UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

CRISTIANO PEGORARO CHENET

ANÁLISE DE SOFT ERRORS EM CONVERSORES ANALÓGICO-DIGITAIS E MITIGAÇÃO UTILIZANDO REDUNDÂNCIA E DIVERSIDADE

Porto Alegre
2015

CRISTIANO PEGORARO CHENET

ANÁLISE DE SOFT ERRORS EM CONVERSORES ANALÓGICO-DIGITAIS E MITIGAÇÃO UTILIZANDO REDUNDÂNCIA E DIVERSIDADE

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Tiago Roberto Balen

Porto Alegre

2015

CRISTIANO PEGORARO CHENET

ANÁLISE DE SOFT ERRORS EM CONVERSORES ANALÓGICO-DIGITAIS E MITIGAÇÃO UTILIZANDO REDUNDÂNCIA E DIVERSIDADE

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo orientador e pela banca examinadora.

Orientador: _____

Prof. Dr. Tiago Roberto Balen, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul

Banca examinadora:

Prof. Dr. Gilson Inácio Wirth, PPGEE - UFRGS

Doutor pela Universitaet Dortmund, Alemanha

Prof. Dr. Paolo Rech, PGMICRO - UFRGS

Doutor pela Università di Padova, Itália

Prof. Dra. Fernanda Gusmão de Lima Kastensmidt, PGMICRO - UFRGS

Doutora pela Universidade Federal do Rio Grande do Sul

Coordenador do PPGEE: _____

Prof. Dr. Luís Fernando Alves Pereira

Porto Alegre, maio de 2015.

Dedico este trabalho à minha nona Edvige (em memória). Uma das pessoas mais admiráveis e inspiradoras que conheci.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Hélio e Zita, e ao meu irmão, Felipe, pelo incentivo, apoio, carinho e parceria ao longo dessa etapa de minha formação e de toda minha vida.

Meu agradecimento e reconhecimento ao orientador, prof. Dr. Tiago Roberto Balen, pela minha acolhida na universidade, e dedicação, suporte e ensinamentos fornecidos durante o mestrado. Agradeço também aos demais professores que tive durante a realização das disciplinas, pelos ensinamentos fornecidos.

Agradeço ao Dr. Odair Lelis Gonçalez, pela oportunidade de realizar os experimentos de irradiação no LRI/IEAv e pelo suporte e ensinamentos prestados. Agradeço também ao Eng. Rafael Galhardo Vaz e ao Eng. Evaldo Carlos Fonseca Pereira Junior, também do LRI/IEAv, pelo suporte e presteza na execução do experimento de irradiação. Aproveito para fazer menção à Cypress Semiconductor, pelo bom suporte prestado durante as implementações com o PSoC.

Agradeço aos colegas do LAPROT, em especial ao M. Sc. Alan Carlos Junior Rossetto, M. Sc. Alisson Jamie Cruz Lanot, M. Sc. Paulo César Comassetto de Aguirre, M. Sc. Thiago Hanna Both e Gustavo dos Santos Fernandes, pelo companheirismo, discussões e conhecimentos trocados.

Agradeço aos meus grandes amigos Felipe de Sousa Gonçalves e Isaque Gomes Correa. Ao Felipe por ser um dos encorajadores do meu mestrado, e ao Isaque pelas diversas horas de bate-papo que ajudaram a amenizar as angústias nesse período. Agradeço também aos demais, que de alguma forma contribuíram comigo.

Por fim, agradeço também à Universidade Federal do Rio Grande do Sul, ao Programa de Pós-Graduação em Engenharia Elétrica, à FAPERGS e à CAPES, por propiciarem essa etapa de minha formação e a realização desse trabalho.

RESUMO

Este trabalho aborda os *soft errors* em conversores de dados analógico-digitais e a mitigação usando redundância e diversidade. Nas tecnologias CMOS recentes, os efeitos singulares (SEEs, *Single Event Effects*) são um grupo de efeitos da radiação espacial que afetam a confiabilidade e disponibilidade dos sistemas. Os *soft errors* são SEEs que não danificam diretamente o sistema e podem ser posteriormente corrigidos. Seus principais subgrupos são o *Single Event Upset* (SEU), o *Single Event Transient* (SET) e o *Single Event Functional Interrupt* (SEFI). Uma das técnicas em nível de sistema amplamente usadas para proteger os circuitos eletrônicos desses efeitos é a Redundância Modular Tripla (TMR, *Triple Modular Redundancy*), que pode ainda ser melhorada com a adição da técnica de diversidade. Nesse contexto, esse trabalho adota um esquema baseado nessas duas técnicas para a implementação de um sistema de aquisição de dados (SAD) analógico-digital. Seus objetivos são observar o comportamento dos conversores de dados frente aos *soft errors* e avaliar a eficácia de um sistema baseado em TMR e diversidade espacial-temporal contra esses efeitos da radiação. A implementação desse SAD em um SoC (*System-on-Chip*) da Cypress Semiconductor, chamado PSoC 5LP e fabricado em tecnologia CMOS de 130 nm, propiciou a realização de dois estudos: no primeiro, é realizada a irradiação com nêutrons, caso de particular interesse para os equipamentos eletrônicos embarcados em aviões; e no segundo, são realizadas injeções de falhas por *software* e em tempo de execução nos registradores de controle dos periféricos e na SRAM do PSoC 5LP. O resultado da irradiação do primeiro estudo foi a não observância de erros, o que impediu cumprir os objetivos propostos para esse teste. Essa situação permitiu duas observações principais: primeiro, o fluxo de nêutrons do experimento é uma característica fundamental que impacta na capacidade de se observar os efeitos da radiação, principalmente quando a seção de choque do circuito em análise é baixa; e segundo, de que a probabilidade de ocorrerem mascaramentos de SETs nos circuitos combinacionais e analógicos é elevada, o que contribui significativamente para reduzir a sensibilidade desses circuitos. Para avaliar a eficácia do sistema baseado em TMR e diversidade espacial-temporal foi então realizada uma investigação teórica baseada em análise combinatória, e os resultados mostraram que a adição de diversidade temporal gera, em comparação ao TMR clássico, um ganho significativo na tolerância de falhas duplas e múltiplas, ao preço de um aumento do atraso do circuito. Os resultados das injeções de falhas por *software* e em tempo de execução nos registradores de controle dos periféricos e na SRAM mostraram que apenas um baixo percentual das falhas injetadas é detectado na forma de erros, convergindo para a justificativa de que os mascaramentos foram determinantes para a não observância de erros no primeiro estudo, de injeção de falhas por radiação. Também verificou-se que os registradores de controle dos periféricos são mais importantes no nível de aplicação do que os dados da memória SRAM. Considerações sobre a auto injeção de falhas e auto monitoramento sugerem que a utilização desses conceitos pode trazer diversas limitações e complicadores aos testes.

Palavras-chave: *Single Event Effects* (SEE). *Soft errors*. *Triple Modular Redundancy* (TMR). Redundância e diversidade. *Programmable System-on-Chip* (PSoC). Conversores analógico-digitais.

ABSTRACT

The present thesis addresses the soft errors in analog-to-digital data converters and mitigation of such errors using redundancy and diversity. In modern CMOS technologies, the Single Event Effects (SEEs) comprises an important group of space radiation effects that influence the reliability and availability of the systems. Soft errors are SEEs that do not directly damage the system and that can be further corrected. Their main subgroups are the Single Event Upset (SEU), the Single Event Transient (SET) and the Single Event Functional Interrupt (SEFI). One of the system level techniques broadly used to protect the electronic circuits against these effects is the Triple Modular Redundancy (TMR), which may be improved with the addition of the diversity technique. In this context, this work proposes a scheme based on these two techniques to implement a tolerant analog-to-digital data acquisition system (DAS). The main objectives are to observe the behavior of the data converters under soft errors, and evaluate the effectiveness of a system based on TMR and spatial-temporal diversity on mitigating these radiation effects. The implementation of this DAS in a Programmable SoC (System-on-Chip) from Cypress Semiconductor (PSoC 5LP) manufactured in 130 nm CMOS, allowed the development of two studies. In the first one, an irradiation with neutrons is performed, case of particular interest to electronic equipment embedded on planes. In the second study, runtime software fault injections are performed at the peripheral control registers and SRAM of the studied device. As a result from irradiation on the first study no errors were found, what does not allowed meet the objectives of this test. This situation allow two main observations: first, the neutron flux of the experiment is a key feature that influences the ability to observe the radiation effects, mainly when the cross section of the circuit in analysis is low; and second, the probability of occurring SETs masking in combinational and analog circuits is high, which contributes significantly to reduce the sensibility of these circuits. To evaluate the effectiveness of a system based on TMR and spatial-temporal diversity then was performed a theoretical investigation based on combinatorial analysis, and the results show that the addition of temporal diversity generates a significant gain in tolerating double and multiple faults, if compared to the classical TMR, at the price of an increase in the circuit delay. The results of the second study, performed by runtime software fault injections at the peripheral control registers and SRAM, showed that only a low percentage of injected faults is detected as errors, according to the justification that no errors were found on irradiation of neutrons due to masking. Also was verified that at the application level the peripheral control registers are more important than the data stored in the SRAM memory. Considerations for faults self-injection and self-monitoring were done, suggesting that the use of these concepts may bring numerous limitations to the test.

Keywords: Single Event Effects (SEE). Soft errors. Triple Modular Redundancy (TMR). Redundancy and diversity. Programmable System-on-Chip (PSoC). Analog-to-digital converters.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADC: Analog to Digital Converter

AE: Auxiliary Equipment

BOX: Buried Oxide

BPSG: Borophosphosilicate Glass

CME: Coronal Mass Ejection

CMOS: Complementary Metal-Oxide-Semiconductor

CRC: Cyclic Redundancy Check

DD: Displacement Damage

DICE: Dual Interlocked Storage Cell

DMA: Direct Memory Access

DMOSFET: Double-Diffused Metal-Oxide-Semiconductor Field Effect Transistor

DRAM: Dynamic Random Access Memory

DUT: Device Under Test

EDAC: Error Detection and Correction

ESA: European Space Agency

FG: Floating Gate

FIT: Failure in Time

FPGA: Field Programmable Gate Array

GSPA: Gerador de Sequência Pseudo-Aleatória

HBD: Hardening by Design

IO: Input-Output

JEDEC: Joint Electron Device Engineering Council

LET: Linear Energy Transfer

MBU: Multiple Bit Upset

MCU: Multiple Cell Upset

MOS: Metal-Oxide-Semiconductor

MOSFET: Metal-Oxide-Semiconductor Field Effect Transistor

NASA: National Aeronautics and Space Administration

OPAMP: Operational Amplifier

PLL: Phase-Locked Loop

PRS: Pseudo Random Sequence

PSoC: Programmable System-on-Chip

RCG: Raios Cósmicos Galácticos

SAD: Sistema de Aquisição de Dados

SAR: Successive Approximations Register

SBU: Single Bit Upset

SEB: Single Event Burnout

SEE: Single Event Effects

SEFI: Single Event Functional Interrupt

SEGR: Single Event Gate Rupture

SEL: Single Event Latchup

SER: Soft Error Rate

SES: Single Event Induced Snap-Back

SET: Single Event Transient

SEU: Single Event Upset

SHE: Single Hard Error

SoC: System-on-Chip

SOI: Silicon on Insulator

SRAM: Static Random Access Memory

SSA: South Atlantic Anomaly

TID: Total Ionizing Dose

TMR: Triple Modular Redundancy

SUMÁRIO

1 INTRODUÇÃO

O estudo dos efeitos da radiação espacial nos circuitos eletrônicos teve início com a exploração espacial. O primeiro veículo espacial em que esses efeitos foram observados foi o satélite Telstar, lançado em julho de 1962. Quatro meses após, alguns transistores falharam e em fevereiro de 1963 o satélite foi perdido devido à degradação de diodos no decodificador de comandos (ECOFFET, 2007). A partir desse período os efeitos da radiação espacial nos circuitos eletrônicos passaram a ser estudados pela comunidade científica, agências espaciais e órgãos militares (BALEN, 2010).

Esses efeitos são basicamente divididos em três grupos: dose ionizante total (TID, *Total Ionizing Dose*), danos por deslocamento (DD, *Displacement Damage*) e efeitos singulares (SEEs, *Single Event Effects*). Atualmente os SEEs têm se tornado o calcanhar de Aquiles para a confiabilidade das tecnologias CMOS (*Complementary Metal-Oxide-Semiconductor*) recentes (DODD et al., 2010), mesmo ao nível do solo, onde a intensidade da radiação espacial é reduzida. Os principais motivadores desse comportamento são: a redução das dimensões dos transistores (*technology scaling*), que faz a quantidade de carga que representa a informação ser reduzida, aumentando a sensibilidade do dispositivo CMOS; o aumento da velocidade de operação dos sistemas, que faz com que um transiente devido a um SEE se propague pelos circuitos com maior facilidade; e a redução da tensão de alimentação dos circuitos integrados, que reduz a carga crítica e aumenta a probabilidade de ocorrência de um erro devido a um SEE (DODD et al., 2010).

Os SEEs são divididos em dois grupos básicos: erros suaves (*soft errors*) e erros catastróficos (*hard errors*). Os principais SEEs no contexto dos *soft errors* são o *Single Event Upset* (SEU), o *Single Event Functional Interrupt* (SEFI) e o *Single Event Transient* (SET). Atualmente os *soft errors* são os fenômenos mais importantes se comparados inclusive a todos os outros fenômenos juntos que afetam a confiabilidade dos circuitos eletrônicos (ZIEGLER apud PETERSEN, 2011). O agravamento da sensibilidade das tecnologias CMOS recentes aos *soft errors* é frequentemente mostrado através da carga mínima necessária para causar um *upset* e da taxa com que os *soft errors* ocorrem, ambas em função das dimensões dos transistores.

A necessidade de proteger os circuitos eletrônicos contra a radiação espacial deu origem a diversas técnicas, as quais são divididas em técnicas em nível de dispositivo, em nível de circuito e em nível de sistema. As técnicas em nível de sistema são indicadas quando: são necessários mecanismos de recuperação *on-line* de erros; as técnicas em nível de dispositivo e circuito, geralmente voltadas para a prevenção de erros, não são suficientes nas tecnologias

CMOS recentes; as técnicas em nível de dispositivo e circuito têm um custo elevado (WANG; AGRAWAL, 2008) e quando não existe o acesso em nível de dispositivo e circuito, por exemplo em caso de uso de componentes comerciais (DODD et al., 2010). Uma das técnicas em nível de sistema amplamente usada é a Redundância Modular Tripla (TMR, *Triple Modular Redundancy*), que pode ainda ser melhorada com a adição da técnica de diversidade.

TMR e diversidade são usados principalmente em aplicações críticas, em que um erro pode resultar na perda de vidas ou na perda de investimentos elevados. A indústria de transporte aéreo comercial é hoje um exemplo clássico de utilização dessas duas técnicas. O computador de controle de voo do Boeing 777 é composto por três variações de *software* e *hardware* projetados para a mesma especificação. Cada variação é chamada de linha (do inglês, *lane*). O *hardware* para cada linha é o mesmo, exceto pelo processador. Os processadores são o AMD29050, o Motorola 68040 e o Intel80486. Cada linha tem programa, dados e entradas e saídas que não são compartilhadas com as outras linhas. A linguagem do código fonte para os três *softwares* é a mesma, exceto para algumas diferenças dependentes do *hardware*, mas a origem dos compiladores é diferente (RITER, 1995). Já o A320 da Airbus usa quatro computadores redundantes no seu sistema de controle de voo. Eles consistem em dois processadores diferentes e quatro *softwares* diferentes, feitos por duas empresas (BRIERE; TRAVERSE, 1993).

Nesse contexto de elevado impacto do SEU, SET e SEFI na confiabilidade e disponibilidade dos circuitos eletrônicos CMOS recentes e de ampla utilização das técnicas de TMR e diversidade em aplicações críticas, esse trabalho adotou um esquema baseado nessas duas técnicas de mitigação para a implementação de um sistema de aquisição de dados (SAD) analógico-digital. Os objetivos com esse SAD são observar o comportamento dos conversores de dados frente a SEU, SET e SEFI e avaliar a eficácia de um sistema baseado em TMR e diversidade espacial-temporal contra esses efeitos da radiação. A implementação desse SAD em um SoC (*System-on-Chip*) da Cypress Semiconductor, chamado PSoC (*Programmable System-on-Chip)* 5LP e fabricado em tecnologia CMOS de 130 nm, propiciou a realização de dois estudos com a intenção de atingir os objetivos mencionados. No primeiro estudo são realizadas injeções de falhas com irradiação de nêutrons. Esse caso é de particular interesse para os equipamentos eletrônicos embarcados em aviões, dado que as altitudes de voo dos aviões comerciais e militares (faixa aproximada de 10.000 a 20.000 m) possui elevado fluxo de nêutrons (TABER; NORMAND, 1993 e FEDERICO, 2011), provenientes da interação dos Raios Cósmicos Galácticos (RCG) com átomos de nitrogênio e oxigênio na atmosfera terrestre. No segundo estudo são realizadas injeções de falhas por *software* e em tempo de execução nos

registradores de controle dos periféricos e na SRAM do PSOC 5LP, uma maneira encontrada para melhorar a controlabilidade e reprodutibilidade dos testes.

Esse trabalho é organizado da seguinte forma: a seção 2 apresenta as fontes da radiação especial, fornece breves noções dos efeitos de TID e DD e discute em detalhes os principais SEEs, tanto os causadores de *soft errors* quanto os causadores de *hard errors*; a seção 3 revisa as principais técnicas de proteção dos circuitos eletrônicos aos *soft errors*, incluindo as técnicas em nível de dispositivo, circuito e sistema; a seção 4 apresenta o estudo com injeção de falhas por irradiação de nêutrons, e uma investigação teórica para avaliar a eficácia de um sistema baseado em TMR e diversidade espacial-temporal; a seção 5 apresenta o estudo com injeções de falhas por *software* e em tempo de execução nos registradores de controle e SRAM do PSoC 5LP; e finalmente a seção 6 apresenta as considerações finais.

2 A RADIAÇÃO ESPACIAL E OS CIRCUITOS ELETRÔNICOS

O conhecimento das fontes da radiação espacial e principalmente dos seus efeitos nos circuitos eletrônicos é primordial para garantir a confiabilidade e disponibilidade dos sistemas modernos, especialmente aqueles dedicados às aplicações críticas. A seção 2.1 apresenta as fontes da radiação espacial, a seção 2.2 introduz de maneira breve todos os possíveis efeitos nos circuitos eletrônicos e a seção 2.3 detalha os efeitos singulares, parte dos quais são o foco desse trabalho.

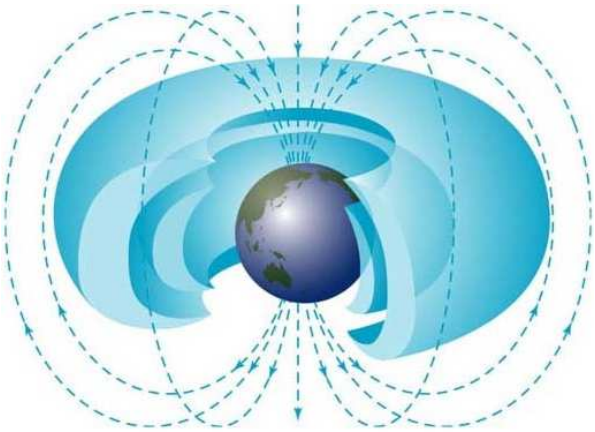
2.1 Fontes da radiação espacial

A radiação espacial que pode causar efeitos indesejáveis nos circuitos eletrônicos é dita ionizante, porque possui energia suficiente para retirar elétrons de átomos, criando íons. Pertencem a esse grupo as partículas como elétrons, prótons, nêutrons, partículas alfa e íons pesados, além de alguns tipos de radiação eletromagnética, como raios-x e raios-gama (BALEN, 2010). As principais fontes dessas partículas e radiações eletromagnéticas são os cinturões de Van Allen, a atividade solar e os RCGs.

2.1.1 Cinturões de Van Allen

Os cinturões de Van Allen são regiões do espaço em torno da terra, que através do campo magnético terrestre, aprisionam prótons e elétrons provenientes do vento solar. Íons pesados também são aprisionados, entretanto sua pequena abundância e baixa energia os fazem

Figura 2.1 - Ilustração dos cinturões de Van Allen.

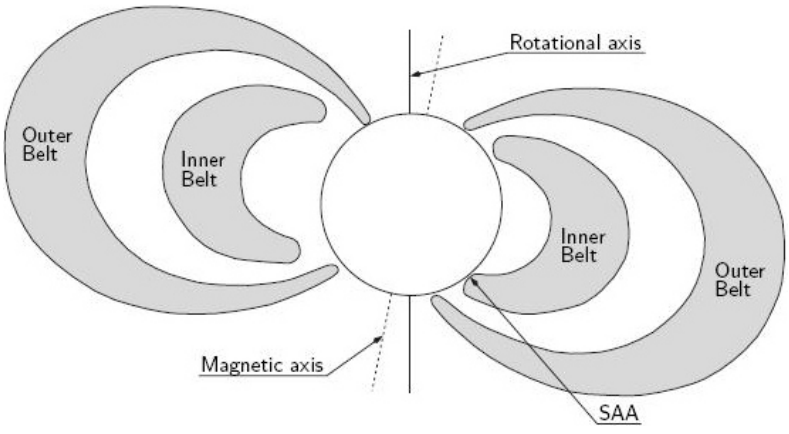


Fonte: Adaptado de NATIONAL AERONAUTICS AND SPACE ADMINISTRATION, 2014a.

desprezíveis na geração de efeitos nos circuitos eletrônicos (BARTH; DYER; STASSINOPOULOS; 2003; PETERSEN, 2011). A figura 2.1 mostra uma ilustração dos cinturões de Van Allen, em que é possível observar um cinturão interno e um externo. O primeiro contém elétrons com espectro de energia de até 5 MeV e localiza-se aproximadamente em altitudes entre 100 km e 10.000 km. Já o cinturão externo contém elétrons de até 7 MeV e localiza-se aproximadamente em altitudes entre 20.000 km e 60.000 km (STASSINOPOULOS; RAYMOND, 1988). Temporariamente, outros cinturões de radiação podem surgir após tempestades magnéticas (BOUDENOT, 2007).

Ainda relacionado ao campo magnético terrestre e aos cinturões de Van Allen, existe a Anomalia Magnética do Atlântico Sul (SSA, *South Atlantic Anomaly*), mostrada na figura 2.2. Consiste em uma aproximação das linhas de campo magnético sobre o sul do Brasil, fazendo com que o cinturão interno de Van Allen se aproxime da superfície da Terra, em altitudes aproximadas de 300 km a 1.000 km (BARTH; DYER; STASSINOPOULOS; 2003). Essa anomalia ocorre devido à diferença entre o centro do dipolo magnético e o centro geográfico da Terra, tendo como referenciais o eixo magnético e o eixo de rotação. As partículas aprisionadas na SAA são elétrons (energia maior que 1 MeV) e prótons (energia maior que 10 MeV) (HEYNDERICKX et al., 1996). Essa anomalia é a maior responsável pela radiação aprisionada recebida pelos satélites de baixa órbita (STASSINOPOULOS; RAYMOND, 1988).

Figura 2.2 - Ilustração dos cinturões de Van Allen e sua aproximação da Terra através da Anomalia Magnética do Atlântico Sul.

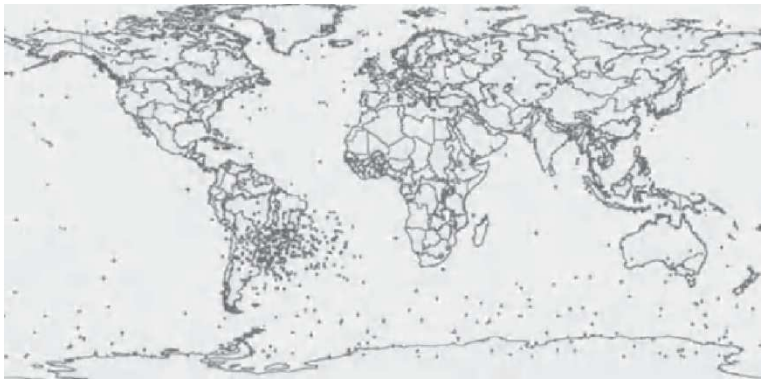


Fonte: EUROPEAN SPACE AGENCY, 2014.

A figura 2.3 mostra a localização em que foram reportados *Single Event Upsets* (SEUs) no satélite TAOS. Os SEUs são efeitos que são detalhados mais adiante neste trabalho. A observação de aproximadamente 1300 SEUs em um computador deste satélite mostra que 50%

ocorreu na SAA, enquanto apenas 5% do tempo em órbita foi gasto nessa região (AEROSPACE apud PETERSEN, 2011). Esse comportamento evidencia o elevado fluxo de partículas ionizantes na região da SAA.

Figura 2.3 - Localização da ocorrência de Single Event Upsets no satélite TAOS, evidenciando o elevado fluxo de partículas ionizantes na região da SAA.



Fonte: AEROSPACE apud PETERSEN, 2011.

2.1.2 Atividade solar

O sol é o grande responsável pelo ambiente de partículas carregadas nas regiões próximas à Terra (PETERSEN, 2011). Dois fenômenos de tempestade que ocorrem nele são os responsáveis pela emissão de partículas: as explosões solares (*solar flares*) e as ejeções de massa coronal (CMEs, *Coronal Mass Ejections*). As explosões solares são intensos lançamentos de energia envolvendo rompimento e religação de fortes linhas de campo magnético (BARTH, 1997). Emitem íons de alta energia (dezenas de MeV a centenas de GeV), além de partículas alfa e elétrons. Já as CMEs são as responsáveis pela formação do vento solar. Nelas, a alta temperatura da coroa solar introduz energia suficiente para que elétrons escapem do campo gravitacional do sol. O efeito dessas ejeções de elétrons é um desbalanceamento de carga que resulta também na ejeção de prótons e íons pesados. A composição do vento solar é de aproximadamente 95% de prótons, 4% de íons de Hélio e 1% de outros íons pesados, além de um número de elétrons necessário para tornar o vento solar neutro (BOUDENOT, 2007).

2.1.3 Raios Cósmicos Galácticos

Embora chamados de raios, os RCGs são partículas com alta energia provenientes de fora

do sistema solar. Existem modelos plausíveis sobre como são produzidos, mas sua origem ainda está em debate. Os cientistas acreditam que essas partículas se propagam através de todo espaço que não é ocupado por material denso. A radiação galáctica consiste de íons de todos os elementos da tabela periódica, e são compostos por aproximadamente 83% de prótons, 13% de partículas alfa (íons de núcleos de Hélio formados por dois prótons e dois nêutrons), 3% de elétrons e 1% de íons de núcleos pesados (BARTH, 1997). Possui energia típica maior que 10 GeV por núcleon e chega às regiões próximas à Terra com aproximadamente 1 GeV por núcleon. O fluxo de RCGs fora da magnetosfera, em distâncias equivalentes à distância Terra-sol, é de aproximadamente 4 partículas/cm^{2}s (STASSINOPOULOS; RAYMOND, 1988).

A atividade do sol é cíclica e tem influência sobre o fluxo dos RCGs. O período dessa atividade dura aproximadamente 11 anos, no qual durante 7 anos a atividade do sol atinge níveis máximos e durante 4 anos atinge níveis mínimos. Como os RCGs precisam “lutar” contra o vento solar para atingir o espaço interplanetário, seu máximo fluxo ocorre durante o período de mínimo da atividade solar (PETERSEN, 2011).

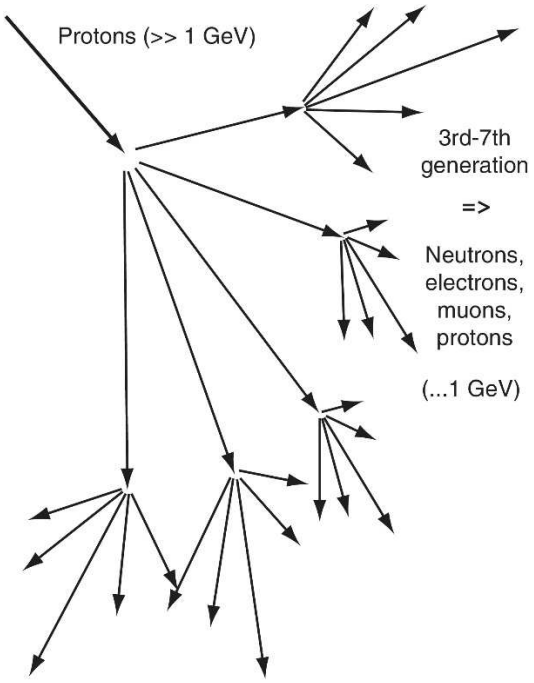
Considerando a composição de partículas dos RCGs, para a geração de efeitos danosos nos dispositivos e circuitos eletrônicos são significativos apenas os prótons e íons. Quanto aos elétrons, sua densidade é ordens de grandeza menor que sua densidade no vento solar, e portanto não são levados em conta (BARTH, 1997).

Os RCGs também dão origem às partículas secundárias quando entram na atmosfera terrestre. Interagem com átomos de nitrogênio e oxigênio e o resultado é uma cascata composta por prótons, elétrons, nêutrons, íons pesados, múons e píons. Esse fenômeno é conhecido como “chuveiro” de partículas, ilustrado na figura 2.4. Em termos de efeitos da radiação na atmosfera, o mais importante produto desse chuveiro de partículas são os nêutrons. Eles são mensuráveis à 330 km de altitude, e sua densidade aumenta com o decremento da altitude, alcançando a saturação em aproximadamente 20 km. Ao nível do mar, a densidade de nêutrons é aproximadamente 1/500 da densidade de saturação (BARTH; DYER; STASSINOPOULOS, 2003).

O fluxo de nêutrons ao nível do mar em função de sua energia é mostrado na figura 2.5. Verifica-se que nêutrons de maiores energias apresentam fluxos reduzidos ao nível do mar. O fluxo de nêutrons ao nível do mar em Nova York (aproximadamente 14 nêutrons/cm^2.h) é usado com frequência como grandeza comparativa a fluxos observados em outras regiões ou altitudes.

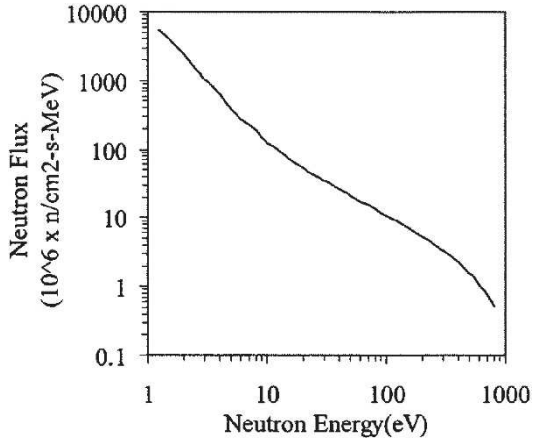
Ao nível do mar, os nêutrons provenientes dos RCGs podem interagir com Boro, usado como dopante tipo p e principalmente no dielétrico BPSG (*Borophosphosilicate glass*), presente no encapsulamento de alguns circuitos integrados. Dessa interação resulta um fóton

Figura 2.4 - Estrutura do chuveiro de partículas formado pelos Raios Cósmicos Galácticos.



Fonte: LERAY apud PETERSEN, 2011.

Figura 2.5 - Fluxo de nêutrons ao nível do mar em função da sua energia.



Fonte: ZIEGLER apud BAUMANN, 2001.

gama, um núcleo de Lítio e uma partícula alfa, sendo estes dois últimos capazes de ionizar materiais do circuito integrado.

2.2 Efeitos da radiação em dispositivos eletrônicos

Os efeitos da radiação em dispositivos eletrônicos são basicamente três:

a) dose ionizante total (TID, *Total Ionizing Dose*): é um efeito de longo prazo, devido

ao acúmulo de energia depositada nos materiais que compõem o circuito integrado. Tipicamente gera falhas ou variações nos parâmetros dos dispositivos, ou falhas funcionais (NATIONAL AERONAUTICS AND SPACE ADMINISTRATION, 2014b);

b) danos por deslocamento (DD, *Displacement Damage*): assim como TID, também é um efeito de longo prazo e cumulativo, mas diferencia-se pelo mecanismo físico distinto. É um efeito resultado da interação entre as partículas incidentes e núcleos atômicos da rede cristalina do semicondutor, deslocando-os de sua posição original (VIRMONTOIS et al., 2010). Podem alterar as propriedades dos materiais e o desempenho dos dispositivos;
c) efeitos singulares (SEEs, *Single Event Effects*): são efeitos que ocorrem devido ao impacto de partículas fortemente ionizantes no semicondutor, ionizando-o densamente e podendo provocar um pulso de corrente (WANG; AGRAWAL, 2008).

Os diversos SEEs podem ser classificados em duas categorias, com base nos erros gerados: erros suaves (*soft errors*) e erros catastróficos (*hard errors*). O termo *soft errors* foi introduzido por May e Woods, em 1978. Ele refere-se a SEEs que não danificam diretamente o sistema e podem ser posteriormente corrigidos. Comumente os *soft errors* referem-se a um erro causado por pulsos de origem radioativa ou eletromagnética, e não associados ao envelhecimento ou aos defeitos físicos introduzidos durante o processo de fabricação (MUNTEANU; AUTRAN, 2008). Os principais SEEs atribuídos a essa categoria são o *Single Event Upset* (SEU), o *Single Event Functional Interrupt* (SEFI) e o *Single Event Transient* (SET).

Hard errors são mudanças irreversíveis na operação que são tipicamente associadas com danos permanentes em um ou mais dispositivo ou circuito. O erro é "*hard*" (catastrófico) porque dados são perdidos e o dispositivo ou circuito não retorna ao funcionamento adequado após um *reset* na alimentação (MUNTEANU; AUTRAN, 2008). Os principais SEEs relacionados a essa categoria são: *Single Event Latchup* (SEL), *Single Event Burnout* (SEB), *Single Event Gate Rupture* (SEGR), *Single Event Induced Snap-Back* (SES) e *Single Hard Error* (SHE).

Os SEEs são de interesse neste trabalho, e portanto são detalhados na seção 2.3. Os efeitos de TID e DD não são abordados. A tabela 2.1 relaciona as fontes de radiação espacial, detalhadas na seção 2.1, com os efeitos da radiação em dispositivos eletrônicos definidos nessa seção.

Tabela 2.1 - Relação entre os efeitos da radiação espacial em dispositivos eletrônicos e sua fonte.

Efeito	Origem	Partícula
Efeitos singulares	Raios Cósmicos Galácticos	Prótons
		Íons
		Nêutrons secundários
	Atividade solar	Prótons
		Íons pesados
	Cinturões de Van Allen	Prótons
		Íons pesados
Danos por deslocamento	Atividade solar	Prótons
	Cinturões de Van Allen	Prótons
Dose ionizante total	Atividade solar	Prótons
	Cinturões de Van Allen	Prótons
		Elétrons

Fonte: Autoria própria. Baseado em BARTH, 1997 e ECOFFET, 2007.

2.3 Efeitos singulares

Os efeitos singulares são causados pela geração de cargas, devido ao impacto de uma única partícula de elevada energia em um nó sensível do circuito eletrônico. Foram mencionados pela primeira vez em um artigo de Wallmark e Marcus, em 1962. Nele os autores preveem a ocorrência de um SEU em dispositivos microeletrônicos devido aos raios cósmicos e estimam a menor dimensão de um semicondutor para contornar esse efeito. Em 1975, Binder, Smith e Holman reportam pela primeira vez a ocorrência de SEU em um satélite em operação, no qual foram observados ao longo de 17 anos 4 SEUs em flip-flops JK bipolares. Nas décadas seguintes um crescente número de pesquisas relacionadas aos SEEs foram realizadas (DODD; MASSENGILL, 2003). A seguir são detalhados os mecanismos físicos que dão origem aos SEEs, possibilitando a melhor compreensão dos motivos que tornam esses efeitos importantes.

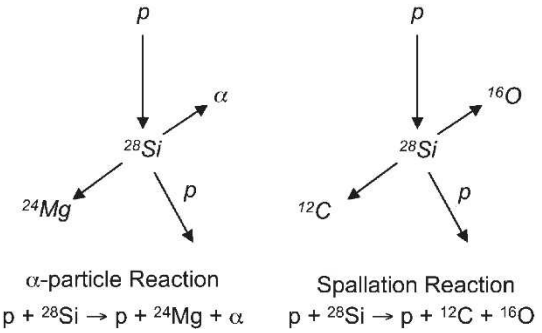
Existem dois processos físicos básicos responsáveis pela geração de SEEs: deposição de carga e coleta de carga. Eles ocorrem em sequência, conforme a ordem anunciada. A deposição de carga, pode ocorrer através de dois métodos diferentes: ionização direta através de partículas radioativas incidentes ou ionização indireta através de partículas secundárias criadas por reações nucleares entre a partícula radioativa incidente e a estrutura do dispositivo (DODD; MASSENGILL, 2003).

Na ionização direta, quando uma partícula energética percorre o semicondutor, ela gera

pares elétron-lacuna ao longo do caminho e gradualmente perde energia. Quando toda sua energia é perdida, a partícula repousa no semicondutor, tendo percorrido um caminho de comprimento determinado. O termo *Linear Energy Transfer* (LET) é utilizado para descrever a quantidade de energia perdida pela partícula por unidade de caminho percorrido. A unidade de LET é o MeV.cm^2/mg, pois a energia perdida por unidade de caminho (em MeV/cm) é normalizada pela densidade do material alvo (em mg/cm^3), assim LET pode ser expressa independentemente do material alvo. Em silício, um LET de 97 MeV.cm^2/mg corresponde à deposição de carga de 1 pC/µm (DODD; MASSENGILL, 2003).

Na ionização indireta, quando prótons ou nêutrons energéticos entram na estrutura cristalina do semicondutor, eles podem sofrer uma colisão inelástica com os núcleos do material alvo, como mostrado na figura 2.6. Como resultado pode haver a emissão de núcleos filhos (magnésio por exemplo) e de partículas alfa ou raios-gama, ou ainda, numa reação conhecida como *spallation*, a fragmentação do silício em íons de carbono e oxigênio (DODD, 2005). Essas reações depositam energia por ionização direta no caminho percorrido.

Figura 2.6 - Deposição de carga através de ionização indireta do silício.

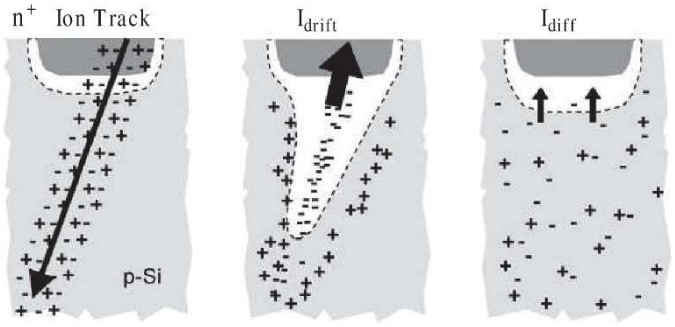


Fonte: DODD, 2005.

Quando partículas energéticas colidem com dispositivos microeletrônicos, as regiões mais sensíveis são geralmente junções p-n reversamente polarizadas. Assim, o processo de deposição de carga produz uma distribuição radial densa de pares elétron-lacuna, conforme mostrado na figura 2.7a. Neste momento dá-se início ao processo de coleta de carga. Se o caminho ionizado atravessa a região de depleção, os portadores são rapidamente coletados pelo campo elétrico, compensando a carga armazenada na junção. Fora da região de depleção, a distribuição de carga desiquilibrada induz a uma distorção de potencial temporária em forma de funil, como mostrado na figura 2.7b, gerando a coleta de carga através de deriva. Quando o funil é deformado, a difusão domina o processo, conforme mostrado na figura 2.7c, até que todos os portadores tenham sido coletados (WANG; AGRAWAL, 2008). A carga coletada é

uma função da energia e trajetória da partícula ionizante, estrutura e dopagem do substrato de silício e do campo elétrico local (BAUMANN apud WANG; AGRAWAL, 2008).

Figura 2.7 - Processos físicos básicos responsáveis pela geração de SEEs. (a) Deposição de carga. (b) Coleta de carga através de deriva. (c) Coleta de carga através de difusão.

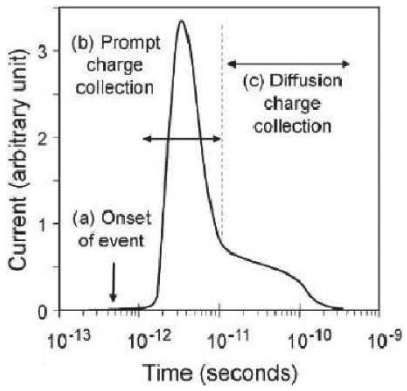


(a) (b) (c)

Fonte: WANG; AGRAWAL, 2008.

A carga transiente coletada devido ao SEE produz um pulso de corrente, como ilustrado na figura 2.8. Sua constante de tempo apresenta valores típicos na faixa de centenas de picossegundos a dezenas de nanossegundos (ZIEGLER; LANFORD, 1981; NARASIMHAM et al., 2007). Para descrever analiticamente o comportamento desse pulso de corrente, em 1982, Messenger propôs um modelo baseado em uma dupla exponencial, conforme mostrado na equação 2.1. I_0 é aproximadamente o máximo valor da corrente devido à coleta de carga, $\tau\alpha$ é a constante de tempo relacionada à coleta de carga da junção e $\tau\beta$ é a constante de tempo relacionada ao estabelecimento do caminho de ionização pela partícula incidente. Nesse modelo de dupla exponencial, a amplitude máxima de corrente é independente da tensão do nó, e assim a coleta de portadores como função da tensão não é precisa. Estimativas no estado da arte

Figura 2.8 - Pulso de corrente induzido pelo SEE como uma função do tempo.



Fonte: BAUMANN apud WANG; AGRAWAL, 2008.

utilizam simulações tridimensionais (YAMAGUSHI et al., 2004; DODD, 1996).

$$I(t) = I_0 \left(e^{-\frac{t}{\tau\alpha}} - e^{-\frac{t}{\tau\beta}} \right) \tag{2.1}$$

A integral do pulso de corrente ocasionado por um SEE representa a carga coletada (Q_{col}) devido a este, conforme representado na equação 2.2. T representa o período de duração do pulso. Se a Q_{col} for maior do que a carga crítica (Q_{crit}), em que a Q_{crit} é usualmente definida como a mínima quantidade de Q_{col} em um nó necessária para causar um *upset* (DODD; MASSENGILL, 2003), um *soft error* ocorre. Além das características do dispositivo e do circuito eletrônico, a Q_{crit} depende também do formato do pulso de corrente.

$$Q_{col} = \int_0^T I(t)dt \tag{2.2}$$

Em termos de unidade de LET existe uma medida análoga à Q_{crit}, conhecida como LET limiar (LET_{th}, *threshold* LET). Representa a mínima carga necessária para causar um *upset* na região mais sensível do circuito (WEAVER; MCMORROW; COHN, 2003). Quando uma partícula atinge um nó sensível do circuito, tal que sua LET seja maior que a LET_{th} daquele nó ($LET_{partícula} > LET_{th}$), a carga coletada pelo nó será maior que sua carga crítica ($Q_{col} > Q_{crit}$), caracterizando um *soft error* (BALEN, 2010).

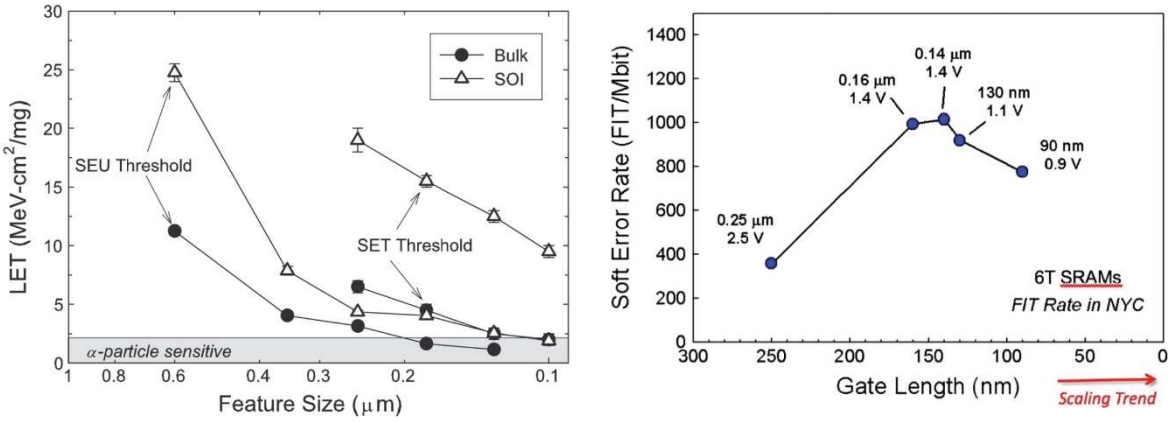
A probabilidade de ocorrência de um SEE em determinado circuito é avaliada através da grandeza conhecida como seção de choque (*cross section*), cujo símbolo é σ. É definida em termos de área, com unidade de cm^2/dispositivo ou cm^2/bit, porque quanto maior o somatório das áreas ativas dos dispositivos, maior a vulnerabilidade do circuito a um SEE. Embora não explícito diretamente, a seção de choque (σ) é também uma função da energia depositada no circuito, e portanto uma função da LET (KOGA; KOLASINSKI; IMAMOTO, 1985). σ é analiticamente expressa conforme a equação 2.3 (KOGA; KOLASINSKI; IMAMOTO, 1985; PETERSEN, 2011), em que N é o número de *upsets* observados e F a fluência do feixe de partículas, que por sua vez, refere-se à quantidade de radiação (partícula) incidente por unidade de tempo. Φ é o ângulo de incidência do feixe de radiação em relação à direção normal. Observa-se pela equação 2.3, que Φ tem grande influência sobre a capacidade da radiação gerar erros devido a SEE.

$$\sigma = \frac{N}{F \cos \Phi} \tag{2.3}$$

A taxa com que os *soft errors* ocorrem é conhecida como *Soft Error Rate* (SER). Quando o ambiente é conhecido ou para um ambiente de referência, a SER pode ser expressa em FIT (*Failure In Time*), sendo que 1 FIT é igual a uma falha em 10^9 horas (GAILLARD, 2011). Em memórias, a SER é geralmente dada em FIT/Mbit ou FIT/dispositivo.

Atualmente os SEEs têm se tornado o calcanhar de Aquiles para a confiabilidade das tecnologias CMOS recentes (DODD et al., 2010), mesmo ao nível do solo, onde a intensidade da radiação espacial é reduzida. A redução das dimensões dos transistores (*technology scaling*) é um dos principais motivos, pois a quantidade de carga que representa a informação é reduzida, assim a sensibilidade do dispositivo CMOS à carga coletada em um SEE aumenta. Outro motivo é o aumento da velocidade de operação dos sistemas, que faz com que um transiente de tensão/corrente induzido pelo SEE tenha maior probabilidade de ser propagado pelo circuito lógico (DODD et al., 2010). Os *soft errors*, de maneira especial, são os fenômenos mais importantes se comparados inclusive a todos os outros fenômenos juntos que afetam a confiabilidade dos circuitos eletrônicos (ZIEGLER apud PETERSEN, 2011). Para se ter uma ideia do agravamento da sensibilidade aos *soft errors* nas tecnologias CMOS recentes, a figura 2.9a mostra a LET limiar em função das dimensões dos transistores, com base em dados extraídos de simulações tridimensionais de circuito de sinais mistos e memória SRAM (*Static*

Figura 2.9 - Aumento da sensibilidade aos soft errors nas tecnologias CMOS recentes. (a) LET limiar em função das dimensões dos transistores. (b) SER em função das dimensões dos transistores.



(a) (b)

Fonte: DODD et al., 2010.

Random Access Memory). A figura 2.9b mostra a SER também em função das dimensões dos transistores para uma memória SRAM comercial exposta a nêutrons no nível do mar.

A seguir são detalhadas as especificidades dos principais SEEs.

2.3.1 Single Event Upset (SEU)

O SEU é caracterizado por uma perturbação induzida pela radiação ionizante que afeta elementos de memória, modificando o estado do bit armazenado. Esse efeito foi reportado pela primeira vez por May e Woods, em 1978, embora a primeira publicação nomeando o efeito ocorreu em 1979, com Guenzer, Wolicki e Allas. Não raras vezes o termo SEU é usado de maneira ambígua (HEIJMEN, 2011): frequentemente refere-se à *soft errors* e eventualmente refere-se aos SEEs, incluindo *soft* e *hard errors*.

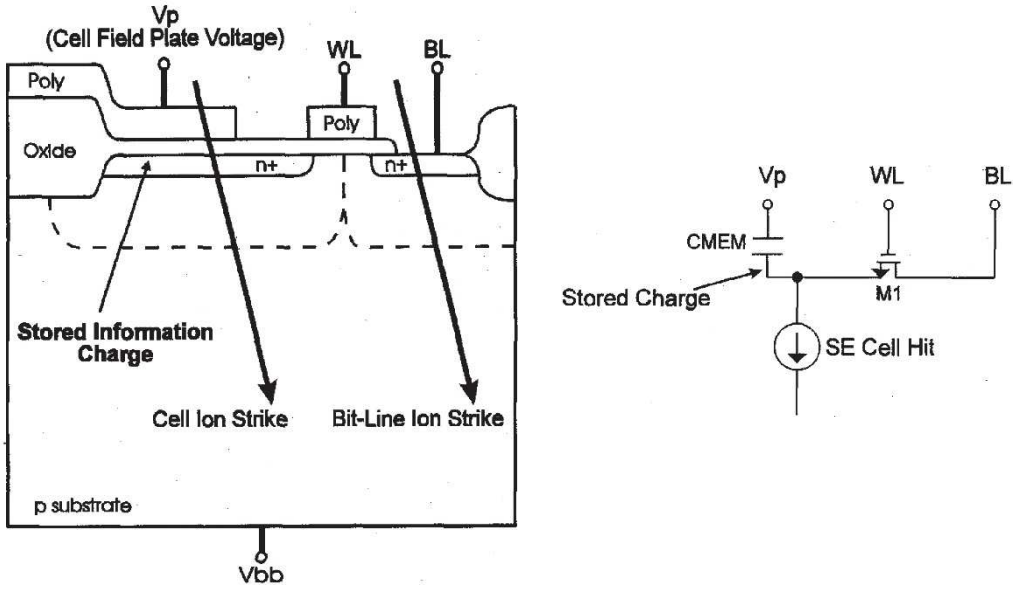
Nos circuitos CMOS, as partes mais sensíveis à SEU são a região do canal de transistores NMOS desligados e a região do dreno de transistores PMOS desligados (WANG; AGRAWAL, 2008). A seguir são detalhados os mecanismos de ação do SEU nas principais memórias.

As memórias DRAM (*Dynamic Random Access Memory*) são baseadas em células binárias, e é no interior destas que se encontra a fonte dominante de erros nesse dispositivo. A informação binária é armazenada em células de 1 transistor através da presença ou ausência de carga em um elemento capacitivo, tendo como referência um elemento idêntico. A figura 2.10 ilustra a célula binária em nível físico e de esquemático. A suscetibilidade da memória DRAM à SEU reside em dois fatores: a inexistência de um mecanismo de atualização (*refreshing*) inerente à carga perdida, ou seja, a inexistência de um mecanismo adicional ao ciclo periódico de atualização característico das memórias DRAM, e o fato de a informação ser armazenada em um elemento passivo, em que não existe regeneração ativa do sinal (MASSENGILL, 1996). Portanto qualquer perturbação se mantém até a correção por um circuito externo.

Os erros nas células DRAM são causados pela colisão de partículas próximo ao capacitor de armazenamento da informação ou da fonte do transistor de acesso ao capacitor. Essa colisão afeta a carga armazenada, através da coleta de elétrons ou lacunas no caminho ionizado e da consequente geração de um pulso de corrente, conforme mostrado na figura 2.10b. Assim, o erro geralmente observado é a mudança de estado de 1 para 0, devido à descarga do capacitor pela corrente estabelecida. Pela natureza do mecanismo, esse erro é independente do *clock* da memória (MASSENGILL, 1996).

Outros mecanismos físicos também provocam *upsets* em memórias DRAM. A colisão de partículas alfa em transistores com canal de dimensões reduzidas induz uma corrente dreno-

Figura 2.10 - Célula DRAM de 1 transistor. (a) Nível físico. (b) Nível de esquemático mostrando o efeito de um SEU.



(a) (b)

Fonte: Adaptado de MASSENGILL, 1996.

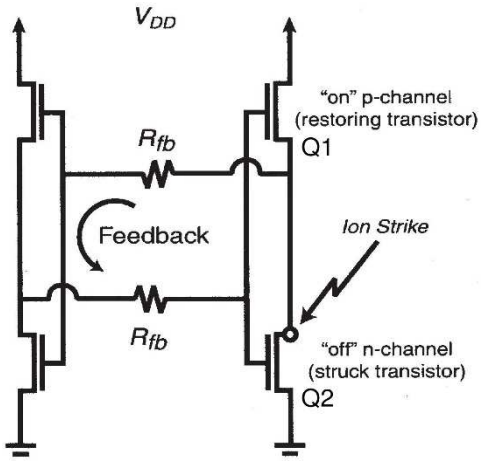
fonte, que provoca a mudança da informação armazenada de estado 0 para 1 (TAKEDA et al., 1989). Erros conhecidos como *Bit-Line Mode Errors* ocorrem devido à coleta de cargas em regiões eletricamente conectadas à célula DRAM, durante o ciclo de leitura da memória (MCPARTLAND, 1981). Neste caso, existe uma dependência dos erros gerados com o *clock* do dispositivo.

Em resumo dois parâmetros relacionam a ocorrência de *upsets* em memórias DRAM: a margem de ruído associada ao sinal representativo do bit e a janela de tempo devido à operação dinâmica. O primeiro é intimamente relacionado ao conceito de carga crítica, explorada no início da seção 2.3 deste trabalho. O segundo relaciona as características temporais da colisão da partícula ionizante com o *clock* da memória. A tecnologia da célula DRAM também é determinante para a suscetibilidade dessa memória à SEU (MASSENGILL, 1996).

O SEU em memórias SRAM é diferente das memórias DRAM em razão da realimentação ativa introduzida pelo par inversor contraposto. No esquemático básico de uma célula SRAM como mostrado na figura 2.11, se uma partícula energética colide próximo ao dreno do transistor Q2, que se encontra desligado, a carga coletada resulta em uma corrente transiente nesse transistor. Pelo transistor Q1, nesse caso denominado de restaurador, flui então uma corrente para balancear a corrente em Q2. Essa corrente que flui através de Q1 induz uma queda de tensão sobre esse transistor, que é a responsável pelo *upset* na célula SRAM (DODD;

MASSENGILL, 2003). Embora nesse exemplo a colisão da partícula ionizante ocorre no dreno de Q2, os drenos dos outros três transistores também são considerados pontos sensíveis. Uma variante significativa é se a junção atingida está dentro de um poço ou do substrato, pois o poço fornece uma barreira de potencial para a carga depositada no substrato. Detalhes do mecanismo em função da localização poço-substrato da junção são encontrados em (DODD et al., 1996).

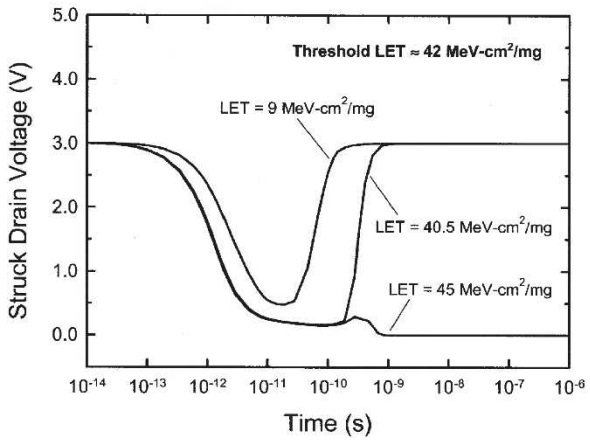
Figura 2.11 - Esquemático básico de uma célula SRAM.



Fonte: DODD; MASSENGILL, 2003.

Partículas abaixo do limiar de *upset* são geralmente suficientes para induzir uma inversão momentânea de uma célula SRAM, embora não caracterize um *upset*. Por exemplo, a figura 2.12 mostra uma tensão de dreno em uma memória SRAM para a colisão de partículas com LET bem abaixo do limiar de *upset*, logo abaixo do limiar e logo acima do limiar. Observa-se

Figura 2.12 - Transiente de tensão de dreno em uma memória SRAM, para colisão com partículas com LET bem abaixo do limiar de upset, logo abaixo do limiar e logo acima do limiar.

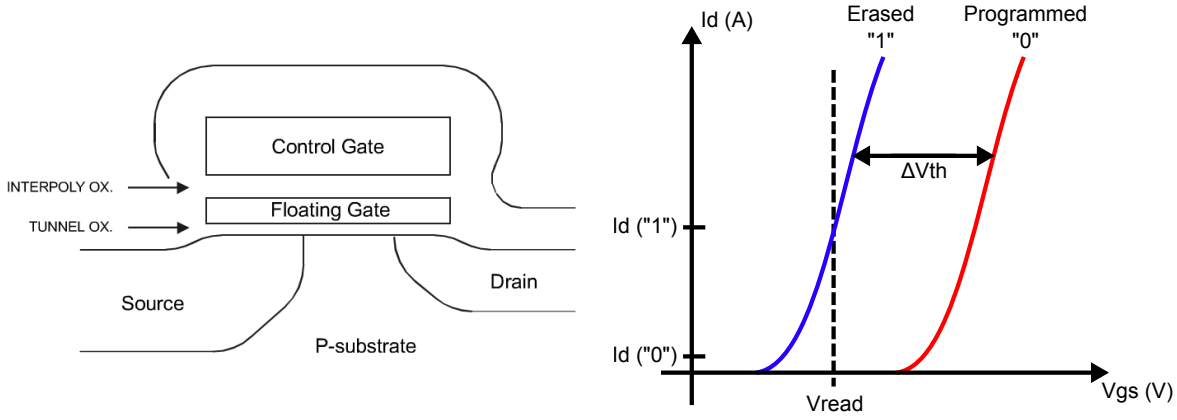


Fonte: DODD; MASSENGILL, 2003.

que mesmo no primeiro caso ocorre um transiente significativo.

Atualmente as memórias Flash são largamente utilizadas, inclusive em aplicações espaciais, devido às suas vantagens de não-volatilidade da informação com a desenergização do dispositivo e elevada densidade de armazenamento de dados por área. Quase a totalidade dessas memórias são baseadas na tecnologia de porta flutuante (FG, Floating Gate) (CAPPELLETTI et al., 1999). Nela a célula de memória consiste em um transistor MOS (*Metal-Oxide-Semiconductor*) com uma FG localizada entre o substrato e a porta, esta última denominada de porta de controle para distinção, conforme mostrado na figura 2.13a. Nos dias atuais a FG é geralmente feita em polisilício, embora também existem dispositivos usando nanocristais ou filme com alta densidade de armadilhas (GERARDIN et al., 2013).

Figura 2.13 - Célula de memória Flash baseada em porta flutuante. (a) Vista em corte da estrutura do dispositivo. (b) Curvas características típicas para o caso de apagado (estado 1) e programado (estado 0).



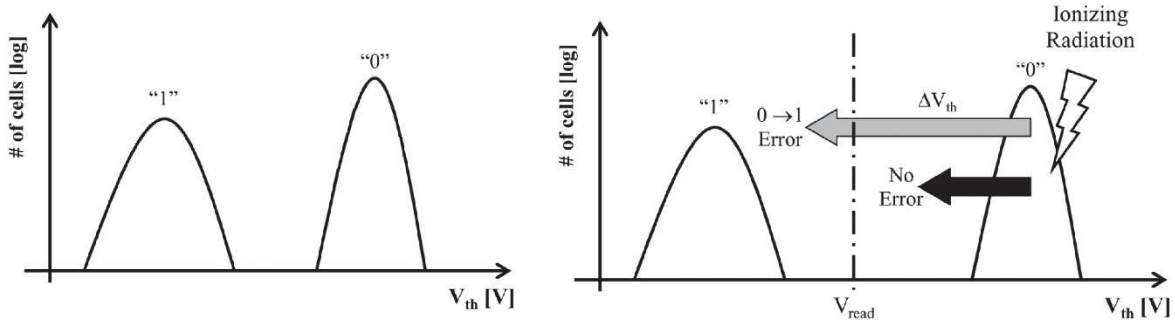
(a) (b)

Fonte: Adaptado de GERARDIN et al., 2013 e PAVAN; LARCHER; MARMIROLLI, 2004.

A FG é um elemento armazenador de carga. Pelo controle da carga armazenada nesse elemento, a tensão de limiar do transistor pode ser alterada, conforme mostrado na figura 2.13b. A situação com carga positiva ou sem carga é convencionalmente referida como apagado (estado 1) e a situação com carga negativa referida como programado (estado 0). A operação de leitura é feita polarizando a porta de controle com uma tensão fixa, entre a tensão de limiar de programado e apagado, e comparando a corrente drenada pela célula com uma célula de referência. A célula apagada drena uma corrente significativa (tipicamente 1 µA) (GERARDIN et al., 2013) enquanto a célula programada drena uma corrente desprezível ou nula. Os parâmetros das células apresentam um espalhamento se levado em consideração uma memória contendo bilhões de células. A figura 2.14a ilustra um exemplo da densidade de células como

uma função da tensão de limiar para programado e apagado.

Figura 2.14 - Alterações das tensões de limiar para apagado (estado 1) e programado (estado 0). (a) Distribuição esperada na análise de muitas células. (b) Influência da radiação ionizante na tensão de limiar.



(a) (b)

Fonte: Adaptado de GERARDIN et al., 2013.

Quando uma célula de porta flutuante de memória Flash é atingida pela radiação ionizante, uma grande alteração da tensão de limiar pode ocasionar um erro. Como mostrado na figura 2.14b, geralmente esse deslocamento traz a tensão de limiar para baixo da tensão de leitura, fazendo a leitura de uma célula que deveria armazenar o estado 0 ser lida como estado 1 (GERARDIN et al., 2013). Existem dois modelos que explicam esse efeito do SEU nas células de FG (GERARDIN et al., 2013). O primeiro é chamado de caminho condutivo transiente (CELLERE et al., 2001; CELLERE et al., 2004), assume que um caminho transiente de fuga é criado e descarrega a FG, e tem encontrado boa concordância com os dados de alteração da tensão de limiar (CELLERE et al., 2004; CELLERE et al., 2006). O segundo modelo, denominado de fluxo de portadores transiente (BUTT; ALAM, 2008), considera a geração de portadores energéticos pela radiação incidente e o fluxo de portadores entrando e saindo da FG devido às correntes de tunelamento. O desbalanceamento entre as correntes geradas acarreta na descarga da FG. Também é encontrada satisfatória concordância desse modelo com dados experimentais (BUTT; ALAM, 2008).

Estudos recentes indicam que o SEU é pouco relevante em memórias Flash produzidas em 50 nm ou em nós tecnológicos mais antigos (CELLERE et al. 2009; GERARDIN et al., 2013; GRÜRMANN et al. 2012). Porém, Grürmann et al. (2012) mostraram uma seção de choque significativa para dispositivos de 25 nm irradiados com íons pesados, sugerindo que o SEU pode representar um problema importante para as memórias Flash de nós tecnológicos recentes. Nos casos em que o SEU é pouco relevante, a maioria das células da memória

experimenta uma alteração da tensão de limiar muito pequena para imediatamente resultar em um *bit-flip*, tornando-se o circuito de controle da memória Flash a maior fonte dos problemas (CELLERE; PACCAGNELLA, 2004). O circuito de controle compreende decodificadores, *buffers*, microcontrolador para gerenciar o complexo algoritmo de programação e circuitos multiplicadores de tensão (*charge pumps*) para gerar as tensões elevadas necessárias à programação e apagamento.

Os SEUs podem ainda ser classificados em subgrupos em função de sua abrangência. *Single Bit Upset* (SBU) ocorre quando a colisão de uma partícula causa um único *bit-flip* (HEIJMEN, 2011). *Multiple Cell Upset* (MCU) causa dois ou mais *bit-flips* e *Multiple Bit Upset* (MBU) causa dois ou mais *bit-flips* na mesma palavra (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Múltiplos erros podem ser criados quando uma partícula cruza regiões sensíveis de diferentes células ou quando os portadores livres gerados por partículas ionizantes são coletados por diferentes junções de transistores de diversas células de memória (GAILLARD, 2011). Neste primeiro caso citado, a incidência da radiação em um ângulo rasante é determinante para potencializar múltiplos erros, conforme reportado por Musseau et al., em 1996. A redução das dimensões dos circuitos integrados nas tecnologias recentes também aumenta a probabilidade de múltiplos erros.

2.3.2 Single Event Functional Interrupt (SEFI)

O termo SEFI foi mencionado pela primeira vez em 1996, pelo Joint Electron Device Engineering Council (JEDEC). É caracterizado pela perda da funcionalidade em circuitos integrados complexos, ocasionada por exemplo, por travamento, ou perturbação em registradores de controle, sinais de *clock* ou sinais de *reset*. Além disso, pertence à classe de *soft errors* pois pode ser recuperado após um *reset* na alimentação, um *reset* funcional ou recarga dos registradores de configuração (GAILLARD, 2011; HEIJMEN, 2011; JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). SEFIs são geralmente reportados em dispositivos como memórias DRAM síncronas, SRAM e Flash, FPGAs (*Field Programmable Gate Array*) baseados em SRAM, microprocessadores e microcontroladores.

Os SEFIs estão associados aos SEUs, porém se diferenciam quando provocam a perda da funcionalidade temporária ou a interrupção da operação normal do circuito afetado. Em alguns casos eles duram por todo o tempo que a alimentação é mantida, enquanto em outros ele duram por um tempo finito.

O artigo de Koga et al., de 1997a, considera que os SEFIs são causados por SEUs em

dispositivos em que não se tem um entendimento detalhado de sua arquitetura, ou em pontos sensíveis do circuito os quais não temos acesso direto. Uma vez que não é possível identificar a localização exata do SEU, é possível apenas observar a falha funcional do dispositivo. Nesse contexto, tem-se uma definição para SEFI um pouco diferente da apresentada inicialmente: é um *upset* para o qual não é possível medir diretamente o bit alterado ou prontamente observar a indicação do *upset*, e consequentemente é detectada apenas a anomalia funcional (KOGA et al., 1997a). Se o mecanismo exato para um *upset* é conhecido, ele não pode ser considerado um SEFI.

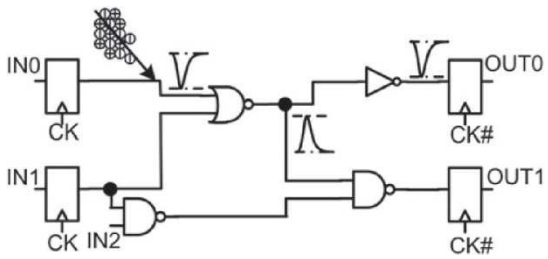
2.3.3 Single Event Transient (SET)

SET refere-se a um pulso transiente em circuitos integrados causado pela passagem de uma única partícula energética (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Pode ou não ser capturado por um elemento de memória (BALEN, 2010), e ocorre tanto em dispositivos digitais (DIEHL et al., 1983) quanto em analógicos (TURFLINGER, 1996). Os erros mais importantes devido à SET ocorrem em circuitos combinacionais e subsistemas analógicos (PETERSEN, 2011).

Nos circuitos combinacionais, mesmo que o pulso transiente não induza uma inversão de bit no nó afetado, ele pode se propagar pelo circuito e quando encontrar um *latch* ou elemento de memória, ser armazenado como um dado incorreto. Tendo como referência o circuito da figura 2.15, o SET deve atender algumas condições para induzir um erro em um elemento de memória (DIEHL et al., 1983):

a) a colisão da partícula ionizante deve produzir um transiente capaz de se propagar no circuito;
b) deve existir um caminho lógico possível para o transiente se propagar até encontrar um *latch* ou elemento de memória;

Figura 2.15 - Ilustração de um SET em um circuito combinacional típico.



Fonte: KARNIK; HAZUCHA; PATEL, 2004.

c) o pulso transiente deve ter suficiente amplitude e duração para alterar o *latch* ou a memória;

d) em lógica síncrona, o transiente deve chegar no *latch* durante a carga deste.

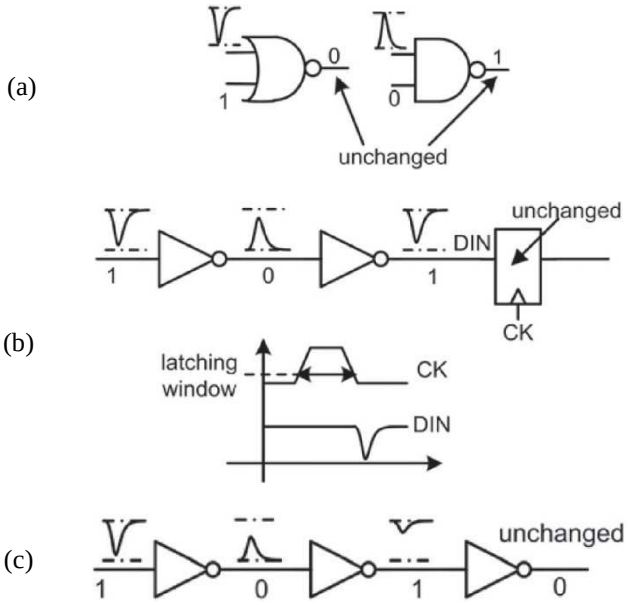
Nos circuitos combinacionais, alguns transientes podem não ser capturados, e portanto não gerar erros. A razão da não captura reside nos mascaramentos, que podem ocorrer das seguintes formas (SHIVAKUMAR et al., 2002):

a) mascaramento lógico: ocorre quando uma partícula colide um nó da lógica combinacional que não afeta a saída, devido ao resultado da porta lógica seguinte ser determinado apenas pelas suas outras entradas. Por exemplo, seguindo a figura 2.16a, se a colisão acontece em uma entrada da porta NAND, e a sua outra entrada estiver no estado dominante 1, a saída permanece inalterada e o transiente é mascarado;

b) mascaramento temporal: ocorre quando o pulso resultante de uma colisão alcança o *latch* em um momento em que ele não captura a sua entrada. Na figura 2.16b, quando o transiente propaga-se em direção ao elemento de memória, a perturbação no nó DIN pode ocorrer fora da janela de captura, e o erro não é capturado;

c) mascaramento elétrico: ocorre quando o pulso resultante de uma colisão é atenuado pelas portas lógicas do caminho percorrido, devido à frequência de corte

Figura 2.16 - Ilustração dos fenômenos de mascaramento em um circuito combinacional. (a) Mascaramento lógico. (b) Mascaramento temporal. (c) Mascaramento elétrico.



Fonte: Adaptado de KARNIK; HAZUCHA; PATEL, 2004.

dos circuitos CMOS ser menor do que a largura de banda do pulso. A amplitude do pulso é reduzida, os tempos de subida e descida aumentam e eventualmente o pulso pode desaparecer, como mostrado na figura 2.16c.

Adicionalmente a esses fenômenos de mascaramento, outros dois fatores são importantes no impacto dos SETs em circuitos combinacionais: a frequência do *clock* e a largura do pulso transiente (BAUMANN apud MUNTEANU; AUTRAN, 2008). Com o aumento da frequência do *clock* existem mais bordas de relógio que podem capturar o pulso (em um dado instante de tempo), e assim a taxa de erros aumenta. A largura do pulso transiente determina a distância que o SET vai percorrer no circuito combinacional e a probabilidade dele ser capturado em um elemento de memória (BUCHNER; BAZE, 2001). Um pulso largo aumenta a probabilidade do SET chegar no *latch* na borda do *clock* de captura. Se o pulso se tornar mais largo que o período do *clock*, então todos os transientes são capturados (GADLAGE et al., 2004).

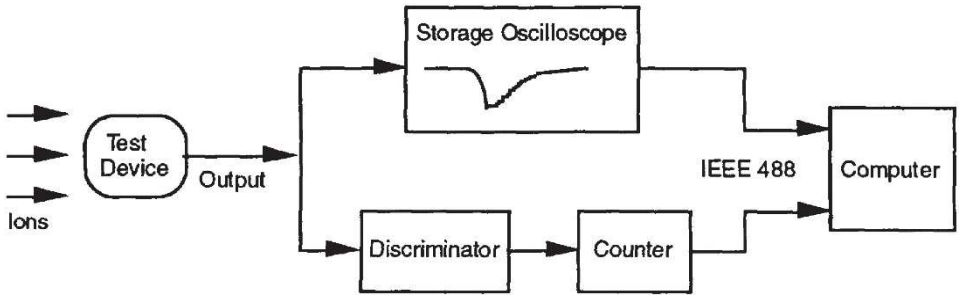
Mitra et al., em 2005, fizeram estimativas quantitativas das contribuições para a SER total em microprocessadores, processadores e controladores de elevada velocidade. Considerando a tecnologia estado da arte do ano em questão, se obteve que aproximadamente 10% dos *soft errors* tinham origem nos circuitos combinacionais, sendo o restante com origem em elementos de memória. A tendência para as tecnologias avançadas, de dimensões reduzidas, é de que o SET em circuitos combinacionais passe a dominar as taxas de erro (BAUMANN apud MUNTEANU; AUTRAN, 2008; DODD et al., 2010). As principais razões para tal são (MUNTEANU; AUTRAN, 2008): tecnologias de dimensões reduzidas permitem que mais transientes tenham largura de pulso e amplitude suficientes para serem capturados, devido à elevada frequência de operação; o aumento da frequência de operação aumenta a habilidade do transiente se propagar pelo circuito; e com a redução das dimensões, a carga representando o nível lógico alto é reduzida, aumentando o número de SETs.

Nos circuitos digitais, o impacto de um SET é mensurado em última instância pela ocorrência de um *bit-flip*, que tem efeito de longa duração e frequentemente a habilidade de alterar a funcionalidade de um sistema, como um *reset* ou modificação de um algoritmo. Já nos circuitos analógicos, o impacto de um SET requer uma dimensão ou atributo adicional para ser adequadamente caracterizado, como por exemplo uma tensão associada, corrente ou dimensão de tempo. Por esse motivo, o estudo do SET em circuitos analógicos é mais recente do que em circuitos digitais, com um crescente de publicações a partir do final dos anos de 1980 (TURFLINGER, 1996).

Para determinar a quantidade de eventos e suas características, a coleta de dados de um SEU analógico utiliza uma configuração de equipamentos conforme mostrado na figura 2.17.

Um osciloscópio é usado para armazenar o tempo de subida do pulso transiente, sua duração, sua amplitude e sua polaridade. O discriminador e o contador são usados para determinar o número de transientes ocorridos à um nível de tensão e polaridade definidos no discriminador (KOGA et al., 1993). Como uma consequência dessa complexidade, diferentemente da determinação da seção de choque em circuitos digitais, como mostrado na equação 2.3, a seção de choque para um dispositivo analógico passa a ser dada em função da condição definida no discriminador no momento da exposição à radiação. Nesse sentido, Balen (2010) afirma que a seção de choque para circuitos analógicos não configura uma grandeza tão significativa quanto a mesma representa para circuitos digitais, além de que as condições dos sinais de entrada, polarização e carga no nó da saída do circuito também interferem. A seção de choque para um circuito analógico pode ser diferente para distintas aplicações e condições consideradas.

Figura 2.17 - Configuração dos equipamentos de teste para a coleta de dados de SETs.



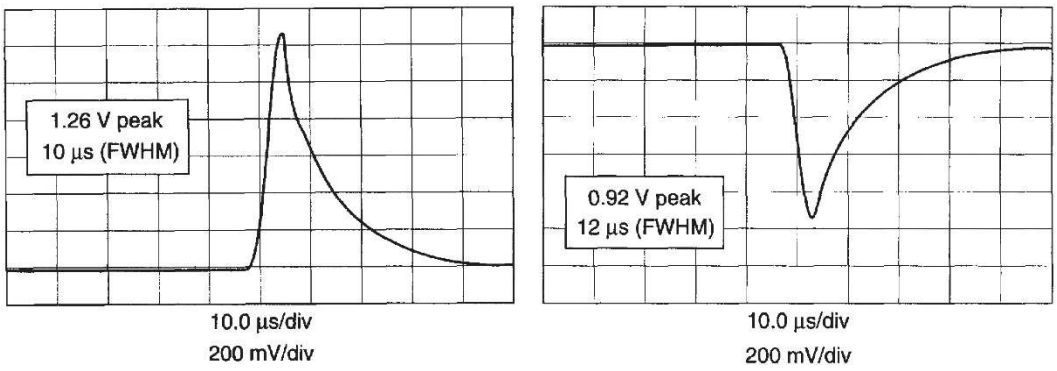
Fonte: KOGA et al., 1993.

Os amplificadores operacionais (OPAMPs, *operational amplifiers*) são usados com frequência na análise do SET em circuitos analógicos devido a sua ampla utilização em uma variedade de sistemas. Considerando um OPAMP como um amplificador de um estágio e aproximando um SET como uma função Delta de Dirac, pela teoria de controle tem-se que a resposta desse OPAMP é um pulso com rápido tempo de subida e decaimento exponencial (TURFLINGER, 1996). Em situações próximas às reais, esse autor apresenta outros dois elementos no comportamento dos OPAMPs frente a um SET. Normalmente um amplificador desses possui mais de um estágio de amplificação, cada um com sua resposta em frequência, geralmente com largura de banda maior do que a do OPAMP como um todo. Como o SET ocorre dentro de um estágio, o resultado é um pulso transiente com amplitude maior do que o esperado no caso do macrocircuito. O segundo elemento é que a análise apresentada, baseada em teoria de controle, desconsidera os transistores de polarização. Esses transistores também são vulneráveis ao SET, existem no circuito em quantidades consideráveis e possuem resposta

em frequência limitada. Com o reestabelecimento lento, os transistores de polarização degradam a resposta dos transistores do caminho de sinal, afetando no decaimento exponencial do pulso transiente. Nesse contexto, as características do pulso transiente resultante são fortemente influenciadas pelas características do circuito analógico.

Com o objetivo de analisar a vulnerabilidade dos circuitos integrados analógicos frente aos SETs, Koga et al. (1993) testaram quatro dispositivos comumente usados em vários sistemas espaciais da época. O OP-15 consiste em um OPAMP afetado por SET no satélite TOPEX da NASA (National Aeronautics and Space Administration), que foi lançado em 1992 (GAY; WELCH; SELBY, 1993). O OP-05 é um amplificador similar ao OP-15, porém diversas vezes mais lento. O HS3530RH é também um OPAMP, tolerante à radiação, de baixa potência e programável. Por último, o LM111H é um comparador de tensão. Esses circuitos foram postos em operação com configuração definida e testados com íons de Xenônio. A figura 2.18 mostra os pulsos transientes gerados pelo amplificador OP-15. Observa-se que a largura desses pulsos (10 µs e 12 µs) é consideravelmente maior do que alargura típica dos pulsos de coleta de carga, o que pode ser devido à saturação do primeiro estágio amplificador ou ao SET nos transistores de polarização.

Figura 2.18 - Pulsos transientes gerados na saída do OP-15 pela colisão de íons de Xenônio.

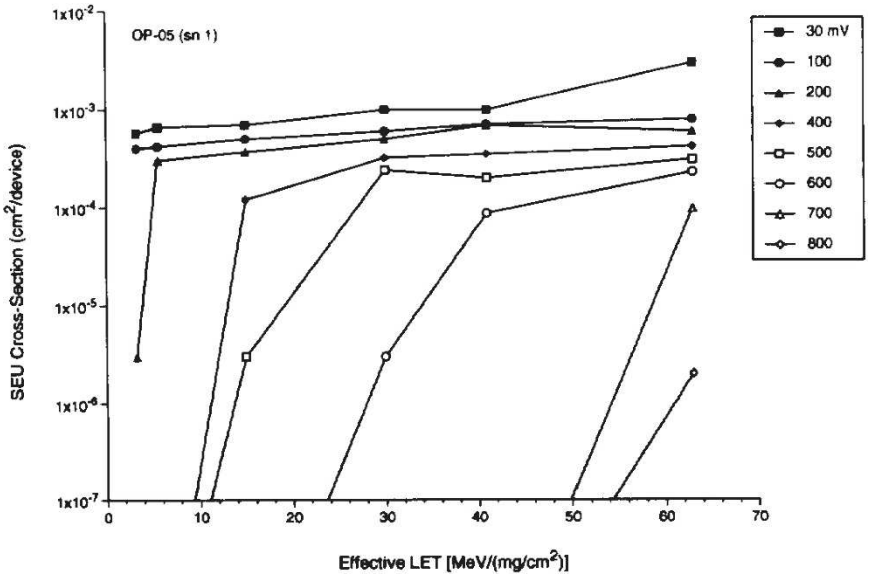


Fonte: KOGA et al., 1993.

A figura 2.19 mostra a relação da seção de choque com a LET efetiva para o amplificador OP-05, baseada nos pulsos positivos. Cada curva refere-se a uma tensão de limiar diferente configurada no discriminador da medida. Neste caso observa-se a forte influência da tensão de limiar com relação à seção de choque, e por este motivo o OP-05 é usado aqui como exemplo. Como pode ser verificado em Koga et al. (1993), os circuitos integrados analisados apresentam diferentes sensibilidades à tensão de limiar na determinação da seção de choque.

A tabela 2.2 mostra os parâmetros dos pulsos transientes gerados pelo SET nos quatro

Figura 2.19 - Seção de choque em função da LET efetiva para a incidência de íons de Xenônio no OP-05, considerando diferentes tensões de limiar no discriminador.



Fonte: KOGA et al., 1993.

circuitos analógicos analisados por Koga et al. (1993). Fica evidente que cada circuito analógico responde de maneira diferente a um SET, devido por exemplo às suas características de polarização e de largura de banda dos estágios de amplificação, conforme explicado por Turfingler (1996) e citado anteriormente neste trabalho. Buscando identificar as regiões mais sensíveis ao SET, Koga et al. (1993) também percorreram o *die* desses OPAMPs com um feixe de laser pontual e chegaram à evidência de que no estágio de entrada do circuito estava a maior sensibilidade. Isso encontra coerência no fato de que o transiente gerado nessa região é amplificado pelo demais estágios do macrocircuito (TURFLINGER, 1996). Em 1997b, Koga et al. investigaram em detalhes os efeitos da polarização na sensibilidade dos circuitos lineares, concluindo que a parte mais sensível é justamente o par diferencial de entrada.

Tabela 2.2 - Parâmetros dos pulsos transientes gerados pelo SET nos quatro circuitos analógicos analisado por Koga et al. (1993).

Device ID	Rise Time (nsec)	Width (FWHM in µsec)	Amplitude (mV)	Polarity
HS3530RH	2500	3	Up to 4000	Positive & Negative
OP-05	2500	15	Up to 800	Mostly Positive
OP-15	3000	12	Up to 1300	Positive & Negative
LM111H	100	0.5	Up to 6000	Mostly Negative

Fonte: KOGA et al., 1993.

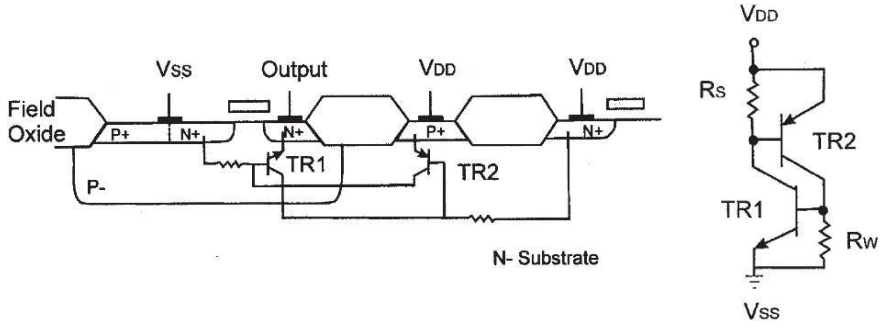
O trabalho de Paulos, Bishop e Turfingler (1987) também traz importantes considerações sobre a resposta de amplificadores operacionais diante de SETs. Existem na literatura diversos trabalhos que analisam o comportamento de outros circuitos analógicos, como conversores de dados (TURFLINGER; DAVEY, 1990; TURFLINGER, 1996; TURFLINGER; DAVEY; BINGS, 1996) e PLLs (Phase-Locked Loop) (CHUNG et al., 2006).

2.3.4 Single Event Latchup (SEL)

SEL refere-se à ocorrência de uma corrente anormal e elevada no dispositivo, causada pela passagem de uma única partícula energética em regiões sensíveis de sua estrutura, e que permanece depois que o disparo é removido. Para reestabelecer a operação normal do dispositivo é necessário um *reset* na alimentação, embora o efeito também possa causar dano permanente (HEIJMEN, 2011; JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012; SEXTON, 2003). Neste último caso, a dissipação térmica devido à elevada corrente gera a vaporização das linhas de metal, fusão dos fios de ligação ou derretimento do silício.

Nos dispositivos CMOS o SEL ocorre através de transistores bipolares parasitas, conforme mostrado na figura 2.20a. O coletor, base e emissor do transistor npn são formados pelo substrato n-, poço p- e difusão n+, respectivamente. Similarmente, coletor, base e emissor do transistor pnp são formados pelo poço p-, substrato n- e difusão p+, respectivamente. O circuito de polarização desses transistores é formado pela resistência entre a base e o emissor do transistor pnp e a resistência entre a base e o emissor do transistor npn. A figura 2.20b mostra um esquema simplificado do circuito parasita. Um *latchup* é disparado quando um SEE liga um

Figura 2.20 - Transistores bipolares parasitas na tecnologia CMOS bulk. (a) Vista em corte para dispositivos sobre substrato do tipo n. (b) Circuito equivalente dos transistores bipolares parasitas.



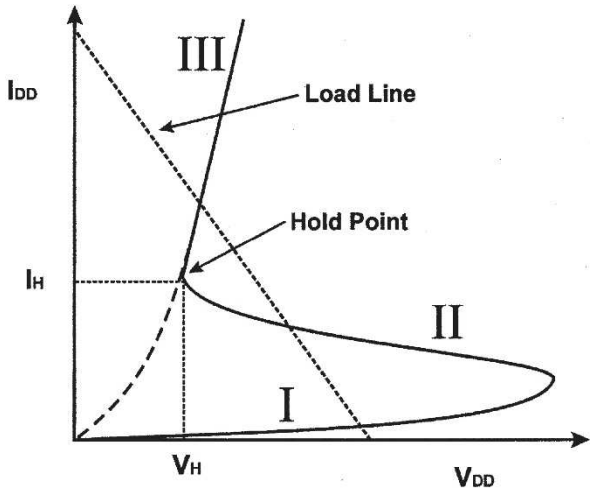
(a) (b)

Fonte: SEXTON, 2003.

dos transistores parasitas. Se isso ocorre no transistor pnp, por exemplo, uma parcela de sua corrente de coletor é direcionada à base do transistor npn, que a amplifica. A corrente de coletor do npn aumenta, a queda de tensão sobre R_S aumenta, a corrente de coletor do transistor pnp aumenta e, consequentemente, a corrente de base do npn aumenta novamente. Esse ciclo se repete indefinidamente, gerando uma corrente elevada que caracteriza o SEL. O *latchup* se mantém apenas quando o produto do ganho dos transistores parasitas excede 1, o circuito de polarização mantém as junções base emissor polarizadas diretamente e as linhas de alimentação podem garantir corrente suficiente para manter a condição (SEXTON, 2003).

Shockley (1950) e Ebers (1952) mostraram que a conexão de um transistor pnp e um transistor npn pode ser equivalente a uma estrutura pnpn, a qual recebe o nome tiristor. Por esse motivo o mecanismo do SEL é frequentemente relacionado ao funcionamento desse dispositivo, que pode ser chaveado de um estado de alta impedância para um estado de baixa impedância e resultar em correntes elevadas. A figura 2.21 mostra a curva corrente versus tensão da estrutura pnpn, na qual podem ser observadas três regiões (SEXTON, 2003). A primeira, a região I, é caracterizada pela baixa corrente em todas as tensões até a ruptura elétrica. Esse é o modo normal de operação de uma junção p-n reversa. Quando um SEL ocorre, o dispositivo passa para um estado de elevada corrente e baixa tensão, como mostrado na região III. A região II conecta as outras duas mais importantes. A mínima tensão e mínima corrente para o *latchup* são definidas como V_H e I_H, respectivamente.

Figura 2.21 - Curva corrente versus tensão para uma estrutura pnpn.



Fonte: SEXTON, 2003.

A região de uma estrutura CMOS mais sensível ao SEL corresponde à região do poço que forma o transistor bipolar npn, pois este é o dispositivo com maior ganho e que possui a maior

área sensível (JOHNSTON; HUGHLOCK, 1990). O comportamento do SEL relativo à temperatura e tensão é diferente do SEU e SET. O aumento da temperatura resulta na redução do limiar de ocorrência de SEL e no aumento da seção de choque (KOLASINSKI et al., 1986). A taxa de ocorrência de SEL reduz com a redução da tensão de alimentação, e a tendência para tecnologias com alimentação próxima à 1 V é não ocorrer *latchup* (SEXTON, 2003). Ao contrário da tecnologia CMOS *bulk*, a tecnologia CMOS SOI (*Silicon on Insulator*) é imune ao SEL, pois a estrutura pnpn da qual resulta a vulnerabilidade ao *latchup* não existe nesta tecnologia, uma vez que o óxido enterrado (BOX, *Buried Oxide*) isola o transistor do substrato (HEIJMEN, 2011).

2.3.5 Single Event Burnout (SEB)

SEB é um evento em que a colisão de uma única partícula energética induz uma corrente elevada que resulta na queima do dispositivo (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Afeta transistores bipolares de potência (GAILLARD, 2011) e principalmente transistores MOSFETs (*Metal-Oxide-Semiconductor Field Effect Transistor*) de potência (BOUDENOT, 2007), como o DMOSFET (*Double-Diffused Metal-Oxide-Semiconductor Field Effect Transistor*) (GALLOWAY; JOHNSON, 1996; TITUS, 2013). Devido a esses transistores não estarem no foco desse trabalho, o SEB não é abordado aqui em maiores detalhes.

2.3.6 Single Event Gate Rupture (SEGR)

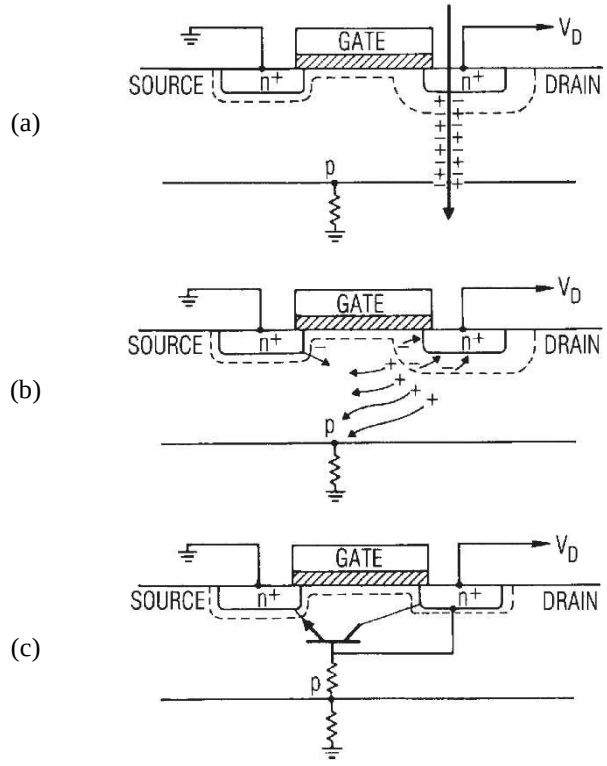
SEGR ocorre quando a colisão de uma única partícula energética provoca a ruptura do dielétrico de porta do transistor MOSFET, criando um caminho de condução através do óxido de porta (JOINT ELECTRON DEVICE ENGINEERING COUNCIL, 2012). Nos transistores MOSFETs de potência, frequentemente ocorre simultaneamente com o SEB (SEXTON, 2003). Da mesma forma que o SEB, o SEGR também não é abordado em profundidade nesse trabalho, dada a relação desse efeito com os transistores de potência, os quais não estão em foco. Mais detalhes podem ser obtidos em obtidos em Brews et al. (1993), Allenspach et al. (1994) e Galloway e Johnson (1996).

2.3.7 Single Event Induced Snap-Back (SES)

Similar ao SEL, SES refere-se à ocorrência de uma corrente anormal e elevada no dispositivo, devido à passagem de uma única partícula energética. Essa corrente permanece depois que o disparo é removido e pode causar dano permanente ao dispositivo. Diferentemente do SEL, em que é necessária a existência de uma estrutura pnpn, o SES ocorre através de um transistor bipolar parasita. Não acontece em transistores MOS de canal p, e além de ionização, pode ser ocasionado por sobretensão (SEXTON, 2003; KOGA; KOLASINSKI, 1989).

A explicação do mecanismo de atuação do SES utiliza a figura 2.22. A passagem de uma partícula ionizante pelo dispositivo gera pares elétron-lacuna, como mostrado na figura 2.22a. Imediatamente após, esses portadores de carga iniciam um movimento em acordo com as linhas de campo elétrico. A maioria dos elétrons movem-se em direção ao dreno, enquanto a maioria das lacunas em direção à fonte. Entretanto, algumas lacunas movem-se pela região p em direção ao plano de terra, como exibido na figura 2.22b. Essas lacunas polarizam o transistor bipolar parasita de maneira a ligá-lo, como mostrado na figura 2.22c. Um processo realimentado mantém esse transistor parasita ligado (KOGA; KOLASINSKI, 1989).

Figura 2.22 - Ilustração do mecanismo de atuação SES.



Fonte: KOGA; KOLASINSKI, 1989.

Transistores baseados na tecnologia CMOS SOI também são susceptíveis à SES (CHEN et al., 1998). A redução do comprimento do canal com as futuras tecnologias agrava a sensibilidade dos transistores MOS ao SES (KOGA; KOLASINSKI, 1989).

2.3.8 Single Hard Error (SHE)

SHE é um *hard error* que está associado à inversão de bits de memória devido à dose total ionizante depositada em dielétricos, através da incidência de uma ou duas partículas energéticas. Pode ser um erro semipermanente, porque em alguns casos pode ser revertido submetendo o dispositivo danificado à radiação ultravioleta e à altas temperaturas (DUFOUR et al., 1992). Embora tenha semelhança com o tradicional efeito de TID, o SHE é diferente pois é um efeito de dose total localizada, em que a degradação ocorre no local da colisão da partícula ionizante. Por causa da natureza localizada do mecanismo, a dose total entregue por uma única partícula é chamada de microdose (GALLOWAY; JOHNSON, 1996).

Os primeiros relatos de SHE em memórias SRAM ocorreram com Koga et al. (1991) e Dufour et al. (1992). Já nas memórias DRAM o primeiro relato ocorreu com Swift, Padgett e Johnston (1994). Para exemplificação, aqui é discutido brevemente o mecanismo que ocorre nas memórias SRAM, o qual é bastante similar ao tradicional efeito de TID e foi detalhado como SHE por Oldham et al. (1993). A colisão de uma partícula cria de algumas centenas a alguns milhares de lacunas aprisionadas no dielétrico de porta do transistor MOS. Essa carga é suficiente para reduzir a tensão de limiar e por isso gerar uma fuga de corrente comparável ao estado ligado do dispositivo. Essa fuga de corrente pode ser suficiente para permanentemente travar uma célula de memória em um estado particular.

3 TÉCNICAS DE PROTEÇÃO AOS SOFT ERRORS

As técnicas de proteção dos circuitos eletrônicos à radiação podem ser divididas em três grupos distintos: as técnicas em nível de dispositivo estão relacionadas ao processo e tecnologia de fabricação; em nível de circuito se referem ao projeto do circuito para reduzir a sensibilidade aos efeitos da radiação; e as técnicas em nível de sistema se preocupam com a arquitetura do sistema. As técnicas abordadas nessa seção dizem respeito apenas à mitigação dos *soft errors*, que são o foco principal desse trabalho. Além disso, como esta é uma vasta área de pesquisa, a seção não tem por objetivo esgotar o assunto, mas apenas realizar uma introdução com algumas exemplificações. A seção 3.1 aborda as técnicas em nível de dispositivo, a seção 3.2 as técnicas em nível de circuito e a seção 3.3 as técnicas em nível de sistema.

3.1 Técnicas em nível de dispositivo

A presença de algumas substâncias específicas nos materiais usados durante a fabricação e principalmente no encapsulamento dos circuitos integrados pode provocar *soft errors*. Isótopos radioativos de Urânio e Tório ocorrem naturalmente e emitem partículas alfa quando seus núcleos decaem para um estado de menor energia, as quais são capazes de ionizar o semicondutor. Já o Boro pode ser usado como dopante do tipo p e na formação do dielétrico BPSG empregado também nos encapsulamentos. O isótopo ^{10}B é instável quando exposto aos nêutrons, e neste caso emite núcleos de Lítio e partículas alfa, ambos também capazes de ionizar o semicondutor (BAUMANN, 2001). Assim, uma melhoria significativa na SER dos dispositivos microeletrônicos pode ser obtida através da pureza do material de fabricação, eliminando o Urânio, Tório e o ^{10}B.

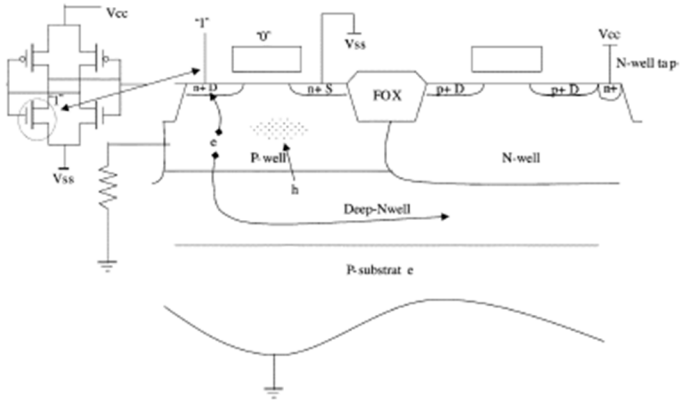
No caso dos isótopos de Urânio e Tório, a utilização de um encapsulamento de elevada pureza reduz a emissão de partículas alfa de 5 a 10 partículas/cm^2.h para aproximadamente 0,001 partículas/cm^2.h (WANG; AGRAWAL, 2008). Efeitos devido ao ^{10}B no BPSG podem ser mitigados eliminando completamente o BPSG do processo ou utilizando um ^{11}BPSG enriquecido ao invés do convencional.

Outra abordagem para a mitigação dos efeitos singulares é baseada na redução da carga coletada após o efeito de funil causado por uma colisão, que pode ser feita pela criação de barreiras de potencial com a adição de poços nos dispositivos CMOS. Uma técnica que implementa essa ideia é conhecida como poço triplo, e consiste em utilizar um poço p para o transistor NMOS e isolá-lo do substrato p com um poço n profundo. Estes dois poços,

juntamente com o poço n do transistor PMOS, formam os três poços que dão nome a técnica.

O princípio de operação do poço triplo na mitigação de efeitos singulares é ilustrado através da figura 3.1, que mostra parte de uma célula de memória SRAM. Pares elétron-lacuna são gerados no poço p do transistor NMOS pela colisão de partículas ionizantes. Os elétrons podem ser coletados pela fonte e dreno do transistor devido ao potencial elétrico. Quando o dreno é nível lógico 1 e os elétrons são coletados, o potencial de dreno reduz. Se carga suficiente é coletada ocorre uma inversão de bit na célula de 1 para 0. A ideia básica da técnica é coletar os elétrons gerados no poço p do dispositivo NMOS antes que eles sejam coletados pelo dreno. A coleta é feita seguindo o caminho poço n profundo, poço n e *tap* poço n para V_{cc}, em que o *tap* poço n é mostrado na extremidade direita da figura 3.1. Já as lacunas são removidas do poço p por um *tap* poço p, o qual não é mostrado na figura em questão, mas que é análogo ao *tap* poço n. A técnica isola o poço p do substrato p, o que é equivalente a uma elevada resistência, como mostrado na figura 3.1 (PUCHNER; RADAELLI; CHATILA, 2004). A técnica de poço triplo também possui a vantagem de oferecer boa isolação entre blocos analógicos e digitais em circuitos de sinal misto e entre memórias dinâmicas de elevada densidade e circuitos lógicos (WESTE; HARRIS, 2010).

Figura 3.1 - Vista em corte de transistores em uma célula SRAM baseada na técnica de poço triplo.

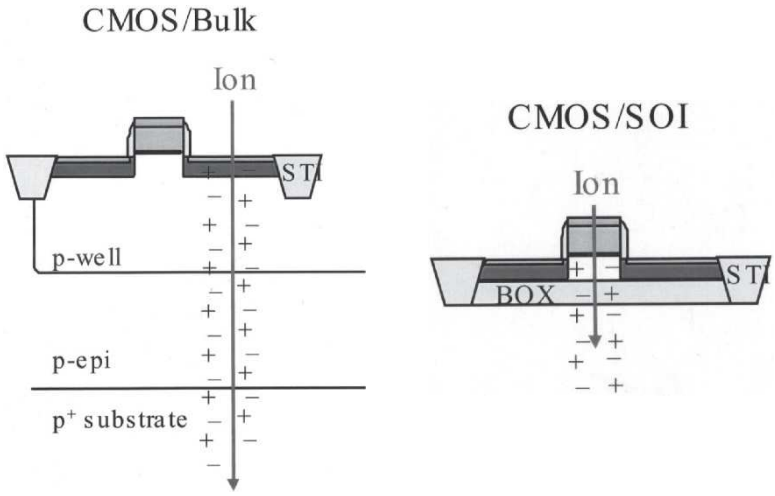


Fonte: PUCHNER; RADAELLI; CHATILA, 2004.

Outra abordagem de proteção aos efeitos singulares que também se baseia na redução da carga coletada, e que atualmente já é utilizada em larga escala, é a tecnologia de silício sobre isolante (SOI, *Silicon on Insulator*). Com base no fato de que em um transistor MOS o transporte de carga no canal ocorre apenas numa região muito superficial do silício e que o restante do volume é usado apenas como suporte mecânico, a tecnologia SOI consiste em

separar a superfície ativa do dispositivo do substrato através de uma camada fina (< 300 nm) de isolante, chamada de óxido enterrado (BOX, *Buried Oxide*). Conforme ilustrado na figura 3.2, nessa tecnologia o BOX teoricamente limita a coleta de carga à região do silício logo abaixo da porta (chamada de corpo flutuante), e faz os circuitos integrados fabricados em SOI consideravelmente menos sensíveis aos SEEs do que os fabricados tradicionalmente em tecnologia *bulk* (SCHWANK et al., 2003).

Figura 3.2 - Comparação entre o volume sensível à coleta de carga. (a) Na tecnologia tradicional bulk. (b) Na tecnologia SOI.



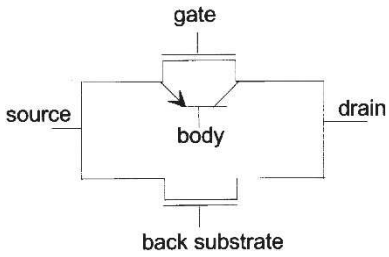
(a) (b)

Fonte: SCHWANK et al., 2003.

Entretanto, estruturas parasitas que surgem na prática quando um transistor MOS SOI é ativado por radiação limitam a efetividade dessa tecnologia na proteção aos SEEs. A figura 3.3 mostra o circuito equivalente para esse caso, em que é possível observar a adição de um transistor bipolar parasita e um transistor MOS parasita, chamado de transistor de porta inferior. O nó formado pelo corpo flutuante constitui a base do transistor bipolar. Esse dispositivo pode ser disparado por condições elétricas específicas (tensão de dreno elevada), radiação, ou cargas elétricas que polarizam diretamente o diodo da junção corpo-fonte. Quanto ao transistor de porta inferior, tem sua porta formada pelo substrato de suporte mecânico e pelo BOX como isolante, e é ativado principalmente por dose total ionizante. Assim, o dispositivo parasita que limita a proteção da tecnologia SOI para os SEEs é o transistor bipolar. Ele amplifica a carga gerada na colisão de uma partícula ionizante, o que anula as vantagens pelo uso da tecnologia SOI. A técnica mais comum usada para reduzir esse efeito é conhecida como ligação direta (*body tie*) (SCHWANK et al., 2003). Consiste em conectar o corpo flutuante a um potencial

fixo, o que fornece um eficiente caminho para os portadores majoritários, reduzindo significativamente a amplificação bipolar.

Figura 3.3 - Estrutura elétrica equivalente quando um transistor MOS SOI é ativado por radiação.



Fonte: SCHWANK et al., 2003.

3.2 Técnicas em nível de circuito

As técnicas de proteção em nível de circuito têm uma vantagem sobre as técnicas em nível de dispositivo no sentido de não necessitarem de mudanças fundamentais no processo de fabricação (DODD et al., 2010). Uma técnica básica em nível de circuito consiste na adição de um capacitor no nó que se deseja proteger. Dado que a carga crítica (Q_{crit}) representa a mínima quantidade de carga coletada (Q_{col}) em um nó necessária para causar um *upset*, a ideia é aumentar a Q_{crit}, de maneira que o pulso de corrente gerado pelo SEE seja dissipado e não se propague pelos estágios seguintes do circuito (BAZE et al., 2002).

A Q_{crit} também pode ser definida conforme a equação 3.1, em que C_{load} é a capacitância do nó e $V_{noise-margim}$ é a tensão de margem de ruído. Assumindo que a tensão transiente de $V_{dd}/2$ caracteriza um *upset*, em que V_{dd} é a tensão de alimentação do circuito, $V_{noise-margim}$ pode ser definida conforme a equação 3.2. Assim, a Q_{crit} pode ser expressa conforme a equação 3.3. Se um capacitor extra é adicionado, C_{load} aumenta e a Q_{crit} aumenta, tornando o circuito mais tolerante à radiação. Se a carga depositada ($Q_{deposited}$) por um SEE é conhecida, a capacitância adicional requerida (C_{add}) pode ser expressa conforme a equação 3.4. A adição de um capacitor pode funcionar como um filtro passa baixa em circuitos integrados com elevadas frequências, prejudicando seu funcionamento. O capacitor adicional também aumenta o atraso de propagação, o consumo de potência e a área em silício. No caso de elevada carga depositada, um capacitor de valor elevado pode ser impraticável (SAYIL, 2010).

$$Q_{crit} = C_{load} * V_{noise-margim} \tag{3.1}$$

$$V_{noise-margin} = \frac{V_{dd}}{2} \tag{3.2}$$

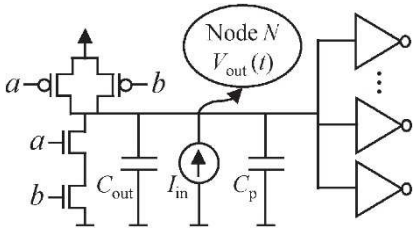
$$Q_{crit} = C_{load} * \frac{V_{dd}}{2} \tag{3.3}$$

$$C_{add} = \frac{C_{load} * (Q_{deposited} - Q_{crit})}{Q_{crit}} \tag{3.4}$$

Outra técnica também baseada no aumento da Q_{crit} é a de dimensionamento da porta dos transistores MOS (ZHOU; MOHANRAM, 2006). Ela consiste em aumentar a Q_{crit} dos nós com maior susceptibilidade ao erro através da alteração da razão de espectro (W/L) dos transistores ligados a eles. Essa medida adiciona um caminho para que o transistor possa drenar rapidamente a carga depositada pelo SEE e prevenir que o transiente alcance magnitude suficiente para se propagar pelos circuitos seguintes. Essa técnica impõe despesas de área, atraso e potência, embora reduzidas se comparadas com outras técnicas. Resultados experimentais para tecnologias de 180 a 70 nm mostram que a SER é reduzida significativamente e com mínimo impacto nos recursos (ZHOU; MOHANRAM, 2004).

Tomando como exemplo o circuito da figura 3.4, em que aparece uma porta NAND com seu nó N de saída, a capacitância total desse nó é dada conforme a equação 3.5. C_p é a

Figura 3.4 - Circuito com porta NAND de duas entradas usado para exemplificar a técnica de dimensionamento da porta dos transistores MOS.



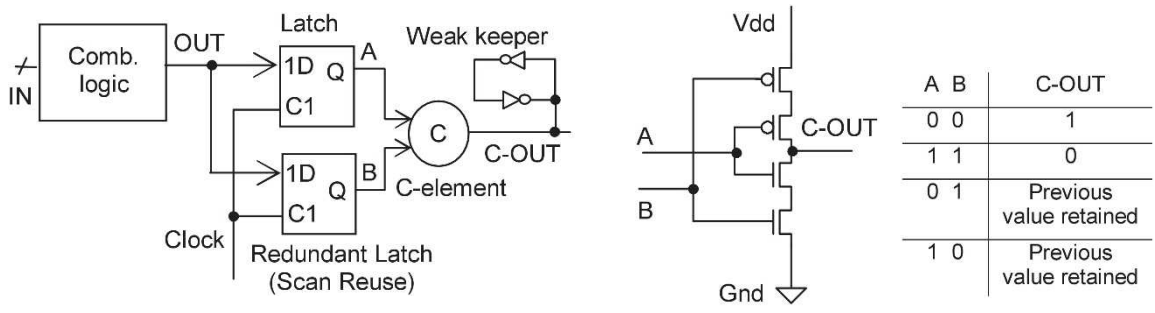
Fonte: ZHOU; MOHANRAM, 2006.

capacitância equivalente que leva em consideração as interconexões e o circuito seguinte ao nó N, (W/L) é o tamanho de um único transistor NMOS da porta NAND e C_{unit} é a capacitância de saída unitária (inclui transistores NMOS e PMOS) obtida pela divisão da capacitância de saída da porta NAND pelo tamanho do transistor NMOS. Aqui o foco é na tensão V_{out} do nó N porque sua magnitude e duração determinam a propagação do transiente. Através da equação em questão é possível verificar que a capacitância de saída pode ser aumentada pelo aumento da razão de espectro dos transistores (ZHOU; MOHANRAM, 2006).

$$C_{total} = C_{unit}\left(\frac{W}{L}\right) + C_p \tag{3.5}$$

Outra técnica em nível de circuito voltada a circuitos combinacionais tem como base o elemento-c (MITRA et al., 2006). Partindo do esquema na figura 3.5a, consiste em através de dois *latches* fazer duas tomadas do sinal que se deseja proteger, e conectar a saída desses *latches* a um elemento-c para gerar o sinal protegido. O elemento-c tem formação e comportamento descritos na figura 3.5b, em que se observa que sua saída é atualizada apenas quando os *latches* geram valores iguais. Em caso de diferença o valor anterior é mantido através do *keeper*. Dado o fato de que *soft errors* em circuitos combinacionais se manifestam como transientes, um erro no circuito combinacional pode ser mascarado pelos *latches*, uma vez que quando o *clock* tem nível lógico 0 os *latches* mantém o valor correto armazenado anteriormente. Um erro em algum dos *latches* também é mascarado, dado que o elemento-c mantém o valor anterior quando eles geram valores diferentes. Já um *soft error* único no *keeper* não tem efeito maior porque a saída do elemento-c é fortemente definida pela saída dos *latches*.

Figura 3.5 - Técnica de proteção baseada no elemento-c. (a) Visão geral da técnica. (b) Formação e comportamento do elemento-c.

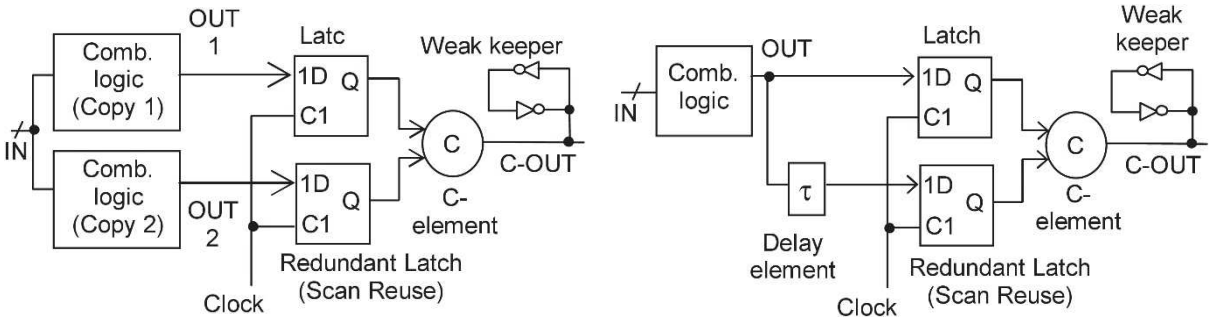


(a) (b)

Fonte: MITRA et al., 2006.

O custo associado com os *latches* redundantes é minimizado com o reuso de recursos já existentes no circuito integrado, como múltiplas funções em vários estágios de fabricação e uso em campo. Resultados mostram que essa técnica melhora a SER em 10 vezes sobre um projeto desprotegido, com baixo prejuízo em área, desempenho e potência (ZHANG et al., 2006). Um aprimoramento a essa técnica pode ser feito com a duplicação do circuito combinacional ou o deslocamento da saída do circuito combinacional no tempo, conforme mostrado na figura 3.6. Com isso se obtém uma significativa redução da SER na lógica combinacional, especialmente útil quando o *clock* dos *latches* tem nível lógico 1, momento em que eles atualizam sua saída. A duplicação do circuito combinacional implica em aumento no custo de área e potência, enquanto o deslocamento do sinal no tempo penaliza o desempenho (MITRA et al., 2006).

Figura 3.6 - Aprimoramento na técnica baseada no elemento-c. (a) Com a duplicação do circuito combinacional. (b) Com o deslocamento do sinal no tempo.



Fonte: MITRA et al., 2006.

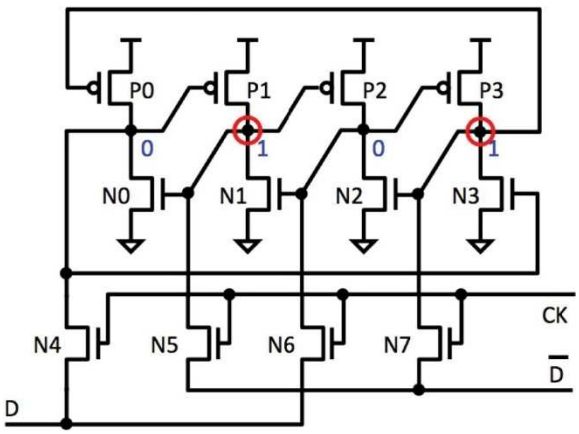
Existem ainda outras técnicas de proteção voltadas para circuitos combinacionais não abordadas neste trabalho. Por exemplo, a técnica baseada no transistor complementar de passagem (KUMAR; TAHOORI apud SAYIL, 2010), a baseada no *buffer* Schmitt Trigger (SASAKI; NAMBA; ITO, 2008) e a baseada em transistores adicionais para construção de portas lógicas, conhecida como CMOS Design Style (ZHANG; SKANBHAG, 2005). Ainda em nível de circuito, existem técnicas específicas para a proteção de memórias, uma vez que elas possuem o maior número e densidade de bits susceptíveis à colisão de partículas. Para exemplificação, a seguir são mostradas duas técnicas bastante conhecidas que podem ser empregadas com memórias SRAM.

Conforme mostrado na seção 2.3.1, uma célula de memória SRAM tem estrutura de acordo com a figura 2.11. Se uma partícula energética colide próximo ao dreno do transistor Q2, que se encontra desligado, a carga coletada resulta em uma corrente transiente nesse

transistor. Pelo transistor Q1, nesse caso denominado de restaurador, flui então uma corrente para balancear a corrente em Q2, resultando na alteração da tensão de porta do par inversor oposto, que tende a desestabilizar o estado lógico armazenado pela célula (ANDREWS et al., 1982). Assim, o processo de *upset* em uma célula SRAM é essencialmente uma disputa entre a realimentação e o processo de restauração. Com base nesse mecanismo, uma técnica de proteção da memória aos SEEs consiste em tornar a realimentação mais lenta, através da adição de resistência (chamada R_{fb} na figura 2.11) ou capacitância na realimentação. A adição de resistência é uma abordagem clássica e é bastante efetiva: estimativas feitas para um microprocessador sem proteção em uma órbita geossíncrona indicam uma taxa de 1 *upset*/dia, enquanto para microprocessador protegido com resistor a taxa é de 1 *upset*/século (DODD; MASSENGILL, 2003). Os problemas e dificuldade advindos dessa técnica são: o atraso proporcionado pelos resistores na realimentação impacta no processo de escrita da memória, e portanto penaliza seu desempenho; os resistores são implementados em polisilício levemente dopado, e como este material tem resistividade muito sensível à concentração de dopantes, é bastante difícil controlar o valor do resistor; e como o polisilício tem uma dependência negativa da temperatura, existe uma dependência do tempo de escrita e da resposta ao SEE com a temperatura (DODD; MASSENGILL, 2003).

Uma abordagem alternativa para a proteção de memórias SRAM é baseada no conceito de HBD (*Hardening by Design*) (ALEXANDER, 1996), que consiste em adicionar elementos de circuito redundantes, construindo a célula de memória com 12 a 16 transistores ao invés dos 6 transistores da célula convencional. Devido ao aumento de transistores por célula, a solução consome mais área e potência. A técnica mais popular que implementa esse conceito é conhecida como DICE (*Dual Interlocked Storage Cell*) (CALIN; NICOLAIDIS; VELAZCO,

Figura 3.7 - Célula de memória SRAM DICE. Os nós circulados referem-se aos nós sensíveis.



Fonte: Adaptado de CALIN; NICOLAIDIS; VELAZCO, 1996.

1996). Conforme ilustrado na figura 3.7, na célula DICE cada nó sensível é controlado por duas realimentações. A vantagem dessa técnica é que requer apenas 12 transistores.

3.3 Técnicas em nível de sistema

A maior parte das técnicas de proteção em nível de dispositivo e em nível de circuito atuam na prevenção de *soft errors*, o que pode não ser suficiente nos circuitos integrados modernos, em virtude de suas características como tensões de alimentação reduzidas, dimensões reduzidas e frequência de *clock* elevada. Nesse contexto, as técnicas em nível de sistema são a alternativa, pois possuem em grande parte mecanismos de recuperação *on-line* (WANG; AGRAWAL, 2008). Elas também são adequadas quando não existe o acesso em nível de dispositivo e circuito (DODD et al., 2010). As principais técnicas em nível de sistema podem ser classificadas em primitivas, baseadas em detecção de erro e correção (EDAC, *Error Detection and Correction*) e baseadas em redundância e votação.

As técnicas primitivas são aplicadas em circuitos integrados complexos, como processadores, memórias e FPGAs. Esses circuitos estão sujeitos à perda da funcionalidade temporária devido à radiação (SEFI), que pode resultar em uma saída inválida ou que não converge, ou no travamento do algoritmo. A mitigação consiste na recuperação da operação normal do circuito e posterior reestabelecimento do fluxo de dados, através da atualização (*refreshing*) dos bits de controle afetados ou *reset* na alimentação. As técnicas primitivas dependem da existência de uma estrutura adequada para rapidamente detectar a anomalia e tomar uma ação, sendo importante que essa estrutura seja independente do processo ou *hardware* monitorado. O monitoramento pode ser feito através de *watchdog timers*, algum sinal vital ou outra estratégia de detecção de falha. As técnicas primitivas devem ser suficientes quando a taxa de dados perdidos não é um problema, e seu custo não é elevado. Apenas perde-se um pouco de eficiência, pois parte do tempo gasto na monitoração pode afetar o tempo de processamento. Além disso, o circuito de monitoramento pode ser suscetível aos erros, causando um *reset* indevido ou perda de dados (LADBURY, 2007).

As técnicas baseadas em EDAC são largamente usadas para monitorar e corrigir erros devido aos efeitos singulares em memórias (SAYIL, 2010), sendo a melhor estratégia em nível de sistema a ser adotada com estes dispositivos (LADBURY, 2007). Essa abordagem requer que bits extra de informação sejam armazenados com os dados para os reconstruir em caso de erro. Os algoritmos usados em EDAC são os mesmos também usados para tratar erros devido ao ruído e perda de dados em sinais de comunicação.

O algoritmo mais simples para detecção de erro se baseia em paridade (CARLSON, 1975). Nela geralmente um bit extra é adicionado à informação, o qual indica se a quantidade de níveis lógicos 1 do conteúdo é par ou ímpar. Esse algoritmo pode detectar um erro apenas quando um número ímpar de bits está em erro, se um número par de bits está em erro, a paridade ainda é correta. Esse algoritmo apenas detecta erros, e não pode corrigi-los. Outro algoritmo para detecção de erro é conhecido como Verificação de Redundância Cíclica (CRC, *Cyclic Redundancy Check*) (SHORT, 1987). Consiste em se anexar ao dado um valor verificador, obtido no resto da divisão polinomial deste dado por um polinômio gerador. Na verificação um cálculo semelhante é repetido, através do qual pode-se verificar uma suposta corrupção de dados.

Existem também algoritmos que possuem a capacidade de corrigir bits, além de detectar bits errados. O Código de Hamming (CARLSON, 1975), amplamente conhecido, é capaz de detectar 2 bits errados e corrigir 1 bit. Consiste em adicionar à informação um código verificador composto por Q bits, gerados por uma matriz verificadora de paridade, e então uma síndrome representada pela palavra de Q dígitos pode descrever a posição do bit errado. O Código de Hamming é recomendado para sistemas com baixa probabilidade de múltiplos erros. Um algoritmo mais robusto é o código Reed-Solomon (REED; SOLOMON, 1960), capaz de detectar e corrigir múltiplos e consecutivos erros em uma estrutura de dados. Essa característica deve-se a este algoritmo dividir os dados em conjuntos de símbolos de m bits, ao invés de conjuntos de bits, e portanto tratar erros em múltiplos bits dentro de um símbolo da mesma forma que inversões de um único bit. O código Reed-Solomon (255,223), implementado em um circuito integrado desenvolvido pelo NASA VLSI Design Center, utiliza 223 bytes de dados e 32 bytes adicionais para gerar uma mensagem de 255 bytes, tornando possível a correção de até 16 bytes consecutivos de erros. Além desses algoritmos mencionados, existem diversos outros, como códigos convolucionais e implementações específicas.

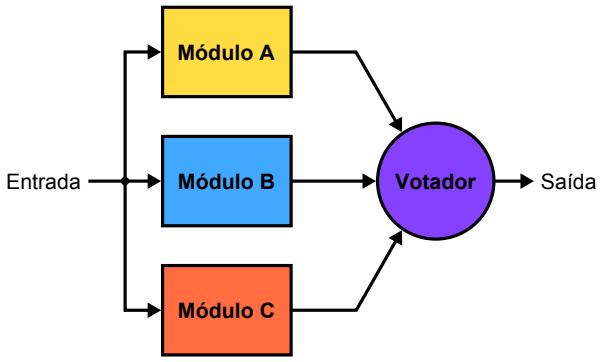
Para melhorar as técnicas baseadas em EDAC e torná-las mais robustas, ainda podem ser usadas duas estratégias. A primeira consiste em intercalar os bits e símbolos de forma que um efeito singular que provoque múltiplos erros não exceda a capacidade do algoritmo. A segunda estratégia leva em consideração a possibilidade nas memórias de acúmulos de bits errados ao longo do tempo. Nesse caso a recomendação consiste em periodicamente testar os dados da memória em busca de erros. Essa estratégia impacta na eficiência do sistema, dada a frequência com que o teste é realizado para manter os dados com baixa quantidade de erros (LADBURY, 2007).

Embora as técnicas baseadas em EDAC são perfeitamente adequadas às memórias, o

mesmo não ocorre em circuitos como processadores, FPGAs e outros de computação intensiva, nos quais o dado entra no dispositivo, uma série de operações é realizada sobre ele e após o dado chega até a saída. Isso porque não existe acesso externo ao dado durante seu processamento para que seja monitorado, e também porque um SEFI pode alterar não apenas o dado, mas as operações que são realizadas sobre ele (LADBURY, 2007). Nessa situação as técnicas adequadas para mitigação de *soft errors* são as baseadas em redundância e votação, as quais possuem tanto a capacidade de detecção de erros como correção.

A técnica baseada em redundância e votação amplamente usada é a Redundância Modular Tripla (TMR, *Triple Modular Redundancy*) (VON NEUMANN, 1956), ilustrada na figura 3.8. Consiste em triplicar o circuito a ser protegido e conectar suas saídas a um votador de maioria. Se algum erro ocorrer em uma das três cópias do circuito e as demais operarem corretamente, o valor correto é escolhido pelo votador. É uma técnica com ênfase em tolerar falhas aleatórias, que presumidamente ocorrem de maneira independente nas cópias redundantes. Os significativos custos de área e potência são evidentes, o que pode restringir a técnica às aplicações críticas, em que um erro pode resultar na perda de vidas ou na perda de investimentos elevados.

Figura 3.8 - Esquema TMR clássico.



Fonte: Autoria própria, 2015.

A técnica de diversidade (AVIZIENIS; KELLY, 1984) pode ser usada juntamente com TMR para aumentar a tolerância à falhas. Nela os elementos de *hardware* e *software* usados para computação múltipla não são cópias, mas projetos independentes que atendem os critérios do sistema. A diversidade pode ocorrer em três domínios: diversidade de *hardware*, também referida como diversidade espacial; diversidade de *software*, também chamada de diversidade de programa; e a diversidade temporal, que consiste na repetição da computação em instantes de tempo distintos ou com frequência diferente. Essa técnica é especialmente útil para lidar com

falhas de modo-comum, devido aos diferentes níveis de resiliência de cada *hardware*, *software* e tempo. Além disso, a diversidade também tem a característica de que diferentes projetistas e ferramentas são empregados em cada cópia, e pontos em comum são sistematicamente evitados, o que geralmente faz com que as falhas de projeto não produzam erros similares. Uma desvantagem dessa técnica é que ela pode adicionar complexidade ao sistema, como por exemplo, a necessidade de um circuito para sincronização dos elementos de computação múltipla.

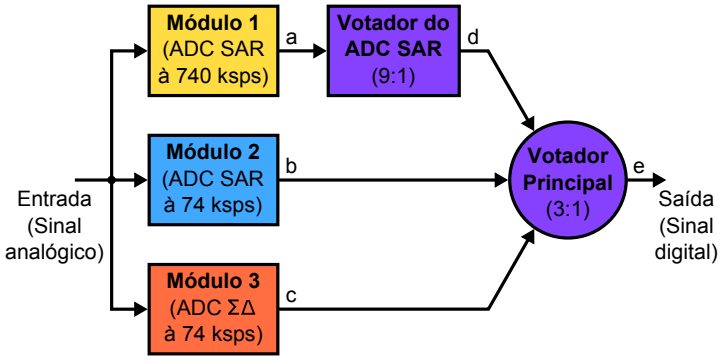
4 SOFT ERRORS EM UM SISTEMA DE AQUISIÇÃO DE DADOS BASEADO EM TMR E DIVERSIDADE ESPACIAL-TEMPORAL: ESTUDO COM INJEÇÃO DE FALHAS POR IRRADIAÇÃO

Os estudos desse trabalho adotaram um esquema baseado em TMR e diversidade espacial-temporal para a implementação de um sistema de aquisição de dados (SAD) analógico-digital. O estudo dessa seção utiliza a injeção de falhas por irradiação de nêutrons para observar o comportamento dos conversores de dados e avaliar a eficácia do esquema adotado baseado em TMR e diversidade. O procedimento de irradiação é considerado nesse trabalho como injeção de falhas no sentido de ser um estímulo artificial para avaliar a resposta de um sistema (HSUEH; TSAI; IYER, 1997). A classificação dada por esses autores mencionados para o procedimento em questão é conhecida como injeção de falhas por *hardware* sem contato, a qual também compreende a injeção de falhas por campos eletromagnéticos. Essa seção está organizada da seguinte forma: em 4.1 é detalhado o esquema adotado; em 4.2 são apresentadas as características do SoC (*System-on-Chip*) em que o SAD foi implementado; em 4.3 o *setup* de teste é explicado, com detalhes da implementação do SAD baseado em TMR e diversidade; em 4.4 é mostrada como foi realizada a validação do *setup*; em 4.5 são apresentados os procedimentos do teste; e em 4.6 apresentados os resultados e as discussões.

4.1 Esquema do sistema de aquisição de dados com base em TMR e diversidade

O sistema de aquisição de dados (SAD) adotado com base em TMR e diversidade é mostrado na figura 4.1. Ele usa três conversores analógico-digitais (ADCs, *Analog to Digital Converters*) em paralelo: 2 com arquitetura de registrador de aproximações sucessivas (SAR, *Successive Approximations Register*) e 1 com arquitetura Sigma-Delta. Um ADC SAR e o ADC Sigma-Delta operam com taxa de amostragem de 74 ksps e o outro ADC SAR opera à 740 ksps. O sistema é composto também por dois votadores: o votador principal, que realiza a votação entre os três conversores e um votador chamado de votador do ADC SAR, que realiza a votação com 9 amostras geradas pelo ADC SAR que opera a 740 ksps. Nesse esquema adotado cada cópia do ADC compõem a abordagem TMR. As duas arquiteturas diferentes dos conversores caracterizam a diversidade espacial, e as duas taxas de amostragem diferentes a diversidade temporal.

Figura 4.1 - Esquema do SAD, com base em TMR e diversidade.

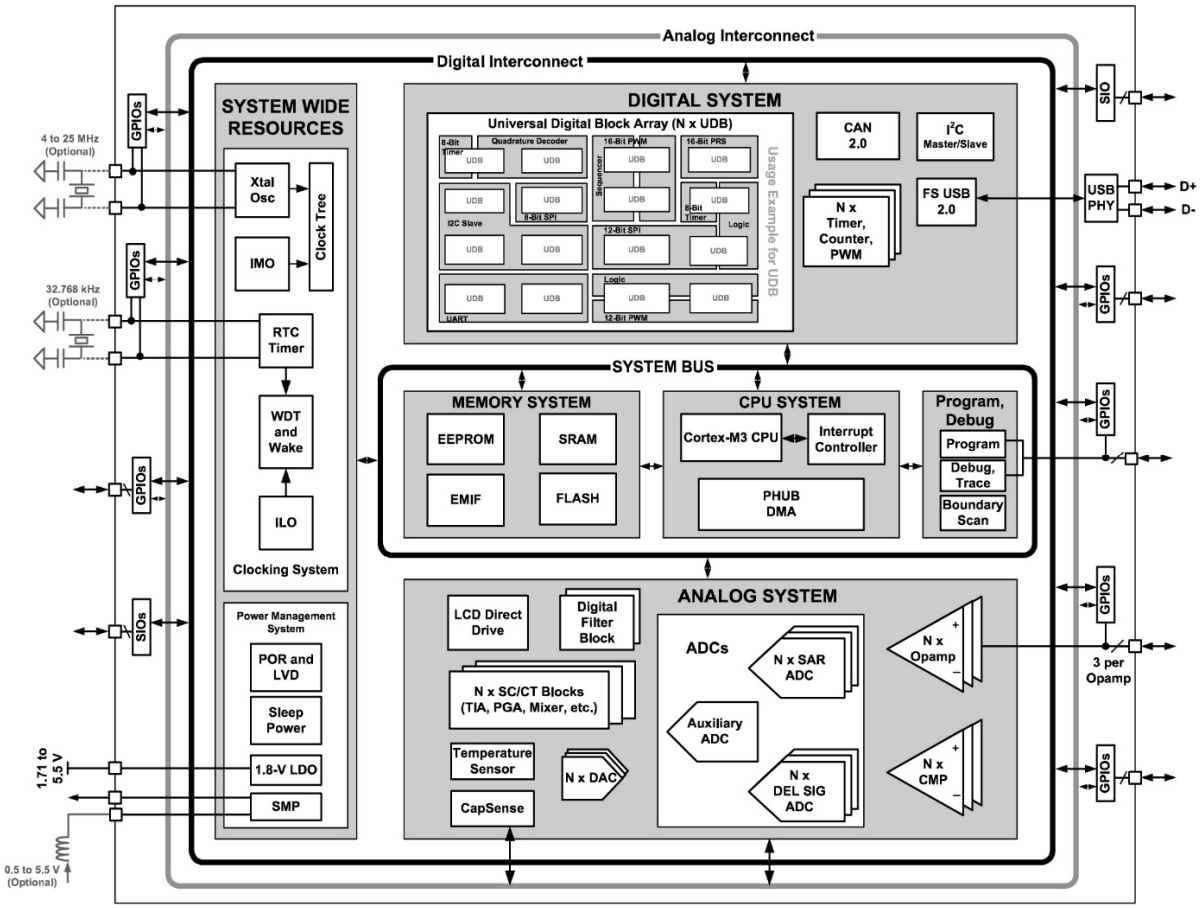


Fonte: Autoria própria, 2015.

4.2 Características do PSoC 5LP

O SAD dos estudos desse trabalho foi totalmente implementado em um SoC comercial,

Figura 4.2 - Diagrama de blocos da arquitetura do PSoC 5LP.



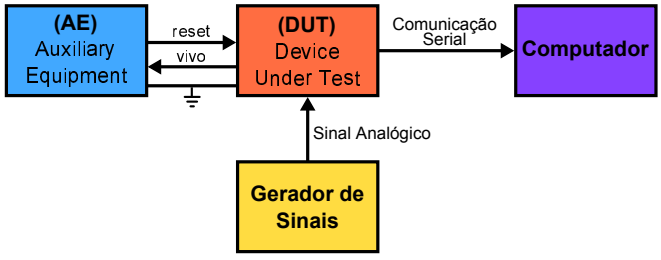
Fonte: CYPRESS SEMICONDUCTOR, 2013a.

programável e de sinais mistos, chamado PSoC 5LP (CYPRESS SEMICONDUCTOR, 2013a), da Cypress Semiconductor. O diagrama de blocos da arquitetura desse circuito integrado é mostrado na figura 4.2. O *part number* do PSoC 5LP usado é o CY8C5588AXI-060. Esse dispositivo é fabricado em tecnologia CMOS de 130 nm e consiste de uma CPU ARM Cortex-M3 de 32 bits e 80 MHz. Memória Flash de 256 KB, memória SRAM de 64 KB, memória EEPROM de 2 KB e 24 canais de DMA (*Direct Memory Access*). Periféricos digitais como interfaces de comunicação e PLDs (*Programmable Logic Devices*) baseados em UDBs (*Universal Digital Blocks*), os quais proporcionam a implementação de diversas funções como temporizadores, contadores e outros. E periféricos analógicos e de sinal misto como 1 conversor Sigma-Delta, 2 conversores SAR, conversores digital-analógico, comparadores, amplificadores operacionais e blocos configuráveis que podem implementar diversões funções analógicas. Todos esses periféricos estão disponíveis em uma biblioteca de componentes pré-construídos presente na ferramenta de *software* PSoC Creator, utilizada em projetos com o PSoC 5LP. A edição de projetos no PSoC Creator é feita por meio de esquemático, usando os componentes pré-construídos, e também por *software* em linguagem C.

4.3 Setup de teste

A figura 4.3 ilustra o *setup* de teste para o estudo com injeção de falhas por irradiação. O SAD implementado no PSoC 5LP consiste no dispositivo sob ensaio (DUT, *Device Under Test*). Um gerador de sinais é conectado ao DUT para gerar o estímulo analógico à entrada do SAD. Uma comunicação serial RS-232/UART conecta um computador ao DUT, para este emitir registro de seu estado e assim viabilizar a análise de seu comportamento. Para garantir que o teste continue caso algum efeito singular gere o travamento do sistema, um equipamento auxiliar (AE, *Auxiliary Equipment*) implementa uma técnica de proteção primitiva em nível de sistema: um sinal que transiciona (chamado de “vivo”) a cada ciclo de votação é monitorado, e

Figura 4.3 - Setup de teste para o estudo com injeção de falhas por irradiação.



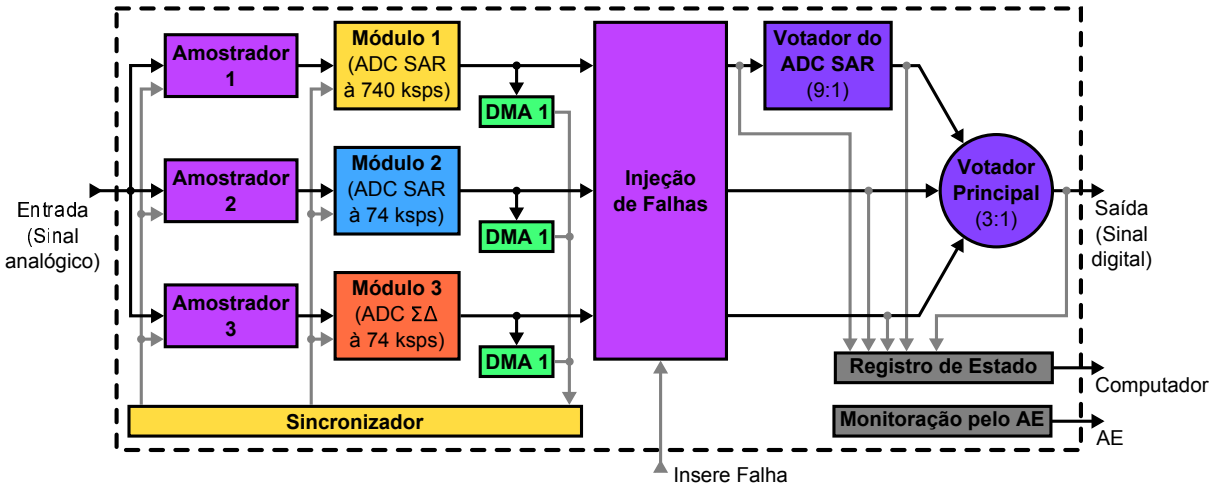
Fonte: Autoria própria, 2015.

caso algum efeito singular faça o DUT parar de enviar esta mensagem, após 30 segundos o AE gera um *reset* no DUT. O AE foi implementado em um PSoC 5, a versão anterior do PSoC 5LP, e desta forma muito semelhante ao descrito na seção 4.2. Nas seções 4.3.1 e 4.3.2 são detalhadas as implementações do DUT e do AE.

4.3.1 Implementação do DUT

A figura 4.4 mostra em detalhes a implementação completa do SAD no PSoC 5LP. Além dos conversores analógico-digitais e votadores inicialmente previstos, a implementação demanda também amostradores, circuitos de acesso direto à memória (DMA, *Direct Memory Access*) e um sincronizador. Além disso, para atender aos objetivos do estudo, são também necessários um bloco de injeção de falhas, um bloco de registro de estado e um bloco para monitoração pelo AE.

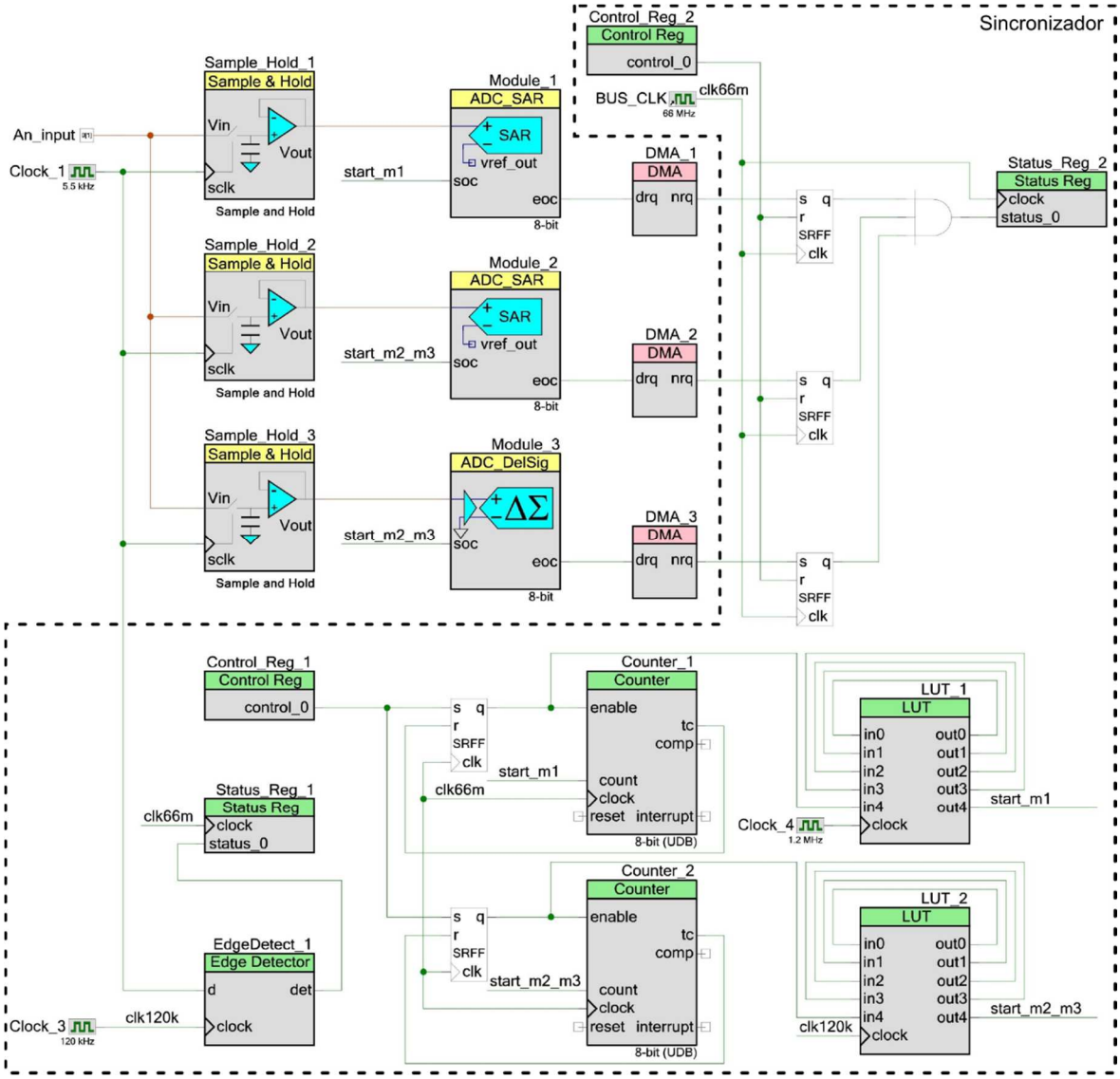
Figura 4.4 - Detalhes da implementação completa do SAD no PSoC 5LP.



Fonte: Autoria própria, 2015.

A figura 4.5 mostra os principais blocos inseridos no esquemático da ferramenta PSoC Creator para a implementação do SAD. Os votadores não são representados porque são implementados inteiramente em *software*. O esquemático completo da implementação encontra-se no apêndice A desse trabalho e o código em linguagem C no apêndice B. Nas seções 4.3.1 a seguir cada bloco é abordado em profundidade.

Figura 4.5 - Principais blocos inseridos no esquemático da ferramenta PSoC Creator para a implementação do SAD.



Fonte: Autoria própria, 2014.

4.3.1.1 Amostradores

Os conversores de dados disponíveis no PSoC 5 LP possuem amostradores internamente, porém não é possível controlar de forma independente seu instante de amostragem, pois ele é iniciado quando se dispara uma conversão. Na implementação em questão é necessário controlar o início da amostragem por dois motivos: porque a amostra fornecida aos três conversores deve ser exatamente a mesma; e porque o ADC SAR que opera à 740 ksps deve realizar 9 conversões de uma mesma amostra, de modo que elas possam ser comparadas entre

si. Por esse motivo usou-se amostradores externos aos conversores, disponíveis na biblioteca de componentes pré-construídos do PSoC 5LP.

O esquema adotado consiste em um amostrador para cada ADC, disparados por um circuito sincronizador, como mostrado na figura 4.4. A primeira vantagem desse esquema é que oferece redundância, uma vez que o uso de um único amostrador gera uma resposta incorreta do sistema no caso de erro nesse bloco. A segunda vantagem do esquema adotado é que a impedância vista pela saída do amostrador é maior do que em um esquema com três ADCs conectados em um único bloco. Assim o esquema adotado é uma condição mais favorável à operação dos amostradores, uma vez que podem manter o dado amostrado por um período maior. Esta análise inclusive expõem um ponto crítico na aplicação de redundância nos circuitos analógicos, pois atenção especial deve ser tomada com os parâmetros analógicos. Por exemplo, deve-se garantir que a impedância de entrada equivalente do sistema TMR analógico é adequada para o circuito que fornece o sinal de entrada.

4.3.1.2 Conversores analógico-digitais

Os ADCs foram implementados com componentes pré-construídos da biblioteca do PSoC 5LP. Foram configurados para 8 bits de resolução, modo de uma conversão por disparo efetuado por *hardware*, fonte de *clock* interna, tensão de referência interna e entrada analógica em modo-comum e na faixa de 0 a 2,048 V. As taxas de amostragem configuradas estão indicadas na seção 4.1.

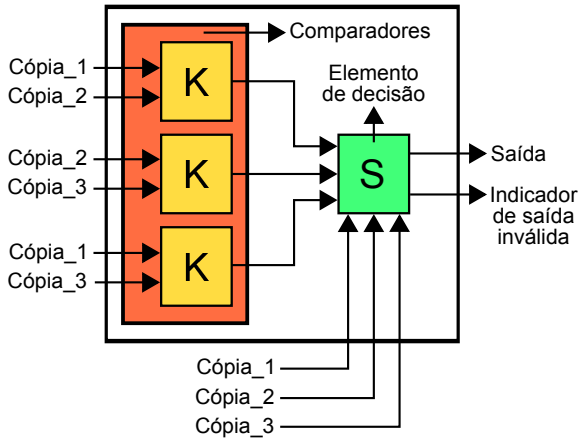
4.3.1.3 Votadores

O conceito básico de votação adotado na implementação do votador principal e do votador do ADC SAR foi o de votação de maioria. Uma discussão sobre algumas das principais técnicas de votação é apresentada em Lorczak, Caglayan e Eckhardt (1989). A votação de maioria é a técnica usada com mais frequência, mesmo com a desvantagem de implicar em uma perda na precisão. Essa perda ocorre devido a necessidade de um limiar para a construção de uma janela de tolerância, que permite determinar a maioria quando o dado a ser votado é aproximado, mas não exatamente o mesmo, como ocorre em grande parte dos sistemas reais.

A implementação de votadores de maioria no domínio digital pode ser feita seguindo os conceitos de votação bit-a-bit ou votação de palavra (MITRA; MCCLUSKEY, 2000). O votador de palavra, como ilustrado na figura 4.6, consiste de três comparadores e um elemento

de decisão. Os comparadores realizam a comparação mútua por subtração decimal entre as saídas das três cópias redundantes (compreendendo todas as combinações possíveis), produzindo três sinais de erro. Com base nesses sinais de erro, o elemento de decisão seleciona o valor correto para a saída do votador. No votador de palavra a integridade dos dados é maior no contexto de falhas de modo-comum e múltiplas falhas, porque ele pode detectar condições errôneas que no bit-a-bit produzem saídas incorretas. Nesse caso, o votador de palavra pode gerar uma indicação para que uma ação apropriada seja iniciada.

Figura 4.6 - Representação do conceito de votação de palavra.



Fonte: Autoria própria. Baseado em BORGES et al., 2010.

O votador principal do SAD foi baseado no conceito de votação de palavra. O código em linguagem C que faz essa implementação é mostrado na figura 4.7. A janela de tolerância foi

Figura 4.7 - Código do votador principal em linguagem C, baseado no conceito de votação de palavra.

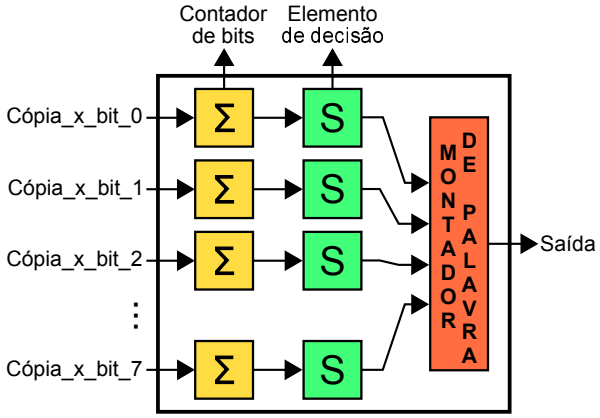
```
1     void main_voter()
2     {
3         error1 = abs (SAR_ADC_voter_data - module2Data[1]);
4         error2 = abs (module2Data[1] - module3Data[1]);
5         error3 = abs (module3Data[1] - SAR_ADC_voter_data);
6
7         if (error1 <= 4)
8             system_output = SAR_ADC_voter_data;
9         else if (error2 <= 4)
10            system_output = module2Data[1];
11        else if (error3 <= 4)
12            system_output = module3Data[1];
13        else
14            system_output = 0;
15
16        if ((error1 > 4) ||  (error2 > 4) || (error3 > 4))
17            main_voter_error_det = 1;
18    }
```

Fonte: Autoria própria, 2014.

configurada como 4 unidades decimais e uma indicação (main_voter_error_det) é gerada no caso de detecção de erro.

Como ilustrado na figura 4.8, a votação bit-a-bit consiste de um contador e um elemento de decisão para cada posição de bit (exemplo, bit mais significativo, bit menos significativo e intermediários), assim como um montador de palavra. O número de estados 1 de cada posição de bit entre todas as palavras é contado e essa informação é levada ao elemento de decisão, que vota se o bit correto é estado 1 ou 0. Posteriormente, cada um desses bits votados é submetido ao montador de palavra, que gera a saída do votador. O votador bit-a-bit possui sobre o votador de palavra a vantagem de ser facilmente implementado em caso de muitas entradas.

Figura 4.8 - Representação do conceito de votação bit-a-bit.



Fonte: Autoria própria. Baseado em BORGES et al., 2010.

O votador do ADC SAR foi baseado no conceito de votação bit-a-bit. Essa escolha foi feita porque esse votador possui 9 entradas e o conceito de votação bit-a-bit é vantajoso neste caso. O código em linguagem C que faz essa implementação é mostrado na figura 4.9. Em caso de em alguma posição de bit serem detectados estados lógicos diferentes entre as palavras, uma indicação (SAR_ADC_voter_error_det) é gerada.

4.3.1.4 Circuitos de acesso direto à memória

O DMA é um recurso de *hardware* para armazenar na memória dados gerados pelos periféricos sem envolver o processador nessa tarefa. As vantagens de usar o DMA são de liberar o processador para outras tarefas e reduzir o atraso imposto pelo processador, uma vez que o DMA é um *hardware* dedicado e portanto mais rápido. Como mostrado na figura 4.4, no SAD foram usados três DMAs, um para cada ADC. Eles geram uma indicação de pronto para que o

Figura 4.9 - Código do votador do ADC SAR em linguagem C, baseado no conceito de votação bit-a-bit.

```
1     void SAR_ADC_voter()
2     {
3         bits = 0;
4         SAR_ADC_voter_data = 0;
5         for (i = 0; i < 8; i++)
6             bit_counter[i] = 0;
7
8         for (i = 0; i < 8; i++)
9         {
10            for (j = 1; j < 10; j++)
11            {
12                bits = (module1Data[j] & mask[i]) != 0;
13                if (bits == 1)
14                    bit_counter[i] = bit_counter[i] + 1;
15            }
16        }
17
18        for (i = 0; i < 8; i++)
19        {
20            if (bit_counter[i] > 4)
21                SAR_ADC_voter_data = SAR_ADC_voter_data + mask[i];
22        }
23
24        bigger = module1Data[1];
25        smaller = module1Data[1];
26        for (pointer = 2; pointer < 10; pointer++)
27        {
28            if (bigger < module1Data[pointer])
29                bigger = module1Data[pointer];
30            if (smaller > module1Data[pointer])
31                smaller = module1Data[pointer];
32        }
33        error = bigger - smaller;
34        if (error > 4)
35            SAR_ADC_voter_error_det = 1;
36    }
```

Fonte: Autoria própria, 2014.

sincronizador inicie um novo ciclo de votação. Sua configuração foi feita através de código, e é mostrada entre as linhas 105 e 148 do apêndice B.

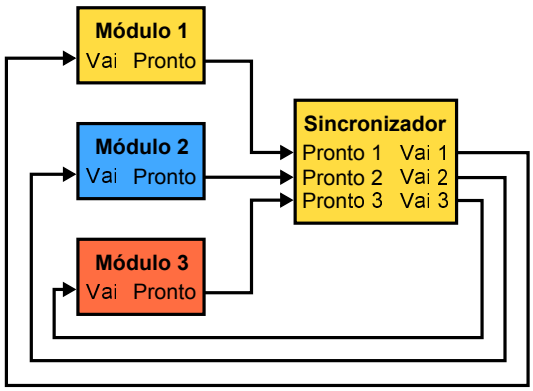
4.3.1.5 Sincronizador

A noção de sincronização usa o termo ciclo de votação. Ele se refere ao intervalo de tempo entre a geração pelas cópias de sucessivas saídas que estão em votação. Para formalizar a noção de sincronização, considera-se um conjunto de duas ou mais cópias, cada uma tendo n ciclos de votação em um intervalo de tempo. Sequencialmente numeram-se os ciclos de votação de cada cópia, começando com 1. Um conjunto de cópias é dito sincronizado se para cada i, com $1 \leq i \leq n$, existe um intervalo de tempo não nulo em que cada uma das cópias está no ciclo

de votação i. Assim, essa noção caracteriza o comportamento temporal das cópias TMR (DAVIES; WAKERLY, 1978). A sincronização é um ponto chave na técnica TMR, principalmente quando utiliza-se diversidade. Ela afeta a controlabilidade e complexidade do sistema. Começando por cópias pré-construídas, projetar um sincronizador efetivo e de baixo custo é provavelmente um dos maiores desafios na construção de um sistema baseado em TMR e diversidade.

Davies e Wakerly (1978) classificam a sincronização em três tipos: bases de tempo independentes e precisas, fundamentado na precisão das bases de tempo individuais; referência externa comum; e realimentação mútua, em que as cópias monitoram umas às outras e corrigem os desvios por elas próprias. A técnica de votação sincronizada é proveniente do tipo realimentação mútua. O modelo de sistema baseado nessa técnica é ilustrado na figura 4.10. Cada cópia TMR tem um sinal de entrada chamado “vai” e um sinal de saída chamado “pronto”. A entrada “vai” informa à cópia que o ciclo de votação iniciou e a saída “pronto” indica que a cópia concluiu o ciclo e está pronta para iniciar o próximo. A saída “vai” pode ser usada por um sincronizador para iniciar a votação e portanto controlar o ciclo de votação.

Figura 4.10 - Modelo de sistema baseado na técnica de votação sincronizada.



Fonte: Autoria própria. Baseado em DAVIES; WAKERLY, 1978.

Devido à flexibilidade de implementar a técnica de votação sincronizada na aplicação e os recursos disponíveis no PSoC 5LP que facilitam a implementação, essa foi a técnica adotada. A figura 4.4 mostra o sincronizador aplicado ao SAD. Como os conversores foram configurados em modo de uma conversão por disparo, o sincronizador dispara os amostradores e os ADCs, com base na indicação de pronto fornecida pelos DMAs. A implementação do sincronizador foi feita parte em *software* e parte em *hardware*. A parte em *software* encontra-se distribuída ao longo do código em C no apêndice B. A parte em *hardware* pode ser visualizada no esquemático presente no apêndice A. De maneira geral seus principais componentes são:

Clock_1, Counter_1 e Counter_2, LUT_1 e LUT_2, flip-flops, porta AND e Status_Reg_2.

4.3.1.6 Bloco de injeção de falhas

Um bloco de injeção de falhas foi implementado para viabilizar a validação do SAD e do *setup* de teste desse estudo. Ele utiliza dois botões conectados ao PSoC 5LP para gerar interrupções e inverter bits do dado gerado na saída dos ADCs. Um dos botões inverte o primeiro bit mais significativo da terceira amostra gerada pelo ADC SAR que opera à 740 ksps, e o outro botão inverte o terceiro bit mais significativo do ADC SAR que opera à 74 ksps. O código em linguagem C que implementa essa inversão de bits encontra-se entre as linhas 607 e 618 do apêndice B.

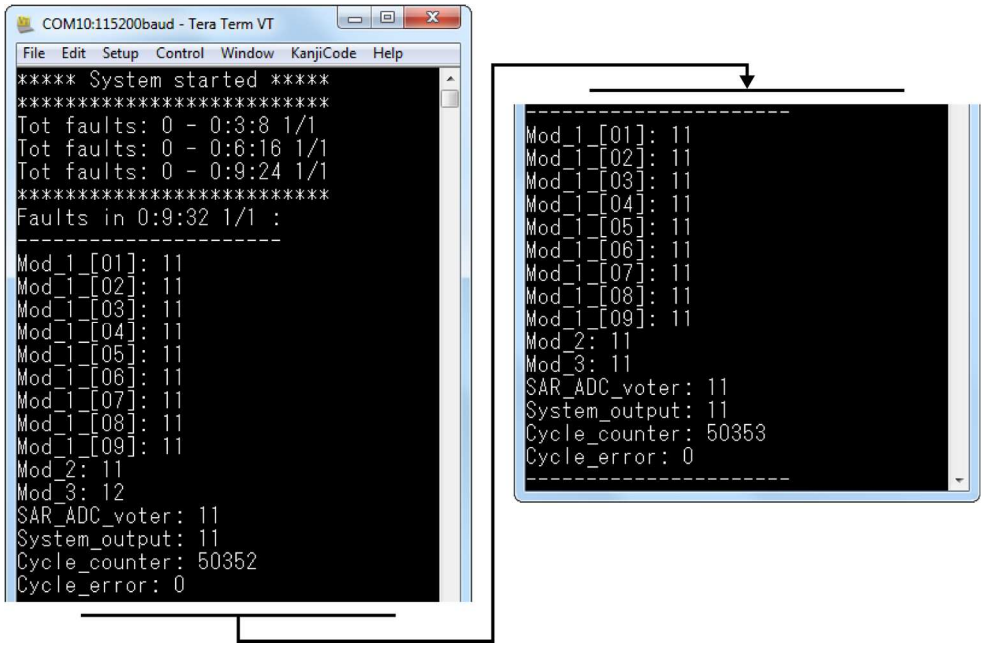
4.3.1.7 Bloco de registro de estado

Um bloco de registro de estado foi implementado com o objetivo de criar um relatório sobre o SAD para análise posterior ao teste. Ele é composto por um *buffer*, um bloco de controle de escrita e uma interface serial RS-232/UART. O *buffer* armazena os dados dos 101 ciclos de votação mais recentes, guardando as seguintes informações: valores de saída dos três conversores, valor de saída do votador principal e do votador do ADC SAR, valor de um contador sequencial de ciclos e o valor do indicador de erro gerado pelos conversores. O bloco de controle de escrita funciona da seguinte forma: se nenhum erro for detectado, aproximadamente a cada 3 minutos as informações de quantidade total de erros e tempo transcorrido de teste são enviadas; caso algum erro é detectado, o conteúdo do *buffer* é enviado. O código em linguagem C que faz essa implementação encontra-se entre as linhas 221 e 499 do apêndice B. O envio de informações é feito através da interface serial RS-232/UART, disponível na biblioteca de componentes pré-construídos do PSoC 5 LP. A interface foi configurada para operar com taxa de 115,2 kbps, dados de 8 bits, 1 bit de parada e sem paridade e controle de fluxo. Um exemplo de parte do relatório criado é mostrado na figura 4.11.

4.3.1.8 Bloco para monitoração pelo AE

Esse bloco tem o objetivo de garantir que o teste continue caso algum efeito singular gere o travamento do DUT. Para isso o DUT gera um sinal digital com nível lógico 1 ou 0 que

Figura 4.11 - Exemplo de parte do relatório criado pelo bloco de registro de estado.



Fonte: Autoria própria, 2015.

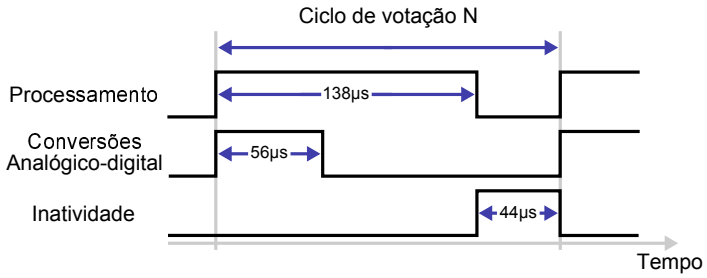
transiciona a cada ciclo de votação (chamado de “vivo”), indicando que o SAD está em execução. A monitoração desse sinal permite o AE reiniciar o DUT se necessário. Em termos de *hardware*, a comunicação entre o DUT e o AE envolve, além dos sinais de “vivo” e *reset*, a conexão do plano de referência (gnd). A implementação no PSoC 5 LP é feita através do registrador Control_Reg_3 e do código em linguagem C entre as linhas 637 e 639 do apêndice B.

4.3.1.9 Paralelismo entre processamento e conversões analógico-digitais

Com a implementação do SAD em estágio avançado foi verificado que o tempo de processamento para um ciclo de votação, que compreende basicamente a execução do votador principal, do votador do ADC SAR e do bloco de registro de estado, era de aproximadamente 138 µs. Já o tempo das conversões analógico-digitais para um ciclo de votação, que compreende as amostragens, conversões e armazenamento dos dados digitais na memória através do DMA, era de aproximadamente 56 µs. A realização destas duas tarefas de maneira sequencial implicaria em um período de tempo grande em que os ADCs ficariam inativos, algo indesejado tendo em vista o objetivo de observar o comportamento dos ADCs frente a SEU, SET e SEFI. Assim, como o processamento é realizado inteiramente em *software* e as conversões analógico-digitais inteiramente em *hardware*, optou-se por paralelizar essas duas tarefas, conforme

mostrado na figura 4.12. O tempo de inatividade do sistema corresponde a uma margem de segurança para garantir o correto funcionamento do sistema.

Figura 4.12 - Paralelismo no tempo entre processamento e conversões analógico-digitais.

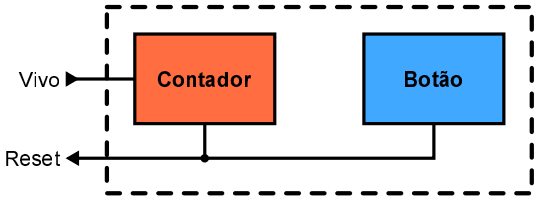


Fonte: Autoria própria, 2015.

4.3.2 Implementação do AE

A implementação do AE é bastante simplificada, e por isso optou-se por mostrar aqui um diagrama funcional ao invés de um diagrama com detalhes da implementação. A figura 4.13 mostra esse diagrama funcional. Basicamente o AE opera de modo semelhante a um *watchdog*. Ele possui um contador que é reiniciado toda vez que ocorrer uma transição de nível lógico no sinal de "vivo". Caso não ocorram transições e o contador chegar a um valor equivalente ao tempo de 30 segundos, esse contador do AE gera o *reset* do DUT. Adicionalmente, um botão é usado para gerar um *reset* manual do DUT através do AE e também para liberar o DUT para o início do teste. O esquemático dessa implementação encontra-se no apêndice C desse trabalho e o código em linguagem C no apêndice D.

Figura 4.13 - Diagrama funcional do AE para o estudo com injeção de falha por irradiação.



Fonte: Autoria própria, 2015.

4.4 Validação do setup

Concluídas as implementações do DUT e do AE, o *setup* de teste foi montado conforme descrito no início da seção 4.3 e na figura 4.3. O sinal analógico usado para estimular a entrada do SAD foi uma senóide com frequência de 120 Hz, nível DC de 0,96 V e amplitude pico-a-pico de 1,88 V. Não se usou toda a faixa de amplitude de entrada dos conversores em função de não-linearidades diferentes entre os conversores SAR e o Sigma-Delta. Para recebimento do registro de estado do DUT e geração de um relatório, foi instalado no computador usado no *setup* o *software* TeraTerm (TERA TERM, 2014). A validação foi realizada através de injeção de falhas por *software* e em tempo de execução, usando o bloco destinado a essa função descrito na seção 4.3.1.6. Foram observados os seguintes critérios para considerar o *setup* como validado:

a) sem a injeção de falhas, os valores gerados pelos ADCs ficam dentro das janelas de tolerâncias definidas nos votadores, e portanto o sistema não reporta erros;
b) sem a injeção de falhas em um longo período de tempo, aproximadamente a cada 3 minutos o sistema reporta a mensagem com a quantidade total de erros e o tempo transcorrido desde o início do teste;
c) em caso de injeção de falha, os votadores detectam a ocorrência de erro e o conteúdo do *buffer* com os valores dos 101 ciclos de conversão mais recentes é reportado;
d) em caso de travamento do SAD, simulado pela desconexão do sinal de “vivo” entre o DUT e o AE, este último gera o *reset* do DUT.

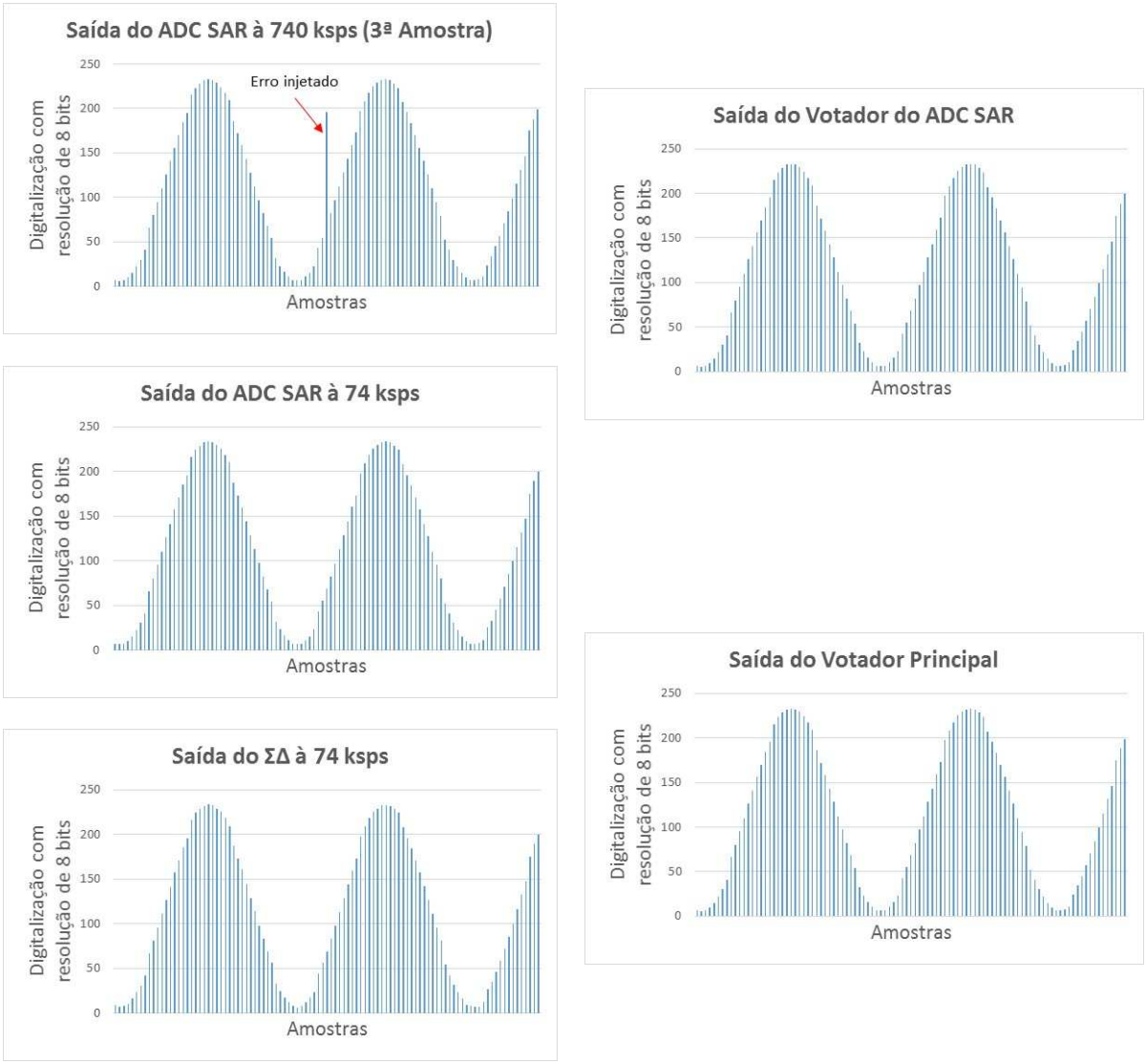
A figura 4.14 mostra os sinais parcialmente reconstruídos gerados pelos conversores e votadores a partir do conteúdo do *buffer* reportado após a detecção de um erro injetado.

Ainda dentro da validação, foi realizado um teste com duração de 168 horas (uma semana) com o *setup* funcionando e sem injeção de falhas, com o objetivo de garantir que durante a realização do teste de injeção de falhas com radiação não fossem reportados erros além dos gerados pelos efeitos singulares.

4.5 Procedimentos de teste

Os testes de injeção de falhas com irradiação foram realizados no Instituto de Estudos Avançados (IEAv), localizado em São José dos Campos (Brasil). A irradiação foi realizada com oito fontes de nêutrons de ^{241}Am-Be de 100 mCi, em que o Curie (Ci) é uma unidade de

Figura 4.14 - Sinais parcialmente reconstruídos gerados pelos conversores e votadores a partir do conteúdo do buffer reportado após a detecção de um erro injetado.

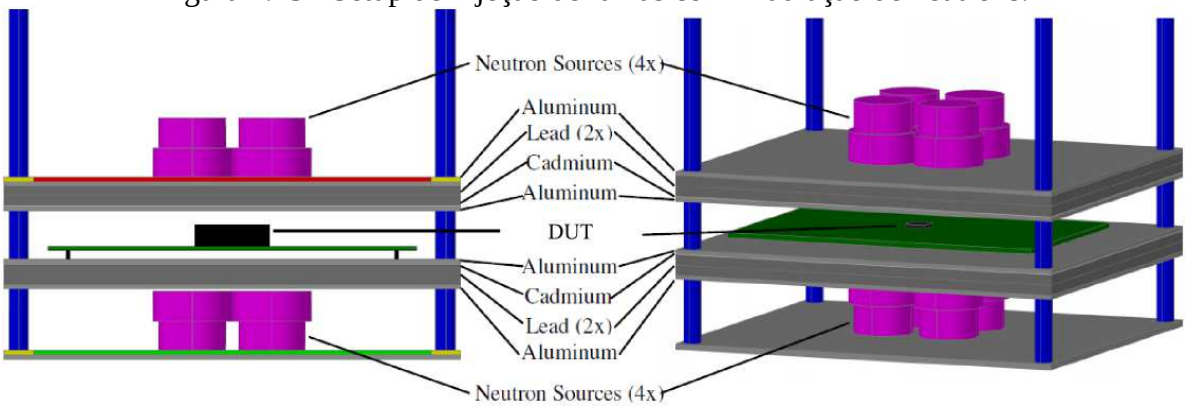


Fonte: Autoria própria, 2015.

atividade radioativa que expressa $3{,}7.10^{10}$ desintegrações por segundo. Essas oito fontes de nêutrons geram na posição do DUT um fluxo total de $(7{,}87 \pm 0{,}24).10^{3}$ n/cm^2.s. Conforme ilustrado na figura 4.15, as fontes são dispostas acima e abaixo do DUT, de maneira a simular neste uma incidência de nêutrons quase-isotrópica. Entre as fontes e o DUT existem duas folhas de Alumínio, duas folhas de Chumbo e uma folha de Cádmio, com espessuras de 2 mm, 4,5 mm e 0,5 mm, respectivamente. Essas folhas têm o objetivo de blindar raios-gama de até 59 keV provenientes do ^{241}Am, e eliminar nêutrons térmicos produzidos por espalhamento e termalização na blindagem de polietileno que protege os equipamentos auxiliares. Portanto a irradiação que chega ao DUT consiste apenas em nêutrons rápidos. Esses nêutrons são distribuídos em uma faixa de energia de 20 keV a 11 MeV, com energia média de 4,16 MeV

(INTERNATIONAL STANDARD ORGANIZATION, 2011).

Figura 4.15 - Setup de injeção de falhas com irradiação de nêutrons.



Fonte: PEREIRA JUNIOR et al., 2013.

A figura 4.16 mostra imagens obtidas na realização do teste do SAD com a injeção de falhas por irradiação.

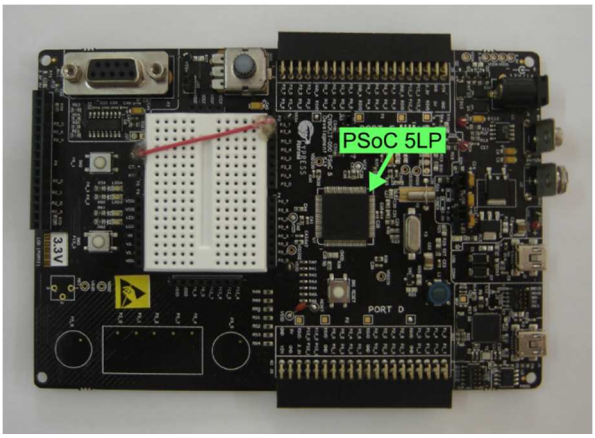
O teste foi executado por um período total de 600 horas (25 dias).

4.6 Resultados e discussões

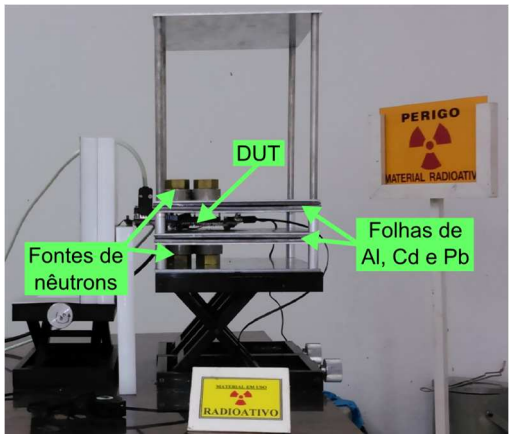
Ao longo das 600 horas (25 dias) do teste de injeção de falhas com irradiação de nêutrons não foram observados erros, o que impediu cumprir os objetivos propostos de através da injeção de falhas por irradiação observar o comportamento dos conversores de dados e avaliar a eficácia do sistema baseado em TMR e diversidade espacial-temporal. Alguns indícios sugerem os motivos pelos quais não foram observados erros, como a pequena sensibilidade a SEU da memória SRAM do SAD, a provável ocorrência de mascaramentos de SETs nos circuitos combinacionais e analógicos (conforme detalhado na seção 2.3.3) e o baixo fluxo de nêutrons do *setup* de injeção de falhas se comparado ao praticado em outras instalações existentes em outros países. A seguir esses indícios são detalhados em profundidade.

Pereira Junior et al. (2013) realizaram testes de injeção de falhas com irradiação de nêutrons em uma memória SRAM, usando as mesmas instalações desse estudo e conforme detalhado na seção 4.5 e figura 4.15. A memória SRAM utilizada corresponde ao *part number* IS62WV25616BLL, do fabricante Integrated Silicon Solution Inc. (ISSI), com tamanho de 4 MB e fabricada em tecnologia CMOS de 130 nm. Resultados dos testes de escrita e leitura mostraram que durante 535039 segundos (6,19 dias) foram observados 138 *bit-flips*, o que corresponde a uma seção de choque de $(3{,}28 \pm 0{,}30).10^{-8}$ cm^2/disp. ou $(9{,}50 \pm 0{,}86).10^{-15}$

Figura 4.16 - Teste do SAD com a injeção de falhas por irradiação. (a) PSoC 5LP. (b) DUT entre as fontes de nêutrons. (c) Setup completo.



(a)



(b)

(c)

Fonte: Autoria própria, 2015.

cm^2/bit. Este trabalho é importante para o presente estudo principalmente por dois motivos: porque a ocorrência de erros na memória devido a SEU indica que o *setup* de irradiação está validado e porque a tecnologia de fabricação de 130 nm pertence ao mesmo nó tecnológico do PSoC 5LP. Este último motivo permite estabelecer comparações entre o comportamento da SRAM do PSoC 5LP e a memória IS62WV25616BLL. A tabela 4.1 resume os parâmetros de comparação. A memória SRAM do PSoC 5LP possui tamanho de 64 KB, porém o compilador do PSoC Creator informou utilização de 36,6%, o que corresponde a 23,997 KB. A aplicação de uma regra de três composta permite estimar que no *setup* de teste desse estudo ocorreu em média 1 *bit-flip* a cada 646263 segundos (7,48 dias). Considerando que o teste de injeção de falhas com irradiação do estudo durou 25 dias, pode-se estimar que nesse período ocorreram na SRAM do PSoC 5LP 3 *bit-flips*, o que representa um número de falhas muito baixo. Estas

possíveis falhas não produziram erros observáveis porque provavelmente foram mascaradas. Por exemplo, os *bit-flips* invertidos podem ter sido sobrescritos antes de sua leitura, e portanto tornam-se insignificantes. Dada a seção de choque obtida por Pereira Junior et al. (2013) e considerando os tamanhos das memórias, através de uma regra de três simples também é possível estimar que a seção de choque da SRAM do PSoC 5LP nesse estudo foi (1,97 ± 0,18).10^{-10} cm^2/disp. ou (5,70 ± 0,51).10^{-17} cm^2/bit, o que representa uma seção de choque pequena. Além disso, é importante notar que no teste realizado por Pereira Junior et al. (2013) a probabilidade de mascaramentos de *bit-flips* é menor do que no estudo com o PSoC 5LP, uma vez que o primeiro consiste em um teste estático, em que são realizadas leituras sistemáticas dos dados armazenados. No teste com o PSoC 5LP as leituras são dinâmicas, e portanto a probabilidade de mascaramentos de *bit-flips* é maior.

Tabela 4.1 - Comparação entre os testes de irradiação da memória IS62WV25616BLL e a SRAM do PSoC 5LP, que permite a estimativa do tempo necessário para a ocorrência de 1 bit-flip no SAD com injeção de falhas por irradiação e a estimativa da seção de choque.

Memória SRAM	Tamanho da memória	Bit-flips	Duração do teste	Seção de choque
IS62WV25616BLL	4×10^6 bytes	138	535039 s ou 6,19 dias	$(3,28 \pm 0,30) \times 10^{-8}$ cm^2/disp. ou $(9,50 \pm 0,86) \times 10^{-15}$ cm^2/bit
Interna PSoC 5LP	$23,997 \times 10^3$ bytes	1	**646263 s** **ou 7,48 dias**	**$(1,97 \pm 0,18) \times 10^{-10}$ cm^2/disp.** **ou $(5,70 \pm 0,51) \times 10^{-17}$ cm^2/bit**

Fonte: Autoria própria, 2015.

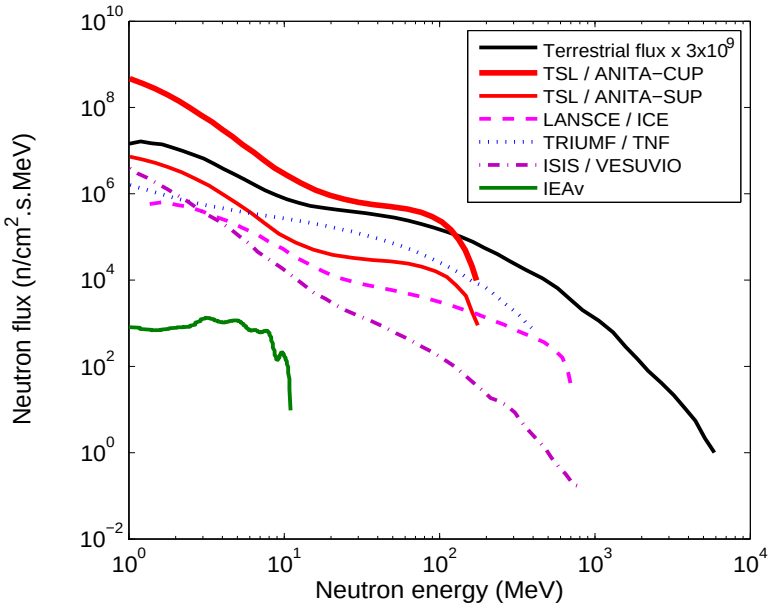
Conforme verificado na seção 2.3.1, as memórias Flash de nó tecnológico 50 nm ou mais antigo são praticamente insuscetíveis à SEU. Portanto, no SAD, além da memória SRAM do PSoC 5LP, os demais circuitos com possibilidade de se observar os efeitos singulares são a CPU ARM Cortex-M3 e os periféricos digitais, analógicos e de sinais mistos. Acredita-se que um dos motivos pelos quais não se observaram erros no teste foram os possíveis mascaramentos nesses circuitos. É difícil mostrar no SAD exatamente em que nós do circuito os mascaramentos podem ocorrer, uma vez que a maior parte do projeto na ferramenta PSoC Creator é feita em *software* e em circuitos em formato de componente fechado. De qualquer forma, além da CPU ARM Cortex-M3, é possível imaginar que podem ocorrer mascaramentos de SETs conforme explicado na seção 2.3.3 nos seguintes blocos formados por periféricos: amostradores, conversores, DMAs, sincronizador e interface serial RS-232/UART. A título de exemplo, com base no esquemático presente no apêndice A, é possível usar o circuito do sincronizador para mostrar a possibilidade de mascaramentos, uma vez que ele é formado por diversos blocos de

periféricos da biblioteca da ferramenta. É possível ocorrer mascaramento temporal de um SET na entrada dos flip-flops usados no sincronizador, e mascaramento lógico de um SET na entrada da porta AND usada próxima ao registrador Status_Reg_2. Evidentemente, o mascaramento elétrico pode ocorrer em ambos os casos desses dois últimos exemplos.

Um mascaramento importante que pode ter ocorrido tem característica diferente dos mascaramentos explicados na seção 2.3.3, porque diz respeito ao mascaramento não em nível de circuito, mas em nível de sistema. Ele consiste no mascaramento temporal dos amostradores, conversores e sincronizador nos instantes de tempo entre o fim de uma conversão pelos ADCs e o início da próxima conversão. Conforme mostrado na figura 4.12, nesse intervalo de tempo esses três blocos ficam inativos aguardando a conclusão do processamento (votações e execução do bloco de registro de estado), e portanto podem mascarar a ocorrência de SETs.

Outro motivo pelo qual não foram observados erros no SAD é o baixo fluxo de nêutrons do *setup* de injeção de falhas se comparado ao praticado em outras instalações existentes em outros países. É importante ter um fluxo de nêutrons elevado porque ele permite a observação de eventos com maior frequência e a obtenção de taxas de eventos justas, possibilitando o acúmulo de suficiente estatística em tempos curtos. A figura 4.17 mostra um comparativo do espectro do fluxo de nêutrons de importantes instalações para teste de irradiação com nêutrons no mundo. Observa-se o baixo fluxo das instalações do IEAv em comparação com os demais.

Figura 4.17 - Comparativo do espectro do fluxo de nêutrons entre diversas instalações para teste de irradiação com nêutrons.



Fonte: Autoria própria. Baseado em PROKOFIEV et al., 2013 e INTERNATIONAL STANDARD ORGANIZATION, 2011.

Portanto a não observância de erros no teste de injeção de falhas com irradiação de nêutrons é creditada aos três motivos discutidos nos parágrafos anteriores: pequena sensibilidade a SEU da memória SRAM, a ocorrência de mascaramentos de SETs nos circuitos combinacionais e analógicos e o baixo fluxo de nêutrons do *setup* onde foram realizadas as injeções de falhas. Como o resultado do teste de injeção de falhas por irradiação não permitiu avaliar a eficácia do SAD adotado com base em TMR e diversidade espacial-temporal contra SEU, SET e SEFI, foi realizada uma investigação teórica baseada em análise combinatória. Essa investigação considerou diretamente a ocorrência de *bit-flips* na saída dos conversores, e portanto avaliou o ganho na tolerância de falhas apenas devido à diversidade temporal, uma vez que o ganho decorrente da diversidade espacial resulta dos diferentes níveis de resiliência de cada *hardware*. Foram analisados casos de falhas simples (1 único *bit-flip*), falhas duplas (2 *bit-flips*) e falhas múltiplas (3 ou mais *bit-flips*), considerando apenas o pior caso, em que falhas ocorrem em distintos domínios espacial e temporal. A tabela 4.2 mostra um resumo da

Tabela 4.2 - Capacidade do SAD adotado em tolerar falhas simples, duplas e múltiplas.

Caso de falha	***Combinação de falha***	***Falha (bit-flip)***			***Tolerância (Sim/Não)***	***Tolerância percentual (%)***
		Módulo 1	***Módulo 2***	***Módulo 3***		
Falhas simples	Primeira	Sem falhas	Sem falhas	Simples	Sim	+33,33
	Segunda	Sem falhas	Simples	Sem falhas	Sim	+33,33
	Terceira	Simples	Sem falhas	Sem falhas	Sim	+33,33
	Tolerância efetiva total com falhas simples					100
Falhas duplas	Quarta	Sem falhas	Simples	Simples	Não	-33,33
	Quinta	Simples	Sem falhas	Simples	Sim	+33,33
	Sexta	Simples	Simples	Sem falhas	Sim	+33,33
	Tolerância efetiva total com falhas duplas					66,66
Falhas múltiplas	Sétima	Duplas ou múltiplas	Sem falhas	Simples	Sim	+16,63
					Não	-16,70
	Oitava	Duplas ou múltiplas	Simples	Sem falhas	Sim	+16,63
					Não	-16,70
	Nona	Simples, dúplas ou múltiplas	Simples	Simples	Não	-33,33
	Tolerância efetiva total com falhas múltiplas					33,26

Fonte: Autoria própria, 2015.

investigação. Nos casos de falhas simples e duplas as tolerâncias quantitativas foram estimadas através de análise intuitiva da operação do SAD em questão. No caso de falhas múltiplas, essas tolerâncias foram estimadas através de uma tabela verdade que considera falhas nas 9 amostras geradas pelo ADC SAR que opera a 740 ksps, e leva em conta que bastam 5 amostras coerentes para que o votador do ADC SAR propicie a tolerância das outras 4 amostras com possíveis erros.

No caso de falhas simples, o SAD é capaz de tolerar 100% das falhas, como já é de conhecimento comum para sistemas baseados em TMR. No caso de falhas duplas, o SAD é capaz de tolerar aproximadamente 66% das falhas, o que já configura um ganho de confiabilidade em relação ao TMR clássico. No caso de falhas múltiplas, o SAD é capaz de tolerar aproximadamente 33% das falhas. É importante notar que esse ganho na tolerância de falhas deve-se à diversidade temporal, implementada pela sobreamostragem realizada pelo ADC SAR à 740 ksps e pelo seu votador. Considerando a baixa probabilidade de ocorrerem múltiplas falhas em distintos domínios espacial e temporal, a tolerância do SAD de 33% no caso de falhas múltiplas pode não ser considerada crítica. Portanto, essa investigação mostra que em comparação com a técnica TMR clássica, a adição de diversidade temporal gera um ganho significativo na tolerância de falhas duplas e múltiplas.

5 SOFT ERRORS EM UM SISTEMA DE AQUISIÇÃO DE DADOS BASEADO EM TMR E DIVERSIDADE ESPACIAL-TEMPORAL: ESTUDO COM INJEÇÃO DE FALHAS POR SOFTWARE E EM TEMPO DE EXECUÇÃO

Esse estudo aborda a injeção de falhas por *software* e em tempo de execução no SAD analógico-digital descrito na seção 4, realizada através de *bit-flips* nos registradores de controle dos periféricos e na SRAM do PSoC 5LP. O objetivo foi avaliar a sensibilidade e o comportamento desse SAD com relação aos *soft errors*, permitindo a comparação dos resultados com os obtidos na injeção de falhas por irradiação. Partiu-se do princípio de aproveitar ao máximo o *setup* de teste desenvolvido no estudo da seção 4, de maneira a também não descaracterizar o SAD irradiado. Essa seção está organizada da seguinte forma: em 5.1 é apresentado o esquema de injeção de falhas adotado; em 5.2 são detalhadas as implementações realizadas no DUT e AE para conformidade com esse esquema; em 5.3 é mostrado como foi realizada a validação do *setup*; em 5.4 são apresentados os procedimentos de teste; e em 5.5 apresentados os resultados e discussões.

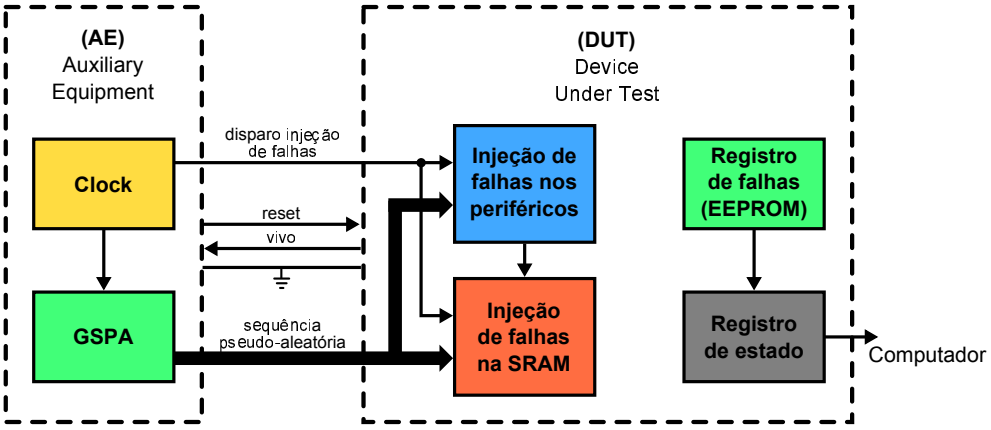
5.1 Esquema de injeção de falhas

O esquema de injeção de falhas adotado é baseado em um gerador de sequência pseudo-aleatória (GSPA) e uma interrupção em *software*. O GSPA tem a função de definir um registrador de controle dos periféricos ou endereço da SRAM alvo para a injeção da falha, e a interrupção a função de proporcionar a execução de um trecho de código de programação para que a falha seja efetivamente inserida. Como o PSoC 5LP possui um GSPA em sua biblioteca de componentes pré-construídos, inicialmente pensou-se em implementar todo o esquema de injeção de falhas internamente no DUT, construindo assim um sistema auto injetor de falhas. Porém o teste com DUT com GSPA interno apresenta a limitação de a sequência gerada ser sempre a mesma a cada reinicialização do GSPA, uma vez que ele é um gerador pseudo-aleatório. Um esquema de injeção de falhas nesse formato seria viciado, dado que os registradores e endereços alvos seriam sempre os mesmos a cada *reset* do DUT. A alternativa de alterar o polinômio do GSPA a cada *reset* do DUT foi descartada, uma vez que seriam necessários muitos polinômios e que estes deveriam ser bem escolhidos para exercitar toda a faixa de alvos desejada. Nesse contexto, a solução adotada foi utilizar um GSPA externo ao DUT, localizado no AE.

O esquema de injeção de falhas adotado é mostrado na figura 5.1. No AE existe um *clock*

que dispara o GSPA. A sequência gerada pelo GSPA é enviada ao DUT através de uma conexão paralela. O mesmo *clock* que dispara o GSPA é também enviado ao DUT, para disparar a interrupção em *software* para a injeção da falha. No DUT existe um bloco de injeção de falhas nos periféricos e um bloco de injeção de falhas na SRAM, ambos construídos usando recursos de *hardware* e principalmente *software*. Ainda no DUT, é implementado em *software* um bloco de registro de falhas. A necessidade desse bloco parte de uma característica do *setup* de teste do estudo com radiação, em que a comunicação com um computador para a geração de registros é feita apenas pelo DUT. Assim, o bloco de registro de falhas tem a função de armazenar em sua EEPROM o número de *resets* do DUT, o número de falhas injetadas até a ocorrência de um erro e o número total de falhas injetadas no decorrer do teste. Essas informações são disponibilizadas ao bloco de registro de estado descrito na seção 4.3.1.7 para a geração do relatório do teste. Na figura 5.1 é mostrada apenas a implementação destinada a injeção e registro de falhas, mas toda a implementação descrita na seção 4 permanece presente no DUT e AE.

Figura 5.1 - Esquema de injeção de falhas por software e em tempo de execução.



Fonte: Autoria própria, 2015.

5.2 Implementações realizadas no DUT e AE

Nas seções 5.2.1 e 5.2.2 são detalhas as implementações realizadas no DUT e AE para a injeção de falhas desse estudo.

5.2.1 Implementações no DUT

O esquemático da implementação do DUT desse estudo encontra-se no apêndice E desse trabalho e o código em linguagem C no apêndice F. Basicamente a implementação do DUT desse estudo consiste no SAD analógico-digital descrito na seção 4, com a adição dos blocos para a injeção de falhas por *software*, os quais são detalhados a seguir.

5.2.1.1 Bloco de injeção de falhas nos periféricos

No PSoC 5LP cada registrador de controle dos periféricos é composto de 8 bits. A tabela 5.1 apresenta a visão geral da organização desses registradores (CYPRESS SEMICONDUCTOR, 2014). Sua faixa nominal de endereços vai de 0x40004000 a 0x5FFFFFFF, o que totalizaria 536.854.527 endereços, porém os registradores existentes fisicamente são aproximadamente 13.000. Esse contexto mostra que dentro da faixa nominal

Tabela 5.1 - Visão geral da organização dos registradores de controle dos periféricos do PSoC 5LP.

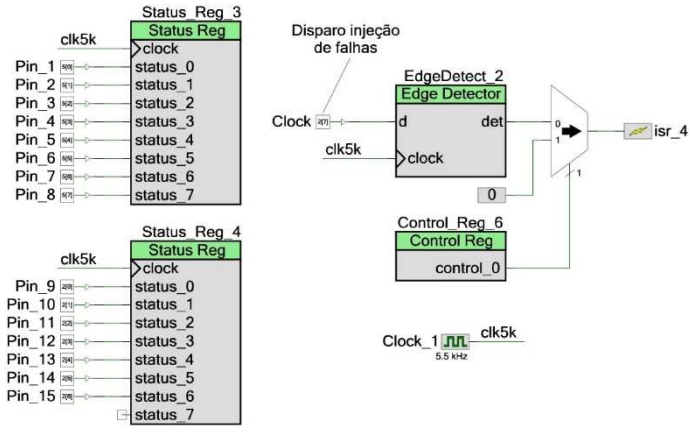
Faixa de endereços	Propósito/Periférico	Priorização para injeção de falhas
0x4000 4000 – 0x4000 42FF	**Clocking, PLLs, and oscillators**	**Sim**
0x4000 4300 – 0x4000 43FF	**Power management**	**Sim**
0x4000 4500 – 0x4000 45FF	Ports interrupt control	Não
0x4000 4700 – 0x4000 47FF	Flash programming interface	Não
0x4000 4900 – 0x4000 49FF	I2C controller	Não
0x4000 4E00 – 0x4000 4EFF	**Decimator**	**Sim**
0x4000 4F00 – 0x4000 4FFF	**Fixed timer/counter/PWMs**	**Sim**
0x4000 5000 – 0x4000 51FF	General purpose I/Os	Não
0x4000 5300 – 0x4000 530F	Output port select register	Não
0x4000 5400 – 0x4000 54FF	External memory interface control registers	Não
0x4000 5800 – 0x4000 5FFF	**Analog subsystem interface**	**Sim**
0x4000 6000 – 0x4000 60FF	USB controller	Não
0x4000 6400 – 0x4000 6FFF	**UDB configuration**	**Sim**
0x4000 7000 – 0x4000 7FFF	**PHUB configuration**	**Sim**
0x4000 8000 – 0x4000 87FF	EEPROM	Não
0x4000 A000 – 0x4000 A400	CAN	Não
0x4000 C000 – 0x4000 C800	Digital filter block	Não
0x4001 0000 – 0x4001 FFFF	**Digital interconnect configuration**	**Sim**
0x4800 0000 – 0x4800 7FFF	Flash ECC bytes	Não

Fonte: Adaptado de CYPRESS SEMICONDUCTOR, 2014.

de endereços existem lacunas, o que torna inviável a injeção de falhas em endereços aleatórios, uma vez que a contabilidade de falhas inseridas seria prejudicada. Como alternativa a essa questão optou-se por colocar todos os endereços válidos de registradores dentro de um vetor existente no bloco de injeção de falhas interno ao DUT. A definição de um desses endereços como alvo para a injeção de *bit-flips* é feita por um número aleatório proveniente do GSPA interno ao AE. Em um teste preliminar observou-se que a memória SRAM do PSoC 5LP não suportaria todos os aproximados 13.000 endereços de registradores. A solução adotada foi escolher endereços de registradores de controle de alguns periféricos específicos, priorizando os periféricos analógicos e digitais utilizados no SAD em questão, o que resultou em 7.892 registradores. A tabela 5.1 mostra também quais desses periféricos foram priorizados.

A entrada no DUT da sequência pseudo-aleatória gerada pelo GSPA e do *clock* que dispara a interrupção para a injeção de uma falha em *software* é feita utilizando recursos de *hardware*, conforme mostrado na figura 5.2. Os registradores Status_Reg_3 e Status_Reg_4 recebem a sequência proveniente da conexão paralela entre o DUT e o AE e a disponibiliza ao *software* para a definição do registrador/endereço alvo. O sinal identificado por *clock*, também proveniente do AE, dispara a interrupção denominada isr_4 para que a falha seja inserida em *software*.

Figura 5.2 - Entrada no DUT da sequência pseudo-aleatória e do clock que dispara a interrupção para a injeção de uma falha.



Fonte: Autoria própria, 2015.

Na injeção de falhas nos periféricos optou-se por utilizar um modelo de falhas denominado neste trabalho de semipermanente. Nesse modelo uma falha é inserida e após aproximadamente 500 milissegundos retirada, ficando portanto ativa por um período consideravelmente maior à duração de um pulso transiente de um *soft error*, o qual é da ordem

de pico a nanossegundos. A escolha desse modelo leva em consideração a possibilidade da duração de um SET ser maior do que a duração do seu pulso transiente, devido a este pulso ser capturado por um *latch*. A opção pela retirada da falha visa evitar o acúmulo de falhas, configurando assim a injeção de falhas simples.

O código em linguagem C que efetivamente insere a falha após a detecção de uma interrupção é mostrado na figura 5.3. A inserção e retirada da falha é realizada pelo ponteiro pointer_addr. A retirada é feita na linha 7, e sempre ocorre após que uma primeira falha é injetada, e a inserção é feita na linha 22. A variável prs_number contém a sequência pseudo-aleatória usada para definir o alvo, e o vetor periph_reg contém o conjunto de endereços dos registradores de controle dos periféricos que se deseja exercitar. A variável state_bit indica um bit para a injeção do *bit-flip*. A função faults_counter_eeprom contabiliza o número total de falhas inseridas.

Figura 5.3 - Código em linguagem C que insere falhas nos periféricos.

```
1     if (flag_2 == 1)
2     {
3         flag_2 = 0;
4         if (first_fault == 1)
5         {
6             updated_value = *pointer_addr;
7             *pointer_addr = updated_value ^ target_bit[state_bit];
8         }
9         prs_number = (Status_Reg_4_Read() << 8) & 0x1f00;
10        prs_number = prs_number + Status_Reg_3_Read();
11        if (prs_number <= 7892)
12        {
13            state_bit++;
14            if (state_bit == 8)
15            {
16                state_bit = 0;
17            }
18            first_fault = 1;
19            faults_counter_eeprom();
20            target_addr = periph_reg[prs_number];
21            pointer_addr = (reg8 *) periph_reg[prs_number];
22            *pointer_addr = *pointer_addr ^ target_bit[state_bit];
23
24            led_fault_inject = !led_fault_inject;
25            Control_Reg_5_Write(led_fault_inject);
26        }
27    }
```

Fonte: Autoria própria, 2015.

5.2.1.2 Bloco de injeção de falhas na SRAM

Esse bloco utiliza o mesmo recurso de *hardware* do bloco de injeção de falhas nos periféricos (mostrado na figura 5.2) para obter do AE a sequência pseudo-aleatória que define o alvo e o *clock* que dispara a interrupção para a injeção da falha.

A SRAM do PSoC 5LP possui endereçamento orientado a byte e é composta por 2 bancos de aproximadamente 32 KB, conforme mostrado na tabela 5.2. Partindo dos endereços bases 0x1FFF8000 e 0x20000000, é possível atingir qualquer endereço de cada um dos bancos a partir de um número de 15 bits. Esta foi a estratégia adotada para a implementação do bloco de injeção de falhas na SRAM, em que esse número de 15 bits é proveniente da sequência pseudo-aleatória gerada no AE.

Tabela 5.2 - Visão geral da organização da SRAM do PSoC 5LP.

Faixa de endereços	Propósito
0x1FFF 8000 – 0x1FFF FFFF	Up to 32 KB SRAM in code region
0x2000 0000 – 0x2000 7FFF	Up to 32 KB SRAM in SRAM region

Fonte: Adaptado de CYPRESS SEMICONDUCTOR, 2013a.

O modelo adotado para a injeção de falhas na SRAM foi o de falhas permanentes, em que uma falha inserida nunca é retirada pelo bloco injetor. Esse modelo visa estar em conformidade com o SEU, que é caracterizado necessariamente como um *bit-flip* permanente até que a célula de memória seja reescrita ou reiniciada. A taxa de injeção de falhas adotada foi de 10 falhas por segundo, o que corresponde a uma falha a cada 100 ms. Considerando que a execução do bloco de injeção de falhas na SRAM dura 27 ms, após a injeção de uma falha e antes da injeção da próxima o teste é executado por 73 ms. Considerando também que, conforme a figura 4.12, o tempo de um ciclo de votação no SAD é de 182 µs, tem-se que são realizados 401 ciclos de votação entre a injeção de uma falha e a próxima. Portanto, considerou-se a taxa de injeção de falhas adotada como adequada, uma vez que permite tanto exercitar a falha injetada por diversos ciclos de votação quanto agilizar a execução do teste. Apesar disso é importante observar que a quantidade de falhas injetadas até a observação de um erro é função dessa taxa de injeção, uma vez que o exercício de uma falha por um tempo tendendo ao infinito permite identificar com maior precisão se a falha injetada é causadora ou não de erro. Ou seja, buscou-se com a taxa de injeção de 10 falhas por segundo equilibrar a relação entre a condição ideal de teste e o tempo necessário para tal.

O código em linguagem C que efetivamente insere a falha na SRAM após a detecção de uma interrupção é mostrado na figura 5.4. A variável prs_number contém um número pseudo-aleatório de 15 bits, ao qual são somados os endereços bases 0x1FFF8000 e 0x20000000 dos bancos da SRAM, constituindo a variável target_addr que contém o endereço alvo para a injeção da falha. A variável state_bit indica um bit para a injeção do *bit-flip*. A inserção da falha é efetivamente realizada pelo ponteiro pointer_addr na linha 22. A função faults_counter_eeprom contabiliza o número total de falhas inseridas. E o teste condicional da linha 18 tem a função de garantir que falhas não sejam inseridas em variáveis do próprio bloco injetor de falhas, o que poderia gerar efeitos indesejados, como por exemplo a inserção de falhas em endereços diferentes aos da SRAM.

Figura 5.4 - Código em linguagem C que insere falhas na SRAM.

```
1     if (flag_2 == 1)
2     {
3         flag_2 = 0;
4         prs_number = (Status_Reg_4_Read() << 8) & 0xff00;
5         prs_number = prs_number + Status_Reg_3_Read();
6
7         if (low_high_addr == 0)
8         {
9             low_high_addr = 1;
10            target_addr = low_addr + prs_number;
11        }
12        else
13        {
14            low_high_addr = 0;
15            target_addr = high_addr + prs_number;
16        }
17
          if ((target_addr != (uint32)&flag_2) &&
              (target_addr != (uint32)&low_addr) &&
18            (target_addr != (uint32)&high_addr) &&
              (target_addr != (uint32)&low_high_addr) &&
              (target_addr != (uint32)&state_bit))
19        {
20            faults_counter_eeprom();
21            pointer_addr = (reg8 *) target_addr;
22            *pointer_addr = *pointer_addr ^ target_bit[state_bit];
23
24            state_bit++;
25            if (state_bit == 8)
26            {
27                state_bit = 0;
28            }
29
30            led_fault_inject = !led_fault_inject;
31            Control_Reg_5_Write(led_fault_inject);
32        }
33    }
```

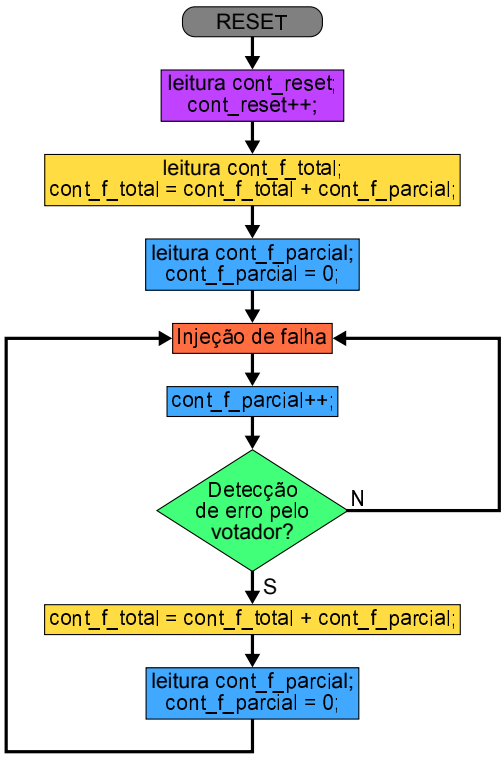
Fonte: Autoria própria, 2015.

No teste de injeção de falhas nos periféricos observou-se uma dificuldade, que é abordada em detalhes na seção 5.5. Basicamente consiste na detecção de muitos erros com poucas falhas inseridas, o que atribui-se à não recuperação do sistema às falhas inseridas. Para eliminar essa dificuldade, o SAD no teste de injeção de falhas na SRAM foi configurado para se reiniciar após uma detecção de falhas pelos votadores e registro desse evento no relatório de estado. Este *reset* foi implementado por meio da função Cysoftwarereset do PSoC 5LP.

5.2.1.3 Bloco de registro de falhas

O bloco de registro de falhas destina-se a fornecer as informações importantes para avaliar a sensibilidade do SAD em questão frente às falhas injetadas. Foram eleitos como informações importantes o número de *resets* do DUT, o número de falhas injetadas até a ocorrência de um erro e o número total de falhas injetadas no decorrer do teste. Informações referentes ao registrador ou endereço em que a injeção de uma falha ocasionou um erro não são de interesse nesse estudo, e portanto não são tratadas. Como esse bloco de registro de falhas é interno ao DUT, os três parâmetros mencionados como relevantes são armazenados na EEPROM do PSoC 5LP. A característica de não-volatilidade dessa memória permite que os dados sejam

Figura 5.5 - Resumo da operação do bloco de registro de falhas.

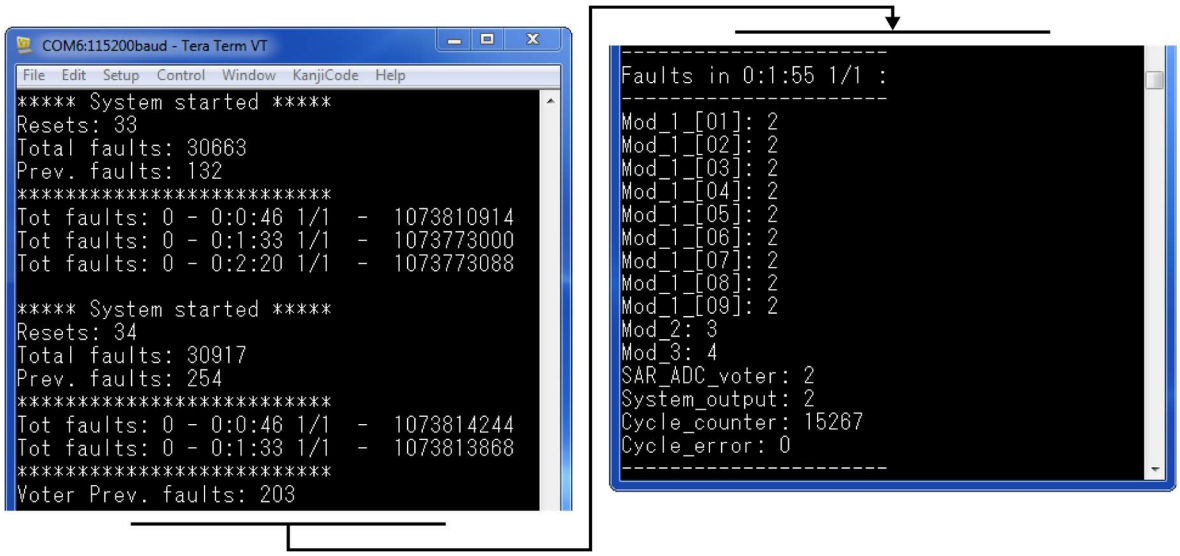


Fonte: Autoria própria, 2015.

recuperados mesmo após uma falha injetada resultar em um desvio na funcionalidade do sistema, como por exemplo um *reset*. Esse bloco de registro é implementado em linguagem C basicamente pelas funções resets_counter_eeprom, calculate_total_faults, faults_counter_eeprom e reinitialize_faults_counter_eeprom, as quais podem ser consultadas no apêndice F. O bloco de registro de estado descrito na seção 4.3.1.7 foi atualizado para reportar esses parâmetros armazenados na EEPROM durante a reinicialização do SAD ou no momento da detecção de um erro pelos votadores. A figura 5.5 resume a operação desse bloco de registro de falhas, em que os parâmetros de interesse são nomeados como cont_reset, cont_f_total e cont_f_parcial.

Um exemplo de parte do relatório criado com as informações do bloco de registro de falhas é mostrado na figura 5.6.

Figura 5.6 - Exemplo de parte do relatório criado com as informações do bloco de registro de falhas.



Fonte: Autoria própria, 2015.

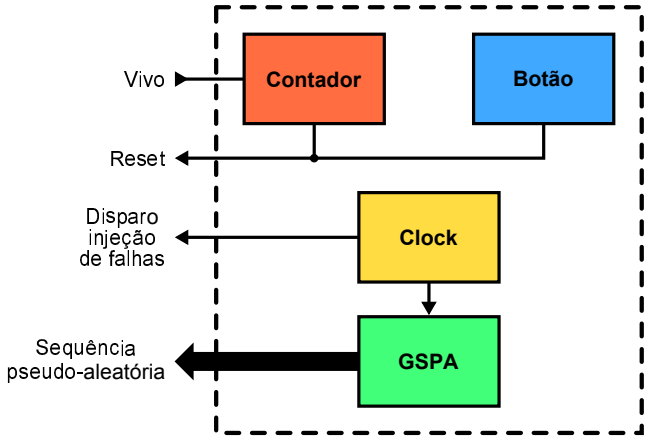
5.2.2 Implementação do AE

O AE utilizado no *setup* de teste da seção 4 foi implementado em um PSoC 5, a versão anterior do PSoC 5LP. Estes dispositivos são funcionalmente muito semelhantes, porém as placas de desenvolvimento disponibilizadas pelo fabricante e utilizadas nesse trabalho são consideravelmente diferentes. A placa de desenvolvimento do PSoC 5LP (CYPRESS SEMICONDUCTOR, 2013b) oferece melhor acesso aos recursos do dispositivo, como por exemplo, um conector USB específico para a comunicação de dados, um conector específico

para a interface serial RS-232, uma área específica para prototipações e conectores do tipo *header* para todos os IOs (*Inputs-Outputs*). Devido a esta última facilidade, como nesse estudo o AE deve conectar 15 bits de dados com o DUT para a transferência da sequência pseudo-aleatória, optou-se por migrar o AE também para um PSoC 5LP.

A nova implementação do AE no PSoC 5LP foi baseada na implementação do AE da seção 4, com a adição do GSPA, *clock* para disparo do GSPA e da interrupção para injeção da falha, e registradores para a interface dessa informação com o DUT. Da mesma forma que realizado na seção 4, na figura 5.7 é mostrado o diagrama funcional do AE implementado para esse estudo. A funcionalidade de monitoração e *reset* (*watchdog*) do DUT opera da mesma forma que o descrito na seção 4.3.2. Além disso, no AE em questão foi implementado um *clock* que dispara o GSPA para gerar a sequência pseudo-aleatória, e que também é enviado ao DUT para disparar a interrupção para a inserção de uma falha. O esquemático da implementação do AE no PSoC 5LP encontra-se no apêndice G desse trabalho e o código em linguagem C no apêndice H.

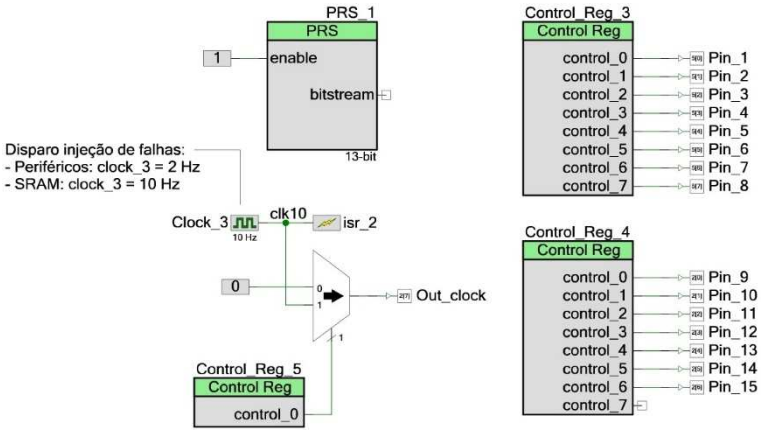
Figura 5.7 - Diagrama funcional do AE para o estudo com injeção de falha por software.



Fonte: Autoria própria, 2015.

A saída do AE da sequência pseudo-aleatória gerada pelo GSPA e do *clock* para disparo da interrupção é realizada de forma semelhante ao DUT, conforme mostrado na figura 5.8. A sequência gerada pelo GSPA, na figura denominado de PRS (*Pseudo Random Sequence*), é enviada para o DUT através dos registradores Status_Reg_3 e Status_Reg_4. O clock_3 é exatamente o disparador do GSPA e da interrupção no DUT para a inserção de falhas. É também por meio dele que se configura a taxa de injeção de falhas, diferente para o teste dos periféricos e da SRAM, conforme mostrado na figura em questão.

Figura 5.8 - GSPA, e saída do AE da sequência pseudo-aleatória e do clock para disparo da interrupção no DUT para a inserção de falhas.



Fonte: Autoria própria, 2015.

5.3 Validação dos setups

Nesse estudo a injeção de falhas nos periféricos e na SRAM foi realizada separadamente, configurando dois *setups* de teste. O objetivo com isso foi facilitar a análise dos resultados e refinar as conclusões. O AE utilizado em ambos os *setups* é semelhante, alterando apenas a frequência do *clock* para disparo da injeção de falhas e o número de bits do GSPA, e o DUT diferenciado pela anulação das linhas de código que implementam o bloco de injeção de falhas nos periféricos ou o bloco de injeção de falhas na SRAM.

Nesse contexto, a validação desses *setups* foi realizada de maneira independente, conforme descrito nas seções 5.3.1 e 5.3.2 a seguir. Os critérios para considerar os *setups* como validados foram os seguintes:

a) as implementações para a injeção de falhas em *software* não alteram o comportamento do SAD implementado e validado na seção 4;
b) com essas implementações rodando, se comentadas as linhas de código que inserem as falhas, erros não são observados;
c) no *setup* de injeção de falhas nos periféricos, o endereço do registrador alvo é adequadamente obtido da sequência pseudo-aleatória, o *bit-flip* é adequadamente inserido de acordo com a posição previamente definida e adequadamente retirado antes de inserir a próxima falha;
d) no *setup* de injeção de falhas na SRAM, o endereço alvo é adequadamente composto a partir da sequência pseudo-aleatória e dos endereços bases dos

bancos, e o *bit-flip* é adequadamente inserido de acordo com a posição previamente definida;

e) o bloco de registro de falhas funciona adequadamente conforme descrito na figura 5.5.

5.3.1 Validação do setup de injeção de falhas nos periféricos

A validação desse *setup* iniciou com o *debug in circuit* do bloco de injeção de falhas nos periféricos, com o objetivo de atender ao critério da alínea “c” da seção 5.3. A figura 5.9 apresenta um breve resumo desse *debug*, usando como referência o código e numeração de linhas da figura 5.3. Com fins didáticos, a análise do *debug* inicia imediatamente após a execução da linha 21, em que é definido como alvo o registrador de endereço 0x40014CD2, cujo conteúdo inicialmente é 0x00. No código da linha 22 um *bit-flip* é inserido no sétimo bit da direita para esquerda, e o conteúdo desse registrador é então alterado para 0x40. Essa falha fica ativa por aproximadamente 500 milissegundos e antes de injetar a próxima falha ela é retirada, através do código da linha 7. Assim, o conteúdo do endereço alvo em questão (0x40014CD2) retorna para seu valor inicial (0x00). Repetindo-se esse *debug* diversas vezes e encontrando coerência, o *setup* foi considerado validado no que concerne ao bloco de injeção de falhas nos periféricos.

A validação do registro de falhas, com o objetivo de atender ao critério da alínea “e” da seção 5.3, foi realizada através de teste funcional do *setup* e posterior análise do relatório criado pelo DUT, comparando-o ao fluxograma da figura 5.5. Eventuais incoerências foram corrigidas. Devido à simplicidade desse procedimento de validação e ao mesmo tempo dificuldade de sistematização, ele não é abordado em detalhes. A validação final do *setup*, principalmente com o objetivo de atender aos critérios das alíneas “a” e “b” da seção 5.3, foi realizada também através de um teste funcional do *setup*. Porém nesse caso se alterou o código do bloco de injeção de falhas nos periféricos para injetar o que foi denominado neste trabalho de pseudo-falhas, conforme explicado a seguir. Na figura 5.10, a linha 22 foi comentada para que não fossem inseridas falhas, a linha 7 também foi comentada para não configurar injeção de falha, e no lugar dela o *software* realizava a leitura do conteúdo do registrador alvo e imediatamente após, a escrita desse conteúdo no mesmo registrador alvo. O objetivo desse teste de validação foi verificar que a implementação para injeção de falhas e principalmente o processo de escrita nos registradores dos periféricos sem alterar seu conteúdo não configuravam injeções de falhas. A execução desse teste permitiu verificar que o processo de leitura dos

Figura 5.9 - Resumo do debug in circuit realizado para validação da injeção de falhas nos periféricos. (a) Posteriormente à definição do registrador e bit alvos (linha 21). (b) Posteriormente à injeção da falha (linha 22). (c) Posteriormente à retirada da falha (linha 7).

(a)

Watch 1

Name	Value	Address	Type	Radix
flag_2	0x00 '\000'	0x1FFFFCED (All)	uint8	Default
first_fault	0x01 '\001'	0x1FFFFCEC (All)	uint8	Default
prs_number	7671	0x1FFFFCDE (All)	uint16	decimal
pointer_addr	0x40014CD2 (All)	0x1FFFFCE8 (All)	reg8 *	Default
*pointer_addr	0x00 '\000'	0x40014CD2 (All)	reg8	Default
state_bit	0x05 '\005'	0x1FFFFCE4 (All)	uint8	Default
updated_value	0x10 '\020'	0x200005EC (All)	uint8	Default

(b)

Watch 1

Name	Value	Address	Type	Radix
flag_2	0x00 '\000'	0x1FFFFCED (All)	uint8	Default
first_fault	0x01 '\001'	0x1FFFFCEC (All)	uint8	Default
prs_number	3567	0x1FFFFCDE (All)	uint16	decimal
pointer_addr	0x40010B18 (All)	0x1FFFFCE8 (All)	reg8 *	Default
*pointer_addr	0x40 '@'	0x40010B18 (All)	reg8	Default
state_bit	0x06 '\006'	0x1FFFFCE4 (All)	uint8	Default
updated_value	0x00 '\000'	0x200005EC (All)	uint8	Default

(c)

Watch 1

Name	Value	Address	Type	Radix
flag_2	0x00 '\000'	0x1FFFFCED (All)	uint8	Default
first_fault	0x01 '\001'	0x1FFFFCEC (All)	uint8	Default
prs_number	3567	0x1FFFFCDE (All)	uint16	decimal
pointer_addr	0x40010B18 (All)	0x1FFFFCE8 (All)	reg8 *	Default
*pointer_addr	0x00 '\000'	0x40010B18 (All)	reg8	Default
state_bit	0x06 '\006'	0x1FFFFCE4 (All)	uint8	Default
updated_value	0x40 '@'	0x200005EC (All)	uint8	Default

Fonte: Autoria própria, 2015.

registradores dos periféricos não resultava em erros, o que não configura injeção de falhas. Porém, verificou-se que a escrita nesses registradores, ainda que do mesmo valor lido imediatamente antes à escrita, ocasionava erros eventuais. Esses erros eram observados, por exemplo, como a mudança da velocidade de escrita da interface serial RS-232 ou a não conclusão de um ciclo de votação pelo *hardware* do SAD, o qual travava a operação do sistema.

Diante da impossibilidade de se atuar para eliminar a injeção de falhas não proposital, realizou-se um teste com o código do bloco em questão modificado, conforme a figura 5.10, no qual foram contabilizadas a injeção de 25.023 pseudo-falhas. Os resultados desse teste de

Figura 5.10 - Alteração do código do bloco de injeção de falhas nos periféricos para a injeção de pseudo-falhas.

```
1     if (flag_2 == 1)
2     {
3         flag_2 = 0;
4         if (first_fault == 1)
5         {
6             updated_value = *pointer_addr;
7             //*pointer_addr = updated_value ^ target_bit[state_bit];
              *pointer_addr = updated_value;
8         }
9         prs_number = (Status_Reg_4_Read() << 8) & 0x1f00;
10        prs_number = prs_number + Status_Reg_3_Read();
11        if (prs_number <= 7892)
12        {
13            state_bit++;
14            if (state_bit == 8)
15            {
16                state_bit = 0;
17            }
18            first_fault = 1;
19            faults_counter_eeprom();
20            target_addr = periph_reg[prs_number];
21            pointer_addr = (reg8 *) periph_reg[prs_number];
22            //*pointer_addr = *pointer_addr ^ target_bit[state_bit];
23
24            led_fault_inject = !led_fault_inject;
25            Control_Reg_5_Write(led_fault_inject);
26        }
27    }
```

Fonte: Autoria própria, 2015.

injeção de pseudo-falhas são mostrados na tabela 5.3. A quantidade de 25 erros observados, que correspondem à 0,1% das falhas injetadas, foi considerada baixa em comparação aos resultados preliminares com injeção de falhas conforme previsto neste estudo. Por esse motivo, optou-se por realizar o estudo mesmo com essa limitação.

Tabela 5.3 - Resultados do teste de injeção de pseudo-falhas.

Total de falhas injetadas		**25023**	**100%**
Total de erros detectados		**25**	**0,10%**
Total de SEFIs (desencadeadores de reset)		25	0,10%
Desconfiguração da interface serial	Reset manual	15	-
Atuação do watchdog do AE	Reset automático	10	-

Fonte: Autoria própria, 2015.

5.3.2 Validação do setup de injeção de falhas na SRAM

A alínea “e” dos critérios de validação da seção 5.3 refere-se ao bloco de registro de falhas. Como esse bloco já foi testado na validação do *setup* de injeção de falhas nos periféricos, e como ele é exatamente o mesmo para ambos os *setups*, o considera-se como validado. Assim, a validação do *setup* de injeção de falhas na SRAM iniciou com o *debug in circuit* do bloco de injeção de falhas, com o objetivo de atender ao critério da alínea “d”.

A figura 5.11 apresenta um breve resumo desse *debug*, usando como referência o código e numeração de linhas da figura 5.4. No código da linha 21 é definido como alvo o endereço 0x1FFFB3D8, cujo conteúdo inicialmente é 0x39. No código da linha 22 um *bit-flip* é inserido no quinto bit da direita para a esquerda, e o conteúdo desse endereço é alterado para 0x29, configurando assim a injeção da falha. Repetindo esse *debug* diversas vezes e encontrando

Figura 5.11 - Resumo do debug in circuit realizado para a validação da injeção de falhas na SRAM. (a) Posteriormente à definição do endereço alvo e antes da injeção da falha (linha 21). (b) Posteriormente à injeção da falha (linha 22).

(a)

Watch 1

Name	Value	Address	Type	Radix
flag_2	0 '\000'	0x1FFF8196 (All)	uint8	Default
prs_number	0x33D8	0x1FFF8198 (All)	uint16	hex
low_high_addr	1 '\001'	0x1FFF819A (All)	uint8	Default
low_addr	0x1FFF8000	0x1FFF8114 (All)	uint32	hex
high_addr	0x20000000	0x1FFF8118 (All)	uint32	hex
target_addr	0x1FFFB3D8	0x1FFF819C (All)	uint32	hex
pointer_addr	0x1FFFB3D8 (All)	0x1FFF81A0 (All)	reg8 *	hex
*pointer_addr	0x39 '9'	0x1FFFB3D8 (All)	reg8	hex
state_bit	4 '\004'	0x1FFF819B (All)	uint8	Default

(b)

Watch 1

Name	Value	Address	Type	Radix
flag_2	0 '\000'	0x1FFF8196 (All)	uint8	Default
prs_number	0x33D8	0x1FFF8198 (All)	uint16	hex
low_high_addr	1 '\001'	0x1FFF819A (All)	uint8	Default
low_addr	0x1FFF8000	0x1FFF8114 (All)	uint32	hex
high_addr	0x20000000	0x1FFF8118 (All)	uint32	hex
target_addr	0x1FFFB3D8	0x1FFF819C (All)	uint32	hex
pointer_addr	0x1FFFB3D8 (All)	0x1FFF81A0 (All)	reg8 *	hex
*pointer_addr	0x29 ')'	0x1FFFB3D8 (All)	reg8	hex
state_bit	4 '\004'	0x1FFF819B (All)	uint8	Default

Fonte: Autoria própria, 2015.

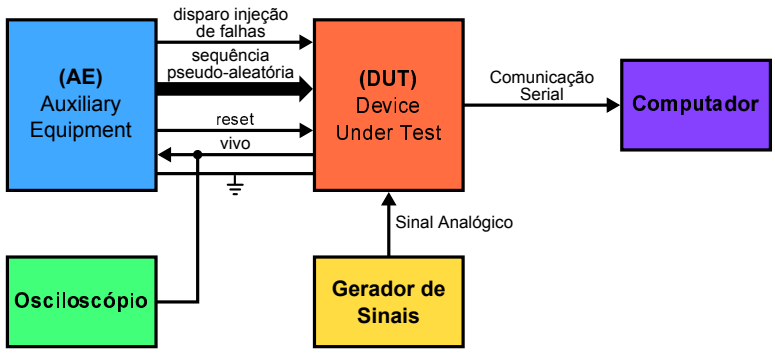
coerência, o *setup* foi considerado validado no que concerne ao bloco de injeção de falhas na SRAM.

A validação final do *setup*, com o objetivo de atender aos critérios das alíneas “a” e “b” da seção 5.3 foi realizada através de um teste funcional do *setup*. Porém nesse caso se alterou o código do bloco de injeção de falhas para não inserir falhas, o que foi feito anulando o código da linha 22 da figura 5.4. Esperava-se com esse teste que o *setup* e o SAD funcionassem sem o registro de erro. O teste foi executado nesse formato com a injeção de aproximadamente 100.000 falhas, e não foram observados erros, o que foi possível considerar o *setup* de injeção de falhas na SRAM como validado.

5.4 Procedimentos de teste

Os procedimentos de teste para a injeção de falhas nos periféricos e na SRAM são semelhantes, e portanto são abordados conjuntamente nesta seção. Características específicas de cada um são indicadas quando necessário. O *setup* de teste é mostrado na figura 5.12. Observa-se que ele é semelhante ao utilizado no estudo da seção 4, mas possui entre o DUT e o AE a adição das conexões para a injeção de falhas, e também a adição de um osciloscópio para a monitoração do sinal de “vivo”. Essa monitoração em tempo real fornece mais um indicativo do estado de operação do SAD em análise.

Figura 5.12 - Setup de teste para a injeção de falhas em software.



Fonte: Autoria própria, 2015.

Antes de iniciar o teste deve-se realizar dois procedimentos importantes: reiniciar os parâmetros cont_reset, cont_f_total e cont_f_parcial do bloco de registro de falhas, e certificar-se do correto funcionamento da conexão entre o DUT e AE para transmissão da sequência pseudo-aleatória gerada pelo GSPA. O primeiro procedimento é realizado apagando toda a

memória EEPROM, o que foi feito programando o dispositivo com o código mostrado na figura 5.13. A certificação da conexão entre o DUT e o AE foi realizada da seguinte forma: com o conhecimento da sequência pseudo-aleatória gerada no AE, alterou-se o DUT para registrar no relatório gerado ao computador cada número pseudo-aleatório recebido do AE, permitindo através de comparação avaliar a coerência da conexão.

Figura 5.13 - Código para apagar a memória EEPROM do PSoC 5LP.

```
1     #include <project.h>
2     int main()
3     {
4         EEPROM_1_Start();
5         EEPROM_1_EraseSector(0);
6         EEPROM_1_EraseSector(1);
7     }
```

Fonte: Autoria própria, 2015.

Durante a realização do teste verificou-se a necessidade de acompanhá-lo em tempo integral, para eventualmente gerar manualmente e através do botão presente no AE o *reset* do SAD. Essa necessidade torna-se evidente na abordagem da seção 5.5, mas deve-se basicamente à ocorrência de SEFIs. Como exemplo é possível citar a mudança da velocidade de escrita da interface serial RS-232, resultante de uma falha injetada, o que inviabiliza a geração do relatório de estado do sistema, essencial para a análise dos resultados. Assim torna-se necessário reiniciar o DUT manualmente para a continuidade do teste. Obviamente que essa limitação do *setup* deve-se à característica do AE de operar apenas como *watchdog*, sem verificar a consistência dos dados reportados.

O teste dos periféricos foi realizado com a injeção total de 23.815 falhas, o que corresponde a aproximadamente 3 vezes o conjunto de registradores de controle alvos desse estudo. Com isso espera-se atingir pelo menos 1 bit da maior parte desses registradores. O teste da SRAM foi realizado com a injeção total de 130.236 falhas, o que corresponde a aproximadamente duas falhas por endereço da SRAM, considerando que esta é composta por 2 bancos de 32 KB. O percentual de ocupação da SRAM informado pelo compilador no projeto do DUT do teste da SRAM é de 36,8%.

5.5 Resultados, discussões e considerações

Nesta seção os resultados e discussões dos testes de injeção de falhas nos periféricos e na SRAM são abordados separadamente. Ao final são realizadas considerações sobre a auto

injeção de falhas e auto monitoramento, baseadas na experiência desse estudo.

5.5.1 Resultados e discussões sobre o teste de injeção de falhas nos periféricos

A tabela 5.4 mostra um resumo dos resultados da injeção de falhas nos periféricos. Foram injetadas 23.815 falhas e detectados 848 erros, o que corresponde a 3,56% das falhas detectadas na forma de erros. No teste para validação do *setup* descrito na seção 5.3.1, foram injetadas 25.023 pseudo-falhas e detectados 25 erros, o que corresponde a 0,1% das pseudo-falhas detectadas na forma de erros. Considerando que o número de falhas e pseudo-falhas injetadas é próximo, tem-se que esse percentual do teste de validação é significativamente menor do que o percentual do teste de injeção de falhas, o que torna possível considerar a influência da limitação do *setup* nos resultados como pequena. Além disso, é positiva a não observância de erros detectados pelos votadores do sistema TMR no teste de injeção de pseudo-falhas, pois permite que esse tipo de erro detectado no teste com falhas seja considerado como resultado de um *bit-flip* injetado.

Tabela 5.4 - Resumo dos resultados da injeção de falhas nos periféricos.

Total de falhas injetadas		**23815**	**100%**
Total de erros detectados		**848**	**3,56%**
Total de erros detectados pelos votadores		528	2,22%
Com zero falhas injetadas*	Não recuperação do sistema	20	-
Com uma falha injetada	Não recuperação do sistema	236	-
Com mais de uma falha injetada	-	272	-
Total de SEFIs (desencadeadores de resets)		320	1,34%
Desconfiguração da interface serial	Reset manual	33	-
Perda da funcionalidade do injetor de falhas	Reset manual	19	-
Detecção permanente de erro pelos votadores	Reset manual	11	-
Atuação do watchdog do AE	Reset automático	257	-

* Atribuídos à não recuperação do sistema à uma falha anterior

Fonte: Autoria própria, 2015.

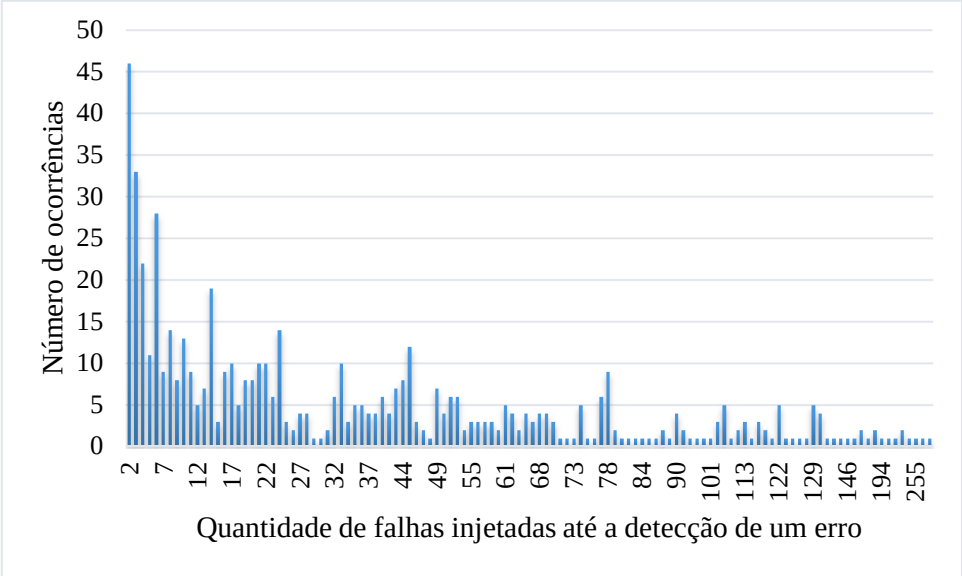
Os erros detectados no teste de injeção de falhas nos periféricos foram classificados em dois grandes grupos: erros detectados pelos votadores e erros devido à SEFIs, sendo estes últimos desencadeadores de *reset*. O primeiro grupo refere-se aos erros detectados pelo votador principal e votador do ADC SAR com base nos códigos implementados para encontrar *bit-flips*, os quais foram detalhados nas figuras 4.7 e 4.9. No total 528 erros foram detectados pelos votadores, dos quais 20 erros ocorreram com 0 falhas injetadas e 236 com uma falha injetada.

A ocorrência de erros sem falhas injetadas ou mesmo a ocorrência elevada de erros (236 no universo de 528) com apenas uma falha injetada são atribuídas à não recuperação do sistema à uma falha anterior, mesmo após a correção do *bit-flip*. No grupo de erros devido à SEFIs foram identificados quatro erros. A desconfiguração da interface serial se manifestou através de escritas incoerentes no relatório do teste, possivelmente geradas pela mudança da velocidade da interface, decorrentes de falhas injetadas em periféricos como PLLs e osciladores. O erro de perda da funcionalidade do injetor de falhas refere-se à situação em que o sistema continuava rodando porém sem entrar no bloco de injeção de falhas, o que acredita-se ocorreu pela incapacidade de o sistema entrar nas interrupções. O erro descrito como detecção permanente de erro pelos votadores está relacionado à não recuperação do sistema à falhas anteriores, como já mencionado, o que gerou a necessidade de um *reset* manual no DUT através do botão específico para a finalidade no AE. Este *reset* manual também foi necessário nos dois erros já mencionados, desconfiguração da serial e perda da funcionalidade do injetor de falhas. O quarto e último erro identificado como decorrente de um SEFI pode ser relacionado a diversos fatores, que geraram o travamento do sistema e a consequência atuação do *watchdog* do AE. Além dessa discussão referente aos tipos de erros identificados, chama a atenção na tabela 5.4 o percentual de apenas 3,56% do total de falhas injetadas detectadas na forma de erros. Da mesma forma que na seção 4.6, em que foram discutidas justificativas para a não observância de erros no estudo de injeção de falhas com irradiação, esse dado mostra que o mascaramento de falhas nos circuitos ou em nível de sistema é elevado. Ou seja, muitas falhas injetadas não resultam na ocorrência de erros.

A figura 5.14 mostra a quantidade de falhas injetadas até a detecção de um erro como uma função do número de ocorrências com que essas quantidades foram observadas. Nessa análise optou-se por desconsiderar a ocorrência de erros sem falhas injetadas e a ocorrência de erros com apenas uma falha injetada devido a serem atribuídas em sua maior parte à não recuperação do sistema à uma falha anterior, conforme já mencionado. E também porque a consideração da ocorrência de erros com apenas uma falha injetada resultaria na perda de detalhes da curva, devido seu número de ocorrências elevado frente aos demais casos. Analisando a figura 5.14 ainda chama atenção o número de ocorrências elevado para os casos com poucas falhas injetadas até a detecção de um erro, como por exemplo, até 6 falhas injetadas. Atribui-se esse comportamento também à não recuperação do sistema às falhas anteriores, da mesma forma que nos casos com zero e uma falhas injetadas. A maneira como isso ocorre não é abordada em detalhes aqui, pois seria necessário um diagrama de tempos, o qual é inexistente devido aos eventos ocorrerem em tempo de execução e de maneira aleatória. A figura 5.14

também mostra que entre 7 e 70 falhas injetadas até a detecção de um erro o número de ocorrências é geralmente maior que 1, evidenciando que quantidades consideráveis de mascaramentos de falhas podem ocorrer até que um erro seja detectado. Ainda ocorreram casos em que foi necessária a injeção de mais de 70 falhas para a detecção de um erro, embora com menor número de ocorrências, geralmente próximo a 1. Esse comportamento mostra que a probabilidade de mascaramentos sem detecção de nenhum erro reduz à medida que o número de falhas injetadas aumenta significativamente.

Figura 5.14 - Quantidade de falhas injetadas até a detecção de um erro como uma função do número de ocorrências com que essas foram observadas.

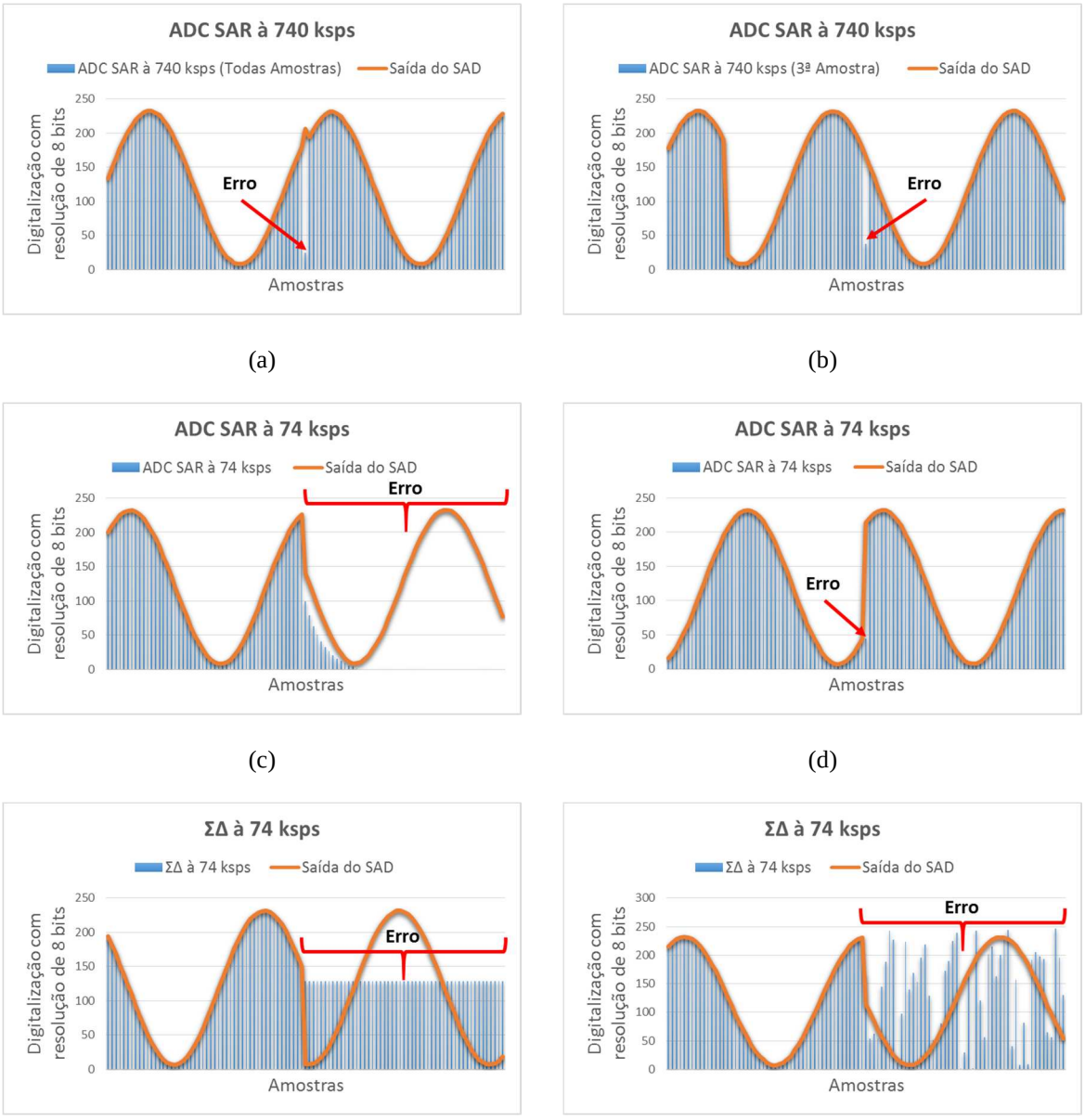


Fonte: Autoria própria, 2015.

Essa discussão da figura 5.14 possibilita a identificação de duas ações necessárias no teste de injeção de falhas nos periféricos, principalmente com o objetivo de melhorar a qualidade da análise da quantidade de falhas injetadas até a detecção de um erro. Essas ações podem ser implementadas futuramente. A primeira consiste em implementar um *reset* do DUT após qualquer detecção de erro, em especial após a detecção de um erro pelos votadores. Essa alteração elimina a incerteza sobre a não recuperação do sistema a uma falha anterior à falha analisada, conforme já mencionado como justificativa para a ocorrência elevada de erros com poucas falhas injetadas. A segunda ação para melhorar a qualidade dos resultados é repetir o teste com a injeção de um número maior de falhas, de maneira a ter mais dados que evidenciem a curva da figura 5.14 e o comportamento do SAD em análise.

A figura 5.15 mostra o comportamento dos ADCs em seis casos em que os votadores

Figura 5.15 - Comportamento dos ADCs em seis casos em que os votadores detectaram erros. (a) ADC SAR à 740 ksps com erro em todas as amostras de um ciclo de votação. (b) ADC SAR à 740 ksps com erro apenas na terceira amostra. (c) ADC SAR à 74 ksps sem recuperação após o primeiro erro. (d) ADC SAR à 74 ksps com recuperação após erro. (e) ADC Sigma-Delta com erro permanente. (f) ADC Sigma-Delta com saída instável.



Fonte: Autoria própria, 2015.

detectaram erros. Como foram realizadas apenas injeções de falhas simples, em todos os casos apenas um ADC falhou e o SAD foi tolerante a essa falha. Assim, por simplicidade optou-se por mostrar na figura 5.15 apenas os dados do ADC que apresentou erro e a saída do SAD. As descontinuidades das curvas estão relacionadas com o momento da injeção da falha, dado que a execução do bloco de injeção de falhas nos periféricos necessita de um tempo

consideravelmente maior do que o tempo de um ciclo de votação, o que atrasa as conversões seguintes a uma injeção de falha. Na figura 5.15a o ADC SAR operando à 740 ksps apresentou erro imediatamente após a injeção da falha, mas no entanto se recuperou a partir do próximo ciclo de votação. É importante notar que todas as nove amostras geradas por esse ADC apresentaram o mesmo erro no ciclo de votação em que se detectou erro. A figura 5.15b mostra outro erro gerado também pelo ADC SAR à 740 ksps. Nesse caso o erro se manifestou alguns ciclos de votação após a injeção da falha e o ADC também se recuperou a partir do próximo ciclo de votação. É importante notar que nesse caso apenas a terceira amostra entre as nove geradas pelo conversor no ciclo de votação apresentou erro, mostrando a utilidade da aplicação de redundância temporal com ADCs.

A figura 5.15c mostra erros apresentados pelo ADC SAR à 74 ksps. Verifica-se que, neste caso esse conversor manifestou o primeiro erro imediatamente após a injeção da falha e não se recuperou nos ciclos de votação seguintes. É interessante notar que nesse caso a saída do conversor tendeu a zero e assim se manteve ao atingir esse valor. Na figura 5.15d é mostrado outro caso em que o ADC SAR à 74 ksps também apresentou erro. O erro se manifestou imediatamente após a injeção da falha, mas o ADC se recuperou a partir do próximo ciclo de votação. A figura 5.15e mostra erros apresentados pelo ADC Sigma-Delta. Observa-se que o primeiro erro se manifestou logo após a injeção da falha e que o conversor não se recuperou nos ciclos de votação seguintes, mantendo a saída com o mesmo erro. Na figura 5.15f é mostrado outro erro apresentado pelo ADC Sigma-Delta. Da mesma forma que no caso anterior discutido, o ADC apresentou erro imediatamente após a injeção da falha e não se recuperou nos ciclos de votação seguintes. É interessante notar que nesse caso o conversor tornou-se instável com a falha injetada.

5.5.2 Resultados e discussões sobre o teste de injeção de falhas na SRAM

A tabela 5.5 mostra um resumo dos resultados da injeção de falhas na SRAM. Foram injetadas 130.236 falhas e detectados 46 erros, o que corresponde a 0,03% das falhas detectadas na forma de erros. Esse percentual indica que o mascaramento de falhas nessa memória é elevado, ou seja, que de maneira geral muitas falhas precisam ser injetadas para que um erro seja observado. Comparando esse resultado (0,03%) com o resultado equivalente do teste de injeção de falhas nos periféricos (3,56%) verifica-se que os dados da memória SRAM são menos importantes no nível de aplicação do que os registradores de controle dos periféricos, o que pode em parte ser justificado pela utilização de apenas 36,8% da capacidade total dessa

memória. Esse resultado também permite afirmar que no teste de injeção de falhas com radiação da seção 4 é possível que ocorreram *bit-flips* na SRAM, porém o elevado mascaramento que o sistema impõe a essa memória fez com que essas falhas não fossem detectadas.

Tabela 5.5 - Resumo dos resultados da injeção de falhas na SRAM.

Total de falhas injetadas	**130236**	**100%**
Total de erros detectados	**46**	**0,03%**
Total de erros detectados pelos votadores	28	60,87% dos erros
Detecção incoerente de erro	3	-
Erro no ADC SAR à 740 ksps	7	-
Erro no ADC SAR à 74 ksps	2	-
Erro no ADC Sigma-Delta	3	-
Erro no votador do ADC SAR	11	-
Erro no votador do ADC SAR e no votador principal	2	-
Total de SEFIs	18	-
Resets pelo watchdog do AE ou próprio DUT	6	13,04% dos erros
Incoerência do relatório de estado	10	21,74% dos erros
Perda do status periódico do relatório de estado	1	2,17% dos erros
Perda da funcionalidade do bloco de registro de falhas	1	2,17% dos erros

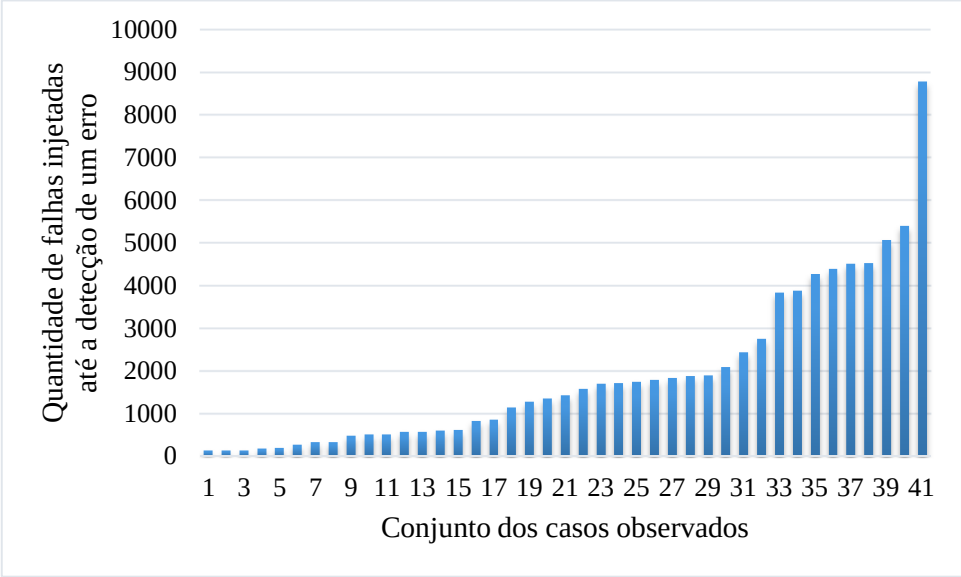
Fonte: Autoria própria, 2015.

Os erros detectados no teste de injeção de falhas na SRAM foram classificados em dois grandes grupos: erros detectados pelos votadores e erros devido à SEFIs. Os primeiros referem-se aos erros detectados pelo votador principal e votador do ADC SAR com base nos códigos implementados para encontrar *bit-flips*, e os erros devido à SEFIs foram classificados em quatro subgrupos detalhados a seguir. Os erros observados na forma de *reset* do DUT são gerados pelo próprio DUT ou gerados no AE, neste caso fruto de um travamento do SAD detectado pelo *watchdog* do AE. Os erros de incoerência do relatório de estado foram observados durante a realização do teste através do acompanhamento das mensagens periódica de status enviadas pelo SAD. Exemplo de parâmetros incoerentes detectados são o contador de falhas detectadas pelos votadores, relógio, data e endereço alvo para injeção da falha. O erro de perda do status periódico do relatório de estado consiste no não envio pelo SAD da mensagem periódica mencionada no erro anterior. E o erro de perda da funcionalidade do bloco de registro de falhas refere-se a não contagem pelo DUT da quantidade de falhas injetadas. Dentre os erros mencionados ocorreram com mais frequência os erros detectados pelos votadores (60,87% do total de erros), incoerência do relatório de estado (21,74%) e *resets* do DUT (13,04%). Dentre os erros detectados pelos votadores aparecem: detecções incoerentes de erro, provavelmente

decorrentes de *bit-flips* na variável indicativa de erro detectado; erros nos valores gerados pelos conversores de dados, provavelmente resultados de *bit-flips* no buffer que armazena os dados convertidos; e erros no valor gerado pelos votadores. É interessante notar que a quantidade de erros no votador do ADC SAR (11) é expressiva se comparada ao total de erros detectados pelos votadores (28), o que indica uma considerável suscetibilidade desse votador à falhas.

A figura 5.16 mostra a quantidade de falhas injetadas até a detecção de um erro no teste de injeção de falhas na SRAM. Ela compreende todos os casos de detecção de erro pelos votadores, *reset* do SAD e incoerência do relatório de estado em que foi possível determinar a quantidade de falhas injetadas até o *reset*. Para facilitar a análise da distribuição das quantidades, os casos foram ordenados de maneira crescente pelas quantidades. Observa-se que em aproximadamente três quartos dos casos erros foram detectados até a injeção de 2.000 falhas, e que dentro desta faixa a quantidade de falhas injetadas para a observação de um erro varia bastante. No restante um quarto dos casos foi necessário injetar mais de 2.000 falhas, em que o máximo observado foram aproximadamente 9.000 injeções de falhas para a observação de um erro.

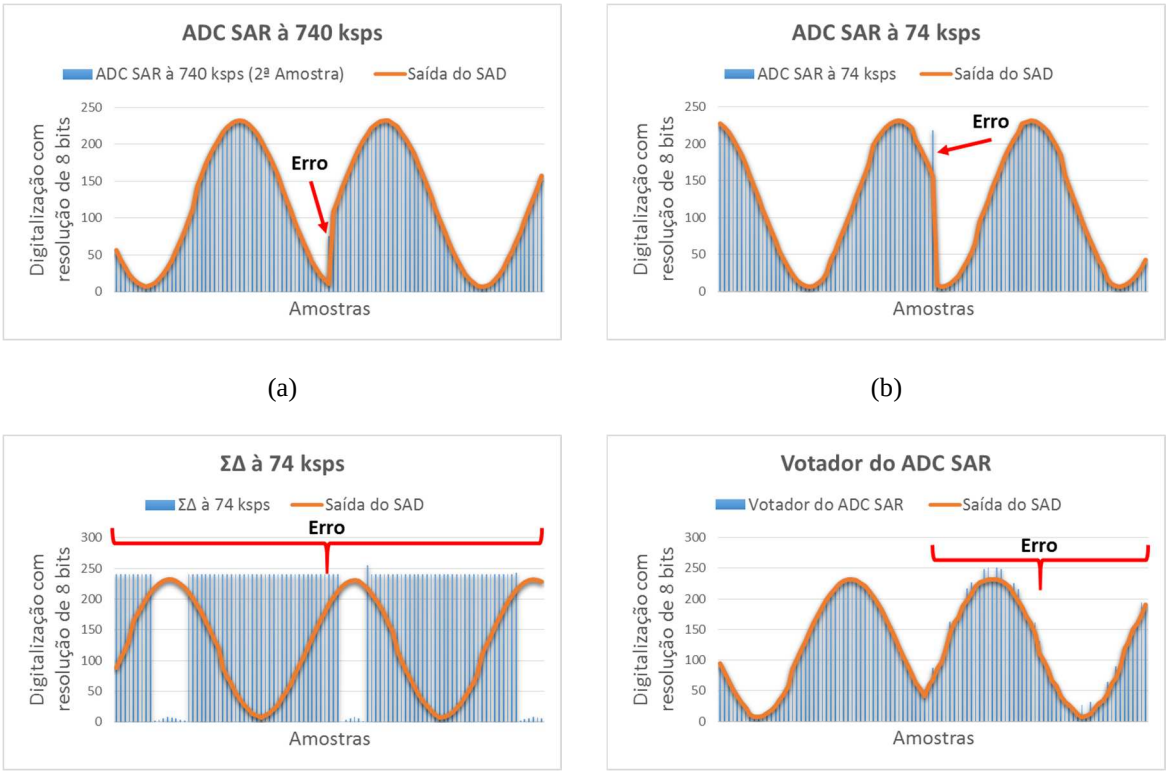
Figura 5.16 - Quantidade de falhas injetadas até a detecção de um erro no teste de injeção de falhas na SRAM.



Fonte: Autoria própria, 2015.

A figura 5.17 mostra o comportamento dos ADCs e do votador do SAR em casos em que os votadores detectaram erros. Novamente por simplicidade optou-se por mostrar apenas os dados dos ADCs ou votador em análise e a saída do SAD. E também, da mesma forma que na

Figura 5.17 - Comportamento dos ADCs e do votador do SAR em casos em que os votadores detectaram erros. (a) Erro no ADC SAR à 740 ksps. (b) Erro no ADC SAR à 74 ksps. (c) Erros no ADC Sigma-Delta. (d) Erros no votador do ADC SAR.



(a) (b)

(c) (d)

Fonte: Autoria própria, 2015.

injeção de falhas nos periféricos, as descontinuidades das curvas estão relacionadas com o momento da injeção da falha. A figura 5.17a mostra um erro em um ciclo de votação na segunda amostra do ADC SAR à 740 ksps, em que observa-se que o erro ocorreu no momento da injeção da falha. Na figura 5.17b é mostrado um erro que ocorreu em um ciclo de votação no ADC SAR à 74 ksps, em que novamente observa-se que o erro ocorreu no momento da injeção da falha. Nestes dois casos acredita-se que a falha foi injetada no endereço em que o dado gerado pelo conversor estava armazenado na SRAM, modificando diretamente seu valor. A figura 5.17c mostra a ocorrência de erros no ADC Sigma-Delta. Observa-se que nesse caso os erros foram reportados em um momento posterior a injeção de uma falha, porque no “valor correto” não aparece nenhuma descontinuidade na curva. Diversas são as possíveis causas desse erro, como por exemplo o funcionamento inadequado do buffer que guarda os dados convertidos, dos votadores ou do bloco de registro de estado. Na figura 5.17d são mostrados erros que ocorreram no votador do ADC SAR, os quais começaram após a injeção de uma falha e permaneceram pelos ciclos de votação seguintes. Esse comportamento sugere uma modificação

no votador que passou a gerar um pequeno erro em sua saída. É importante registrar que nenhum erro no votar principal foi observado.

5.5.3 Considerações sobre a auto injeção de falhas e auto monitoramento

O sistema estudado nessa seção 5 foi o SAD irradiado e descrito na seção 4. Assim na injeção de falhas por *software* foi necessário lidar com as características desse SAD. As implementações para a injeção de falhas por *software* partiram das seguintes premissas: aproveitar ao máximo a implementação descrita na seção 4; intervir minimamente nessa implementação para possibilitar a comparação dos resultados com os resultados da irradiação; e viabilizar com pouco tempo de projeto os testes. Nesse contexto foi adotado o conceito de injeção de falhas denominado nesse trabalho de auto injetor, em que o próprio DUT em estudo realiza a injeção de falhas. Além disso, a implementação do SAD descrita na seção 4 também utilizou o conceito de auto monitoração, em que o próprio DUT gera o relatório de seu estado para que a análise dos resultados seja feita.

A primeira dificuldade encontrada com a auto injeção de falhas foi com relação à reinicialização do GSPA. Verificou-se que um gerador de sequência pseudo-aleatória embarcado no DUT impactava na limitação de ser reinicializado a cada *reset* do DUT, o qual poderia acontecer em função de um erro provocado por uma falha injetada. O resultado disso é que a cada *reset* do DUT os alvos para a injeção da falha se repetiriam, tornando o teste viciado. A alternativa adotada foi implementar o GSPA no AE, o qual não é reiniciado durante todo o período de execução do teste. Essa solução demonstra uma primeira limitação em utilizar a injeção de falhas totalmente embarcada no DUT. Outras limitações da auto injeção foram verificadas no decorrer do teste desse estudo, e estão relacionadas com perda da funcionalidade do injetor de falhas. Como o disparo da injeção de falhas baseou-se em interrupções de *software*, no teste de injeção de falhas nos periféricos foi observado que eventualmente as interrupções não ocorriam, provavelmente devido a um erro causado por uma falha injetada. O resultado disso foi a perda da funcionalidade do injetor de falhas, o que foi contornado nesse estudo através do acompanhamento *on-line* do teste, reiniciando o DUT a cada evento desses. Já no decorrer do teste de injeção de falhas na SRAM, foi observado que, diferentemente do planejado, falhas estavam sendo injetadas em endereços não correspondentes aos da SRAM, criando assim um desvio relevante aos resultados do teste. Verificou-se que esse evento ocorria devido a uma falha injetada causar um erro em variáveis utilizadas pelo injeto de falhas que estavam armazenadas na própria SRAM, como por exemplo alterando os endereços bases dos

bancos da memória utilizados para definir o endereço alvo. A solução adotada para essa limitação foi testar o endereço alvo antes de injetar a falha se ele não correspondia ao endereço em que estavam armazenadas variáveis importantes do injetor de falhas.

Além da auto injeção também foram verificadas limitações do auto monitoramento, conceito empregado no DUT dos estudos em questão referente à geração de dados para a análise dos resultados. No teste de injeção de falhas nos periféricos foi observado que eventualmente ocorria a desconfiguração da interface serial utilizada pelo DUT para envio do relatório de seu estado. Acredita-se que essa desconfiguração tenha ocorrido por um erro decorrente de uma falha injetada em registradores de controle dos PLLs e osciladores. Mas o importante de se observar nesse caso é que uma falha inserida resulta na perda da monitoração do sistema em análise, o que compromete a análise dos resultados. O contorno usado a essa limitação também foi o acompanhamento *on-line* do teste e o reinício do DUT a cada evento desses. Outra limitação do auto monitoramento observada foi a dificuldade em registrar eventos, uma vez que uma falha inserida pode resultar na reinicialização do DUT e consequente perda de informação útil. Por exemplo, nos testes de injeção de falhas por *software* desse trabalho, foi necessário armazenar na memória EEPROM (não-volátil) do dispositivo os contadores de *reset*, falhas inseridas até a observação de um erro e total de falhas inseridas, para recuperar a informação no próximo *reset*.

As limitações da auto injeção e do auto monitoramento mencionadas criam dificuldades e variáveis adicionais ao teste, e podem comprometer os resultados. Por esse motivo sugere-se avaliar os impactos e viabilidade da aplicação desses conceitos em testes de injeção de falhas. Uma alternativa a esse conceito consiste em utilizar um injetor de falhas externo ao DUT, o qual também pode ser usado para fazer a monitoração do sistema em análise.

6 CONSIDERAÇÕES FINAIS

O estudo dos efeitos da radiação espacial nos circuitos eletrônicos iniciou na década de 1960 e desde então esses efeitos vêm apresentando comportamentos diferentes à medida que a eletrônica evolui. Atualmente os efeitos singulares, em especial os *soft errors*, são a maior preocupação para a garantia da confiabilidade e disponibilidade das tecnologias CMOS, dado a redução das dimensões dos transistores, o aumento da velocidade de operação dos sistemas e a redução da tensão de alimentação dos circuitos integrados. TMR e diversidade são técnicas em nível de sistema de proteção contra os *soft errors*, usadas geralmente em aplicações críticas. Nesse contexto, esse trabalho adotou um esquema baseado nessas duas técnicas para a implementação de um sistema de aquisição de dados (SAD) analógico-digital, com o objetivo de avaliar o comportamento dos conversores de dados frente a SEU, SET e SEFI, e avaliar a eficácia de um sistema baseado em TMR e diversidade espacial-temporal contra esses efeitos da radiação. A implementação foi feita no SoC PSoC 5LP, da Cypress Semiconductor, fabricado em tecnologia CMOS de 130 nm.

A utilização de TMR e diversidade ao longo desse trabalho propiciou esclarecer alguns aspectos importantes dessas técnicas, os quais não são inéditos, mas que são ressaltados com pouca frequência na literatura do assunto. São eles:

a) a técnica TMR clássica é adequada apenas para tolerar falhas aleatórias que presumidamente ocorrem de maneira independente nas cópias redundantes;
b) a técnica de diversidade é especialmente útil para lidar com falhas de modo-comum, que é quando múltiplas cópias de um sistema redundante sofrem falhas simultaneamente, geralmente devido a uma única causa. Essa característica da diversidade é devido aos diferentes níveis de resiliência de cada *hardware*, *software* e tempo, que fazem com que erros devido a pontos em comum sejam sistematicamente evitados;
c) a proteção com TMR e diversidade implica em um aumento da complexidade de projeto do sistema. Exemplos de fatores que contribuem para isso são: a necessidade de um sincronizador para controlar as cópias e os votadores, e a necessidade de atenção com os parâmetros analógicos quando se replica e conecta circuitos analógicos. O sincronizador e os votadores são elementos-chave em TMR e diversidade;
d) devido ao seu custo elevado, TMR e diversidade são geralmente mais apropriadas para aplicações críticas, que requerem elevada confiabilidade e disponibilidade.

Como exemplo, em naves espaciais, aviões comerciais e instalações nucleares.

Nesse trabalho a implementação do esquema baseado em TMR e diversidade foi feita com um SoC programável de sinais mistos. Esses dispositivos são uma alternativa para reduzir os custos da replicação decorrentes da aplicação dessas técnicas de proteção, principalmente em comparação aos casos em que TMR e diversidade são aplicadas em um nível de sistema, em que cada elemento usado para computação múltipla consiste em uma placa. Os recentes e significativos avanços dos SoCs programáveis de sinais mistos oferecem a possibilidade de uso de FPGAs, microprocessadores, blocos analógicos programáveis e diferentes arquiteturas de conversores de dados embarcados em um mesmo circuito integrado. Assim, os recursos que não são originalmente usados no projeto podem ser empregados para adicionar um grau extra de confiabilidade com um custo de área virtualmente zero, embora às custas de um aumento no consumo de potência e complexidade do sistema. Com o contínuo avanço desses dispositivos, a implementação de sistemas de controle e instrumentação confiáveis baseados em TMR e diversidade vai se tornar mais viável, além de se beneficiar da rápida prototipação oferecida pelos SoCs programáveis.

O esquema específico adotado para a implementação do SAD analógico-digital dos estudos desse trabalho, baseado em TMR e diversidade espacial-temporal, foi quantitativamente avaliado através da investigação teórica baseada em análise combinatória. Essa investigação considerou diretamente a ocorrência de *bit-flips* na saída dos conversores, e portanto avaliou o ganho na tolerância de falhas apenas devido à diversidade temporal, uma vez que o ganho decorrente da diversidade espacial resulta dos diferentes níveis de resiliência de cada *hardware*. A diversidade temporal é implementada pela sobreamostragem do ADC SAR à 740 ksps e pelo seu votador. Os resultados mostraram que a adição de diversidade temporal gera em comparação ao TMR clássico um ganho significativo na tolerância de falhas duplas e múltiplas. O preço a ser pago por esse ganho é o aumento do atraso do circuito, que nos sistemas de aquisição de dados como o dos estudos desse trabalho se traduz na redução da taxa de amostragem. A necessidade de análise do ponto de equilíbrio é então evidente, e deve ser feita em função dos requisitos da aplicação e dos recursos disponíveis.

A injeção de falhas por irradiação de nêutrons no SAD foi abordada na seção 4. A não observância de erros no período testado impediu cumprir os objetivos propostos de através da injeção de falhas por irradiação observar o comportamento dos conversores de dados e avaliar a eficácia do sistema baseado em TMR e diversidade espacial-temporal. Essa situação permite duas observações principais. Primeira, de que é importante ter um fluxo de nêutrons elevado para observar os efeitos da radiação, e em um passo adiante para gerar estatísticas,

principalmente quando a seção de choque do circuito em análise é baixa. E segunda, de que a probabilidade de ocorrerem mascaramentos de SETs nos circuitos combinacionais e analógicos é elevada, o que contribui significativamente para reduzir a sensibilidade desses circuitos, ou em outras palavras, reduzir sua seção de choque.

A seção 5 abordou a injeção de falhas no SAD por *software* e em tempo de execução. Foram injetadas separadamente falhas nos registradores de controle dos periféricos e na memória SRAM. Os resultados mostraram que apenas um baixo percentual das falhas injetadas é detectado na forma de erros, devido aos mascaramentos de *soft errors* que ocorrem nos circuitos combinacionais e analógicos, na memória e ao nível de sistema. Esses resultados confirmam a hipótese da seção 4 de que no teste de injeção de falhas por irradiação os mascaramentos foram determinantes para a não observância de erros, considerando também que, possivelmente, poucas falhas foram efetivamente injetadas, devido ao baixo fluxo de nêutrons do experimento. Comparando os resultados das injeções de falhas nos periféricos com os das injeções de falhas na SRAM verificou-se que os registradores de controle dos periféricos são mais importantes no nível de aplicação do que os dados da memória SRAM, o que foi contribuído pela baixa utilização da capacidade total dessa memória.

A análise da quantidade de falhas que precisam ser injetadas até a observação de um erro mostrou que essa quantidade é aleatória, e que em alguns casos é necessária a injeção de muitas falhas até que um erro seja verificado. A análise dos erros detectados pelos votadores mostrou que a injeção de falhas nos periféricos gera erros nos conversores de dados e que a injeção de falhas na SRAM gera tanto erros nos dados dos conversores quanto nos votadores, fato observado no votador do ADC SAR. Um primeiro erro nos ADCs devido à falhas nos periféricos pode ocorrer no ciclo de votação da inserção da falha ou em um ciclo posterior, e pode ocorrer de erros se manifestarem também nos ciclos de votação posteriores. Ainda, esses erros se configuram pelo travamento da saída do conversor ou pela sua instabilidade. Os erros nos dados dos votadores devido à falhas na SRAM geralmente ocorrem apenas em um ciclo de votação. A falha exemplificada do votador do ADC SAR se manifestou semelhantemente a um ruído por diversos ciclos de votação. Por fim, fruto da experiência adquirida na injeção de falhas por *software*, ainda na seção 5 foram realizadas considerações sobre a auto injeção de falhas e auto monitoramento, sugerindo que a utilização desses conceitos pode trazer diversas limitações e complicadores aos testes.

Algumas partes dessa dissertação foram publicadas e outras atualmente encontram-se em fase de avaliação pelo meio de divulgação científica. As referências desses trabalhos encontram-se no apêndice I. Como sugestões para trabalhos futuros são mencionados:

a) testar o SAD implementado nesse trabalho em instalações de teste com maior fluxo de partículas, conforme discutido na seção 4;
b) testar o SAD proposto com base em TMR e diversidade em dispositivos fabricados em tecnologias CMOS mais recentes que 130 nm. Embora no estudo com irradiação de nêutrons desse trabalho não foram observados erros, em tecnologias mais recentes é possível que eles sejam observados, dada a discussão apresentada nesse trabalho sobre o aumento da sensibilidade com a redução das dimensões dos transistores, e portanto o esquema proposto com base em TMR e diversidade pode ser importante no futuro;
c) testar o SAD implementado novamente com a injeção de falhas por *software* nos registradores de controle dos periféricos, reiniciando o DUT a cada detecção de uma falha pelos votadores. Essa medida elimina a limitação verificada nesse trabalho de não recuperação do sistema à falhas anteriores, refinando os resultados obtidos;
d) analisar os testes de injeção de falhas por *software* nos registradores de controle dos periféricos e na SRAM com foco quantitativo nas detecções de erro feitas pelos votadores. Essa análise permite um comparativo detalhado entre o comportamento das diferentes arquiteturas de conversores de dados e votadores.

REFERÊNCIAS

ALEXANDER, D. R. Design issues for radiation tolerant microcircuits for space. **IEEE NSREC Short Course Notes**, Indian Wells, p. V 1-54, July 1996.

ALLENSPACH, M. et al. Evaluation of SEGR threshold in power MOSFETs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 41, n. 6, p. 2160-2166, Dec. 1994.

ANDREWS, J. L. et al. Single event error immune CMOS RAM. **IEEE Transactions on Nuclear Science**, [S. l.], v. 29, n. 6, p. 2040-2043, Dec. 1982.

AVIZIENIS, A.; KELLY, J. P. J. Fault tolerance by design diversity: concepts and experiments. **Computer**, [S. l.], v. 17, n. 8, p. 67-80, Aug. 1984.

BALEN, T. R. **Efeitos da radiação em dispositivos analógicos programáveis (FPAAs) e técnicas de proteção**. 2010. 205 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.

BARTH, J. L.; DYER, C. S.; STASSINOPOULOS, E. G. Space, atmospheric, and terrestrial radiation environments. **IEEE Transactions on Nuclear Science**, [S. l.], v. 50, n. 3, p. 466-482, June 2003.

BARTH, J. Modeling space radiation environments. **IEEE NSREC Short Course Notes**, Snowmass Village, p. I 1-83, July 1997.

BAUMANN, R. C. Soft errors in advanced semiconductor devices-part I: the three radiation sources. **IEEE Transactions on Device and Materials Reliability**, [S. l.], v. 1, n. 1, p. 17-22, Mar. 2001.

BAZE, M. P. et al. SEU hardening techniques for retargetable, scalable, sub-micron digital circuits and libraries. In: SINGLE EVENT EFFECTS SYMPOSIUM, 2002, Manhattan Beach. **Proceedings…** [S. l.]: klabs. Org, 2002. p. 1-21.

BINDER, D.; SMITH, E. C.; HOLMAN, A. B. Satellite anomalies from galactic cosmic rays. **IEEE Transactions on Nuclear Science**, [S. l.], v. 22, n. 6, p. 2675-2680, Dec. 1975.

BORGES, de M. et al. Increasing reliability of programmable mixed-signal systems by applying design diversity redundancy. In: IEEE EUROPEAN TEST SYMPOSIUM, 15., 2010, Prague. **Proceedings…** Piscataway: IEEE, 2010. p. 261-261.

BOUDENOT, J. C. Radiation space environment. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation effects on embedded systems**. Dordrecht: Springer, 2007. p. 1-9.

BREWS, J. R. et al. A conceptual model of a single-event gate-rupture in power MOSFETs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 40, n. 6, p. 1959-1966, Dec. 1993.

BRIERE, D.; TRAVERSE, P. AIRBUS A320/A330/A340 electrical flight controls: a family of fault-tolerant systems. In: INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT

COMPUTING, 23., 1993, Toulouse. **Proceedings…** Piscataway: IEEE, 1993. p. 616-623.

BUCHNER, S. P.; BAZE, M. P. Single-event transients in fast electronic circuits. **IEEE NSREC Short Course Notes**, Vancouver, p. V 1-105, July 2001.

BUTT, N. Z.; ALAM, M. Modeling single event upsets in floating gate memory cells. In: RELIABILITY PHYSICS SYMPOSIUM, 2008, Phoenix. **Proceedings…** Piscataway: IEEE, 2008. p. 547-555.

CALIN, T.; NICOLAIDIS, M.; VELAZCO, R. Upset hardened memory design for submicron CMOS technology. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 6, p. 2874-2878, Dec. 1996.

CAPPELLETTI, P. et al. **Flash Memories**. Dordrecht: Kluwer Academic Publishers, 1999. 540p.

CARLSON, A. B. **Communication systems**: an introduction to signals and noise in electrical communication. 2 nd ed. New York: McGraw Hill, 1975. 510p.

CELLERE, G. et al. Can atmospheric neutrons induce soft errors in nand floating gate memories? **IEEE Electron Device Letters**, [S. l.], v. 30, n. 2, p. 178-180, Feb. 2009.

CELLERE, G. et al. Radiation effects on floating-gate memory cells. **IEEE Transactions on Nuclear Science**, [S. l.], v. 48, n. 6, p. 2222-2228, Dec. 2001.

CELLERE, G. et al. Sub-picosecond conduction through thin SiO_2 layers triggered by heavy ions. **Journal of applied physics**, [S. l.], v. 99, p. 74101, Apr. 2006.

CELLERE, G. et al. Transient conductive path induced by a Single ion in 10 nm SiO_2 Layers. **IEEE Transactions on Nuclear Science**, [S. l.], v. 51, n. 6, p. 3304-3311, Dec. 2004.

CELLERE, G.; PACCAGNELLA, A. A review of ionizing radiation effects in floating gate memories. **IEEE Transactions on Device and Materials Reliability**, [S. l.], v. 4, n. 3, p. 359-370, Sept. 2004.

CHEN, C. -E. D. et al. Single-transistor latch in SOI MOSFETs. **IEEE Electron Device Letters**, [S. l.], v. 9, n. 12, p. 636-638, Dec. 1988.

CHUNG, H. H. et al. Analysis of single events effects on monolithic PLL frequency synthesizers. **IEEE Transactions on Nuclear Science**, [S. l.], v. 53, n. 6, p. 3539-3543, Dec. 2006.

CYPRESS SEMICONDUCTOR. **PSoC 5LP architecture:** technical reference manual. San Jose, 2013a, 440 f. Disponível em: <http://www.cypress.com/?docID=46050>. Acesso em: 5 jan. 2015.

CYPRESS SEMICONDUCTOR. **PSoC 5LP development kit guide**. San Jose, 2013b, 58 f. Disponível em: <http://www.cypress.com/?docID=46949>. Acesso em: 6 jan. 2014.

CYPRESS SEMICONDUCTOR. **PSoC 5LP registers:** technical reference manual. San Jose,

2014, 2106 f. Disponível em: <http://www.cypress.com/?docID=46051>. Acesso em: 5 jan. 2015.

DAVIES, D.; WAKERLY, J. F. Synchronization and matching in redundant systems. **IEEE Transactions on Computers**, [S. l.], v. C-27, n. 6, p. 531-539, June 1978.

DIEHL, S. E. Considerations for single event immune VLSI logic. **IEEE Transactions on Nuclear Science**, [S. l.], v. 30, n. 6, p. 4501-4507, Dec. 1983.

DODD, P. E. Device simulation of charge collection and single-event upset. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 2, p. 561-575, Apr. 1996.

DODD, P. E. et al. Current and future challenges in radiation effects on CMOS electronics. **IEEE Transactions on Nuclear Science**, [S. l.], v. 57, n. 4, p. 1747-1763, Aug. 2010.

DODD, P. E. et al. Impact of technology trends on SEU in CMOS SRAMs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 6, p. 2797-2804, Dec. 1996.

DODD, P. E. et al. Neutron-induced latchup in SRAMs at ground level. In: ANNUAL IEEE INTERNATIONAL RELIABILITY PHYSICS SYMPOSIUM, 41., 2013, Dallas. **Proceedings…** Piscataway: IEEE, 2013. p. 51-55.

DODD, P. E. et al. Production and propagation of single-event transients in high-speed digital logic ICs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 51, n. 6, p. 3278-3284, Dec. 2004.

DODD, P. E. Physics-based simulation of single-event effects. **IEEE Transactions on Device and Materials Reliability**, [S. l.], v. 5, n. 3, p. 343-357, Sept. 2005.

DODD, P. E.; MASSENGILL, L. W. Basic mechanisms and modeling of single-event upset in digital microelectronics. **IEEE Transactions on Nuclear Science**, [S. l.], v. 50, n. 3, p. 583-602, June 2003.

DUFOUR, C. et al. Heavy ion induced single hard errors on submicronic memories [for space application]. **IEEE Transactions on Nuclear Science**, [S. l.], v. 39, n. 6, p. 1693-1697, Dec. 1992.

EBERS, J. J. Four-terminal P-N-P-N transistors. **Proceedings of the IRE**, [S. l.], v. 40, n. 11, p. 1361-1364, Nov. 1952.

ECOFFET, R. In-flight anomalies on electronic devices. In: VELAZCO, R.; FOUILLAT, P.; REIS, R. (Ed.). **Radiation effects on embedded systems**. Dordrecht: Springer, 2007. p. 31-68.

EUROPEAN SPACE AGENCY. **Ilustração dos cinturões de Van Allen e sua aproximação da Terra através da Anomalia Magnética do Atlântico Sul**. Disponível em: <http://www.esa.int/spaceinimages/Images/2007/03/ Earth_radiation_belts_with_the_South_ Atlantic_Anomaly_indicated>. Acesso em: 18 ago. 2014.

FEDERICO, C. A. **Dosimetria da radiação cósmica no interior de aeronaves no espaço**

aéreo Brasileiro. 2011. 172 f. Tese (Doutorado em Tecnologia Nuclear) – Universidade de São Paulo, São Paulo, 2011.

GADLAGE, M. J. et al. Single event transient pulse widths in digital microcircuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 51, n. 6, p. 3285-3290, Dec. 2004.

GAILLARD, R. Single event effects: mechanisms and classification. In: NICOLAIDIS, M. (Ed.). S**oft errors in modern electronic systems**. Dordrecht: Springer, 2011. p. 27-54.

GALLOWAY, K. F.; JOHNSON, G. H. Catastrophic single-event effects in the natural space environment. **IEEE NSREC Short Course Notes**, Monterey, p. IV 1-72, July 2003.

GAY C.; WELCH R.; SELBY V. Living with on-orbit anomalies in the TOPEX/POSEIDON earth sensors. In: ANNUAL AAS GUIDANCE CONTROL CONFERENCE, 16., 1993, Keystone. **Proceedings…** [S. l.: s. n.], 1993. p. 361-373.

GERARDIN, S. Radiation effects in Flash memories. **IEEE Transactions on Nuclear Science**, [S. l.], v. 60, n. 3, p. 1953-1969, June 2013.

GREGORY, B. L.; SHAFER, B. D. Latch-up in CMOS integrated Circuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 20, n. 6, p. 293-299, Dec. 1973.

GRÜRMANN, K. et al. Heavy ion sensitivity of 16/32-Gbit NAND-Flash and 4-Gbit DDR3 SDRAM. In: RADIATION EFFECTS DATA WORKSHOP, 2012, Miami. **Proceedings…** Piscataway: IEEE, 2012. p. 1-6.

GUENZER, C. S.; WOLICKI, E. A.; ALLAS, R. G. Single event upset of dynamic Rams by neutrons and protons. **IEEE Transactions on Nuclear Science**, [S. l.], v. 26, n. 6, p. 5048-5052, Dec. 1979.

HEIJMEN, T. Soft Errors from space to ground: historical overview, empirical evidence, and future trends. In: NICOLAIDIS, M. (Ed.). **Soft Errors in Modern Electronic Systems**. Dordrecht: Springer, 2011. p. 1-25.

HSUEH, M.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. **Computer**, [S. l.], v. 30, n. 4, p. 75-82, Apr. 1997.

INTERNATIONAL STANDARD ORGANIZATION. **ISO 8259-1**: reference neutron radiations - part 1: characteristics and methods of production. [S. l.], 2001.

JOHNSTON, A. H.; HUGHLOCK, B. W. Latchup in CMOS from single particles. **IEEE Transactions on Nuclear Science**, [S. l.], v. 37, n. 6, p. 1886-1893, Dec. 1990.

JOINT ELECTRON DEVICE ENGINEERING COUNCIL. **JESD57**: test procedures for the measurement of single-event effects in semiconductor devices from heavy ion irradiation. Arlington, 1996.

JOINT ELECTRON DEVICE ENGINEERING COUNCIL. **JESD89A**: measurement and reporting of alpha particle and terrestrial cosmic ray-induced soft errors in semiconductor devices. Arlington, 2006.

KARNIK, T.; HAZUCHA, P.; PATEL, J. Characterization of soft errors caused by single event upsets in CMOS processes. **IEEE Transactions on Dependable and Secure Computing**, [S. l.], v. 1, n. 2, p. 128-143, Apr./June 2004.

KOGA, R. et al. Observation of single event upsets in analog microcircuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 40, n. 6, p. 1838-1844, Dec. 1993.

KOGA, R. et al. On the suitability of non-hardened high density SRAMs for space applications. **IEEE Transactions on Nuclear Science**, [S. l.], v. 38, n. 6, p. 1507-1513, Dec. 1991.

KOGA, R. et al. Single event functional interrupt (SEFI) sensitivity in microcircuits. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, 4., 1997, Cannes. **Proceedings…** Piscataway: IEEE, 1997a. p. 311-318.

KOGA, R. et al. Single event upset (SEU) sensitivity dependence of linear integrated circuits (ICs) on bias conditions. **IEEE Transactions on Nuclear Science**, [S. l.], v. 44, n. 6, p. 2325-2332, Dec. 1997b.

KOGA, R.; KOLASINSKI, W. A. Heavy ion induced snapback in CMOS devices. **IEEE Transactions on Nuclear Science**, [S. l.], v. 36, n. 6, p. 2367-2374, Dec. 1989.

KOGA, R.; KOLASINSKI, W. A.; IMAMOTO, S. Heavy ion induced upsets in semiconductor devices. **IEEE Transactions on Nuclear Science**, [S. l.], v. 32, n. 1, p. 159-162, Feb. 1985.

KOLASINSKI, W. A. The effect of elevated temperature on latchup and bit errors in CMOS devices. **IEEE Transactions on Nuclear Science**, [S. l.], v. 33, n. 6, p. 1605-1609, Dec. 1986.

LADBURY, R. Radiation hardening at the system level. **IEEE NSREC Short Course Notes**, Waikiki, p. IV 1-94, July 2007.

LORCZAK, P. R.; CAGLAYAN, A. K.; ECKHARDT, D. E. A theoretical investigation of generalized voters for redundant systems. In: INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, 19., 1989, Chicago. **Proceedings…** Piscataway: IEEE, 1989. p. 444-451.

MASSENGILL, L. W. Cosmic and terrestrial single-event radiation effects in dynamic random access memories. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 2, p. 576-593, Apr. 1996.

MAY, T. C.; WOODS, M. H. A new physical mechanism for soft errors in dynamic memories. In: ANNUAL RELIABILITY PHYSICS SYMPOSIUM, 16., 1978, San Diego. **Proceedings…** Piscataway: IEEE, 1978. p. 33-40.

MCPARTLAND, R. J. Circuit simulations of alpha-particle-induced soft errors in MOS dynamic RAMs. **IEEE Journal of Solid-State Circuits**, [S. l.], v. 16, n. 1, p. 31-34, Feb. 1981.

MESSENGER, G. C. Collection of charge on junction nodes from ion tracks. **IEEE Transactions on Nuclear Science**, Snowmass Village, v. 29, n. 6, p. 2024-2031, Dec. 1982.

MITRA, S. et al. Robust system design with built-in soft-error resilience. **Computer**, [S. l.], v. 38, n. 2, p. 43-52, Feb. 2005.

MITRA, S. et al. Soft error resilient system design through error correction. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 2006, Nice. **Proceedings…** Piscataway: IEEE, 2006. p. 332-337.

MITRA, S.; MCCLUSKEY, E. J. Word-voter: a new voter design for triple modular redundant systems. In: IEEE VLSI TEST SYMPOSIUM, 18., 2000, Montréal. **Proceedings…** Piscataway: IEEE, 2000. p. 465-470.

MUNTEANU, D.; AUTRAN, J. L. Modeling and simulation of single-event effects in digital devices and ICs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 55, n. 4, p. 1854-1878, Aug. 2008.

MUSSEAU, O. et al. Analysis of multiple bit upsets (MBU) in CMOS SRAM. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 6, p. 2879-2888, Dec. 1996.

NARASIMHAM, B. et al. Characterization of digital single event transient pulse-widths in 130-nm and 90-nm CMOS technologies. **IEEE Transactions on Nuclear Science**, [S. l.], v. 54, n. 6, p. 2506-2511, Dec. 2007.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. **Ilustração dos cinturões de Van Allen**. Disponível em: <http://image.gsfc.nasa.gov/poetry/venus/a11789.html>. Acesso em: 18 ago. 2014a.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. **Natural space radiation effects on technology**. Disponível em: < http://radhome.gsfc.nasa.gov/radhome/Nat_Space_Rad_Tech.htm>. Acesso em: 18 ago. 2014b.

OLDHAM, T. R. et al. Total dose failures in advanced electronics from single ions. **IEEE Transactions on Nuclear Science**, [S. l.], v. 40, n. 6, p. 1820-1830, Dec. 1993.

PAULOS, J. J.; BISHOP, R. J.; TURFLINGER, T. L. Radiation-induced response of operational amplifiers in low-level transient radiation environments. **IEEE Transactions on Nuclear Science**, [S. l.], v. 34, n. 6, p. 1442-1447, Dec. 1987.

PAVAN, P.; LARCHER, L.; MARMIROLI, A. **Floating gate devices**: operation and compact modeling. Dordrecht: Kluwer Academic, 2004. 131p.

PEREIRA JUNIOR, E. C. F. et al. The fast neutron SEU cross section of a 4 Mb SRAM memory. In: INTERNATIONAL NUCLEAR ATLANTIC CONFERENCE, 2013, Recife. **Proceedings…** Vienna: IAEA INIS, 2013. Não paginado.

PETERSEN, E. **Single event effects in aerospace**. Hoboken: John Wiley & Sons, 2011. 520p.

PROKOFIEV, A. V. et al. CUP: a new high-flux irradiation position at the ANITA neutron facility at TSL. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, 14., 2013, Oxford. **Proceedings…** Piscataway: IEEE, 2013. p. 1-8.

PUCHNER, H.; RADAELLI, D.; CHATILA, A. Alpha-particle SEU performance of SRAM with triple well. **IEEE Transactions on Nuclear Science**, [S. l.], v. 51, n. 6, p. 3525-3528, Dec. 2004

REED, I. S.; SOLOMON, G. Polynomial codes over certain finite fields. **Journal of the Society for Industrial & Applied Mathematics**, [S. l.], v. 8, n. 2, p. 300-304, 1960.

RITER, R. Modeling and testing a critical fault-tolerant multi-process system. In: INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, 25., 1995, Pasadena. **Proceedings…** Piscataway: IEEE, 1995. p. 516-521.

SASAKI, Y.; NAMBA, K.; ITO H. Circuit and latch capable of masking soft errors with schmitt trigger. **Journal of Electronic Testing**, [S. l.], v. 24, p. 11-19, Jan. 2008.

SAYIL, S. Space radiation effects on technology and human biology and proper mitigation techniques. **Texas Space Grant Consortium Higher Education Grant Final Report**, [S. l.], p. 1-105, July 2010. Disponível em: <http://ee.lamar.edu/sayil/rad-eff.pdf>. Acesso em: 18 ago. 2014.

SCHWANK, J. R. et al. Radiation effects in SOI technologies. **IEEE Transactions on Nuclear Science**, [S. l.], v. 50, n. 3, p. 522-538, June 2003.

SEXTON, F. W. Destructive single-event effects in semiconductor devices and ICs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 50, n. 3, p. 603-621, June 2003.

SHIVAKUMAR, P. et al. Modeling the effect of technology trends on the soft error rate of combinational logic. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2002, Washington, D.C. **Proceedings…** Piscataway: IEEE, 2002. p. 389-398.

SHOCKLEY, W. **Electrons and holes in semiconductors**. Princeton: Van Nostrand, 1950. 112p.

SHORT, K. L. **Microprocessors and programmed logic**. 2 nd ed. Englewood Cliffs: Prentice Hall, 1987. 624p.

STASSINOPOULOS, E. G.; RAYMOND, J. P. The space radiation environment for electronics. **Proceedings of the IEEE**, [S. l.], v. 76, n. 11, p. 1423-1442, Nov. 1988.

SWIFT, G. M.; PADGETT, D. J.; JOHNSTON, A. H. A new class of single event hard errors [DRAM cells]. **IEEE Transactions on Nuclear Science**, [S. l.], v. 41, n. 6, p. 2043-2048, Dec. 1994.

TABER, A.; NORMAND, E. Single event upset in avionics. **IEEE Transactions on Nuclear Science**, [S. l.], v. 40, n. 2, p. 120-126, Apr. 1993.

TAKEDA, E. et al. A cross section of alpha-particle-induced soft-error phenomena in VLSIs. **IEEE Transactions on Electron Devices**, [S. l.], v. 36, n. 11, p. 2567-2575, Nov. 1989.

TERA TERM, version 4.82. [S. l.]: Tera Term open source project, 2014. Disponível em: <http://ttssh2.sourceforge.jp/index.html.en>. Acesso em: 5 jan. 2015.

TITUS, J. L. An updated perspective of single event gate rupture and single event burnout in power MOSFETs. **IEEE Transactions on Nuclear Science**, [S. l.], v. 60, n. 3, p. 1912-1928, June 2013.

TURFLINGER, T. L. Single-event effects in analog and mixed-signal integrated circuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 43, n. 2, p. 594-602, Apr. 1996.

TURFLINGER, T. L.; DAVEY, M. V. Understanding single event phenomena in complex analog and digital integrated circuits. **IEEE Transactions on Nuclear Science**, [S. l.], v. 37, n. 6, p. 1832-1838, Dec. 1990.

TURFLINGER, T. L.; DAVEY, M. V.; BINGS, J. P. Radiation effects in analog CMOS analog-to-digital converters. In: RADIATION EFFECTS DATA WORKSHOP, 1996, Indian Wells. **Proceedings…** Piscataway: IEEE, 1996. p. 6-12.

VIRMONTOIS, C. et al. Displacement damage effects due to neutron and proton irradiations on CMOS image sensors manufactured in deep submicron technology. **IEEE Transactions on Nuclear Science**, [S. l.], v. 57, n. 6, p. 3101-3108, Dec. 2010.

VON NEUMANN, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. **Automata studies**, [S. l.], v. 34, p. 43-98, 1956.

WALLMARK, J. T.; MARCUS, S. M. Minimum size and maximum packing density of nonredundant semiconductor devices. **Proceedings of the IRE**, [S. l.], v. 50, n. 3, p. 286-298, Mar. 1962.

WANG, F.; AGRAWAL, V. D. Single event upset: an embedded tutorial. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 21., 2008, Hyderabad. **Proceedings…** Piscataway: IEEE, 2008. p. 429-434.

WEAVER, B. D.; MCMORROW, D.; COHN, L. M. Radiation effects in III-V semiconductor electronics. In: VU, T. T. (Ed.). **Compound semiconductor integrated circuits.** Singapore: World Scientific, 2003. p. 293-326.

WESTE, N.; HARRIS, D. **CMOS VLSI design:** a circuits and systems perspective. 4 th ed. [S. l.]: Addison-Wesley, 2010. 864p.

YAMAGUCHI, K. et al. 3-D device modeling for SRAM soft-error immunity and tolerance analysis. **IEEE Transactions on Electron Devices**, [S. l.], v. 51, n. 3, p. 378-388, Mar. 2004.

ZHANG, M. et al. Sequential element design with built-in soft error resilience. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S. l.], v. 14, n. 12, p. 1368-1378, Dec. 2006.

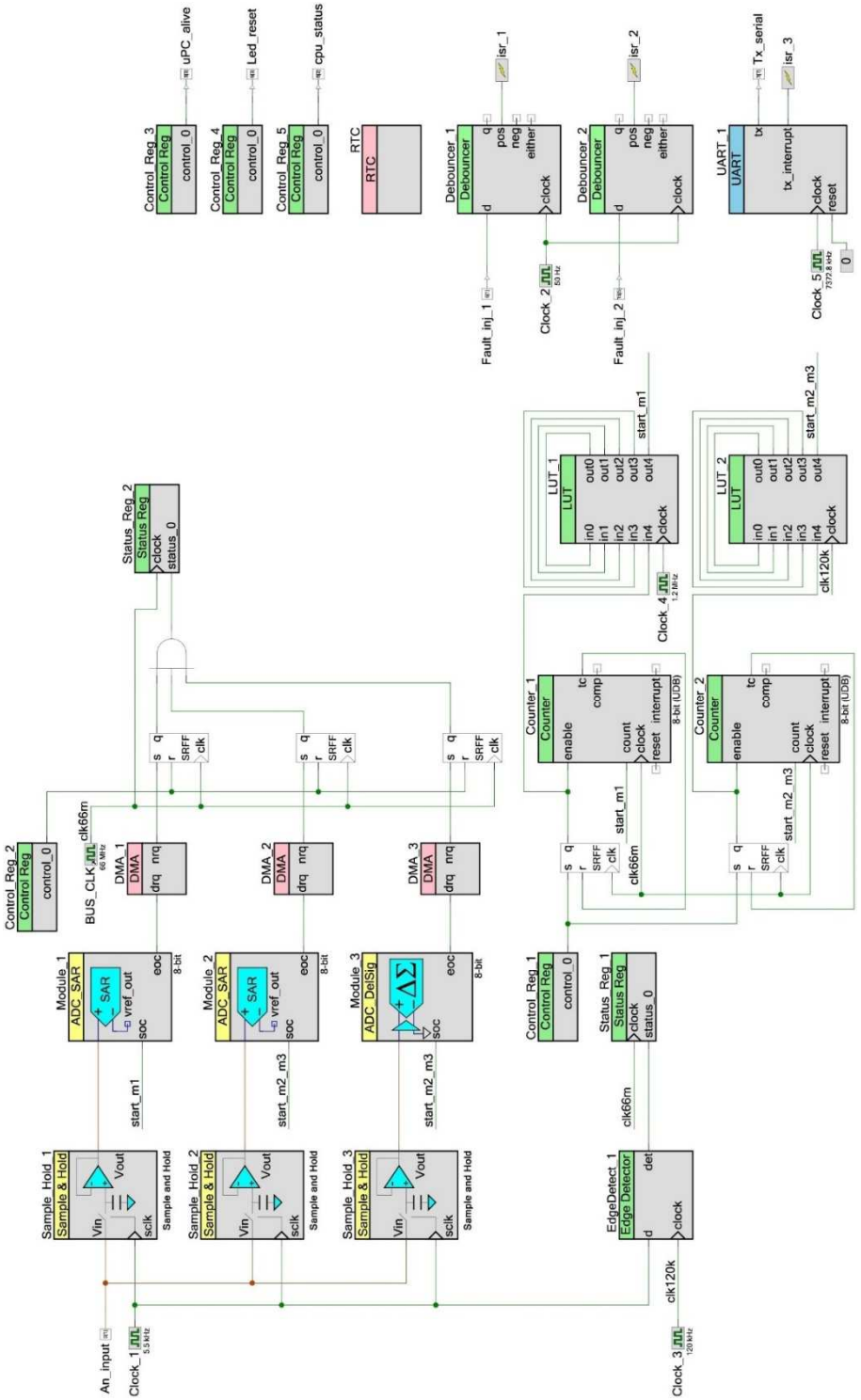
ZHANG, M.; SHANBHAG, N. R. A CMOS design style for logic circuit hardening. In: ANNUAL IEEE INTERNATIONAL RELIABILITY PHYSICS SYMPOSIUM, 43., 2005, San Jose. **Proceedings…** Piscataway: IEEE, 2005. p. 223-229.

ZHOU, Q.; MOHANRAM, K. Cost-effective radiation hardening technique for combinational logic. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 2004, San Jose. **Proceedings…** Piscataway: IEEE, 2004. p. 100-106.

ZHOU, Q.; MOHANRAM, K. Gate sizing to radiation harden combinational logic. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S. l.], v. 25, n. 1, p. 155-166, Jan. 2006.

ZIEGLER, J. F.; LANFORD, W. A. The effect of sea level cosmic rays on electronic devices. **Journal of applied physics**, [S. l.], v. 52, n. 6, p. 4305-4312, June 1981.

APÊNDICE A – ESQUEMÁTICO DA IMPLEMENTAÇÃO DO SAD DO ESTUDO DA SEÇÃO 4



APÊNDICE B – CÓDIGO QUE IMPLEMENTA O SAD DO ESTUDO DA SEÇÃO 4

```
#include <project.h>
#include <stdlib.h>
#include <stdio.h>

/* main function's variables*/
uint8 DMA_module1Data[10];
uint8 DMA_module2Data[2];
uint8 DMA_module3Data[2];
uint8 module1Data[10] = {0,0,0,0,0,0,0,0,0,0};
uint8 module2Data[2] = {0,0};
uint8 module3Data[2] = {0,0};
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];
uint8 DMA_2_Chan;
uint8 DMA_2_TD[1];
uint8 DMA_3_Chan;
uint8 DMA_3_TD[1];
uint16 cycle_counter = 0;
uint8 tmpStat;
uint8 running_bit = 0;
uint8 status_cpu_bit = 0;
uint16 one_sec_counter = 0;

/* SAR_ADC_voter function's variables*/
uint8 i, j;
uint8 bits;
uint8 mask[8] = {128,64,32,16,8,4,2,1};
uint8 bit_counter[8] = {0,0,0,0,0,0,0,0};
uint8 SAR_ADC_voter_data = 0;
uint8 SAR_ADC_voter_error_det = 0;
uint8 bigger;
uint8 smaller;
uint8 pointer;
uint8 error;

/* main_voter function's variables */
uint8 error1, error2, error3;
uint8 system_output = 0;
uint8 data_invalid_flag = 0;
uint8 main_voter_error_det = 0;

/* buffer's variables */
uint16 buf_1[909];
uint8 buf_2[101];
uint8 buf_3[101];
uint8 buf_4[101];
uint8 buf_5[101];
uint16 buf_6[101];
uint8 buf_7[101];
uint16 buf_pointer_1 = 0;
uint8 buf_pointer_2 = 0;

/* uart_print_values function's variables */
char8 *module1Data_1[3];
char8 *module1Data_2[3];
char8 *module1Data_3[3];
char8 *module1Data_4[3];
char8 *module1Data_5[3];
char8 *module1Data_6[3];
char8 *module1Data_7[3];
char8 *module1Data_8[3];
char8 *module1Data_9[3];
char8 *module2Data_1[3];
char8 *module3Data_1[3];
char8 *SAR_ADC_voter_data_uart[3];
char8 *system_output_uart[3];
char8 *cycle_counter_uart[5];
char8 *cycle_error_uart[1];
uint16 uart_pointer_1 = 0;
uint8 uart_pointer_2 = 0;

/* uart_control_write function's variables */
uint32 count_alive = 0;
uint8 error_capture = 0;
uint8 buffer_counter = 0;
uint32 fault_counter = 0;
```

```
77      char8 *fault_counter_uart[10];
78
79      /* fault injection */
80      uint8 flag_1 = 0;
81      uint8 flag_2 = 0;
82
83      /* real time clock */
84      uint8 second = 0u;
85      uint8 minute = 0u;
86      uint8 hour = 0u;
87      uint8 day_month = 0u;
88      uint8 month = 0u;
89      char8 *second_uart[2];
90      char8 *minute_uart[2];
91      char8 *hour_uart[2];
92      char8 *day_month_uart[2];
93      char8 *month_uart[2];
94
95      CY_ISR(Interrupt_1)
96      {
97          flag_1 = 1;
98      }
99
100     CY_ISR(Interrupt_2)
101     {
102         flag_2 = 1;
103     }
104
105     void DMA_1_Config(void)
106     {
107             #define DMA_1_BYTES_PER_BURST 1
108             #define DMA_1_REQUEST_PER_BURST 1
109             #define DMA_1_SRC_BASE (CYDEV_PERIPH_BASE)
110             #define DMA_1_DST_BASE (CYDEV_SRAM_BASE)
111
112             DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST, HI16(DMA_1_SRC_BASE),
    HI16(DMA_1_DST_BASE));
113             DMA_1_TD[0] = CyDmaTdAllocate();
114             CyDmaTdSetConfiguration(DMA_1_TD[0], 10, DMA_1_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
115             CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)Module_1_SAR_WRK0_PTR), LO16((uint32)DMA_module1Data));
116             CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
117             CyDmaChEnable(DMA_1_Chan, 1);
118     }
119
120     void DMA_2_Config(void)
121     {
122             #define DMA_2_BYTES_PER_BURST 1
123             #define DMA_2_REQUEST_PER_BURST 1
124             #define DMA_2_SRC_BASE (CYDEV_PERIPH_BASE)
125             #define DMA_2_DST_BASE (CYDEV_SRAM_BASE)
126
127             DMA_2_Chan = DMA_2_DmaInitialize(DMA_2_BYTES_PER_BURST, DMA_2_REQUEST_PER_BURST, HI16(DMA_2_SRC_BASE),
    HI16(DMA_2_DST_BASE));
128             DMA_2_TD[0] = CyDmaTdAllocate();
129             CyDmaTdSetConfiguration(DMA_2_TD[0], 2, DMA_2_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
130             CyDmaTdSetAddress(DMA_2_TD[0], LO16((uint32)Module_2_SAR_WRK0_PTR), LO16((uint32)DMA_module2Data));
131             CyDmaChSetInitialTd(DMA_2_Chan, DMA_2_TD[0]);
132             CyDmaChEnable(DMA_2_Chan, 1);
133     }
134
135     void DMA_3_Config(void)
136     {
137             #define DMA_3_BYTES_PER_BURST 1
138             #define DMA_3_REQUEST_PER_BURST 1
139             #define DMA_3_SRC_BASE (CYDEV_PERIPH_BASE)
140             #define DMA_3_DST_BASE (CYDEV_SRAM_BASE)
141
142             DMA_3_Chan = DMA_3_DmaInitialize(DMA_3_BYTES_PER_BURST, DMA_3_REQUEST_PER_BURST, HI16(DMA_3_SRC_BASE),
    HI16(DMA_3_DST_BASE));
143             DMA_3_TD[0] = CyDmaTdAllocate();
144             CyDmaTdSetConfiguration(DMA_3_TD[0], 2, DMA_3_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
145             CyDmaTdSetAddress(DMA_3_TD[0], LO16((uint32)Module_3_DEC_SAMP_PTR), LO16((uint32)DMA_module3Data));
146             CyDmaChSetInitialTd(DMA_3_Chan, DMA_3_TD[0]);
147             CyDmaChEnable(DMA_3_Chan, 1);
148     }
149
150     void SAR_ADC_voter()
151     {
152         bits = 0;
153         SAR_ADC_voter_data = 0;
154         for (i = 0; i < 8; i++)
```

```
155             bit_counter[i] = 0;
156
157         for (i = 0; i < 8; i++)
158         {
159             for (j = 1; j < 10; j++)
160             {
161                 bits = (module1Data[j] & mask[i]) != 0;
162                 if (bits == 1)
163                     bit_counter[i] = bit_counter[i] + 1;
164             }
165         }
166
167         for (i = 0; i < 8; i++)
168         {
169             if (bit_counter[i] > 4)
170             {
171                 SAR_ADC_voter_data = SAR_ADC_voter_data + mask[i];
172             }
173         }
174
175         bigger = module1Data[1];
176         smaller = module1Data[1];
177         for (pointer = 2; pointer < 10; pointer++)
178         {
179             if (bigger < module1Data[pointer])
180                 bigger = module1Data[pointer];
181
182             if (smaller > module1Data[pointer])
183                 smaller = module1Data[pointer];
184         }
185         error = bigger - smaller;
186         if (error > 4)
187             SAR_ADC_voter_error_det = 1;
188     }
189
190     void main_voter()
191     {
192         error1 = abs (SAR_ADC_voter_data - module2Data[1]);
193         error2 = abs (module2Data[1] - module3Data[1]);
194         error3 = abs (module3Data[1] - SAR_ADC_voter_data);
195
196         if (error1 <= 4)
197         {
198             system_output = SAR_ADC_voter_data;
199             data_invalid_flag = 0;
200         }
201         else if (error2 <= 4)
202         {
203             system_output = module2Data[1];
204             data_invalid_flag = 0;
205         }
206         else if (error3 <= 4)
207         {
208             system_output = module3Data[1];
209             data_invalid_flag = 0;
210         }
211         else
212         {
213             system_output = 0;
214             data_invalid_flag = 1;
215         }
216
217         if ((error1 > 4) ||  (error2 > 4) || (error3 > 4))
218             main_voter_error_det = 1;
219     }
220
221     void buffer()
222     {
223         if (buf_pointer_2 > 100)
224         {
225             buf_pointer_1 = 0;
226             buf_pointer_2 = 0;
227         }
228
229         buf_1[buf_pointer_1] = module1Data[1];
230         buf_pointer_1++;
231         buf_1[buf_pointer_1] = module1Data[2];
232         buf_pointer_1++;
233         buf_1[buf_pointer_1] = module1Data[3];
234         buf_pointer_1++;
```

```
235         buf_1[buf_pointer_1] = module1Data[4];
236         buf_pointer_1++;
237         buf_1[buf_pointer_1] = module1Data[5];
238         buf_pointer_1++;
239         buf_1[buf_pointer_1] = module1Data[6];
240         buf_pointer_1++;
241         buf_1[buf_pointer_1] = module1Data[7];
242         buf_pointer_1++;
243         buf_1[buf_pointer_1] = module1Data[8];
244         buf_pointer_1++;
245         buf_1[buf_pointer_1] = module1Data[9];
246         buf_pointer_1++;
247 
248         buf_2[buf_pointer_2] = module2Data[1];
249         buf_3[buf_pointer_2] = module3Data[1];
250         buf_4[buf_pointer_2] = SAR_ADC_voter_data;
251         buf_5[buf_pointer_2] = system_output;
252         buf_6[buf_pointer_2] = cycle_counter;
253         buf_7[buf_pointer_2] = SAR_ADC_voter_error_det || main_voter_error_det;
254         buf_pointer_2++;
255     }
256 
257     void uart_print_time()
258     {
259         second = RTC_ReadSecond();
260         minute = RTC_ReadMinute();
261         hour = RTC_ReadHour();
262         day_month = RTC_ReadDayOfMonth();
263         month = RTC_ReadMonth();
264 
265         sprintf(second_uart, "%d", second);
266         sprintf(minute_uart, "%d", minute);
267         sprintf(hour_uart, "%d", hour);
268         sprintf(day_month_uart, "%d", day_month);
269         sprintf(month_uart, "%d", month);
270 
271         UART_1_PutString(&hour_uart);
272         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
273         UART_1_PutString(":");
274         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
275         UART_1_PutString(&minute_uart);
276         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
277         UART_1_PutString(":");
278         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
279         UART_1_PutString(&second_uart);
280         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
281         UART_1_PutString(" ");
282         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
283         UART_1_PutString(&month_uart);
284         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
285         UART_1_PutString("/");
286         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
287         UART_1_PutString(&day_month_uart);
288         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
289     }
290 
291     void uart_print_values()
292     {
293 
294         UART_1_PutString("**************************");
295         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
296         UART_1_PutString("\n\r");
297         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
298 
299         UART_1_PutString("Faults in ");
300         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
301         uart_print_time();
302         UART_1_PutString(" :");
303         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
304         UART_1_PutString("\n\r");
305         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
306 
307         UART_1_PutString("----------------------");
308         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
309         UART_1_PutString("\n\r");
310         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
311         uart_pointer_1 = 0;
312 
313         for (uart_pointer_2=0; uart_pointer_2 < 101; uart_pointer_2++)
314         {
```

```
315             sprintf(module1Data_1, "%d", buf_1[uart_pointer_1]);
316             uart_pointer_1++;
317             sprintf(module1Data_2, "%d", buf_1[uart_pointer_1]);
318             uart_pointer_1++;
319             sprintf(module1Data_3, "%d", buf_1[uart_pointer_1]);
320             uart_pointer_1++;
321             sprintf(module1Data_4, "%d", buf_1[uart_pointer_1]);
322             uart_pointer_1++;
323             sprintf(module1Data_5, "%d", buf_1[uart_pointer_1]);
324             uart_pointer_1++;
325             sprintf(module1Data_6, "%d", buf_1[uart_pointer_1]);
326             uart_pointer_1++;
327             sprintf(module1Data_7, "%d", buf_1[uart_pointer_1]);
328             uart_pointer_1++;
329             sprintf(module1Data_8, "%d", buf_1[uart_pointer_1]);
330             uart_pointer_1++;
331             sprintf(module1Data_9, "%d", buf_1[uart_pointer_1]);
332             uart_pointer_1++;
333
334             sprintf(module2Data_1, "%d", buf_2[uart_pointer_2]);
335             sprintf(module3Data_1, "%d", buf_3[uart_pointer_2]);
336             sprintf(SAR_ADC_voter_data_uart, "%d", buf_4[uart_pointer_2]);
337             sprintf(system_output_uart, "%d", buf_5[uart_pointer_2]);
338             sprintf(cycle_counter_uart, "%d", buf_6[uart_pointer_2]);
339             sprintf(cycle_error_uart, "%d", buf_7[uart_pointer_2]);
340
341             UART_1_PutString("Mod_1_[01]: ");
342             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
343             UART_1_PutString(&module1Data_1);
344             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
345             UART_1_PutString("\n\r");
346             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
347
348             UART_1_PutString("Mod_1_[02]: ");
349             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
350             UART_1_PutString(&module1Data_2);
351             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
352             UART_1_PutString("\n\r");
353             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
354
355             UART_1_PutString("Mod_1_[03]: ");
356             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
357             UART_1_PutString(&module1Data_3);
358             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
359             UART_1_PutString("\n\r");
360             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
361
362             UART_1_PutString("Mod_1_[04]: ");
363             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
364             UART_1_PutString(&module1Data_4);
365             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
366             UART_1_PutString("\n\r");
367             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
368
369             UART_1_PutString("Mod_1_[05]: ");
370             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
371             UART_1_PutString(&module1Data_5);
372             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
373             UART_1_PutString("\n\r");
374             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
375
376             UART_1_PutString("Mod_1_[06]: ");
377             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
378             UART_1_PutString(&module1Data_6);
379             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
380             UART_1_PutString("\n\r");
381             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
382
383             UART_1_PutString("Mod_1_[07]: ");
384             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
385             UART_1_PutString(&module1Data_7);
386             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
387             UART_1_PutString("\n\r");
388             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
389
390             UART_1_PutString("Mod_1_[08]: ");
391             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
392             UART_1_PutString(&module1Data_8);
393             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
394             UART_1_PutString("\n\r");
```

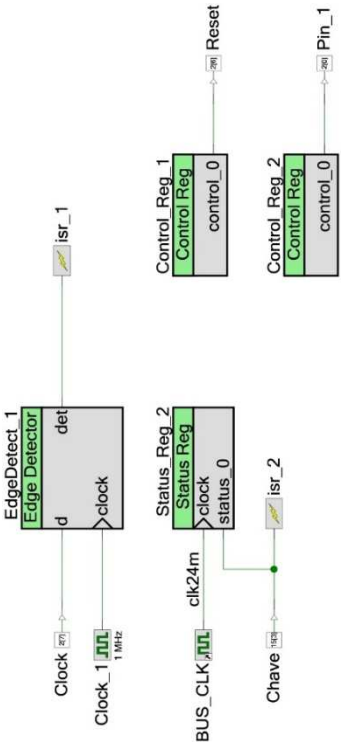
```
395             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
396
397             UART_1_PutString("Mod_1_[09]: ");
398             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
399             UART_1_PutString(&module1Data_9);
400             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
401             UART_1_PutString("\n\r");
402             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
403
404             UART_1_PutString("Mod_2: ");
405             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
406             UART_1_PutString(&module2Data_1);
407             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
408             UART_1_PutString("\n\r");
409             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
410
411             UART_1_PutString("Mod_3: ");
412             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
413             UART_1_PutString(&module3Data_1);
414             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
415             UART_1_PutString("\n\r");
416             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
417
418             UART_1_PutString("SAR_ADC_voter: ");
419             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
420             UART_1_PutString(&SAR_ADC_voter_data_uart);
421             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
422             UART_1_PutString("\n\r");
423             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
424
425             UART_1_PutString("System_output: ");
426             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
427             UART_1_PutString(&system_output_uart);
428             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
429             UART_1_PutString("\n\r");
430             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
431
432             UART_1_PutString("Cycle_counter: ");
433             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
434             UART_1_PutString(&cycle_counter_uart);
435             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
436             UART_1_PutString("\n\r");
437             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
438
439             UART_1_PutString("Cycle_error: ");
440             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
441             UART_1_PutString(&cycle_error_uart);
442             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
443             UART_1_PutString("\n\r");
444             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
445
446             UART_1_PutString("----------------------");
447             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
448             UART_1_PutString("\n\r");
449             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
450         }
451     }
452
453     void uart_print_status()
454     {
455         sprintf(fault_counter_uart, "%d", fault_counter);
456
457         UART_1_PutString("Tot faults: ");
458         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
459         UART_1_PutString(&fault_counter_uart);
460         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
461         UART_1_PutString(" - ");
462         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
463
464         uart_print_time();
465
466         UART_1_PutString("\n\r");
467         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
468     }
469
470     void uart_control_write()
471     {
472         if (error_capture == 1)
473         {
474             buffer_counter++;
```

```
475             if (buffer_counter == 50)
476             {
477                 uart_print_values();
478                 error_capture = 0;
479                 buffer_counter = 0;
480             }
481         }
482         if ((SAR_ADC_voter_error_det == 1) || (main_voter_error_det == 1))
483         {
484             error_capture = 1;
485             fault_counter++;
486             SAR_ADC_voter_error_det = 0;
487             main_voter_error_det = 0;
488             count_alive = 0;
489         }
490         else
491         {
492             count_alive++;
493             if (count_alive == 943200)
494             {
495                 uart_print_status();
496                 count_alive = 0;
497             }
498         }
499     }
500
501     int main()
502     {
503         /* Enable led of reset */
504         Control_Reg_4_Write(1);
505
506         /* Disable led cpu status */
507         Control_Reg_5_Write(status_cpu_bit);
508
509         /* Initialize interruptions */
510         isr_1_StartEx(Interrupt_1);
511         isr_2_StartEx(Interrupt_2);
512         isr_3_Start();
513         CyDelay(100);
514         isr_1_ClearPending();
515         isr_2_ClearPending();
516         isr_3_ClearPending();
517         CyGlobalIntEnable;
518
519         /* Initialize UART */
520         UART_1_Start();
521
522         /* Initialize DMA */
523               DMA_1_Config();
524               DMA_2_Config();
525               DMA_3_Config();
526
527         /* Initialize Sample_hold */
528         Sample_Hold_1_Start();
529         Sample_Hold_1_Enable();
530         Sample_Hold_2_Start();
531         Sample_Hold_2_Enable();
532         Sample_Hold_3_Start();
533         Sample_Hold_3_Enable();
534
535         /* Initialize modules */
536               Module_1_Start();
537               Module_2_Start();
538               Module_3_Start();
539
540         /* Initialize counters */
541         Counter_1_Start();
542         Counter_2_Start();
543
544         /* Initialize clock_1 */
545         Clock_1_Start();
546
547         /* Delay to stabilize operation */
548         CyDelay(10000);
549
550         /* Initialize Real time Clock */
551         RTC_TIME_DATE Start;
552
553         Start.Sec = 0u;
554         Start.Min = 0u;
```

```
555         Start.Hour = 0u;
556         Start.DayOfMonth = 1u;
557         Start.Month = 1u;
558         Start.Year = 2014u;
559 
560         RTC_WriteTime(&Start);
561         RTC_Start();
562 
563         /* Disable led of UART */
564         Control_Reg_4_Write(0);
565 
566         /* First write in UART */
567         UART_1_PutString("***** System started *****");
568         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
569         UART_1_PutString("\n\r");
570         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
571         UART_1_PutString("**************************");
572         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
573         UART_1_PutString("\n\r");
574         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
575 
576         for(;;)
577         {
578             /* Reset of flip-flops */
579             Control_Reg_2_Write(1);
580             CyDelayCycles(5);
581             Control_Reg_2_Write(0);
582 
583             /* Wait for sampling control */
584             while (Status_Reg_1_Read() == 0)
585             {
586             }
587 
588             /* Wait for the end of the sampling */
589             CyDelayCycles(660);
590 
591             /* Start conversions */
592             Control_Reg_1_Write(1);
593             CyDelayCycles(5);
594             Control_Reg_1_Write(0);
595 
596             /* Processing */
597             SAR_ADC_voter();
598             main_voter();
599             buffer();
600             uart_control_write();
601 
602             /* Wait for the end of the conversions */
603             while (Status_Reg_2_Read() == 0)
604             {
605             }
606 
607             /* Fault injection */
608             if (flag_1 == 1)
609             {
610                 flag_1 = 0;
611                 DMA_module1Data[3] = DMA_module1Data[3] ^ 128;
612             }
613 
614             if (flag_2 == 1)
615             {
616                 flag_2 = 0;
617                 DMA_module2Data[1] = DMA_module2Data[1] ^ 32;
618             }
619 
620             /* Move DMA to temporary buffer */
621 
622             module1Data[1] = DMA_module1Data[1];
623             module1Data[2] = DMA_module1Data[2];
624             module1Data[3] = DMA_module1Data[3];
625             module1Data[4] = DMA_module1Data[4];
626             module1Data[5] = DMA_module1Data[5];
627             module1Data[6] = DMA_module1Data[6];
628             module1Data[7] = DMA_module1Data[7];
629             module1Data[8] = DMA_module1Data[8];
630             module1Data[9] = DMA_module1Data[9];
631             module2Data[1] = DMA_module2Data[1];
632             module3Data[1] = DMA_module3Data[1];
633 
634             /* Increments cycle_counter */
```

```
635             cycle_counter++;
636 
637             /* Sends the running bit */
638             running_bit = !running_bit;
639             Control_Reg_3_Write(running_bit);
640 
641             /* Led cpu status */
642             one_sec_counter++;
643             if (one_sec_counter == 5680)
644             {
645                 one_sec_counter = 0;
646                 status_cpu_bit = !status_cpu_bit;
647                 Control_Reg_5_Write(status_cpu_bit);
648             }
649         }
650     }
```

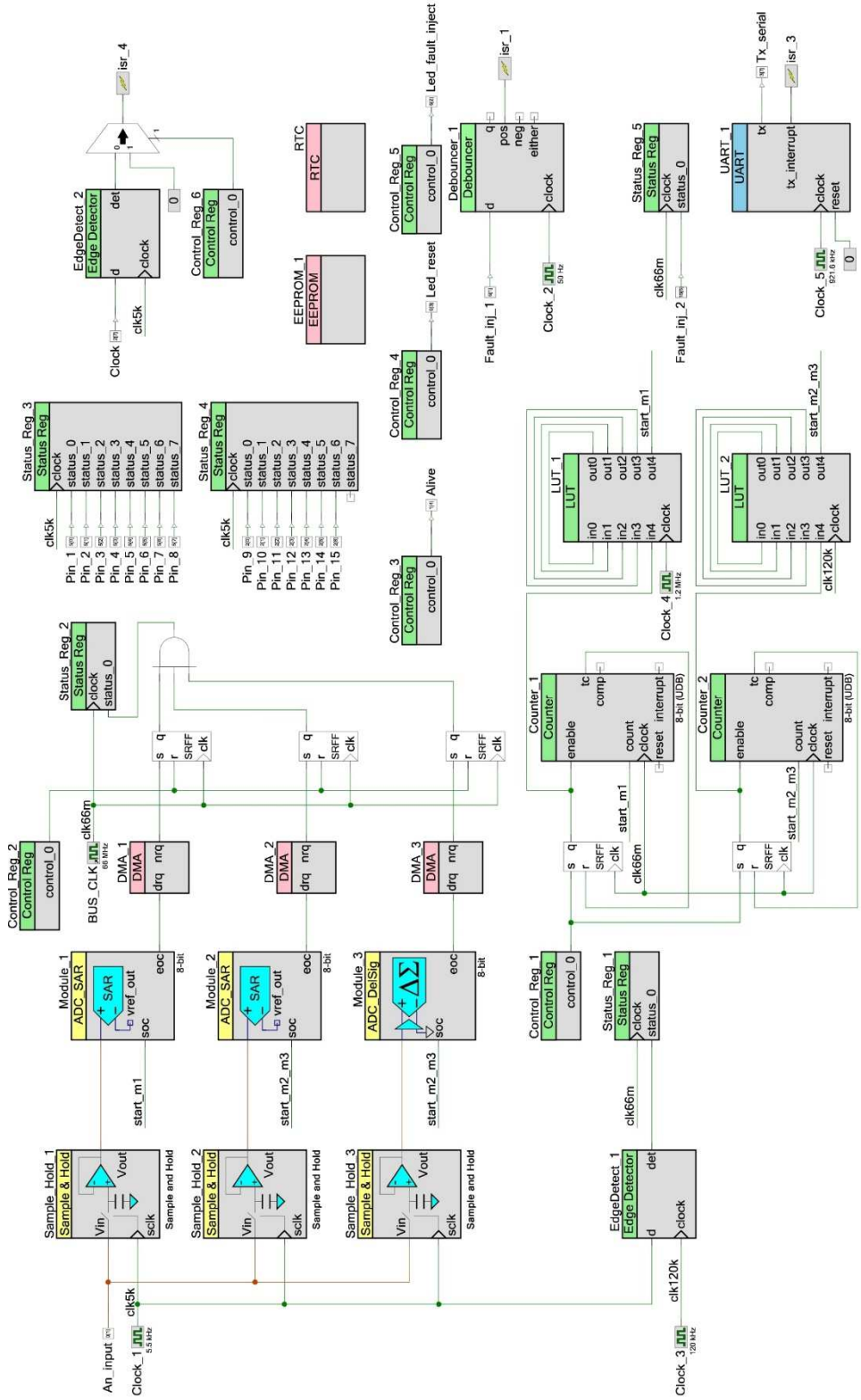
APÊNDICE C – ESQUEMÁTICO DA IMPLEMENTAÇÃO DO AE DO ESTUDO DA SEÇÃO 4



APÊNDICE D – CÓDIGO QUE IMPLEMENTA O AE DO ESTUDO DA SEÇÃO 4

```
#include <device.h>

uint8 contador = 0;

CY_ISR(Interrupt_1)
{
    contador = 0;
}

CY_ISR(Interrupt_2)
{
    Control_Reg_1_Write(0);
    Control_Reg_2_Write(0);
    CyDelay(1000);
    Control_Reg_1_Write(1);
    Control_Reg_2_Write(1);
    contador = 0;
}

void main()
{
    Control_Reg_1_Write(0);
    Control_Reg_2_Write(0);

    while (Status_Reg_2_Read() == 0)
    {
    }

    Control_Reg_2_Write(1);
    Control_Reg_1_Write(1);

    isr_1_StartEx(Interrupt_1);
    isr_2_StartEx(Interrupt_2);
    CyDelay(100);
    isr_1_ClearPending();
    isr_2_ClearPending();
    CyGlobalIntEnable;

    for(;;)

    {
        contador++;
        CyDelay(1000);
        if (contador == 30)
        {
            Control_Reg_1_Write(0);
            Control_Reg_2_Write(0);
            CyDelay(1000);
            Control_Reg_1_Write(1);
            Control_Reg_2_Write(1);
            contador = 0;
        }
    }
}
```

APÊNDICE E – ESQUEMÁTICO DA IMPLEMENTAÇÃO DO SAD DO ESTUDO DA SEÇÃO 5



APÊNDICE F – CÓDIGO QUE IMPLEMENTA O SAD DO ESTUDO DA SEÇÃO 5

```
#include <project.h>
#include <stdlib.h>
#include <stdio.h>

/* main function's variables*/
uint8 DMA_module1Data[10];
uint8 DMA_module2Data[2];
uint8 DMA_module3Data[2];
uint8 module1Data[10] = {0,0,0,0,0,0,0,0,0,0};
uint8 module2Data[2] = {0,0};
uint8 module3Data[2] = {0,0};
uint8 DMA_1_Chan;
uint8 DMA_1_TD[1];
uint8 DMA_2_Chan;
uint8 DMA_2_TD[1];
uint8 DMA_3_Chan;
uint8 DMA_3_TD[1];
uint16 cycle_counter = 0;
uint8 tmpStat;
uint8 running_bit = 0;
uint8 led_fault_inject = 0;

/* SAR_ADC_voter function's variables*/
uint8 i, j;
uint8 bits;
uint8 mask[8] = {128,64,32,16,8,4,2,1};
uint8 bit_counter[8] = {0,0,0,0,0,0,0,0};
uint8 SAR_ADC_voter_data = 0;
uint8 SAR_ADC_voter_error_det = 0;
uint8 bigger;
uint8 smaller;
uint8 pointer;
uint8 error;

/* main_voter function's variables */
uint8 error1, error2, error3;
uint8 system_output = 0;
uint8 data_invalid_flag = 0;
uint8 main_voter_error_det = 0;

/* buffer's variables */
uint16 buf_1[909];
uint8 buf_2[101];
uint8 buf_3[101];
uint8 buf_4[101];
uint8 buf_5[101];
uint16 buf_6[101];
uint8 buf_7[101];
uint16 buf_pointer_1 = 0;
uint8 buf_pointer_2 = 0;

/* uart_print_values function's variables */
char8 *module1Data_1[3];
char8 *module1Data_2[3];
char8 *module1Data_3[3];
char8 *module1Data_4[3];
char8 *module1Data_5[3];
char8 *module1Data_6[3];
char8 *module1Data_7[3];
char8 *module1Data_8[3];
char8 *module1Data_9[3];
char8 *module2Data_1[3];
char8 *module3Data_1[3];
char8 *SAR_ADC_voter_data_uart[3];
char8 *system_output_uart[3];
char8 *cycle_counter_uart[5];
char8 *cycle_error_uart[1];
uint16 uart_pointer_1 = 0;
uint8 uart_pointer_2 = 0;

/* uart_control_write function's variables */
uint32 count_alive = 0;
uint8 error_capture = 0;
uint8 buffer_counter = 0;
uint32 fault_counter = 0;
char8 *fault_counter_uart[10];
```

```
77
78      /* fault injection through button */
79      uint8 flag_1 = 0;
80
81      /* real time clock */
82      uint8 second = 0u;
83      uint8 minute = 0u;
84      uint8 hour = 0u;
85      uint8 day_month = 0u;
86      uint8 month = 0u;
87      char8 *second_uart[2];
88      char8 *minute_uart[2];
89      char8 *hour_uart[2];
90      char8 *day_month_uart[2];
91      char8 *month_uart[2];
92
93      /* SRAM fault injection */
94      uint8 flag_2 = 0;
95      uint16 prs_number = 0;
96      uint8 low_high_addr = 0;
97      uint32 low_addr = 0x1FFF8000;
98      uint32 high_addr = 0x20000000;
99      uint8 target_bit[8] = {1,2,4,8,16,32,64,128};
100     uint8 state_bit = 0;
101     uint32 target_addr = 0;
102     char8 target_addr_uart[9];
103     reg8 *pointer_addr = 0;
104
105     /* Peripheral registers fault injection */
106     /*uint16 prs_number = 0;
107     uint32 target_addr = 0;
108     char8 target_addr_uart[9];
109     uint8 target_bit[8] = {1,2,4,8,16,32,64,128};
110     uint8 state_bit = 0;
111     reg8 *pointer_addr = 0;
112     uint32 periph_reg[7892] =
        {0x00000000,0x40004000,0x40004001,0x40004002,0x40004003,0x40004004,0x40004005,0x40004006,0x40004007,0x40004008,0x4
        0004009,0x4000400a,0x4000400b,0x40004010,0x40004014,0x40004080,0x40004084,0x40004088,0x4000408c,0x40004090,0x40004
        094,0x40004098,0x4000409c,0x40004081,0x40004085,0x40004089,0x4000408d,0x40004091,0x40004095,0x40004099,0x4000409d,
        0x40004082,0x40004086,0x4000408a,0x4000408e,0x40004092,0x40004096,0x4000409a,0x4000409e,0x40004100,0x40004104,0x40
        004108,0x4000410c,0x40004101,0x40004105,0x40004109,0x4000410d,0x40004102,0x40004106,0x4000410a,


113
        0x4000410e,0x40004103,0x40004107,0x4000410b,0x4000410f,0x40004200,0x40004210,0x40004211,0x40004212,0x40004213,0x40
        004220,0x40004221,0x40004222,0x40004223,0x40004224,0x40004225,0x40004300,0x40004301,0x40004308,0x40004309,0x400043
        0a,0x40004320,0x40004321,0x40004322,0x40004323,0x40004324,0x40004325,0x40004326,0x40004330,0x40004331,0x40004380,0
        x40004381,0x40004382,0x40004383,0x40004384,0x40004390,0x40004391,0x40004392,0x40004393,0x40004394,0x40004398,0x400
        04399,0x4000439a,0x400043a0,0x400043a1,0x400043a2,0x400043a3,0x400043a4,0x400043a5,0x400043a6,


114
        0x400043a7,0x400043a8,0x400043a9,0x400043aa,0x400043ab,0x400043ac,0x400043ad,0x400043b0,0x400043b1,0x400043b2,0x40
        0043b3,0x400043b4,0x400043b5,0x400043b6,0x400043b7,0x400043b8,0x400043b9,0x400043ba,0x400043bb,0x400043bc,0x400043
        bd,0x400043c0,0x400043c1,0x400043c2,0x400043c3,0x400043c4,0x400043c5,0x400043c6,0x400043d0,0x400043d1,0x400043d2,0
        x400043d3,0x400043d4,0x400043d5,0x400043d6,0x40004e00,0x40004e01,0x40004e02,0x40004e03,0x40004e04,0x40004e05,0x400
        04e06,0x40004e08,0x40004e09,0x40004e0a,0x40004e0c,0x40004e0d,0x40004e0e,0x40004e10,0x40004e11,


115
        0x40004e12,0x40004e13,0x40004e14,0x40004f00,0x40004f18,0x40004f0c,0x40004f24,0x40004f01,0x40004f19,0x40004f0d,0x40
        004f25,0x40004f02,0x40004f1a,0x40004f0e,0x40004f26,0x40004f03,0x40004f1b,0x40004f0f,0x40004f27,0x40004f04,0x40004f
        1c,0x40004f10,0x40004f28,0x40004f05,0x40004f1d,0x40004f11,0x40004f29,0x40004f06,0x40004f1e,0x40004f12,0x40004f2a,0
        x40004f07,0x40004f1f,0x40004f13,0x40004f2b,0x40004f08,0x40004f20,0x40004f14,0x40004f2c,0x40004f09,0x40004f21,0x400
        04f15,0x40004f2d,0x40004f0a,0x40004f22,0x40004f16,0x40004f2e,0x40004f0b,0x40004f23,0x40004f17,

116
        0x40004f2f,0x40005800,0x40005808,0x40005804,0x4000580c,0x40005801,0x40005809,0x40005805,0x4000580d,0x40005802,0x40
        00580a,0x40005806,0x4000580e,0x40005820,0x40005828,0x40005824,0x4000582c,0x40005821,0x40005829,0x40005825,0x400058
        2d,0x40005822,0x4000582a,0x40005826,0x4000582e,0x40005840,0x40005842,0x40005841,0x40005843,0x40005848,0x4000584c,0
        x4000584a,0x4000584e,0x40005849,0x4000584d,0x4000584b,0x4000584f,0x40005858,0x4000585c,0x4000585a,0x4000585e,0x400
        05859,0x4000585d,0x4000585b,0x4000585f,0x40005868,0x40005869,0x4000586a,0x4000586b,0x4000586c,

117
        0x4000586d,0x4000586e,0x4000586f,0x40005870,0x40005871,0x40005872,0x40005873,0x40005876,0x40005877,0x40005878,0x40
        005879,0x4000587a,0x4000587b,0x4000587c,0x40005880,0x40005881,0x40005882,0x40005883,0x40005884,0x40005885,0x400058
        86,0x40005887,0x40005888,0x40005889,0x4000588a,0x4000588b,0x4000588c,0x4000588d,0x4000588e,0x4000588f,0x40005890,0
        x40005891,0x40005892,0x40005893,0x40005894,0x40005895,0x40005896,0x40005897,0x40005898,0x40005899,0x4000589a,0x400
        0589b,0x4000589c,0x4000589d,0x4000589e,0x4000589f,0x40005900,0x40005908,0x40005901,0x40005909,

118
        0x40005902,0x4000590a,0x40005903,0x4000590b,0x40005904,0x4000590c,0x40005905,0x4000590d,0x40005906,0x4000590e,0x40
        005a00,0x40005a02,0x40005a03,0x40005a04,0x40005a06,0x40005a07,0x40005a08,0x40005a0a,0x40005a0b,0x40005a0c,0x40005a
        10,0x40005a12,0x40005a13,0x40005a14,0x40005a16,0x40005a17,0x40005a18,0x40005a1a,0x40005a1b,0x40005a1c,0x40005a20,0
        x40005a22,0x40005a23,0x40005a24,0x40005a26,0x40005a27,0x40005a28,0x40005a2a,0x40005a2b,0x40005a2c,0x40005a30,0x400
        05a32,0x40005a33,0x40005a34,0x40005a36,0x40005a37,0x40005a38,0x40005a3a,0x40005a3b,0x40005a3c,
```

119

0x40005a80,0x40005a82,0x40005a83,0x40005a84,0x40005a87,0x40005a88,0x40005a8a,0x40005a8b,0x40005a8c,0x40005a8f,0x40005a90,0x40005a92,0x40005a93,0x40005a94,0x40005a97,0x40005a98,0x40005a9a,0x40005a9b,0x40005a9c,0x40005a9f,0x40005ac0,0x40005ac2,0x40005ac3,0x40005ac4,0x40005ac6,0x40005ac7,0x40005ac8,0x40005aca,0x40005acb,0x40005acc,0x40005ace,0x40005acf,0x40005ad0,0x40005ad2,0x40005ad3,0x40005ad4,0x40005ad6,0x40005ad7,0x40005ad8,0x40005ada,0x40005adb,0x40005adc,0x40005ade,0x40005adf,0x40005b00,0x40005b02,0x40005b03,0x40005b04,0x40005b06,0x40005b07,

120

0x40005b20,0x40005b22,0x40005b23,0x40005b24,0x40005b26,0x40005b27,0x40005b28,0x40005b2a,0x40005b2b,0x40005b2c,0x40005b2e,0x40005b2f,0x40005b40,0x40005b41,0x40005b42,0x40005b43,0x40005b44,0x40005b45,0x40005b46,0x40005b47,0x40005b50,0x40005b51,0x40005b52,0x40005b53,0x40005b54,0x40005b56,0x40005b58,0x40005b5a,0x40005b5b,0x40005b5c,0x40005b5d,0x40005b5e,0x40005b5f,0x40005b60,0x40005b61,0x40005b80,0x40005b83,0x40005b81,0x40005b82,0x40005b88,0x40005b89,0x40005b90,0x40005b91,0x40005b92,0x40005b93,0x40005b94,0x40005b96,0x40005b97,0x40005b98,0x40005b99,

121

0x40005b9a,0x40005b9b,0x40005b9c,0x40005ba0,0x40005ba2,0x40005ba1,0x40005ba3,0x40005ba8,0x40006400,0x40006401,0x40006402,0x40006403,0x40006404,0x40006504,0x40006405,0x40006505,0x40006406,0x40006506,0x40006407,0x40006507,0x40006408,0x40006508,0x40006409,0x40006509,0x4000640a,0x4000650a,0x4000640b,0x4000650b,0x4000640c,0x4000640d,0x4000640e,0x4000640f,0x40006410,0x40006411,0x40006412,0x40006413,0x40006414,0x40006514,0x40006415,0x40006515,0x40006416,0x40006516,0x40006417,0x40006517,0x40006418,0x40006518,0x40006419,0x40006519,0x4000641a,0x4000651a,

122

0x4000641b,0x4000651b,0x4000641c,0x4000641d,0x4000641e,0x4000641f,0x40006420,0x40006421,0x40006422,0x40006423,0x40006424,0x40006524,0x40006425,0x40006525,0x40006426,0x40006526,0x40006427,0x40006527,0x40006428,0x40006528,0x40006429,0x40006529,0x4000642a,0x4000652a,0x4000642b,0x4000652b,0x4000642c,0x4000642d,0x4000642e,0x4000642f,0x40006430,0x40006431,0x40006432,0x40006433,0x40006434,0x40006534,0x40006435,0x40006535,0x40006436,0x40006536,0x40006437,0x40006537,0x40006438,0x40006538,0x40006439,0x40006539,0x4000643a,0x4000653a,0x4000643b,0x4000653b,

123

0x4000643c,0x4000643d,0x4000643e,0x4000643f,0x40006440,0x40006441,0x40006442,0x40006443,0x40006444,0x40006544,0x40006445,0x40006545,0x40006446,0x40006546,0x40006447,0x40006547,0x40006448,0x40006548,0x40006449,0x40006549,0x4000644a,0x4000654a,0x4000644b,0x4000654b,0x4000644c,0x4000644d,0x4000644e,0x4000644f,0x40006450,0x40006451,0x40006452,0x40006453,0x40006454,0x40006554,0x40006455,0x40006555,0x40006456,0x40006556,0x40006457,0x40006557,0x40006458,0x40006558,0x40006459,0x40006559,0x4000645a,0x4000655a,0x4000645b,0x4000655b,0x4000645c,0x4000645d,

124

0x4000645e,0x4000645f,0x40006460,0x40006461,0x40006462,0x40006463,0x40006464,0x40006564,0x40006465,0x40006565,0x40006466,0x40006566,0x40006467,0x40006567,0x40006468,0x40006568,0x40006469,0x40006569,0x4000646a,0x4000656a,0x4000646b,0x4000656b,0x4000646c,0x4000646d,0x4000646e,0x4000646f,0x40006470,0x40006471,0x40006472,0x40006473,0x40006474,0x40006574,0x40006475,0x40006575,0x40006476,0x40006576,0x40006477,0x40006577,0x40006478,0x40006578,0x40006479,0x40006579,0x4000647a,0x4000657a,0x4000647b,0x4000657b,0x4000647c,0x4000647d,0x4000647e,0x4000647f,

125

0x40006480,0x40006481,0x40006482,0x40006483,0x40006484,0x40006584,0x40006485,0x40006585,0x40006486,0x40006586,0x40006487,0x40006587,0x40006488,0x40006588,0x40006489,0x40006589,0x4000648a,0x4000658a,0x4000648b,0x4000658b,0x4000648c,0x4000648d,0x4000648e,0x4000648f,0x40006490,0x40006491,0x40006492,0x40006493,0x40006494,0x40006594,0x40006495,0x40006595,0x40006496,0x40006596,0x40006497,0x40006597,0x40006498,0x40006598,0x40006499,0x40006599,0x4000649a,0x40006659a,0x4000649b,0x4000659b,0x4000649c,0x4000649d,0x4000649e,0x4000649f,0x400064a0,0x400064a1,

126

0x400064a2,0x400064a3,0x400064a4,0x400065a4,0x400064a5,0x400065a5,0x400064a6,0x400065a6,0x400064a7,0x400065a7,0x400064a8,0x400065a8,0x400064a9,0x400065a9,0x400064aa,0x400065aa,0x400064ab,0x400065ab,0x400064ac,0x400064ad,0x400064ae,0x400064af,0x40006800,0x40006802,0x40006804,0x40006806,0x40006808,0x40006a08,0x40006a0a,0x4000680a,0x4000680c,0x40006a0c,0x4000680e,0x40006a0e,0x40006810,0x40006a10,0x40006812,0x40006a12,0x40006814,0x40006a14,0x40006816,0x40006a16,0x40006818,0x4000681a,0x4000681c,0x40006820,0x40006822,0x40006824,0x40006826,0x40006828,

127

0x40006a28,0x4000682a,0x40006a2a,0x4000682c,0x40006a2c,0x4000682e,0x40006a2e,0x40006830,0x40006a30,0x40006832,0x40006a32,0x40006834,0x40006a34,0x40006836,0x40006a36,0x40006838,0x4000683a,0x4000683c,0x40006840,0x40006842,0x40006844,0x40006846,0x40006848,0x40006a48,0x4000684a,0x40006a4a,0x4000684c,0x40006a4c,0x4000684e,0x40006a4e,0x40006850,0x40006a50,0x40006852,0x40006a52,0x40006854,0x40006a54,0x40006856,0x40006a56,0x40006858,0x4000685a,0x4000685c,0x40006860,0x40006862,0x40006864,0x40006866,0x40006868,0x40006a68,0x4000686a,0x40006a6a,0x4000686c,

128

0x40006a6c,0x4000686e,0x40006a6e,0x40006870,0x40006a70,0x40006872,0x40006a72,0x40006874,0x40006a74,0x40006876,0x40006a76,0x40006878,0x4000687a,0x4000687c,0x40006880,0x40006882,0x40006884,0x40006886,0x40006888,0x40006a88,0x400068a,0x40006a8a,0x4000688c,0x40006a8c,0x4000688e,0x40006a8e,0x40006890,0x40006a90,0x40006892,0x40006a92,0x40006894,0x40006a94,0x40006896,0x40006a96,0x40006898,0x4000689a,0x4000689c,0x400068a0,0x400068a2,0x400068a4,0x400068a6,0x400068a8,0x40006aa8,0x400068aa,0x40006aaa,0x400068ac,0x40006aac,0x400068ae,0x40006aae,0x400068b0,

129

0x40006ab0,0x400068b2,0x40006ab2,0x400068b4,0x40006ab4,0x400068b6,0x40006ab6,0x400068b8,0x400068ba,0x400068bc,0x400068c0,0x400068c2,0x400068c4,0x400068c6,0x400068c8,0x40006ac8,0x400068ca,0x40006aca,0x400068cc,0x40006acc,0x400068ce,0x40006ace,0x400068d0,0x40006ad0,0x400068d2,0x40006ad2,0x400068d4,0x40006ad4,0x400068d6,0x40006ad6,0x400068d8,0x400068da,0x400068dc,0x400068e0,0x400068e2,0x400068e4,0x400068e6,0x400068e8,0x40006ae8,0x400068ea,0x40006aea,0x400068ec,0x40006aec,0x400068ee,0x40006aee,0x400068f0,0x40006af0,0x400068f2,0x40006af2,0x400068f4,

130

0x40006af4,0x400068f6,0x40006af6,0x400068f8,0x400068fa,0x400068fc,0x40006900,0x40006902,0x40006904,0x40006906,0x40006908,0x40006b08,0x4000690a,0x40006b0a,0x4000690c,0x40006b0c,0x4000690e,0x40006b0e,0x40006910,0x40006b10,0x40006912,0x40006b12,0x40006914,0x40006b14,0x40006916,0x40006b16,0x40006918,0x4000691a,0x4000691c,0x40006920,0x40006922,0x40006924,0x40006926,0x40006928,0x40006b28,0x4000692a,0x40006b2a,0x4000692c,0x40006b2c,0x4000692e,0x40006b2e,0x40006930,0x40006b30,0x40006932,0x40006b32,0x40006934,0x40006b34,0x40006936,0x40006b36,0x40006938,

131

0x4000693a,0x4000693c,0x40006940,0x40006942,0x40006944,0x40006946,0x40006948,0x40006b48,0x4000694a,0x40006b4a,0x40006b4c,0x4000694e,0x40006b4e,0x40006950,0x40006b50,0x40006952,0x40006b52,0x40006954,0x40006b54,0x40006956,0x40006b56,0x40006958,0x4000695a,0x4000695c,0x40006800,0x40006802,0x40006804,0x40006806,0x40006808,0x40006a08,0x4000680a,0x40006a0a,0x4000680c,0x40006a0c,0x4000680e,0x40006a0e,0x40006810,0x40006a10,0x40006812,0x40006a12,0x40006814,0x40006a14,0x40006816,0x40006a16,0x40006818,0x4000681a,0x4000681c,0x4000681e,0x40006840,

132

0x40006842,0x40006844,0x40006846,0x40006848,0x40006a48,0x4000684a,0x40006a4a,0x4000684c,0x40006a4c,0x4000684e,0x40006a4e,0x40006850,0x40006a50,0x40006852,0x40006a52,0x40006854,0x40006a54,0x40006856,0x40006a56,0x40006858,0x400685a,0x4000685c,0x4000685e,0x40006880,0x40006882,0x40006884,0x40006886,0x40006888,0x40006a88,0x4000688a,0x40006a8a,0x4000688c,0x40006a8c,0x4000688e,0x40006a8e,0x40006890,0x40006a90,0x40006892,0x40006a92,0x40006894,0x40006a94,0x40006896,0x40006a96,0x40006898,0x4000689a,0x4000689c,0x4000689e,0x400068c0,0x400068c2,0x400068c4,

133

0x400068c6,0x400068c8,0x40006ac8,0x400068ca,0x40006aca,0x400068cc,0x40006acc,0x400068ce,0x40006ace,0x400068d0,0x40006ad0,0x400068d2,0x40006ad2,0x400068d4,0x40006ad4,0x400068d6,0x40006ad6,0x400068d8,0x400068da,0x400068dc,0x400068de,0x40006900,0x40006902,0x40006904,0x40006906,0x40006908,0x40006b08,0x4000690a,0x40006b0a,0x4000690c,0x40006b0c,0x4000690e,0x40006b0e,0x40006910,0x40006b10,0x40006912,0x40006b12,0x40006914,0x40006b14,0x40006916,0x40006b16,0x40006918,0x4000691a,0x4000691c,0x4000691e,0x40006940,0x40006942,0x40006944,0x40006946,0x40006948,

134

0x40006b48,0x4000694a,0x40006b4a,0x4000694c,0x40006b4c,0x4000694e,0x40006b4e,0x40006950,0x40006b50,0x40006952,0x40006b52,0x40006954,0x40006b54,0x40006956,0x40006b56,0x40006958,0x4000695a,0x4000695c,0x4000695e,0x40007000,0x40007004,0x40007008,0x40007014,0x40007034,0x40007054,0x40007074,0x40007094,0x400070b4,0x400070d4,0x400070f4,0x40007114,0x40007134,0x40007154,0x40007174,0x40007024,0x40007044,0x40007064,0x40007084,0x400070a4,0x400070c4,0x400070e4,0x40007104,0x40007124,0x40007144,0x40007164,0x40007184,0x40007018,0x40007028,0x40007038,0x40007048,

135

0x40007058,0x40007068,0x40007078,0x40007088,0x40007098,0x400070a8,0x400070b8,0x400070c8,0x400070d8,0x400070e8,0x400070f8,0x40007108,0x40007118,0x40007128,0x40007138,0x40007148,0x40007158,0x40007168,0x40007178,0x40007188,0x40007600,0x40007608,0x40007610,0x40007618,0x40007620,0x40007628,0x40007630,0x40007638,0x40007640,0x40007648,0x40007650,0x40007658,0x40007660,0x40007668,0x40007670,0x40007678,0x40007680,0x40007688,0x40007690,0x40007698,0x400076a0,0x400076a8,0x400076b0,0x400076b8,0x40007604,0x4000760c,0x40007614,0x4000761c,0x40007624,0x4000762c,

136

0x40007634,0x4000763c,0x40007644,0x4000764c,0x40007654,0x4000765c,0x40007664,0x4000766c,0x40007674,0x4000767c,0x40007684,0x4000768c,0x40007694,0x4000769c,0x400076a4,0x400076ac,0x400076b4,0x400076bc,0x40007800,0x40007808,0x40007810,0x40007818,0x40007820,0x40007828,0x40007830,0x40007838,0x40007840,0x40007848,0x40007850,0x40007858,0x40007860,0x40007868,0x40007870,0x40007878,0x40007880,0x40007888,0x40007890,0x40007898,0x400078a0,0x400078a8,0x400078b0,0x400078b8,0x400078c0,0x400078c8,0x400078d0,0x400078d8,0x400078e0,0x400078e8,0x400078f0,0x400078f8,

137

0x40007900,0x40007908,0x40007910,0x40007918,0x40007920,0x40007928,0x40007930,0x40007938,0x40007940,0x40007948,0x40007950,0x40007958,0x40007960,0x40007968,0x40007970,0x40007978,0x40007980,0x40007988,0x40007990,0x40007998,0x400079a0,0x400079a8,0x400079b0,0x400079b8,0x400079c0,0x400079c8,0x400079d0,0x400079d8,0x400079e0,0x400079e8,0x400079f0,0x400079f8,0x40007a00,0x40007a08,0x40007a10,0x40007a18,0x40007a20,0x40007a28,0x40007a30,0x40007a38,0x40007a40,0x40007a48,0x40007a50,0x40007a58,0x40007a60,0x40007a68,0x40007a70,0x40007a78,0x40007a80,0x40007a88,

138

0x40007a90,0x40007a98,0x40007aa0,0x40007aa8,0x40007ab0,0x40007ab8,0x40007ac0,0x40007ac8,0x40007ad0,0x40007ad8,0x40007ae0,0x40007ae8,0x40007af0,0x40007af8,0x40007b00,0x40007b08,0x40007b10,0x40007b18,0x40007b20,0x40007b28,0x40007b30,0x40007b38,0x40007b40,0x40007b48,0x40007b50,0x40007b58,0x40007b60,0x40007b68,0x40007b70,0x40007b78,0x40007b80,0x40007b88,0x40007b90,0x40007b98,0x40007ba0,0x40007ba8,0x40007bb0,0x40007bb8,0x40007bc0,0x40007bc8,0x40007bd0,0x40007bd8,0x40007be0,0x40007be8,0x40007bf0,0x40007bf8,0x40007804,0x4000780c,0x40007814,0x4000781c,

139

0x40007824,0x4000782c,0x40007834,0x4000783c,0x40007844,0x4000784c,0x40007854,0x4000785c,0x40007864,0x4000786c,0x40007874,0x4000787c,0x40007884,0x4000788c,0x40007894,0x4000789c,0x400078a4,0x400078ac,0x400078b4,0x400078bc,0x400078c4,0x400078cc,0x400078d4,0x400078dc,0x400078e4,0x400078ec,0x400078f4,0x400078fc,0x40007904,0x4000790c,0x40007914,0x4000791c,0x40007924,0x4000792c,0x40007934,0x4000793c,0x40007944,0x4000794c,0x40007954,0x4000795c,0x40007964,0x40007960796c,0x40007974,0x4000797c,0x40007984,0x4000798c,0x40007994,0x4000799c,0x400079a4,0x400079ac,

140

0x400079b4,0x400079bc,0x400079c4,0x400079cc,0x400079d4,0x400079dc,0x400079e4,0x400079ec,0x400079f4,0x400079fc,0x40007a04,0x40007a0c,0x40007a14,0x40007a1c,0x40007a24,0x40007a2c,0x40007a34,0x40007a3c,0x40007a44,0x40007a4c,0x40007a54,0x40007a5c,0x40007a64,0x40007a6c,0x40007a74,0x40007a7c,0x40007a84,0x40007a8c,0x40007a94,0x40007a9c,0x40007aa4,0x40007aac,0x40007ab4,0x40007abc,0x40007ac4,0x40007acc,0x40007ad4,0x40007adc,0x40007ae4,0x40007aec,0x40007af4,0x40007afc,0x40007b04,0x40007b0c,0x40007b14,0x40007b1c,0x40007b24,0x40007b2c,0x40007b34,0x40007b3c,

141

0x40007b44,0x40007b4c,0x40007b54,0x40007b5c,0x40007b64,0x40007b6c,0x40007b74,0x40007b7c,0x40007b84,0x40007b8c,0x40007b94,0x40007b9c,0x40007ba4,0x40007bac,0x40007bb4,0x40007bbc,0x40007bc4,0x40007bcc,0x40007bd4,0x40007bdc,0x40007be4,0x40007bec,0x40007bf4,0x40007bfc,0x40010000,0x40010004,0x40010008,0x4001000c,0x40010010,0x40010014,0x40010018,0x4001001c,0x40010020,0x40010024,0x40010028,0x4001002c,0x40010080,0x40010084,0x40010088,0x4001008c,0x40010090,0x40010094,0x40010098,0x4001009c,0x400100a0,0x400100a4,0x400100a8,0x400100ac,0x40010200,0x40010204,

142

0x40010208,0x4001020c,0x40010210,0x40010214,0x40010218,0x4001021c,0x40010220,0x40010224,0x40010228,0x4001022c,0x40010280,0x40010284,0x40010288,0x4001028c,0x40010290,0x40010294,0x40010298,0x4001029c,0x400102a0,0x400102a4,0x400102a8,0x400102ac,0x40010400,0x40010404,0x40010408,0x4001040c,0x40010410,0x40010414,0x40010418,0x4001041c,0x40010420,0x40010424,0x40010428,0x4001042c,0x40010480,0x40010484,0x40010488,0x4001048c,0x40010490,0x40010494,0x40010498,0x4001049c,0x400104a0,0x400104a4,0x400104a8,0x400104ac,0x40010600,0x40010604,0x40010608,0x4001060c,

143

0x40010610,0x40010614,0x40010618,0x4001061c,0x40010620,0x40010624,0x40010628,0x4001062c,0x40010680,0x40010684,0x40010688,0x4001068c,0x40010690,0x40010694,0x40010698,0x4001069c,0x400106a0,0x400106a4,0x400106a8,0x400106ac,0x40010800,0x40010804,0x40010808,0x4001080c,0x40010810,0x40010814,0x40010818,0x4001081c,0x40010820,0x40010824,0x40010828,0x4001082c,0x40010880,0x40010884,0x40010888,0x4001088c,0x40010890,0x40010894,0x40010898,0x4001089c,0x400108a0,0x400108a4,0x400108a8,0x400108ac,0x40010a00,0x40010a04,0x40010a08,0x40010a0c,0x40010a10,0x40010a14,

144

0x40010a18,0x40010a1c,0x40010a20,0x40010a24,0x40010a28,0x40010a2c,0x40010a80,0x40010a84,0x40010a88,0x40010a8c,0x40010a90,0x40010a94,0x40010a98,0x40010a9c,0x40010aa0,0x40010aa4,0x40010aa8,0x40010aac,0x40010c00,0x40010c04,0x40010c08,0x40010c0c,0x40010c10,0x40010c14,0x40010c18,0x40010c1c,0x40010c20,0x40010c24,0x40010c28,0x40010c2c,0x40010c80,0x40010c84,0x40010c88,0x40010c8c,0x40010c90,0x40010c94,0x40010c98,0x40010c9c,0x40010ca0,0x40010ca4,0x40010ca8,0x40010cac,0x40010e00,0x40010e04,0x40010e08,0x40010e0c,0x40010e10,0x40010e14,0x40010e18,0x40010e1c,

145

0x40010e20,0x40010e24,0x40010e28,0x40010e2c,0x40010e80,0x40010e84,0x40010e88,0x40010e8c,0x40010e90,0x40010e94,0x40010e98,0x40010e9c,0x40010ea0,0x40010ea4,0x40010ea8,0x40010eac,0x40011400,0x40011404,0x40011408,0x4001140c,0x40011410,0x40011414,0x40011418,0x4001141c,0x40011420,0x40011424,0x40011428,0x4001142c,0x40011480,0x40011484,0x40011488,0x4001148c,0x40011490,0x40011494,0x40011498,0x4001149c,0x400114a0,0x400114a4,0x400114a8,0x400114ac,0x40011600,0x40011604,0x40011608,0x4001160c,0x40011610,0x40011614,0x40011618,0x4001161c,0x40011620,0x40011624,

146

0x40011628,0x4001162c,0x40011680,0x40011684,0x40011688,0x4001168c,0x40011690,0x40011694,0x40011698,0x4001169c,0x40
0116a0,0x400116a4,0x400116a8,0x400116ac,0x40011800,0x40011804,0x40011808,0x4001180c,0x40011810,0x40011814,0x400118
18,0x4001181c,0x40011820,0x40011824,0x40011828,0x4001182c,0x40011880,0x40011884,0x40011888,0x4001188c,0x40011890,0
x40011894,0x40011898,0x4001189c,0x400118a0,0x400118a4,0x400118a8,0x400118ac,0x40011a00,0x40011a04,0x40011a08,0x400
11a0c,0x40011a10,0x40011a14,0x40011a18,0x40011a1c,0x40011a20,0x40011a24,0x40011a28,0x40011a2c,

147

0x40011a80,0x40011a84,0x40011a88,0x40011a8c,0x40011a90,0x40011a94,0x40011a98,0x40011a9c,0x40011aa0,0x40011aa4,0x40
011aa8,0x40011aac,0x40010030,0x40010032,0x40010034,0x40010036,0x400100b0,0x400100b2,0x400100b4,0x400100b6,0x400102
30,0x40010232,0x40010234,0x40010236,0x400102b0,0x400102b2,0x400102b4,0x400102b6,0x40010430,0x40010432,0x40010434,0
x40010436,0x400104b0,0x400104b2,0x400104b4,0x400104b6,0x40010630,0x40010632,0x40010634,0x40010636,0x400106b0,0x400
106b2,0x400106b4,0x400106b6,0x40010830,0x40010832,0x40010834,0x40010836,0x400108b0,0x400108b2,

148

0x400108b4,0x400108b6,0x40010a30,0x40010a32,0x40010a34,0x40010a36,0x40010ab0,0x40010ab2,0x40010ab4,0x40010ab6,0x40
010c30,0x40010c32,0x40010c34,0x40010c36,0x40010cb0,0x40010cb2,0x40010cb4,0x40010cb6,0x40010e30,0x40010e32,0x40010e
34,0x40010e36,0x40010eb0,0x40010eb2,0x40010eb4,0x40010eb6,0x40011430,0x40011432,0x40011434,0x40011436,0x400114b0,0
x400114b2,0x400114b4,0x400114b6,0x40011630,0x40011632,0x40011634,0x40011636,0x400116b0,0x400116b2,0x400116b4,0x400
116b6,0x40011830,0x40011832,0x40011834,0x40011836,0x400118b0,0x400118b2,0x400118b4,0x400118b6,

149

0x40011a30,0x40011a32,0x40011a34,0x40011a36,0x40011ab0,0x40011ab2,0x40011ab4,0x40011ab6,0x40010038,0x400100b8,0x40
010238,0x400102b8,0x40010438,0x400104b8,0x40010638,0x400106b8,0x40010838,0x400108b8,0x40010a38,0x40010ab8,0x40010c
38,0x40010cb8,0x40010e38,0x40010eb8,0x40011438,0x400114b8,0x40011638,0x400116b8,0x40011838,0x400118b8,0x40011a38,0
x40011ab8,0x4001003a,0x400100ba,0x4001023a,0x400102ba,0x4001043a,0x400104ba,0x4001063a,0x400106ba,0x4001083a,0x400
108ba,0x40010a3a,0x40010aba,0x40010c3a,0x40010cba,0x40010e3a,0x40010eba,0x4001143a,0x400114ba,

150

0x4001163a,0x400116ba,0x4001183a,0x400118ba,0x40011a3a,0x40011aba,0x4001003c,0x400100bc,0x4001023c,0x400102bc,0x40
01043c,0x400104bc,0x4001063c,0x400106bc,0x4001083c,0x400108bc,0x40010a3c,0x40010abc,0x40010c3c,0x40010cbc,0x40010e
3c,0x40010ebc,0x4001143c,0x400114bc,0x4001163c,0x400116bc,0x4001183c,0x400118bc,0x40011a3c,0x40011abc,0x4001003e,0
x400100be,0x4001023e,0x400102be,0x4001043e,0x400104be,0x4001063e,0x400106be,0x4001083e,0x400108be,0x40010a3e,0x400
10abe,0x40010c3e,0x40010cbe,0x40010e3e,0x40010ebe,0x4001143e,0x400114be,0x4001163e,0x400116be,

151

0x4001183e,0x400118be,0x40011a3e,0x40011abe,0x40010040,0x40010240,0x40010440,0x40010640,0x40010840,0x40010a40,0x40
010c40,0x40010e40,0x40011440,0x40011640,0x40011840,0x40011a40,0x400100c0,0x400102c0,0x400104c0,0x400106c0,0x400108
c0,0x40010ac0,0x40010cc0,0x40010ec0,0x400114c0,0x400116c0,0x400118c0,0x40011ac0,0x40010041,0x40010241,0x40010441,0
x40010641,0x40010841,0x40010a41,0x40010c41,0x40010e41,0x40011441,0x40011641,0x40011841,0x40011a41,0x400100c1,0x400
102c1,0x400104c1,0x400106c1,0x400108c1,0x40010ac1,0x40010cc1,0x40010ec1,0x400114c1,0x400116c1,

152

0x400118c1,0x40011ac1,0x40010042,0x40010242,0x40010442,0x40010642,0x40010842,0x40010a42,0x40010c42,0x40010e42,0x40
011442,0x40011642,0x40011842,0x40011a42,0x400100c2,0x400102c2,0x400104c2,0x400106c2,0x400108c2,0x40010ac2,0x40010c
c2,0x40010ec2,0x400114c2,0x400116c2,0x400118c2,0x40011ac2,0x40010043,0x40010243,0x40010443,0x40010643,0x40010843,0
x40010a43,0x40010c43,0x40010e43,0x40011443,0x40011643,0x40011843,0x40011a43,0x400100c3,0x400102c3,0x400104c3,0x400
106c3,0x400108c3,0x40010ac3,0x40010cc3,0x40010ec3,0x400114c3,0x400116c3,0x400118c3,0x40011ac3,

153

0x40010044,0x40010244,0x40010444,0x40010644,0x40010844,0x40010a44,0x40010c44,0x40010e44,0x40011444,0x40011644,0x40
011844,0x40011a44,0x400100c4,0x400102c4,0x400104c4,0x400106c4,0x400108c4,0x40010ac4,0x40010cc4,0x40010ec4,0x400114
c4,0x400116c4,0x400118c4,0x40011ac4,0x40010045,0x40010245,0x40010445,0x40010645,0x40010845,0x40010a45,0x40010c45,0
x40010e45,0x40011445,0x40011645,0x40011845,0x40011a45,0x400100c5,0x400102c5,0x400104c5,0x400106c5,0x400108c5,0x400
10ac5,0x40010cc5,0x40010ec5,0x400114c5,0x400116c5,0x400118c5,0x40011ac5,0x40010046,0x40010246,

154

0x40010446,0x40010646,0x40010846,0x40010a46,0x40010c46,0x40010e46,0x40011446,0x40011646,0x40011846,0x40011a46,0x40
0100c6,0x400102c6,0x400104c6,0x400106c6,0x400108c6,0x40010ac6,0x40010cc6,0x40010ec6,0x400114c6,0x400116c6,0x400118
c6,0x40011ac6,0x40010047,0x40010247,0x40010447,0x40010647,0x40010847,0x40010a47,0x40010c47,0x40010e47,0x40011447,0
x40011647,0x40011847,0x40011a47,0x400100c7,0x400102c7,0x400104c7,0x400106c7,0x400108c7,0x40010ac7,0x40010cc7,0x400
10ec7,0x400114c7,0x400116c7,0x400118c7,0x40011ac7,0x40010048,0x40010248,0x40010448,0x40010648,

155

0x40010848,0x40010a48,0x40010c48,0x40010e48,0x40011448,0x40011648,0x40011848,0x40011a48,0x400100c8,0x400102c8,0x40
0104c8,0x400106c8,0x400108c8,0x40010ac8,0x40010cc8,0x40010ec8,0x400114c8,0x400116c8,0x400118c8,0x40011ac8,0x400100
49,0x40010249,0x40010449,0x40010649,0x40010849,0x40010a49,0x40010c49,0x40010e49,0x40011449,0x40011649,0x40011849,0
x40011a49,0x400100c9,0x400102c9,0x400104c9,0x400106c9,0x400108c9,0x40010ac9,0x40010cc9,0x40010ec9,0x400114c9,0x400
116c9,0x400118c9,0x40011ac9,0x4001004a,0x4001024a,0x4001044a,0x4001064a,0x4001084a,0x40010a4a,

156

0x40010c4a,0x40010e4a,0x4001144a,0x4001164a,0x4001184a,0x40011a4a,0x400100ca,0x400102ca,0x400104ca,0x400106ca,0x40
0108ca,0x40010aca,0x40010cca,0x40010eca,0x400114ca,0x400116ca,0x400118ca,0x40011aca,0x4001004b,0x4001024b,0x400104
4b,0x4001064b,0x4001084b,0x40010a4b,0x40010c4b,0x40010e4b,0x4001144b,0x4001164b,0x4001184b,0x40011a4b,0x400100cb,0
x400102cb,0x400104cb,0x400106cb,0x400108cb,0x40010acb,0x40010ccb,0x40010ecb,0x400114cb,0x400116cb,0x400118cb,0x400
11acb,0x4001004c,0x4001024c,0x4001044c,0x4001064c,0x4001084c,0x40010a4c,0x40010c4c,0x40010e4c,

157

0x4001144c,0x4001164c,0x4001184c,0x40011a4c,0x400100cc,0x400102cc,0x400104cc,0x400106cc,0x400108cc,0x40010acc,0x40
010ccc,0x40010ecc,0x400114cc,0x400116cc,0x400118cc,0x40011acc,0x4001004d,0x4001024d,0x4001044d,0x4001064d,0x400108
4d,0x40010a4d,0x40010c4d,0x40010e4d,0x4001144d,0x4001164d,0x4001184d,0x40011a4d,0x400100cd,0x400102cd,0x400104cd,0
x400106cd,0x400108cd,0x40010acd,0x40010ccd,0x40010ecd,0x400114cd,0x400116cd,0x400118cd,0x40011acd,0x4001004e,0x400
1024e,0x4001044e,0x4001064e,0x4001084e,0x40010a4e,0x40010c4e,0x40010e4e,0x4001144e,0x4001164e,

158

0x4001184e,0x40011a4e,0x400100ce,0x400102ce,0x400104ce,0x400106ce,0x400108ce,0x40010ace,0x40010cce,0x40010ece,0x40
0114ce,0x400116ce,0x400118ce,0x40011ace,0x4001004f,0x4001024f,0x4001044f,0x4001064f,0x4001084f,0x40010a4f,0x40010c
4f,0x40010e4f,0x4001144f,0x4001164f,0x4001184f,0x40011a4f,0x400100cf,0x400102cf,0x400104cf,0x400106cf,0x400108cf,0
x40010acf,0x40010ccf,0x40010ecf,0x400114cf,0x400116cf,0x400118cf,0x40011acf,0x40010050,0x40010250,0x40010450,0x400
10650,0x40010850,0x40010a50,0x40010c50,0x40010e50,0x40011450,0x40011650,0x40011850,0x40011a50,

159

0x400100d0,0x400102d0,0x400104d0,0x400106d0,0x400108d0,0x40010ad0,0x40010cd0,0x40010ed0,0x400114d0,0x400116d0,0x40
0118d0,0x40011ad0,0x40010051,0x40010251,0x40010451,0x40010651,0x40010851,0x40010a51,0x40010c51,0x40010e51,0x400114
51,0x40011651,0x40011851,0x40011a51,0x400100d1,0x400102d1,0x400104d1,0x400106d1,0x400108d1,0x40010ad1,0x40010cd1,0
x40010ed1,0x400114d1,0x400116d1,0x400118d1,0x40011ad1,0x40010052,0x40010252,0x40010452,0x40010652,0x40010852,0x400
10a52,0x40010c52,0x40010e52,0x40011452,0x40011652,0x40011852,0x40011a52,0x400100d2,0x400102d2,

160

0x400104d2,0x400106d2,0x400108d2,0x40010ad2,0x40010cd2,0x40010ed2,0x400114d2,0x400116d2,0x400118d2,0x40011ad2,0x40
010053,0x40010253,0x40010453,0x40010653,0x40010853,0x40010a53,0x40010c53,0x40010e53,0x40011453,0x40011653,0x400118
53,0x40011a53,0x400100d3,0x400102d3,0x400104d3,0x400106d3,0x400108d3,0x40010ad3,0x40010cd3,0x40010ed3,0x400114d3,0
x400116d3,0x400118d3,0x40011ad3,0x40010054,0x40010254,0x40010454,0x40010654,0x40010854,0x40010a54,0x40010c54,0x400
10e54,0x40011454,0x40011654,0x40011854,0x40011a54,0x400100d4,0x400102d4,0x400104d4,0x400106d4,

161

0x400108d4,0x40010ad4,0x40010cd4,0x40010ed4,0x400114d4,0x400116d4,0x400118d4,0x40011ad4,0x40010055,0x40010255,0x40
010455,0x40010655,0x40010855,0x40010a55,0x40010c55,0x40010e55,0x40011455,0x40011655,0x40011855,0x40011a55,0x400100
d5,0x400102d5,0x400104d5,0x400106d5,0x400108d5,0x40010ad5,0x40010cd5,0x40010ed5,0x400114d5,0x400116d5,0x400118d5,0
x40011ad5,0x40010056,0x40010256,0x40010456,0x40010656,0x40010856,0x40010a56,0x40010c56,0x40010e56,0x40011456,0x400
11656,0x40011856,0x40011a56,0x400100d6,0x400102d6,0x400104d6,0x400106d6,0x400108d6,0x40010ad6,

162

0x40010cd6,0x40010ed6,0x400114d6,0x400116d6,0x400118d6,0x40011ad6,0x40010057,0x40010257,0x40010457,0x40010657,0x40
010857,0x40010a57,0x40010c57,0x40010e57,0x40011457,0x40011657,0x40011857,0x40011a57,0x400100d7,0x400102d7,0x400104
d7,0x400106d7,0x400108d7,0x40010ad7,0x40010cd7,0x40010ed7,0x400114d7,0x400116d7,0x400118d7,0x40011ad7,0x40010058,0
x40010258,0x40010458,0x40010658,0x40010858,0x40010a58,0x40010c58,0x40010e58,0x40011458,0x40011658,0x40011858,0x400
11a58,0x400100d8,0x400102d8,0x400104d8,0x400106d8,0x400108d8,0x40010ad8,0x40010cd8,0x40010ed8,

163

0x400114d8,0x400116d8,0x400118d8,0x40011ad8,0x40010059,0x40010259,0x40010459,0x40010659,0x40010859,0x40010a59,0x40
010c59,0x40010e59,0x40011459,0x40011659,0x40011859,0x40011a59,0x400100d9,0x400102d9,0x400104d9,0x400106d9,0x400108
d9,0x40010ad9,0x40010cd9,0x40010ed9,0x400114d9,0x400116d9,0x400118d9,0x40011ad9,0x4001005a,0x4001025a,0x4001045a,0
x4001065a,0x4001085a,0x40010a5a,0x40010c5a,0x40010e5a,0x4001145a,0x4001165a,0x4001185a,0x40011a5a,0x400100da,0x400
102da,0x400104da,0x400106da,0x400108da,0x40010ada,0x40010cda,0x40010eda,0x400114da,0x400116da,

164

0x400118da,0x40011ada,0x4001005b,0x4001025b,0x4001045b,0x4001065b,0x4001085b,0x40010a5b,0x40010c5b,0x40010e5b,0x40
01145b,0x4001165b,0x4001185b,0x40011a5b,0x400100db,0x400102db,0x400104db,0x400106db,0x400108db,0x40010adb,0x40010c
db,0x40010edb,0x400114db,0x400116db,0x400118db,0x40011adb,0x4001005c,0x4001025c,0x4001045c,0x4001065c,0x4001085c,0
x40010a5c,0x40010c5c,0x40010e5c,0x4001145c,0x4001165c,0x4001185c,0x40011a5c,0x400100dc,0x400102dc,0x400104dc,0x400
106dc,0x400108dc,0x40010adc,0x40010cdc,0x40010edc,0x400114dc,0x400116dc,0x400118dc,0x40011adc,

165

0x4001005d,0x4001025d,0x4001045d,0x4001065d,0x4001085d,0x40010a5d,0x40010c5d,0x40010e5d,0x4001145d,0x4001165d,0x40
01185d,0x40011a5d,0x400100dd,0x400102dd,0x400104dd,0x400106dd,0x400108dd,0x40010add,0x40010cdd,0x40010edd,0x400114
dd,0x400116dd,0x400118dd,0x40011add,0x4001005e,0x4001025e,0x4001045e,0x4001065e,0x4001085e,0x40010a5e,0x40010c5e,0
x40010e5e,0x4001145e,0x4001165e,0x4001185e,0x40011a5e,0x400100de,0x400102de,0x400104de,0x400106de,0x400108de,0x400
10ade,0x40010cde,0x40010ede,0x400114de,0x400116de,0x400118de,0x40011ade,0x4001005f,0x4001025f,

166

0x4001045f,0x4001065f,0x4001085f,0x40010a5f,0x40010c5f,0x40010e5f,0x4001145f,0x4001165f,0x4001185f,0x40011a5f,0x40
0100df,0x400102df,0x400104df,0x400106df,0x400108df,0x40010adf,0x40010cdf,0x40010edf,0x400114df,0x400116df,0x400118
df,0x40011adf,0x40010060,0x40010064,0x40010068,0x4001006c,0x400100e0,0x400100e4,0x400100e8,0x400100ec,0x40010260,0
x40010264,0x40010268,0x4001026c,0x400102e0,0x400102e4,0x400102e8,0x400102ec,0x40010460,0x40010464,0x40010468,0x400
1046c,0x400104e0,0x400104e4,0x400104e8,0x400104ec,0x40010660,0x40010664,0x40010668,0x4001066c,

167

0x400106e0,0x400106e4,0x400106e8,0x400106ec,0x40010860,0x40010062,0x40010066,0x4001006a,0x4001006e,0x400100e2,0x40
0100e6,0x400100ea,0x400100ee,0x40010262,0x40010266,0x4001026a,0x4001026e,0x400102e2,0x400102e6,0x400102ea,0x400102
ee,0x40010462,0x40010466,0x4001046a,0x4001046e,0x400104e2,0x400104e6,0x400104ea,0x400104ee,0x40010662,0x40010666,0
x4001066a,0x4001066e,0x400106e2,0x400106e6,0x400106ea,0x400106ee,0x40010862,0x40010864,0x40010868,0x4001086c,0x400
108e0,0x400108e4,0x400108e8,0x400108ec,0x40010a60,0x40010a64,0x40010a68,0x40010a6c,0x40010ae0,

168

0x40010ae4,0x40010ae8,0x40010aec,0x40010c60,0x40010c64,0x40010c68,0x40010c6c,0x40010ce0,0x40010ce4,0x40010ce8,0x40
010cec,0x40010e60,0x40010e64,0x40010e68,0x40010e6c,0x40010ee0,0x40010ee4,0x40010ee8,0x40010eec,0x40011460,0x400114
64,0x40011468,0x4001146c,0x400114e0,0x40010866,0x4001086a,0x4001086e,0x400108e2,0x400108e6,0x400108ea,0x400108ee,0
x40010a62,0x40010a66,0x40010a6a,0x40010a6e,0x40010ae2,0x40010ae6,0x40010aea,0x40010aee,0x40010c62,0x40010c66,0x400
10c6a,0x40010c6e,0x40010ce2,0x40010ce6,0x40010cea,0x40010cee,0x40010e62,0x40010e66,0x40010e6a,

169

0x40010e6e,0x40010ee2,0x40010ee6,0x40010eea,0x40010eee,0x40011462,0x40011466,0x4001146a,0x4001146e,0x400114e2,0x40
0114e4,0x400114e8,0x400114ec,0x40011660,0x40011664,0x40011668,0x4001166c,0x400116e0,0x400116e4,0x400116e8,0x400116
ec,0x40011860,0x40011864,0x40011868,0x4001186c,0x400118e0,0x400118e4,0x400118e8,0x400118ec,0x40011a60,0x40011a64,0
x40011a68,0x40011a6c,0x40011ae0,0x40011ae4,0x40011ae8,0x40011aec,0x400114e6,0x400114ea,0x400114ee,0x40011662,0x400
11666,0x4001166a,0x4001166e,0x400116e2,0x400116e6,0x400116ea,0x400116ee,0x40011862,0x40011866,

170

0x4001186a,0x4001186e,0x400118e2,0x400118e6,0x400118ea,0x400118ee,0x40011a62,0x40011a66,0x40011a6a,0x40011a6e,0x40
011ae2,0x40011ae6,0x40011aea,0x40011aee,0x40010100,0x40010102,0x40010104,0x40010106,0x40010108,0x4001010a,0x400101
0c,0x4001010e,0x40010110,0x40010112,0x40010114,0x40010116,0x40010118,0x4001011a,0x4001011c,0x4001011e,0x40010120,0
x40010122,0x40010124,0x40010126,0x40010128,0x4001012a,0x4001012c,0x4001012e,0x40010130,0x40010132,0x40010134,0x400
10136,0x40010138,0x4001013a,0x4001013c,0x4001013e,0x40010140,0x40010101,0x40010103,0x40010105,

171

0x40010107,0x40010109,0x4001010b,0x4001010d,0x4001010f,0x40010111,0x40010113,0x40010115,0x40010117,0x40010119,0x40
01011b,0x4001011d,0x4001011f,0x40010121,0x40010123,0x40010125,0x40010127,0x40010129,0x4001012b,0x4001012d,0x400101
2f,0x40010131,0x40010133,0x40010135,0x40010137,0x40010139,0x4001013b,0x4001013d,0x4001013f,0x40010141,0x40010142,0
x40010144,0x40010146,0x40010148,0x4001014a,0x4001014c,0x4001014e,0x40010150,0x40010152,0x40010154,0x40010156,0x400
10158,0x4001015a,0x4001015c,0x4001015e,0x40010160,0x40010162,0x40010164,0x40010166,0x40010168,

172

0x4001016a,0x4001016c,0x4001016e,0x40010170,0x40010172,0x40010174,0x40010176,0x40010178,0x4001017a,0x4001017c,0x40
01017e,0x40010300,0x40010302,0x40010304,0x40010306,0x40010308,0x40010143,0x40010145,0x40010147,0x40010149,0x400101
4b,0x4001014d,0x4001014f,0x40010151,0x40010153,0x40010155,0x40010157,0x40010159,0x4001015b,0x4001015d,0x4001015f,0
x40010161,0x40010163,0x40010165,0x40010167,0x40010169,0x4001016b,0x4001016d,0x4001016f,0x40010171,0x40010173,0x400
10175,0x40010177,0x40010179,0x4001017b,0x4001017d,0x4001017f,0x40010301,0x40010303,0x40010305,

173

0x40010307,0x40010309,0x4001030a,0x4001030c,0x4001030e,0x40010310,0x40010312,0x40010314,0x40010316,0x40010318,0x40
01031a,0x4001031c,0x4001031e,0x40010320,0x40010322,0x40010324,0x40010326,0x40010328,0x4001032a,0x4001032c,0x400103
2e,0x40010330,0x40010332,0x40010334,0x40010336,0x40010338,0x4001033a,0x4001033c,0x4001033e,0x40010340,0x40010342,0
x40010344,0x40010346,0x40010348,0x4001034a,0x4001034c,0x4001034e,0x40010350,0x4001030b,0x4001030d,0x4001030f,0x400
10311,0x40010313,0x40010315,0x40010317,0x40010319,0x4001031b,0x4001031d,0x4001031f,0x40010321,

174

0x40010323,0x40010325,0x40010327,0x40010329,0x4001032b,0x4001032d,0x4001032f,0x40010331,0x40010333,0x40010335,0x40010337,0x40010339,0x4001033b,0x4001033d,0x4001033f,0x40010341,0x40010343,0x40010345,0x40010347,0x40010349,0x4001034b,0x4001034d,0x4001034f,0x40010351,0x40010352,0x40010354,0x40010356,0x40010358,0x4001035a,0x4001035c,0x4001035e,0x40010360,0x40010362,0x40010364,0x40010366,0x40010368,0x4001036a,0x4001036c,0x4001036e,0x40010370,0x40010372,0x40010374,0x40010376,0x40010378,0x4001037a,0x4001037c,0x4001037e,0x40010500,0x40010502,0x40010504,

175

0x40010506,0x40010508,0x4001050a,0x4001050c,0x4001050e,0x40010510,0x40010512,0x40010514,0x40010516,0x40010518,0x40010353,0x40010355,0x40010357,0x40010359,0x4001035b,0x4001035d,0x4001035f,0x40010361,0x40010363,0x40010365,0x40010367,0x40010369,0x4001036b,0x4001036d,0x4001036f,0x40010371,0x40010373,0x40010375,0x40010377,0x40010379,0x4001037b,0x4001037d,0x4001037f,0x40010501,0x40010503,0x40010505,0x40010507,0x40010509,0x4001050b,0x4001050d,0x4001050f,0x40010511,0x40010513,0x40010515,0x40010517,0x40010519,0x4001051a,0x4001051c,0x4001051e,0x40010520,

176

0x40010522,0x40010524,0x40010526,0x40010528,0x4001052a,0x4001052c,0x4001052e,0x40010530,0x40010532,0x40010534,0x40010536,0x40010538,0x4001053a,0x4001053c,0x4001053e,0x40010540,0x40010542,0x40010544,0x40010546,0x40010548,0x4001054a,0x4001054c,0x4001054e,0x40010550,0x40010552,0x40010554,0x40010556,0x40010558,0x4001055a,0x4001055c,0x4001055e,0x40010560,0x4001051b,0x4001051d,0x4001051f,0x40010521,0x40010523,0x40010525,0x40010527,0x40010529,0x4001052b,0x4001052d,0x4001052f,0x40010531,0x40010533,0x40010535,0x40010537,0x40010539,0x4001053b,0x4001053d,

177

0x4001053f,0x40010541,0x40010543,0x40010545,0x40010547,0x40010549,0x4001054b,0x4001054d,0x4001054f,0x40010551,0x40010553,0x40010555,0x40010557,0x40010559,0x4001055b,0x4001055d,0x4001055f,0x40010561,0x40010562,0x40010564,0x40010566,0x40010568,0x4001056a,0x4001056c,0x4001056e,0x40010570,0x40010572,0x40010574,0x40010576,0x40010578,0x4001057a,0x4001057c,0x4001057e,0x40010700,0x40010702,0x40010704,0x40010706,0x40010708,0x4001070a,0x4001070c,0x4001070e,0x40010710,0x40010712,0x40010714,0x40010716,0x40010718,0x4001071a,0x4001071c,0x4001071e,0x40010720,

178

0x40010722,0x40010724,0x40010726,0x40010728,0x40010563,0x40010565,0x40010567,0x40010569,0x4001056b,0x4001056d,0x4001056f,0x40010571,0x40010573,0x40010575,0x40010577,0x40010579,0x4001057b,0x4001057d,0x4001057f,0x40010701,0x40010703,0x40010705,0x40010707,0x40010709,0x4001070b,0x4001070d,0x4001070f,0x40010711,0x40010713,0x40010715,0x40010717,0x40010719,0x4001071b,0x4001071d,0x4001071f,0x40010721,0x40010723,0x40010725,0x40010727,0x40010729,0x4001072a,0x4001072c,0x4001072e,0x40010730,0x40010732,0x40010734,0x40010736,0x40010738,0x4001073a,0x4001073c,

179

0x4001073e,0x40010740,0x40010742,0x40010744,0x40010746,0x40010748,0x4001074a,0x4001074c,0x4001074e,0x40010750,0x40010752,0x40010754,0x40010756,0x40010758,0x4001075a,0x4001075c,0x4001075e,0x40010760,0x40010762,0x40010764,0x40010766,0x40010768,0x4001076a,0x4001076c,0x4001076e,0x40010770,0x4001072b,0x4001072d,0x4001072f,0x40010731,0x40010733,0x40010735,0x40010737,0x40010739,0x4001073b,0x4001073d,0x4001073f,0x40010741,0x40010743,0x40010745,0x40010747,0x40010749,0x4001074b,0x4001074d,0x4001074f,0x40010751,0x40010753,0x40010755,0x40010757,0x40010759,

180

0x4001075b,0x4001075d,0x4001075f,0x40010761,0x40010763,0x40010765,0x40010767,0x40010769,0x4001076b,0x4001076d,0x40010771,0x40010772,0x40010774,0x40010776,0x40010778,0x4001077a,0x4001077c,0x4001077e,0x40010900,0x40010902,0x40010904,0x40010906,0x40010908,0x4001090a,0x4001090c,0x4001090e,0x40010910,0x40010912,0x40010914,0x40010916,0x40010918,0x4001091a,0x4001091c,0x4001091e,0x40010920,0x40010922,0x40010924,0x40010926,0x40010928,0x4001092a,0x4001092c,0x4001092e,0x40010930,0x40010932,0x40010934,0x40010936,0x40010938,0x40010773,0x40010775,

181

0x40010777,0x40010779,0x4001077b,0x4001077d,0x4001077f,0x40010901,0x40010903,0x40010905,0x40010907,0x40010909,0x4001090b,0x4001090d,0x4001090f,0x40010911,0x40010913,0x40010915,0x40010917,0x40010919,0x4001091b,0x4001091d,0x4001091f,0x40010921,0x40010923,0x40010925,0x40010927,0x40010929,0x4001092b,0x4001092d,0x4001092f,0x40010931,0x40010933,0x40010935,0x40010937,0x40010939,0x4001093a,0x4001093c,0x4001093e,0x40010940,0x40010942,0x40010944,0x40010946,0x40010948,0x4001094a,0x4001094c,0x4001094e,0x40010950,0x40010952,0x40010954,0x40010956,0x40010958,

182

0x4001095a,0x4001095c,0x4001095e,0x40010960,0x40010962,0x40010964,0x40010966,0x40010968,0x4001096a,0x4001096c,0x4001096e,0x40010970,0x40010972,0x40010974,0x40010976,0x40010978,0x4001097a,0x4001097c,0x4001097e,0x40010b00,0x40010b3b,0x4001093d,0x4001093f,0x40010941,0x40010943,0x40010945,0x40010947,0x40010949,0x4001094b,0x4001094d,0x4001094f,0x40010951,0x40010953,0x40010955,0x40010957,0x40010959,0x4001095b,0x4001095d,0x4001095f,0x40010961,0x40010963,0x40010965,0x40010967,0x40010969,0x4001096b,0x4001096d,0x4001096f,0x40010971,0x40010973,0x40010975,

183

0x40010977,0x40010979,0x4001097b,0x4001097d,0x4001097f,0x40010b01,0x40010b02,0x40010b04,0x40010b06,0x40010b08,0x40010b0a,0x40010b0c,0x40010b0e,0x40010b10,0x40010b12,0x40010b14,0x40010b16,0x40010b18,0x40010b1a,0x40010b1c,0x40010b1e,0x40010b20,0x40010b22,0x40010b24,0x40010b26,0x40010b28,0x40010b2a,0x40010b2c,0x40010b2e,0x40010b30,0x40010b32,0x40010b34,0x40010b36,0x40010b38,0x40010b3a,0x40010b3c,0x40010b3e,0x40010b40,0x40010b42,0x40010b44,0x40010b46,0x40010b48,0x40010b03,0x40010b05,0x40010b07,0x40010b09,0x40010b0b,0x40010b0d,0x40010b0f,0x40010b11,

184

0x40010b13,0x40010b15,0x40010b17,0x40010b19,0x40010b1b,0x40010b1d,0x40010b1f,0x40010b21,0x40010b23,0x40010b25,0x40010b27,0x40010b29,0x40010b2b,0x40010b2d,0x40010b2f,0x40010b31,0x40010b33,0x40010b35,0x40010b37,0x40010b39,0x40010b3b,0x40010b3d,0x40010b3f,0x40010b41,0x40010b43,0x40010b45,0x40010b47,0x40010b49,0x40010b4a,0x40010b4c,0x40010b4e,0x40010b50,0x40010b52,0x40010b54,0x40010b56,0x40010b58,0x40010b5a,0x40010b5c,0x40010b5e,0x40010b60,0x40010b62,0x40010b64,0x40010b66,0x40010b68,0x40010b6a,0x40010b6c,0x40010b6e,0x40010b70,0x40010b72,0x40010b74,

185

0x40010b76,0x40010b78,0x40010b7a,0x40010b7c,0x40010b7e,0x40010d00,0x40010d02,0x40010d04,0x40010d06,0x40010d08,0x40010d0a,0x40010d0c,0x40010d0e,0x40010d10,0x40010b4b,0x40010b4d,0x40010b4f,0x40010b51,0x40010b53,0x40010b55,0x40010b57,0x40010b59,0x40010b5b,0x40010b5d,0x40010b5f,0x40010b61,0x40010b63,0x40010b65,0x40010b67,0x40010b69,0x40010b6b,0x40010b6d,0x40010b6f,0x40010b71,0x40010b73,0x40010b75,0x40010b77,0x40010b79,0x40010b7b,0x40010b7d,0x40010b7f,0x40010d01,0x40010d03,0x40010d05,0x40010d07,0x40010d09,0x40010d0b,0x40010d0d,0x40010d0f,0x40010d11,

186

0x40010d12,0x40010d14,0x40010d16,0x40010d18,0x40010d1a,0x40010d1c,0x40010d1e,0x40010d20,0x40010d22,0x40010d24,0x40010d26,0x40010d28,0x40010d2a,0x40010d2c,0x40010d2e,0x40010d30,0x40010d32,0x40010d34,0x40010d36,0x40010d38,0x40010d3a,0x40010d3c,0x40010d3e,0x40010d40,0x40010d42,0x40010d44,0x40010d46,0x40010d48,0x40010d4a,0x40010d4c,0x40010d4e,0x40010d50,0x40010d52,0x40010d54,0x40010d56,0x40010d58,0x40010d13,0x40010d15,0x40010d17,0x40010d19,0x40010d1b,0x40010d1d,0x40010d1f,0x40010d21,0x40010d23,0x40010d25,0x40010d27,0x40010d29,0x40010d2b,0x40010d2d,

187

0x40010d2f,0x40010d31,0x40010d33,0x40010d35,0x40010d37,0x40010d39,0x40010d3b,0x40010d3d,0x40010d3f,0x40010d41,0x40010d43,0x40010d45,0x40010d47,0x40010d49,0x40010d4b,0x40010d4d,0x40010d4f,0x40010d51,0x40010d53,0x40010d55,0x40010d57,0x40010d59,0x40010d5a,0x40010d5c,0x40010d5e,0x40010d60,0x40010d62,0x40010d64,0x40010d66,0x40010d68,0x40010d6a,0x40010d6c,0x40010d6e,0x40010d70,0x40010d72,0x40010d74,0x40010d76,0x40010d78,0x40010d7a,0x40010d7c,0x40010d7e,0x40010f00,0x40010f02,0x40010f04,0x40010f06,0x40010f08,0x40010f0a,0x40010f0c,0x40010f0e,0x40010f10,

188

0x40010f12,0x40010f14,0x40010f16,0x40010f18,0x40010f1a,0x40010f1c,0x40010f1e,0x40010f20,0x40010d5b,0x40010d5d,0x40
010d5f,0x40010d61,0x40010d63,0x40010d65,0x40010d67,0x40010d69,0x40010d6b,0x40010d6d,0x40010d6f,0x40010d71,0x40010d
73,0x40010d75,0x40010d77,0x40010d79,0x40010d7b,0x40010d7d,0x40010d7f,0x40010f01,0x40010f03,0x40010f05,0x40010f07,0
x40010f09,0x40010f0b,0x40010f0d,0x40010f0f,0x40010f11,0x40010f13,0x40010f15,0x40010f17,0x40010f19,0x40010f1b,0x400
10f1d,0x40010f1f,0x40010f21,0x40010f22,0x40010f24,0x40010f26,0x40010f28,0x40010f2a,0x40010f2c,

189

0x40010f2e,0x40010f30,0x40010f32,0x40010f34,0x40010f36,0x40010f38,0x40010f3a,0x40010f3c,0x40010f3e,0x40010f40,0x40
010f42,0x40010f44,0x40010f46,0x40010f48,0x40010f4a,0x40010f4c,0x40010f4e,0x40010f50,0x40010f52,0x40010f54,0x40010f
56,0x40010f58,0x40010f5a,0x40010f5c,0x40010f5e,0x40010f60,0x40010f62,0x40010f64,0x40010f66,0x40010f68,0x40010f23,0
x40010f25,0x40010f27,0x40010f29,0x40010f2b,0x40010f2d,0x40010f2f,0x40010f31,0x40010f33,0x40010f35,0x40010f37,0x400
10f39,0x40010f3b,0x40010f3d,0x40010f3f,0x40010f41,0x40010f43,0x40010f45,0x40010f47,0x40010f49,

190

0x40010f4b,0x40010f4d,0x40010f4f,0x40010f51,0x40010f53,0x40010f55,0x40010f57,0x40010f59,0x40010f5b,0x40010f5d,0x40
010f5f,0x40010f61,0x40010f63,0x40010f65,0x40010f67,0x40010f69,0x40010f6a,0x40010f6c,0x40010f6e,0x40010f70,0x40010f
72,0x40010f74,0x40010f76,0x40010f78,0x40010f7a,0x40010f7c,0x40010f7e,0x40011500,0x40011502,0x40011504,0x40011506,0
x40011508,0x4001150a,0x4001150c,0x4001150e,0x40011510,0x40011512,0x40011514,0x40011516,0x40011518,0x4001151a,0x400
1151c,0x4001151e,0x40011520,0x40011522,0x40011524,0x40011526,0x40011528,0x4001152a,0x4001152c,

191

0x4001152e,0x40011530,0x40010f6b,0x40010f6d,0x40010f6f,0x40010f71,0x40010f73,0x40010f75,0x40010f77,0x40010f79,0x40
010f7b,0x40010f7d,0x40010f7f,0x40011501,0x40011503,0x40011505,0x40011507,0x40011509,0x4001150b,0x4001150d,0x400115
0f,0x40011511,0x40011513,0x40011515,0x40011517,0x40011519,0x4001151b,0x4001151d,0x4001151f,0x40011521,0x40011523,0
x40011525,0x40011527,0x40011529,0x4001152b,0x4001152d,0x4001152f,0x40011531,0x40011532,0x40011534,0x40011536,0x400
11538,0x4001153a,0x4001153c,0x4001153e,0x40011540,0x40011542,0x40011544,0x40011546,0x40011548,

192

0x4001154a,0x4001154c,0x4001154e,0x40011550,0x40011552,0x40011554,0x40011556,0x40011558,0x4001155a,0x4001155c,0x40
01155e,0x40011560,0x40011562,0x40011564,0x40011566,0x40011568,0x4001156a,0x4001156c,0x4001156e,0x40011570,0x400115
72,0x40011574,0x40011576,0x40011578,0x40011533,0x40011535,0x40011537,0x40011539,0x4001153b,0x4001153d,0x4001153f,0
x40011541,0x40011543,0x40011545,0x40011547,0x40011549,0x4001154b,0x4001154d,0x4001154f,0x40011551,0x40011553,0x400
11555,0x40011557,0x40011559,0x4001155b,0x4001155d,0x4001155f,0x40011561,0x40011563,0x40011565,

193

0x40011567,0x40011569,0x4001156b,0x4001156d,0x4001156f,0x40011571,0x40011573,0x40011575,0x40011577,0x40011579,0x40
01157a,0x4001157c,0x4001157e,0x40011700,0x40011702,0x40011704,0x40011706,0x40011708,0x4001170a,0x4001170c,0x400117
0e,0x40011710,0x40011712,0x40011714,0x40011716,0x40011718,0x4001171a,0x4001171c,0x4001171e,0x40011720,0x40011722,0
x40011724,0x40011726,0x40011728,0x4001172a,0x4001172c,0x4001172e,0x40011730,0x40011732,0x40011734,0x40011736,0x400
11738,0x4001173a,0x4001173c,0x4001173e,0x40011740,0x4001157b,0x4001157d,0x4001157f,0x40011701,

194

0x40011703,0x40011705,0x40011707,0x40011709,0x4001170b,0x4001170d,0x4001170f,0x40011711,0x40011713,0x40011715,0x40
011717,0x40011719,0x4001171b,0x4001171d,0x4001171f,0x40011721,0x40011723,0x40011725,0x40011727,0x40011729,0x400117
2b,0x4001172d,0x4001172f,0x40011731,0x40011733,0x40011735,0x40011737,0x40011739,0x4001173b,0x4001173d,0x4001173f,0
x40011741,0x40011742,0x40011744,0x40011746,0x40011748,0x4001174a,0x4001174c,0x4001174e,0x40011750,0x40011752,0x400
11754,0x40011756,0x40011758,0x4001175a,0x4001175c,0x4001175e,0x40011760,0x40011762,0x40011764,

195

0x40011766,0x40011768,0x4001176a,0x4001176c,0x4001176e,0x40011770,0x40011772,0x40011774,0x40011776,0x40011778,0x40
01177a,0x4001177c,0x4001177e,0x40011900,0x40011902,0x40011904,0x40011906,0x40011908,0x40011743,0x40011745,0x400117
47,0x40011749,0x4001174b,0x4001174d,0x4001174f,0x40011751,0x40011753,0x40011755,0x40011757,0x40011759,0x4001175b,0
x4001175d,0x4001175f,0x40011761,0x40011763,0x40011765,0x40011767,0x40011769,0x4001176b,0x4001176d,0x4001176f,0x400
11771,0x40011773,0x40011775,0x40011777,0x40011779,0x4001177b,0x4001177d,0x4001177f,0x40011901,

196

0x40011903,0x40011905,0x40011907,0x40011909,0x4001190a,0x4001190c,0x4001190e,0x40011910,0x40011912,0x40011914,0x40
011916,0x40011918,0x4001191a,0x4001191c,0x4001191e,0x40011920,0x40011922,0x40011924,0x40011926,0x40011928,0x400119
2a,0x4001192c,0x4001192e,0x40011930,0x40011932,0x40011934,0x40011936,0x40011938,0x4001193a,0x4001193c,0x4001193e,0
x40011940,0x40011942,0x40011944,0x40011946,0x40011948,0x4001194a,0x4001194c,0x4001194e,0x40011950,0x4001190b,0x400
1190d,0x4001190f,0x40011911,0x40011913,0x40011915,0x40011917,0x40011919,0x4001191b,0x4001191d,

197

0x4001191f,0x40011921,0x40011923,0x40011925,0x40011927,0x40011929,0x4001192b,0x4001192d,0x4001192f,0x40011931,0x40
011933,0x40011935,0x40011937,0x40011939,0x4001193b,0x4001193d,0x4001193f,0x40011941,0x40011943,0x40011945,0x400119
47,0x40011949,0x4001194b,0x4001194d,0x4001194f,0x40011951,0x40011952,0x40011954,0x40011956,0x40011958,0x4001195a,0
x4001195c,0x4001195e,0x40011960,0x40011962,0x40011964,0x40011966,0x40011968,0x4001196a,0x4001196c,0x4001196e,0x400
11970,0x40011972,0x40011974,0x40011976,0x40011978,0x4001197a,0x4001197c,0x4001197e,0x40011b00,

198

0x40011b02,0x40011b04,0x40011b06,0x40011b08,0x40011b0a,0x40011b0c,0x40011b0e,0x40011b10,0x40011b12,0x40011b14,0x40
011b16,0x40011b18,0x40011953,0x40011955,0x40011957,0x40011959,0x4001195b,0x4001195d,0x4001195f,0x40011961,0x400119
63,0x40011965,0x40011967,0x40011969,0x4001196b,0x4001196d,0x4001196f,0x40011971,0x40011973,0x40011975,0x40011977,0
x40011979,0x4001197b,0x4001197d,0x4001197f,0x40011b01,0x40011b03,0x40011b05,0x40011b07,0x40011b09,0x40011b0b,0x400
11b0d,0x40011b0f,0x40011b11,0x40011b13,0x40011b15,0x40011b17,0x40011b19,0x40011b1a,0x40011b1c,

199

0x40011b1e,0x40011b20,0x40011b22,0x40011b24,0x40011b26,0x40011b28,0x40011b2a,0x40011b2c,0x40011b2e,0x40011b30,0x40
011b32,0x40011b34,0x40011b36,0x40011b38,0x40011b3a,0x40011b3c,0x40011b3e,0x40011b40,0x40011b42,0x40011b44,0x40011b
46,0x40011b48,0x40011b4a,0x40011b4c,0x40011b4e,0x40011b50,0x40011b52,0x40011b54,0x40011b56,0x40011b58,0x40011b5a,0
x40011b5c,0x40011b5e,0x40011b60,0x40011b1b,0x40011b1d,0x40011b1f,0x40011b21,0x40011b23,0x40011b25,0x40011b27,0x400
11b29,0x40011b2b,0x40011b2d,0x40011b2f,0x40011b31,0x40011b33,0x40011b35,0x40011b37,0x40011b39,

200

0x40011b3b,0x40011b3d,0x40011b3f,0x40011b41,0x40011b43,0x40011b45,0x40011b47,0x40011b49,0x40011b4b,0x40011b4d,0x40
011b4f,0x40011b51,0x40011b53,0x40011b55,0x40011b57,0x40011b59,0x40011b5b,0x40011b5d,0x40011b5f,0x40011b61,0x40011b
62,0x40011b64,0x40011b66,0x40011b68,0x40011b6a,0x40011b6c,0x40011b6e,0x40011b70,0x40011b72,0x40011b74,0x40011b76,0
x40011b78,0x40011b7a,0x40011b7c,0x40011b7e,0x40011b63,0x40011b65,0x40011b67,0x40011b69,0x40011b6b,0x40011b6d,0x400
11b6f,0x40011b71,0x40011b73,0x40011b75,0x40011b77,0x40011b79,0x40011b7b,0x40011b7d,0x40011b7f,

201

0x40010180,0x40010181,0x40010182,0x40010183,0x40010184,0x40010185,0x40010186,0x40010187,0x40010188,0x40010189,0x40
01018a,0x4001018b,0x4001018c,0x4001018d,0x4001018e,0x4001018f,0x40010380,0x40010381,0x40010382,0x40010383,0x400103
84,0x40010385,0x40010386,0x40010387,0x40010388,0x40010389,0x4001038a,0x4001038b,0x4001038c,0x4001038d,0x4001038e,0
x4001038f,0x40010580,0x40010581,0x40010582,0x40010583,0x40010584,0x40010585,0x40010586,0x40010587,0x40010588,0x400
10589,0x4001058a,0x4001058b,0x4001058c,0x4001058d,0x4001058e,0x4001058f,0x40010780,0x40010781,

202

0x40010782,0x40010783,0x40010784,0x40010785,0x40010786,0x40010787,0x40010788,0x40010789,0x4001078a,0x4001078b,0x40
01078c,0x4001078d,0x4001078e,0x4001078f,0x40010980,0x40010981,0x40010982,0x40010983,0x40010984,0x40010985,0x400109
86,0x40010987,0x40010988,0x40010989,0x4001098a,0x4001098b,0x4001098c,0x4001098d,0x4001098e,0x4001098f,0x40010b80,0
x40010b81,0x40010b82,0x40010b83,0x40010b84,0x40010b85,0x40010b86,0x40010b87,0x40010b88,0x40010b89,0x40010b8a,0x400
10b8b,0x40010b8c,0x40010b8d,0x40010b8e,0x40010b8f,0x40010d80,0x40010d81,0x40010d82,0x40010d83,

203

0x40010d84,0x40010d85,0x40010d86,0x40010d87,0x40010d88,0x40010d89,0x40010d8a,0x40010d8b,0x40010d8c,0x40010d8d,0x40
010d8e,0x40010d8f,0x40010f80,0x40010f81,0x40010f82,0x40010f83,0x40010f84,0x40010f85,0x40010f86,0x40010f87,0x40010f
88,0x40010f89,0x40010f8a,0x40010f8b,0x40010f8c,0x40010f8d,0x40010f8e,0x40010f8f,0x40011580,0x40011581,0x40011582,0
x40011583,0x40011584,0x40011585,0x40011586,0x40011587,0x40011588,0x40011589,0x4001158a,0x4001158b,0x4001158c,0x400
1158d,0x4001158e,0x4001158f,0x40011780,0x40011781,0x40011782,0x40011783,0x40011784,0x40011785,

204

0x40011786,0x40011787,0x40011788,0x40011789,0x4001178a,0x4001178b,0x4001178c,0x4001178d,0x4001178e,0x4001178f,0x40
011980,0x40011981,0x40011982,0x40011983,0x40011984,0x40011985,0x40011986,0x40011987,0x40011988,0x40011989,0x400119
8a,0x4001198b,0x4001198c,0x4001198d,0x4001198e,0x4001198f,0x40011b80,0x40011b81,0x40011b82,0x40011b83,0x40011b84,0
x40011b85,0x40011b86,0x40011b87,0x40011b88,0x40011b89,0x40011b8a,0x40011b8b,0x40011b8c,0x40011b8d,0x40011b8e,0x400
11b8f,0x40010190,0x40010192,0x40010194,0x40010196,0x40010198,0x4001019a,0x4001019c,0x4001019e,

205

0x400101a0,0x400101a2,0x400101a4,0x400101a6,0x40010390,0x40010392,0x40010394,0x40010396,0x40010398,0x4001039a,0x40
01039c,0x4001039e,0x400103a0,0x400103a2,0x400103a4,0x400103a6,0x40010590,0x40010592,0x40010594,0x40010596,0x400105
98,0x4001059a,0x4001059c,0x4001059e,0x400105a0,0x40010191,0x40010193,0x40010195,0x40010197,0x40010199,0x4001019b,0
x4001019d,0x4001019f,0x400101a1,0x400101a3,0x400101a5,0x400101a7,0x40010391,0x40010393,0x40010395,0x40010397,0x400
10399,0x4001039b,0x4001039d,0x4001039f,0x400103a1,0x400103a3,0x400103a5,0x400103a7,0x40010591,

206

0x40010593,0x40010595,0x40010597,0x40010599,0x4001059b,0x4001059d,0x4001059f,0x400105a1,0x400105a2,0x400105a4,0x40
0105a6,0x40010790,0x40010792,0x40010794,0x40010796,0x40010798,0x4001079a,0x4001079c,0x4001079e,0x400107a0,0x400107
a2,0x400107a4,0x400107a6,0x40010990,0x40010992,0x40010994,0x40010996,0x40010998,0x4001099a,0x4001099c,0x4001099e,0
x400109a0,0x400109a2,0x400109a4,0x400109a6,0x40010b90,0x40010b92,0x40010b94,0x40010b96,0x40010b98,0x40010b9a,0x400
10b9c,0x40010b9e,0x40010ba0,0x400105a3,0x400105a5,0x400105a7,0x40010791,0x40010793,0x40010795,

207

0x40010797,0x40010799,0x4001079b,0x4001079d,0x4001079f,0x400107a1,0x400107a3,0x400107a5,0x400107a7,0x40010991,0x40
010993,0x40010995,0x40010997,0x40010999,0x4001099b,0x4001099d,0x4001099f,0x400109a1,0x400109a3,0x400109a5,0x400109
a7,0x40010b91,0x40010b93,0x40010b95,0x40010b97,0x40010b99,0x40010b9b,0x40010b9d,0x40010b9f,0x40010ba1,0x40010ba2,0
x40010ba4,0x40010ba6,0x40010d90,0x40010d92,0x40010d94,0x40010d96,0x40010d98,0x40010d9a,0x40010d9c,0x40010d9e,0x400
10da0,0x40010da2,0x40010da4,0x40010da6,0x40010f90,0x40010f92,0x40010f94,0x40010f96,0x40010f98,

208

0x40010f9a,0x40010f9c,0x40010f9e,0x40010fa0,0x40010fa2,0x40010fa4,0x40010fa6,0x40011590,0x40011592,0x40011594,0x40
011596,0x40011598,0x4001159a,0x4001159c,0x4001159e,0x400115a0,0x40010ba3,0x40010ba5,0x40010ba7,0x40010d91,0x40010d
93,0x40010d95,0x40010d97,0x40010d99,0x40010d9b,0x40010d9d,0x40010d9f,0x40010da1,0x40010da3,0x40010da5,0x40010da7,0
x40010f91,0x40010f93,0x40010f95,0x40010f97,0x40010f99,0x40010f9b,0x40010f9d,0x40010f9f,0x40010fa1,0x40010fa3,0x400
10fa5,0x40010fa7,0x40011591,0x40011593,0x40011595,0x40011597,0x40011599,0x4001159b,0x4001159d,

209

0x4001159f,0x400115a1,0x400115a2,0x400115a4,0x400115a6,0x40011790,0x40011792,0x40011794,0x40011796,0x40011798,0x40
01179a,0x4001179c,0x4001179e,0x400117a0,0x400117a2,0x400117a4,0x400117a6,0x40011990,0x40011992,0x40011994,0x400119
96,0x40011998,0x4001199a,0x4001199c,0x4001199e,0x400119a0,0x400119a2,0x400119a4,0x400119a6,0x40011b90,0x40011b92,0
x40011b94,0x40011b96,0x40011b98,0x40011b9a,0x40011b9c,0x40011b9e,0x40011ba0,0x400115a3,0x400115a5,0x400115a7,0x400
11791,0x40011793,0x40011795,0x40011797,0x40011799,0x4001179b,0x4001179d,0x4001179f,0x400117a1,

210

0x400117a3,0x400117a5,0x400117a7,0x40011991,0x40011993,0x40011995,0x40011997,0x40011999,0x4001199b,0x4001199d,0x40
01199f,0x400119a1,0x400119a3,0x400119a5,0x400119a7,0x40011b91,0x40011b93,0x40011b95,0x40011b97,0x40011b99,0x40011b
9b,0x40011b9d,0x40011b9f,0x40011ba1,0x40011ba2,0x40011ba4,0x40011ba6,0x40011ba3,0x40011ba5,0x40011ba7,0x400101a8,0
x400101a9,0x400101aa,0x400101ab,0x400101ac,0x400101ad,0x400101ae,0x400101af,0x400101b0,0x400101b1,0x400101b2,0x400
101b3,0x400101b4,0x400101b5,0x400101b6,0x400101b7,0x400103a8,0x400103a9,0x400103aa,0x400103ab,

211

0x400103ac,0x400103ad,0x400103ae,0x400103af,0x400103b0,0x400103b1,0x400103b2,0x400103b3,0x400103b4,0x400103b5,0x40
0103b6,0x400103b7,0x400105a8,0x400105a9,0x400105aa,0x400105ab,0x400105ac,0x400105ad,0x400105ae,0x400105af,0x400105
b0,0x400105b1,0x400105b2,0x400105b3,0x400105b4,0x400105b5,0x400105b6,0x400105b7,0x400107a8,0x400107a9,0x400107aa,0
x400107ab,0x400107ac,0x400107ad,0x400107ae,0x400107af,0x400107b0,0x400107b1,0x400107b2,0x400107b3,0x400107b4,0x400
107b5,0x400107b6,0x400107b7,0x400109a8,0x400109a9,0x400109aa,0x400109ab,0x400109ac,0x400109ad,

212

0x400109ae,0x400109af,0x400109b0,0x400109b1,0x400109b2,0x400109b3,0x400109b4,0x400109b5,0x400109b6,0x400109b7,0x40
010ba8,0x40010ba9,0x40010baa,0x40010bab,0x40010bac,0x40010bad,0x40010bae,0x40010baf,0x40010bb0,0x40010bb1,0x40010b
b2,0x40010bb3,0x40010bb4,0x40010bb5,0x40010bb6,0x40010bb7,0x40010da8,0x40010da9,0x40010daa,0x40010dab,0x40010dac,0
x40010dad,0x40010dae,0x40010daf,0x40010db0,0x40010db1,0x40010db2,0x40010db3,0x40010db4,0x40010db5,0x40010db6,0x400
10db7,0x40010fa8,0x40010fa9,0x40010faa,0x40010fab,0x40010fac,0x40010fad,0x40010fae,0x40010faf,

213

0x40010fb0,0x40010fb1,0x40010fb2,0x40010fb3,0x40010fb4,0x40010fb5,0x40010fb6,0x40010fb7,0x400115a8,0x400115a9,0x40
0115aa,0x400115ab,0x400115ac,0x400115ad,0x400115ae,0x400115af,0x400115b0,0x400115b1,0x400115b2,0x400115b3,0x400115
b4,0x400115b5,0x400115b6,0x400115b7,0x400117a8,0x400117a9,0x400117aa,0x400117ab,0x400117ac,0x400117ad,0x400117ae,0
x400117af,0x400117b0,0x400117b1,0x400117b2,0x400117b3,0x400117b4,0x400117b5,0x400117b6,0x400117b7,0x400119a8,0x400
119a9,0x400119aa,0x400119ab,0x400119ac,0x400119ad,0x400119ae,0x400119af,0x400119b0,0x400119b1,

214

0x400119b2,0x400119b3,0x400119b4,0x400119b5,0x400119b6,0x400119b7,0x40011ba8,0x40011ba9,0x40011baa,0x40011bab,0x40
011bac,0x40011bad,0x40011bae,0x40011baf,0x40011bb0,0x40011bb1,0x40011bb2,0x40011bb3,0x40011bb4,0x40011bb5,0x40011b
b6,0x40011bb7,0x400101c0,0x400103c0,0x400105c0,0x400107c0,0x400109c0,0x40010bc0,0x40010dc0,0x40010fc0,0x400115c0,0
x400117c0,0x400119c0,0x40011bc0,0x400101c2,0x400103c2,0x400105c2,0x400107c2,0x400109c2,0x40010bc2,0x40010dc2,0x400
10fc2,0x400115c2,0x400117c2,0x400119c2,0x40011bc2,0x400101c4,0x400103c4,0x400105c4,0x400107c4,

215

0x400109c4,0x40010bc4,0x40010dc4,0x40010fc4,0x400115c4,0x400117c4,0x400119c4,0x40011bc4,0x400101ca,0x400103ca,0x40
0105ca,0x400107ca,0x400109ca,0x40010bca,0x40010dca,0x40010fca,0x400115ca,0x400117ca,0x400119ca,0x40011bca,0x400101
cc,0x400103cc,0x400105cc,0x400107cc,0x400109cc,0x40010bcc,0x40010dcc,0x40010fcc,0x400115cc,0x400117cc,0x400119cc,0
x40011bcc,0x400101ce,0x400103ce,0x400105ce,0x400107ce,0x400109ce,0x40010bce,0x40010dce,0x40010fce,0x400115ce,0x400
117ce,0x400119ce,0x40011bce,0x400101d0,0x400103d0,0x400105d0,0x400107d0,0x400109d0,0x40010bd0,

216

0x40010dd0,0x40010fd0,0x400115d0,0x400117d0,0x400119d0,0x40011bd0,0x400101d2,0x400103d2,0x400105d2,0x400107d2,0x40
0109d2,0x40010bd2,0x40010dd2,0x40010fd2,0x400115d2,0x400117d2,0x400119d2,0x40011bd2,0x400101d6,0x400103d6,0x400105
d6,0x400107d6,0x400109d6,0x40010bd6,0x40010dd6,0x40010fd6,0x400115d6,0x400117d6,0x400119d6,0x40011bd6,0x400101d8,0
x400103d8,0x400105d8,0x400107d8,0x400109d8,0x40010bd8,0x40010dd8,0x40010fd8,0x400115d8,0x400117d8,0x400119d8,0x400
11bd8,0x400101de,0x400103de,0x400105de,0x400107de,0x400109de,0x40010bde,0x40010dde,0x40010fde,

217

0x400115de,0x400117de,0x400119de,0x40011bde,0x400101e0,0x400105e0,0x400109e0,0x40010de0,0x400115e0,0x400119e0,0x40
0103e0,0x400107e0,0x40010be0,0x40010fe0,0x400117e0,0x40011be0,0x400101e2,0x400105e2,0x400109e2,0x40010de2,0x400115
e2,0x400119e2,0x400103e2,0x400107e2,0x40010be2,0x40010fe2,0x400117e2,0x40011be2,0x400101e4,0x400105e4,0x400109e4,0
x40010de4,0x400115e4,0x400119e4,0x400103e4,0x400107e4,0x40010be4,0x40010fe4,0x400117e4,0x40011be4,0x400101e6,0x400
105e6,0x400109e6,0x40010de6,0x400115e6,0x400119e6,0x400103e6,0x400107e6,0x40010be6,0x40010fe6,

218

0x400117e6,0x40011be6,0x400101e8,0x400105e8,0x400109e8,0x40010de8,0x400115e8,0x400119e8,0x400103e8,0x400107e8,0x40
010be8,0x40010fe8,0x400117e8,0x40011be8,0x400101ea,0x400105ea,0x400109ea,0x40010dea,0x400115ea,0x400119ea,0x400103
ea,0x400107ea,0x40010bea,0x40010fea,0x400117ea,0x40011bea,0x400101ec,0x400105ec,0x400109ec,0x40010dec,0x400115ec,0
x400119ec,0x400103ec,0x400107ec,0x40010bec,0x40010fec,0x400117ec,0x40011bec,0x400101ee,0x400105ee,0x400109ee,0x400
10dee,0x400115ee,0x400119ee,0x400103ee,0x400107ee,0x40010bee,0x40010fee,0x400117ee,0x40011bee,

219

0x40014000,0x40014002,0x40014004,0x40014006,0x40014008,0x4001400a,0x4001400c,0x4001400e,0x40014010,0x40014012,0x40
014014,0x40014016,0x40014018,0x4001401a,0x4001401c,0x4001401e,0x40014020,0x40014022,0x40014024,0x40014026,0x400140
28,0x4001402a,0x4001402c,0x4001402e,0x40014030,0x40014032,0x40014034,0x40014036,0x40014038,0x4001403a,0x4001403c,0
x4001403e,0x40014040,0x40014001,0x40014003,0x40014005,0x40014007,0x40014009,0x4001400b,0x4001400d,0x4001400f,0x400
14011,0x40014013,0x40014015,0x40014017,0x40014019,0x4001401b,0x4001401d,0x4001401f,0x40014021,

220

0x40014023,0x40014025,0x40014027,0x40014029,0x4001402b,0x4001402d,0x4001402f,0x40014031,0x40014033,0x40014035,0x40
014037,0x40014039,0x4001403b,0x4001403d,0x4001403f,0x40014041,0x40014042,0x40014044,0x40014046,0x40014048,0x400140
4a,0x4001404c,0x4001404e,0x40014050,0x40014052,0x40014054,0x40014056,0x40014058,0x4001405a,0x4001405c,0x4001405e,0
x40014060,0x40014062,0x40014064,0x40014066,0x40014068,0x4001406a,0x4001406c,0x4001406e,0x40014070,0x40014072,0x400
14074,0x40014076,0x40014078,0x4001407a,0x4001407c,0x4001407e,0x40014100,0x40014102,0x40014104,

221

0x40014106,0x40014108,0x40014043,0x40014045,0x40014047,0x40014049,0x4001404b,0x4001404d,0x4001404f,0x40014051,0x40
014053,0x40014055,0x40014057,0x40014059,0x4001405b,0x4001405d,0x4001405f,0x40014061,0x40014063,0x40014065,0x400140
67,0x40014069,0x4001406b,0x4001406d,0x4001406f,0x40014071,0x40014073,0x40014075,0x40014077,0x40014079,0x4001407b,0
x4001407d,0x4001407f,0x40014101,0x40014103,0x40014105,0x40014107,0x40014109,0x4001410a,0x4001410c,0x4001410e,0x400
14110,0x40014112,0x40014114,0x40014116,0x40014118,0x4001411a,0x4001411c,0x4001411e,0x40014120,

222

0x40014122,0x40014124,0x40014126,0x40014128,0x4001412a,0x4001412c,0x4001412e,0x40014130,0x40014132,0x40014134,0x40
014136,0x40014138,0x4001413a,0x4001413c,0x4001413e,0x40014140,0x40014142,0x40014144,0x40014146,0x40014148,0x400141
4a,0x4001414c,0x4001414e,0x40014150,0x4001410b,0x4001410d,0x4001410f,0x40014111,0x40014113,0x40014115,0x40014117,0
x40014119,0x4001411b,0x4001411d,0x4001411f,0x40014121,0x40014123,0x40014125,0x40014127,0x40014129,0x4001412b,0x400
1412d,0x4001412f,0x40014131,0x40014133,0x40014135,0x40014137,0x40014139,0x4001413b,0x4001413d,

223

0x4001413f,0x40014141,0x40014143,0x40014145,0x40014147,0x40014149,0x4001414b,0x4001414d,0x4001414f,0x40014151,0x40
014152,0x40014154,0x40014156,0x40014158,0x4001415a,0x4001415c,0x4001415e,0x40014160,0x40014162,0x40014164,0x400141
66,0x40014168,0x4001416a,0x4001416c,0x4001416e,0x40014170,0x40014172,0x40014174,0x40014176,0x40014178,0x4001417a,0
x4001417c,0x4001417e,0x40014200,0x40014202,0x40014204,0x40014206,0x40014208,0x4001420a,0x4001420c,0x4001420e,0x400
14210,0x40014212,0x40014214,0x40014216,0x40014218,0x40014153,0x40014155,0x40014157,0x40014159,

224

0x4001415b,0x4001415d,0x4001415f,0x40014161,0x40014163,0x40014165,0x40014167,0x40014169,0x4001416b,0x4001416d,0x40
01416f,0x40014171,0x40014173,0x40014175,0x40014177,0x40014179,0x4001417b,0x4001417d,0x4001417f,0x40014201,0x400142
03,0x40014205,0x40014207,0x40014209,0x4001420b,0x4001420d,0x4001420f,0x40014211,0x40014213,0x40014215,0x40014217,0
x40014219,0x4001421a,0x4001421c,0x4001421e,0x40014220,0x40014222,0x40014224,0x40014226,0x40014228,0x4001422a,0x400
1422c,0x4001422e,0x40014230,0x40014232,0x40014234,0x40014236,0x40014238,0x4001423a,0x4001423c,

225

0x4001423e,0x40014240,0x40014242,0x40014244,0x40014246,0x40014248,0x4001424a,0x4001424c,0x4001424e,0x40014250,0x40
014252,0x40014254,0x40014256,0x40014258,0x4001425a,0x4001425c,0x4001425e,0x40014260,0x4001421b,0x4001421d,0x400142
1f,0x40014221,0x40014223,0x40014225,0x40014227,0x40014229,0x4001422b,0x4001422d,0x4001422f,0x40014231,0x40014233,0
x40014235,0x40014237,0x40014239,0x4001423b,0x4001423d,0x4001423f,0x40014241,0x40014243,0x40014245,0x40014247,0x400
14249,0x4001424b,0x4001424d,0x4001424f,0x40014251,0x40014253,0x40014255,0x40014257,0x40014259,

226

0x4001425b,0x4001425d,0x4001425f,0x40014261,0x40014262,0x40014264,0x40014266,0x40014268,0x4001426a,0x4001426c,0x40
01426e,0x40014270,0x40014272,0x40014274,0x40014276,0x40014278,0x4001427a,0x4001427c,0x4001427e,0x40014300,0x400143
02,0x40014304,0x40014306,0x40014308,0x4001430a,0x4001430c,0x4001430e,0x40014310,0x40014312,0x40014314,0x40014316,0
x40014318,0x4001431a,0x4001431c,0x4001431e,0x40014320,0x40014322,0x40014324,0x40014326,0x40014328,0x40014263,0x400
14265,0x40014267,0x40014269,0x4001426b,0x4001426d,0x4001426f,0x40014271,0x40014273,0x40014275,

227

0x40014277,0x40014279,0x4001427b,0x4001427d,0x4001427f,0x40014301,0x40014303,0x40014305,0x40014307,0x40014309,0x40
01430b,0x4001430d,0x4001430f,0x40014311,0x40014313,0x40014315,0x40014317,0x40014319,0x4001431b,0x4001431d,0x400143
1f,0x40014321,0x40014323,0x40014325,0x40014327,0x40014329,0x4001432a,0x4001432c,0x4001432e,0x40014330,0x40014332,0
x40014334,0x40014336,0x40014338,0x4001433a,0x4001433c,0x4001433e,0x40014340,0x40014342,0x40014344,0x40014346,0x400
14348,0x4001434a,0x4001434c,0x4001434e,0x40014350,0x40014352,0x40014354,0x40014356,0x40014358,

228

0x4001435a,0x4001435c,0x4001435e,0x40014360,0x40014362,0x40014364,0x40014366,0x40014368,0x4001436a,0x4001436c,0x40
01436e,0x40014370,0x4001432b,0x4001432d,0x4001432f,0x40014331,0x40014333,0x40014335,0x40014337,0x40014339,0x400143
3b,0x4001433d,0x4001433f,0x40014341,0x40014343,0x40014345,0x40014347,0x40014349,0x4001434b,0x4001434d,0x4001434f,0
x40014351,0x40014353,0x40014355,0x40014357,0x40014359,0x4001435b,0x4001435d,0x4001435f,0x40014361,0x40014363,0x400
14365,0x40014367,0x40014369,0x4001436b,0x4001436d,0x4001436f,0x40014371,0x40014372,0x40014374,

229

0x40014376,0x40014378,0x4001437a,0x4001437c,0x4001437e,0x40014400,0x40014402,0x40014404,0x40014406,0x40014408,0x40
01440a,0x4001440c,0x4001440e,0x40014410,0x40014412,0x40014414,0x40014416,0x40014418,0x4001441a,0x4001441c,0x400144
1e,0x40014420,0x40014422,0x40014424,0x40014426,0x40014428,0x4001442a,0x4001442c,0x4001442e,0x40014430,0x40014432,0
x40014434,0x40014436,0x40014438,0x40014373,0x40014375,0x40014377,0x40014379,0x4001437b,0x4001437d,0x4001437f,0x400
14401,0x40014403,0x40014405,0x40014407,0x40014409,0x4001440b,0x4001440d,0x4001440f,0x40014411,

230

0x40014413,0x40014415,0x40014417,0x40014419,0x4001441b,0x4001441d,0x4001441f,0x40014421,0x40014423,0x40014425,0x40014427,0x40014429,0x4001442b,0x4001442d,0x4001442f,0x40014431,0x40014433,0x40014435,0x40014437,0x40014439,0x4001443a,0x4001443c,0x4001443e,0x40014440,0x40014442,0x40014444,0x40014446,0x40014448,0x4001444a,0x4001444c,0x4001444e,0x40014450,0x40014452,0x40014454,0x40014456,0x40014458,0x4001445a,0x4001445c,0x4001445e,0x40014460,0x40014462,0x40014464,0x40014466,0x40014468,0x4001446a,0x4001446c,0x4001446e,0x40014470,0x40014472,0x40014474,

231

0x40014476,0x40014478,0x4001447a,0x4001447c,0x4001447e,0x40014500,0x4001443b,0x4001443d,0x4001443f,0x40014441,0x40014443,0x40014445,0x40014447,0x40014449,0x4001444b,0x4001444d,0x4001444f,0x40014451,0x40014453,0x40014455,0x40014457,0x40014459,0x4001445b,0x4001445d,0x4001445f,0x40014461,0x40014463,0x40014465,0x40014467,0x40014469,0x4001446b,0x4001446d,0x4001446f,0x40014471,0x40014473,0x40014475,0x40014477,0x40014479,0x4001447b,0x4001447d,0x4001447f,0x40014501,0x40014502,0x40014504,0x40014506,0x40014508,0x4001450a,0x4001450c,0x4001450e,0x40014510,

232

0x40014512,0x40014514,0x40014516,0x40014518,0x4001451a,0x4001451c,0x4001451e,0x40014520,0x40014522,0x40014524,0x40014526,0x40014528,0x4001452a,0x4001452c,0x4001452e,0x40014530,0x40014532,0x40014534,0x40014536,0x40014538,0x4001453a,0x4001453c,0x4001453e,0x40014540,0x40014542,0x40014544,0x40014546,0x40014548,0x40014503,0x40014505,0x40014507,0x40014509,0x4001450b,0x4001450d,0x4001450f,0x40014511,0x40014513,0x40014515,0x40014517,0x40014519,0x4001451b,0x4001451d,0x4001451f,0x40014521,0x40014523,0x40014525,0x40014527,0x40014529,0x4001452b,0x4001452d,

233

0x4001452f,0x40014531,0x40014533,0x40014535,0x40014537,0x40014539,0x4001453b,0x4001453d,0x4001453f,0x40014541,0x40014543,0x40014545,0x40014547,0x40014549,0x4001454a,0x4001454c,0x4001454e,0x40014550,0x40014552,0x40014554,0x40014556,0x40014558,0x4001455a,0x4001455c,0x4001455e,0x40014560,0x40014562,0x40014564,0x40014566,0x40014568,0x4001456a,0x4001456c,0x4001456e,0x40014570,0x40014572,0x40014574,0x40014576,0x40014578,0x4001457a,0x4001457c,0x4001457e,0x40014600,0x40014602,0x40014604,0x40014606,0x40014608,0x4001460a,0x4001460c,0x4001460e,0x40014610,

234

0x4001454b,0x4001454d,0x4001454f,0x40014551,0x40014553,0x40014555,0x40014557,0x40014559,0x4001455b,0x4001455d,0x40014455f,0x40014561,0x40014563,0x40014565,0x40014567,0x40014569,0x4001456b,0x4001456d,0x4001456f,0x40014571,0x40014573,0x40014575,0x40014577,0x40014579,0x4001457b,0x4001457d,0x4001457f,0x40014601,0x40014603,0x40014605,0x40014607,0x40014609,0x4001460b,0x4001460d,0x4001460f,0x40014611,0x40014612,0x40014614,0x40014616,0x40014618,0x4001461a,0x40014610,0x4001461c,0x4001461e,0x40014620,0x40014622,0x40014624,0x40014626,0x40014628,0x4001462a,0x4001462c,

235

0x4001462e,0x40014630,0x40014632,0x40014634,0x40014636,0x40014638,0x4001463a,0x4001463c,0x4001463e,0x40014640,0x40014642,0x40014644,0x40014646,0x40014648,0x4001464a,0x4001464c,0x4001464e,0x40014650,0x40014652,0x40014654,0x40014656,0x40014658,0x40014613,0x40014615,0x40014617,0x40014619,0x4001461b,0x4001461d,0x4001461f,0x40014621,0x40014623,0x40014625,0x40014627,0x40014629,0x4001462b,0x4001462d,0x4001462f,0x40014631,0x40014633,0x40014635,0x40014637,0x40014639,0x4001463b,0x4001463d,0x4001463f,0x40014641,0x40014643,0x40014645,0x40014647,0x40014649,

236

0x4001464b,0x4001464d,0x4001464f,0x40014651,0x40014653,0x40014655,0x40014657,0x40014659,0x4001465a,0x4001465c,0x40014465e,0x40014660,0x40014662,0x40014664,0x40014666,0x40014668,0x4001466a,0x4001466c,0x4001466e,0x40014670,0x40014672,0x40014674,0x40014676,0x40014678,0x4001467a,0x4001467c,0x4001467e,0x40014700,0x40014702,0x40014704,0x40014706,0x40014708,0x4001470a,0x4001470c,0x4001470e,0x40014710,0x40014712,0x40014714,0x40014716,0x40014718,0x4001471a,0x40014720,0x4001471c,0x4001471e,0x4001465b,0x4001465d,0x4001465f,0x40014661,0x40014663,0x40014665,

237

0x40014667,0x40014669,0x4001466b,0x4001466d,0x4001466f,0x40014671,0x40014673,0x40014675,0x40014677,0x40014679,0x40014678,0x4001467d,0x4001467f,0x40014701,0x40014703,0x40014705,0x40014707,0x40014709,0x4001470b,0x4001470d,0x4001470f,0x40014711,0x40014713,0x40014715,0x40014717,0x40014719,0x4001471b,0x4001471d,0x4001471f,0x40014721,0x40014722,0x40014724,0x40014726,0x40014728,0x4001472a,0x4001472c,0x4001472e,0x40014730,0x40014732,0x40014734,0x40014736,0x40014738,0x4001473a,0x4001473c,0x4001473e,0x40014740,0x40014742,0x40014744,0x40014746,0x40014748,

238

0x4001474a,0x4001474c,0x4001474e,0x40014750,0x40014752,0x40014754,0x40014756,0x40014758,0x4001475a,0x4001475c,0x40014750,0x4001475e,0x40014760,0x40014762,0x40014764,0x40014766,0x40014768,0x40014723,0x40014725,0x40014727,0x40014729,0x4001472b,0x4001472d,0x4001472f,0x40014731,0x40014733,0x40014735,0x40014737,0x40014739,0x4001473b,0x4001473d,0x4001473f,0x40014741,0x40014743,0x40014745,0x40014747,0x40014749,0x4001474b,0x4001474d,0x4001474f,0x40014751,0x40014753,0x40014755,0x40014757,0x40014759,0x4001475b,0x4001475d,0x4001475f,0x40014761,0x40014763,0x40014765,

239

0x40014767,0x40014769,0x4001476a,0x4001476c,0x4001476e,0x40014770,0x40014772,0x40014774,0x40014776,0x40014778,0x4001477a,0x4001477c,0x4001477e,0x40014800,0x40014802,0x40014804,0x40014806,0x40014808,0x4001480a,0x4001480c,0x4001480e,0x40014810,0x40014812,0x40014814,0x40014816,0x40014818,0x4001481a,0x4001481c,0x4001481e,0x40014820,0x40014822,0x40014824,0x40014826,0x40014828,0x4001482a,0x4001482c,0x4001482e,0x40014830,0x4001476b,0x4001476d,0x4001476f,0x40014771,0x40014773,0x40014775,0x40014777,0x40014779,0x4001477b,0x4001477d,0x4001477f,0x40014801,

240

0x40014803,0x40014805,0x40014807,0x40014809,0x4001480b,0x4001480d,0x4001480f,0x40014811,0x40014813,0x40014815,0x40014817,0x40014819,0x4001481b,0x4001481d,0x4001481f,0x40014821,0x40014823,0x40014825,0x40014827,0x40014829,0x4001482b,0x4001482d,0x4001482f,0x40014831,0x40014832,0x40014834,0x40014836,0x40014838,0x4001483a,0x4001483c,0x4001483e,0x40014840,0x40014842,0x40014844,0x40014846,0x40014848,0x4001484a,0x4001484c,0x4001484e,0x40014850,0x40014852,0x40014854,0x40014856,0x40014858,0x4001485a,0x4001485c,0x4001485e,0x40014860,0x40014862,0x40014864,

241

0x40014866,0x40014868,0x4001486a,0x4001486c,0x4001486e,0x40014870,0x40014872,0x40014874,0x40014876,0x40014878,0x40014833,0x40014835,0x40014837,0x40014839,0x4001483b,0x4001483d,0x4001483f,0x40014841,0x40014843,0x40014845,0x40014847,0x40014849,0x4001484b,0x4001484d,0x4001484f,0x40014851,0x40014853,0x40014855,0x40014857,0x40014859,0x4001485b,0x4001485d,0x4001485f,0x40014861,0x40014863,0x40014865,0x40014867,0x40014869,0x4001486b,0x4001486d,0x4001486f,0x40014871,0x40014873,0x40014875,0x40014877,0x40014879,0x4001487a,0x4001487c,0x4001487e,0x40014900,

242

0x40014902,0x40014904,0x40014906,0x40014908,0x4001490a,0x4001490c,0x4001490e,0x40014910,0x40014912,0x40014914,0x40014916,0x40014918,0x4001491a,0x4001491c,0x4001491e,0x40014920,0x40014922,0x40014924,0x40014926,0x40014928,0x4001492a,0x4001492c,0x4001492e,0x40014930,0x40014932,0x40014934,0x40014936,0x40014938,0x4001493a,0x4001493c,0x4001493e,0x40014940,0x4001487b,0x4001487d,0x4001487f,0x40014901,0x40014903,0x40014905,0x40014907,0x40014909,0x4001490b,0x4001490d,0x4001490f,0x40014911,0x40014913,0x40014915,0x40014917,0x40014919,0x4001491b,0x4001491d,

243

0x4001491f,0x40014921,0x40014923,0x40014925,0x40014927,0x40014929,0x4001492b,0x4001492d,0x4001492f,0x40014931,0x40014933,0x40014935,0x40014937,0x40014939,0x4001493b,0x4001493d,0x4001493f,0x40014941,0x40014942,0x40014944,0x40014946,0x40014948,0x4001494a,0x4001494c,0x4001494e,0x40014950,0x40014952,0x40014954,0x40014956,0x40014958,0x4001495a,0x4001495c,0x4001495e,0x40014960,0x40014962,0x40014964,0x40014966,0x40014968,0x4001496a,0x4001496c,0x4001496e,0x40014970,0x40014972,0x40014974,0x40014976,0x40014978,0x4001497a,0x4001497c,0x4001497e,0x40014c00,

244

0x40014c02,0x40014c04,0x40014c06,0x40014c08,0x40014943,0x40014945,0x40014947,0x40014949,0x4001494b,0x4001494d,0x40
01494f,0x40014951,0x40014953,0x40014955,0x40014957,0x40014959,0x4001495b,0x4001495d,0x4001495f,0x40014961,0x400149
63,0x40014965,0x40014967,0x40014969,0x4001496b,0x4001496d,0x4001496f,0x40014971,0x40014973,0x40014975,0x40014977,0
x40014979,0x4001497b,0x4001497d,0x4001497f,0x40014c01,0x40014c03,0x40014c05,0x40014c07,0x40014c09,0x40014c0a,0x400
14c0c,0x40014c0e,0x40014c10,0x40014c12,0x40014c14,0x40014c16,0x40014c18,0x40014c1a,0x40014c1c,

245

0x40014c1e,0x40014c20,0x40014c22,0x40014c24,0x40014c26,0x40014c28,0x40014c2a,0x40014c2c,0x40014c2e,0x40014c30,0x40
014c32,0x40014c34,0x40014c36,0x40014c38,0x40014c3a,0x40014c3c,0x40014c3e,0x40014c40,0x40014c42,0x40014c44,0x40014c
46,0x40014c48,0x40014c4a,0x40014c4c,0x40014c4e,0x40014c50,0x40014c0b,0x40014c0d,0x40014c0f,0x40014c11,0x40014c13,0
x40014c15,0x40014c17,0x40014c19,0x40014c1b,0x40014c1d,0x40014c1f,0x40014c21,0x40014c23,0x40014c25,0x40014c27,0x400
14c29,0x40014c2b,0x40014c2d,0x40014c2f,0x40014c31,0x40014c33,0x40014c35,0x40014c37,0x40014c39,

246

0x40014c3b,0x40014c3d,0x40014c3f,0x40014c41,0x40014c43,0x40014c45,0x40014c47,0x40014c49,0x40014c4b,0x40014c4d,0x40
014c4f,0x40014c51,0x40014c52,0x40014c54,0x40014c56,0x40014c58,0x40014c5a,0x40014c5c,0x40014c5e,0x40014c60,0x40014c
62,0x40014c64,0x40014c66,0x40014c68,0x40014c6a,0x40014c6c,0x40014c6e,0x40014c70,0x40014c72,0x40014c74,0x40014c76,0
x40014c78,0x40014c7a,0x40014c7c,0x40014c7e,0x40014d00,0x40014d02,0x40014d04,0x40014d06,0x40014d08,0x40014d0a,0x400
14d0c,0x40014d0e,0x40014d10,0x40014d12,0x40014d14,0x40014d16,0x40014d18,0x40014c53,0x40014c55,

247

0x40014c57,0x40014c59,0x40014c5b,0x40014c5d,0x40014c5f,0x40014c61,0x40014c63,0x40014c65,0x40014c67,0x40014c69,0x40
014c6b,0x40014c6d,0x40014c6f,0x40014c71,0x40014c73,0x40014c75,0x40014c77,0x40014c79,0x40014c7b,0x40014c7d,0x40014c
7f,0x40014d01,0x40014d03,0x40014d05,0x40014d07,0x40014d09,0x40014d0b,0x40014d0d,0x40014d0f,0x40014d11,0x40014d13,0
x40014d15,0x40014d17,0x40014d19,0x40014d1a,0x40014d1c,0x40014d1e,0x40014d20,0x40014d22,0x40014d24,0x40014d26,0x400
14d28,0x40014d2a,0x40014d2c,0x40014d2e,0x40014d30,0x40014d32,0x40014d34,0x40014d36,0x40014d38,

248

0x40014d3a,0x40014d3c,0x40014d3e,0x40014d40,0x40014d42,0x40014d44,0x40014d46,0x40014d48,0x40014d4a,0x40014d4c,0x40
014d4e,0x40014d50,0x40014d52,0x40014d54,0x40014d56,0x40014d58,0x40014d5a,0x40014d5c,0x40014d5e,0x40014d60,0x40014d
1b,0x40014d1d,0x40014d1f,0x40014d21,0x40014d23,0x40014d25,0x40014d27,0x40014d29,0x40014d2b,0x40014d2d,0x40014d2f,0
x40014d31,0x40014d33,0x40014d35,0x40014d37,0x40014d39,0x40014d3b,0x40014d3d,0x40014d3f,0x40014d41,0x40014d43,0x400
14d45,0x40014d47,0x40014d49,0x40014d4b,0x40014d4d,0x40014d4f,0x40014d51,0x40014d53,0x40014d55,

249

0x40014d57,0x40014d59,0x40014d5b,0x40014d5d,0x40014d5f,0x40014d61,0x40014d62,0x40014d64,0x40014d66,0x40014d68,0x40
014d6a,0x40014d6c,0x40014d6e,0x40014d70,0x40014d72,0x40014d74,0x40014d76,0x40014d78,0x40014d7a,0x40014d7c,0x40014d
7e,0x40014d63,0x40014d65,0x40014d67,0x40014d69,0x40014d6b,0x40014d6d,0x40014d6f,0x40014d71,0x40014d73,0x40014d75,0
x40014d77,0x40014d79,0x40014d7b,0x40014d7d,0x40014d7f,0x40014080,0x40014082,0x40014084,0x40014086,0x40014088,0x400
1408a,0x4001408c,0x4001408e,0x40014180,0x40014182,0x40014184,0x40014186,0x40014188,0x4001418a,

250

0x4001418c,0x4001418e,0x40014280,0x40014282,0x40014284,0x40014286,0x40014288,0x4001428a,0x4001428c,0x4001428e,0x40
014380,0x40014382,0x40014384,0x40014386,0x40014388,0x4001438a,0x4001438c,0x4001438e,0x40014480,0x40014081,0x400140
83,0x40014085,0x40014087,0x40014089,0x4001408b,0x4001408d,0x4001408f,0x40014181,0x40014183,0x40014185,0x40014187,0
x40014189,0x4001418b,0x4001418d,0x4001418f,0x40014281,0x40014283,0x40014285,0x40014287,0x40014289,0x4001428b,0x400
1428d,0x4001428f,0x40014381,0x40014383,0x40014385,0x40014387,0x40014389,0x4001438b,0x4001438d,

251

0x4001438f,0x40014481,0x40014482,0x40014484,0x40014486,0x40014488,0x4001448a,0x4001448c,0x4001448e,0x40014580,0x40
014582,0x40014584,0x40014586,0x40014588,0x4001458a,0x4001458c,0x4001458e,0x40014680,0x40014682,0x40014684,0x400146
86,0x40014688,0x4001468a,0x4001468c,0x4001468e,0x40014780,0x40014782,0x40014784,0x40014786,0x40014788,0x4001478a,0
x4001478c,0x4001478e,0x40014880,0x40014882,0x40014884,0x40014886,0x40014888,0x40014483,0x40014485,0x40014487,0x400
14489,0x4001448b,0x4001448d,0x4001448f,0x40014581,0x40014583,0x40014585,0x40014587,0x40014589,

252

0x4001458b,0x4001458d,0x4001458f,0x40014681,0x40014683,0x40014685,0x40014687,0x40014689,0x4001468b,0x4001468d,0x40
01468f,0x40014781,0x40014783,0x40014785,0x40014787,0x40014789,0x4001478b,0x4001478d,0x4001478f,0x40014881,0x400148
83,0x40014885,0x40014887,0x40014889,0x4001488a,0x4001488c,0x4001488e,0x40014980,0x40014982,0x40014984,0x40014986,0
x40014988,0x4001498a,0x4001498c,0x4001498e,0x40014c80,0x40014c82,0x40014c84,0x40014c86,0x40014c88,0x40014c8a,0x400
14c8c,0x40014c8e,0x40014d80,0x40014d82,0x40014d84,0x40014d86,0x40014d88,0x40014d8a,0x40014d8c,

253

0x40014d8e,0x4001488b,0x4001488d,0x4001488f,0x40014981,0x40014983,0x40014985,0x40014987,0x40014989,0x4001498b,0x40
01498d,0x4001498f,0x40014c81,0x40014c83,0x40014c85,0x40014c87,0x40014c89,0x40014c8b,0x40014c8d,0x40014c8f,0x40014d
81,0x40014d83,0x40014d85,0x40014d87,0x40014d89,0x40014d8b,0x40014d8d,0x40014d8f,0x40014090,0x40014092,0x40014094,0
x40014096,0x40014098,0x4001409a,0x4001409c,0x4001409e,0x400140a0,0x400140a2,0x400140a4,0x400140a6,0x40014190,0x400
14192,0x40014194,0x40014196,0x40014198,0x4001419a,0x4001419c,0x4001419e,0x400141a0,0x400141a2,

254

0x400141a4,0x400141a6,0x40014290,0x40014292,0x40014294,0x40014296,0x40014298,0x4001429a,0x4001429c,0x4001429e,0x40
0142a0,0x40014091,0x40014093,0x40014095,0x40014097,0x40014099,0x4001409b,0x4001409d,0x4001409f,0x400140a1,0x400140
a3,0x400140a5,0x400140a7,0x40014191,0x40014193,0x40014195,0x40014197,0x40014199,0x4001419b,0x4001419d,0x4001419f,0
x400141a1,0x400141a3,0x400141a5,0x400141a7,0x40014291,0x40014293,0x40014295,0x40014297,0x40014299,0x4001429b,0x400
1429d,0x4001429f,0x400142a1,0x400142a2,0x400142a4,0x400142a6,0x40014390,0x40014392,0x40014394,

255

0x40014396,0x40014398,0x4001439a,0x4001439c,0x4001439e,0x400143a0,0x400143a2,0x400143a4,0x400143a6,0x40014490,0x40
014492,0x40014494,0x40014496,0x40014498,0x4001449a,0x4001449c,0x4001449e,0x400144a0,0x400144a2,0x400144a4,0x400144
a6,0x40014590,0x40014592,0x40014594,0x40014596,0x40014598,0x4001459a,0x4001459c,0x4001459e,0x400145a0,0x400142a3,0
x400142a5,0x400142a7,0x40014391,0x40014393,0x40014395,0x40014397,0x40014399,0x4001439b,0x4001439d,0x4001439f,0x400
143a1,0x400143a3,0x400143a5,0x400143a7,0x40014491,0x40014493,0x40014495,0x40014497,0x40014499,

256

0x4001449b,0x4001449d,0x4001449f,0x400144a1,0x400144a3,0x400144a5,0x400144a7,0x40014591,0x40014593,0x40014595,0x40
014597,0x40014599,0x4001459b,0x4001459d,0x4001459f,0x400145a1,0x400145a2,0x400145a4,0x400145a6,0x40014690,0x400146
92,0x40014694,0x40014696,0x40014698,0x4001469a,0x4001469c,0x4001469e,0x400146a0,0x400146a2,0x400146a4,0x400146a6,0
x40014790,0x40014792,0x40014794,0x40014796,0x40014798,0x4001479a,0x4001479c,0x4001479e,0x400147a0,0x400147a2,0x400
147a4,0x400147a6,0x40014890,0x40014892,0x40014894,0x40014896,0x40014898,0x4001489a,0x4001489c,

257

0x4001489e,0x400148a0,0x400145a3,0x400145a5,0x400145a7,0x40014691,0x40014693,0x40014695,0x40014697,0x40014699,0x40
01469b,0x4001469d,0x4001469f,0x400146a1,0x400146a3,0x400146a5,0x400146a7,0x40014791,0x40014793,0x40014795,0x400147
97,0x40014799,0x4001479b,0x4001479d,0x4001479f,0x400147a1,0x400147a3,0x400147a5,0x400147a7,0x40014891,0x40014893,0
x40014895,0x40014897,0x40014899,0x4001489b,0x4001489d,0x4001489f,0x400148a1,0x400148a2,0x400148a4,0x400148a6,0x400
14990,0x40014992,0x40014994,0x40014996,0x40014998,0x4001499a,0x4001499c,0x4001499e,0x400149a0,

```
258
        0x400149a2,0x400149a4,0x400149a6,0x40014c90,0x40014c92,0x40014c94,0x40014c96,0x40014c98,0x40014c9a,0x40014c9c,0x40
        014c9e,0x40014ca0,0x40014ca2,0x40014ca4,0x40014ca6,0x40014d90,0x40014d92,0x40014d94,0x40014d96,0x40014d98,0x40014d
        9a,0x40014d9c,0x40014d9e,0x40014da0,0x400148a3,0x400148a5,0x400148a7,0x40014991,0x40014993,0x40014995,0x40014997,0
        x40014999,0x4001499b,0x4001499d,0x4001499f,0x400149a1,0x400149a3,0x400149a5,0x400149a7,0x40014c91,0x40014c93,0x400
        14c95,0x40014c97,0x40014c99,0x40014c9b,0x40014c9d,0x40014c9f,0x40014ca1,0x40014ca3,0x40014ca5,

259
        0x40014ca7,0x40014d91,0x40014d93,0x40014d95,0x40014d97,0x40014d99,0x40014d9b,0x40014d9d,0x40014d9f,0x40014da1,0x40
        014da2,0x40014da4,0x40014da6,0x40014da3,0x40014da5,0x40014da7,0x400140a8,0x400140aa,0x400140ac,0x400140ae,0x400140
        b0,0x400140b2,0x400140b4,0x400140b6,0x400141a8,0x400141aa,0x400141ac,0x400141ae,0x400141b0,0x400141b2,0x400141b4,0
        x400141b6,0x400142a8,0x400142aa,0x400142ac,0x400142ae,0x400142b0,0x400142b2,0x400142b4,0x400142b6,0x400143a8,0x400
        143aa,0x400143ac,0x400143ae,0x400143b0,0x400143b2,0x400143b4,0x400143b6,0x400144a8,0x400140a9,

260
        0x400140ab,0x400140ad,0x400140af,0x400140b1,0x400140b3,0x400140b5,0x400140b7,0x400141a9,0x400141ab,0x400141ad,0x40
        0141af,0x400141b1,0x400141b3,0x400141b5,0x400141b7,0x400142a9,0x400142ab,0x400142ad,0x400142af,0x400142b1,0x400142
        b3,0x400142b5,0x400142b7,0x400143a9,0x400143ab,0x400143ad,0x400143af,0x400143b1,0x400143b3,0x400143b5,0x400143b7,0
        x400144a9,0x400144aa,0x400144ac,0x400144ae,0x400144b0,0x400144b2,0x400144b4,0x400144b6,0x400145a8,0x400145aa,0x400
        145ac,0x400145ae,0x400145b0,0x400145b2,0x400145b4,0x400145b6,0x400146a8,0x400146aa,0x400146ac,

261
        0x400146ae,0x400146b0,0x400146b2,0x400146b4,0x400146b6,0x400147a8,0x400147aa,0x400147ac,0x400147ae,0x400147b0,0x40
        0147b2,0x400147b4,0x400147b6,0x400148a8,0x400148aa,0x400148ac,0x400148ae,0x400148b0,0x400144ab,0x400144ad,0x400144
        af,0x400144b1,0x400144b3,0x400144b5,0x400144b7,0x400145a9,0x400145ab,0x400145ad,0x400145af,0x400145b1,0x400145b3,0
        x400145b5,0x400145b7,0x400146a9,0x400146ab,0x400146ad,0x400146af,0x400146b1,0x400146b3,0x400146b5,0x400146b7,0x400
        147a9,0x400147ab,0x400147ad,0x400147af,0x400147b1,0x400147b3,0x400147b5,0x400147b7,0x400148a9,

262
        0x400148ab,0x400148ad,0x400148af,0x400148b1,0x400148b2,0x400148b4,0x400148b6,0x400149a8,0x400149aa,0x400149ac,0x40
        0149ae,0x400149b0,0x400149b2,0x400149b4,0x400149b6,0x40014ca8,0x40014caa,0x40014cac,0x40014cae,0x40014cb0,0x40014c
        b2,0x40014cb4,0x40014cb6,0x40014da8,0x40014daa,0x40014dac,0x40014dae,0x40014db0,0x40014db2,0x40014db4,0x40014db6,0
        x400148b3,0x400148b5,0x400148b7,0x400149a9,0x400149ab,0x400149ad,0x400149af,0x400149b1,0x400149b3,0x400149b5,0x400
        149b7,0x40014ca9,0x40014cab,0x40014cad,0x40014caf,0x40014cb1,0x40014cb3,0x40014cb5,0x40014cb7,

263
        0x40014da9,0x40014dab,0x40014dad,0x40014daf,0x40014db1,0x40014db3,0x40014db5,0x40014db7,0x400140c0,0x400142c0,0x40
        0144c0,0x400146c0,0x400148c0,0x40014cc0,0x400141c0,0x400143c0,0x400145c0,0x400147c0,0x400149c0,0x40014dc0,0x400140
        c2,0x400142c2,0x400144c2,0x400146c2,0x400148c2,0x40014cc2,0x400141c2,0x400143c2,0x400145c2,0x400147c2,0x400149c2,0
        x40014dc2,0x400140c4,0x400142c4,0x400144c4,0x400146c4,0x400148c4,0x40014cc4,0x400141c4,0x400143c4,0x400145c4,0x400
        147c4,0x400149c4,0x40014dc4,0x400140c6,0x400142c6,0x400144c6,0x400146c6,0x400148c6,0x40014cc6,

264
        0x400141c6,0x400143c6,0x400145c6,0x400147c6,0x400149c6,0x40014dc6,0x400140c8,0x400142c8,0x400144c8,0x400146c8,0x40
        0148c8,0x40014cc8,0x400141c8,0x400143c8,0x400145c8,0x400147c8,0x400149c8,0x40014dc8,0x400140ca,0x400142ca,0x400144
        ca,0x400146ca,0x400148ca,0x40014cca,0x400141ca,0x400143ca,0x400145ca,0x400147ca,0x400149ca,0x40014dca,0x400140cc,0
        x400142cc,0x400144cc,0x400146cc,0x400148cc,0x40014ccc,0x400141cc,0x400143cc,0x400145cc,0x400147cc,0x400149cc,0x400
        14dcc,0x400140ce,0x400142ce,0x400144ce,0x400146ce,0x400148ce,0x40014cce,0x400141ce,0x400143ce,

265
        0x400145ce,0x400147ce,0x400149ce,0x40014dce,0x400140d0,0x400142d0,0x400144d0,0x400146d0,0x400148d0,0x40014cd0,0x40
        0141d0,0x400143d0,0x400145d0,0x400147d0,0x400149d0,0x40014dd0,0x400140d2,0x400142d2,0x400144d2,0x400146d2,0x400148
        d2,0x40014cd2,0x400141d2,0x400143d2,0x400145d2,0x400147d2,0x400149d2,0x40014dd2,0x400140d4,0x400142d4,0x400144d4,0
        x400146d4,0x400148d4,0x40014cd4,0x400141d4,0x400143d4,0x400145d4,0x400147d4,0x400149d4,0x40014dd4,0x400140d6,0x400
        142d6,0x400144d6,0x400146d6,0x400148d6,0x40014cd6,0x400141d6,0x400143d6,0x400145d6,0x400147d6,

266
        0x400149d6,0x40014dd6,0x400140d8,0x400142d8,0x400144d8,0x400146d8,0x400148d8,0x40014cd8,0x400141d8,0x400143d8,0x40
        0145d8,0x400147d8,0x400149d8,0x40014dd8,0x400140da,0x400142da,0x400144da,0x400146da,0x400148da,0x40014cda,0x400141
        da,0x400143da,0x400145da,0x400147da,0x400149da,0x40014dda,0x400140dc,0x400142dc,0x400144dc,0x400146dc,0x400148dc,0
        x40014cdc,0x400141dc,0x400143dc,0x400145dc,0x400147dc,0x400149dc,0x40014ddc,0x400140de,0x400142de,0x400144de,0x400
        146de,0x400148de,0x40014cde,0x400141de,0x400143de,0x400145de,0x400147de,0x400149de,0x40014dde,

267
        0x400140e0,0x400142e0,0x400144e0,0x400146e0,0x400148e0,0x40014ce0,0x400141e0,0x400143e0,0x400145e0,0x400147e0,0x40
        0149e0,0x40014de0,0x400140e2,0x400142e2,0x400144e2,0x400146e2,0x400148e2,0x40014ce2,0x400141e2,0x400143e2,0x400145
        e2,0x400147e2,0x400149e2,0x40014de2,0x400140e4,0x400142e4,0x400144e4,0x400146e4,0x400148e4,0x40014ce4,0x400141e4,0
        x400143e4,0x400145e4,0x400147e4,0x400149e4,0x40014de4,0x400140e6,0x400142e6,0x400144e6,0x400146e6,0x400148e6,0x400
        14ce6,0x400141e6,0x400143e6,0x400145e6,0x400147e6,0x400149e6,0x40014de6,0x400140e8,0x400142e8,

268
        0x400144e8,0x400146e8,0x400148e8,0x40014ce8,0x400141e8,0x400143e8,0x400145e8,0x400147e8,0x400149e8,0x40014de8,0x40
        0140ea,0x400142ea,0x400144ea,0x400146ea,0x400148ea,0x40014cea,0x400141ea,0x400143ea,0x400145ea,0x400147ea,0x400149
        ea,0x40014dea,0x400140ec,0x400142ec,0x400144ec,0x400146ec,0x400148ec,0x40014cec,0x400141ec,0x400143ec,0x400145ec,0
        x400147ec,0x400149ec,0x40014dec,0x400140ee,0x400142ee,0x400144ee,0x400146ee,0x400148ee,0x40014cee,0x400141ee,0x400
        143ee,0x400145ee,0x400147ee,0x400149ee,0x40014dee,0x40015000,0x40015010,0x40015001,0x40015011,

269
        0x40015002,0x40015012,0x40015003,0x40015013,0x40015004,0x40015014,0x40015005,0x40015015,0x40015006,0x40015016,0x40
        015007,0x40015017,0x40015008,0x40015018,0x40015009,0x40015019,0x4001500a,0x4001501a,0x4001500b,0x4001501b,0x400150
        0c,0x4001501c,0x4001500d,0x4001501d,0x4001500e,0x4001501e,0x4001500f,0x4001501f,0x40015100,0x40015102,0x40015104,0
        x40015106,0x40015101,0x40015103,0x40015105,0x40015107,0x40015110,0x40015112,0x40015114,0x40015111,0x40015113,0x400
        15115};
270     uint8 updated_value;
271     uint8 first_fault = 0;
272     uint8 flag_2 = 0;*/
273
274     /* Reset and fault injection counters */
275     uint16 num_resets;
276     char8 num_resets_uart[5];
277     uint32 prev_faults = 0;
278     char8 prev_faults_uart[10];
279     uint32 data_read_A;
280     uint32 data_read_B;
281     uint16 data_read_C;
282     uint8 data_read_D;
283     uint8 data_write[4] = {0,0,0,0};
284     uint32 total_faults = 0;
```

```
285     char8 total_faults_uart[10];
286
287     void DMA_1_Config(void)
288     {
289               #define DMA_1_BYTES_PER_BURST 1
290               #define DMA_1_REQUEST_PER_BURST 1
291               #define DMA_1_SRC_BASE (CYDEV_PERIPH_BASE)
292               #define DMA_1_DST_BASE (CYDEV_SRAM_BASE)
293
294               DMA_1_Chan = DMA_1_DmaInitialize(DMA_1_BYTES_PER_BURST, DMA_1_REQUEST_PER_BURST, HI16(DMA_1_SRC_BASE),
    HI16(DMA_1_DST_BASE));
295               DMA_1_TD[0] = CyDmaTdAllocate();
296               CyDmaTdSetConfiguration(DMA_1_TD[0], 10, DMA_1_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
297               CyDmaTdSetAddress(DMA_1_TD[0], LO16((uint32)Module_1_SAR_WRK0_PTR), LO16((uint32)DMA_module1Data));
298               CyDmaChSetInitialTd(DMA_1_Chan, DMA_1_TD[0]);
299               CyDmaChEnable(DMA_1_Chan, 1);
300     }
301
302     void DMA_2_Config(void)
303     {
304               #define DMA_2_BYTES_PER_BURST 1
305               #define DMA_2_REQUEST_PER_BURST 1
306               #define DMA_2_SRC_BASE (CYDEV_PERIPH_BASE)
307               #define DMA_2_DST_BASE (CYDEV_SRAM_BASE)
308
309               DMA_2_Chan = DMA_2_DmaInitialize(DMA_2_BYTES_PER_BURST, DMA_2_REQUEST_PER_BURST, HI16(DMA_2_SRC_BASE),
    HI16(DMA_2_DST_BASE));
310               DMA_2_TD[0] = CyDmaTdAllocate();
311               CyDmaTdSetConfiguration(DMA_2_TD[0], 2, DMA_2_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
312               CyDmaTdSetAddress(DMA_2_TD[0], LO16((uint32)Module_2_SAR_WRK0_PTR), LO16((uint32)DMA_module2Data));
313               CyDmaChSetInitialTd(DMA_2_Chan, DMA_2_TD[0]);
314               CyDmaChEnable(DMA_2_Chan, 1);
315     }
316
317     void DMA_3_Config(void)
318     {
319               #define DMA_3_BYTES_PER_BURST 1
320               #define DMA_3_REQUEST_PER_BURST 1
321               #define DMA_3_SRC_BASE (CYDEV_PERIPH_BASE)
322               #define DMA_3_DST_BASE (CYDEV_SRAM_BASE)
323
324               DMA_3_Chan = DMA_3_DmaInitialize(DMA_3_BYTES_PER_BURST, DMA_3_REQUEST_PER_BURST, HI16(DMA_3_SRC_BASE),
    HI16(DMA_3_DST_BASE));
325               DMA_3_TD[0] = CyDmaTdAllocate();
326               CyDmaTdSetConfiguration(DMA_3_TD[0], 2, DMA_3_TD[0], DMA_1__TD_TERMOUT_EN | TD_INC_DST_ADR);
327               CyDmaTdSetAddress(DMA_3_TD[0], LO16((uint32)Module_3_DEC_SAMP_PTR), LO16((uint32)DMA_module3Data));
328               CyDmaChSetInitialTd(DMA_3_Chan, DMA_3_TD[0]);
329               CyDmaChEnable(DMA_3_Chan, 1);
330     }
331
332     void calculate_total_faults()
333     {
334         /* Read eeprom (from 8 to 32) */
335         data_read_A = CY_GET_REG8(CYDEV_EE_BASE + 32);
336         data_read_A = data_read_A << 24;
337         data_read_B = CY_GET_REG8(CYDEV_EE_BASE + 33);
338         data_read_B = data_read_B << 16;
339         data_read_C = CY_GET_REG8(CYDEV_EE_BASE + 34);
340         data_read_C = data_read_C << 8;
341         data_read_D = CY_GET_REG8(CYDEV_EE_BASE + 35);
342         total_faults = data_read_A + data_read_B + data_read_C + data_read_D;
343
344         total_faults = total_faults + prev_faults;
345
346         /* Write eeprom (from 32 to 8) */
347         data_write[3] = total_faults & 0xff;
348         data_write[2] = (total_faults >> 8) & 0xff;
349         data_write[1] = (total_faults >> 16) & 0xff;
350         data_write[0] = (total_faults >> 24) & 0xff;
351         if(CySetTemp() == CYRET_SUCCESS)
352         {
353             EEPROM_1_Write(data_write, 2);
354         }
355     }
356
357     void SAR_ADC_voter()
358     {
359         bits = 0;
360         SAR_ADC_voter_data = 0;
361         for (i = 0; i < 8; i++)
362             bit_counter[i] = 0;
```

```
363
364         for (i = 0; i < 8; i++)
365         {
366             for (j = 1; j < 10; j++)
367             {
368                 bits = (module1Data[j] & mask[i]) != 0;
369                 if (bits == 1)
370                     bit_counter[i] = bit_counter[i] + 1;
371             }
372         }
373
374         for (i = 0; i < 8; i++)
375         {
376             if (bit_counter[i] > 4)
377             {
378                 SAR_ADC_voter_data = SAR_ADC_voter_data + mask[i];
379             }
380         }
381
382         bigger = module1Data[1];
383         smaller = module1Data[1];
384         for (pointer = 2; pointer < 10; pointer++)
385         {
386             if (bigger < module1Data[pointer])
387                 bigger = module1Data[pointer];
388
389             if (smaller > module1Data[pointer])
390                 smaller = module1Data[pointer];
391         }
392         error = bigger - smaller;
393         if (error > 4)
394             SAR_ADC_voter_error_det = 1;
395     }
396
397     void main_voter()
398     {
399         error1 = abs (SAR_ADC_voter_data - module2Data[1]);
400         error2 = abs (module2Data[1] - module3Data[1]);
401         error3 = abs (module3Data[1] - SAR_ADC_voter_data);
402
403         if (error1 <= 4)
404         {
405             system_output = SAR_ADC_voter_data;
406             data_invalid_flag = 0;
407         }
408         else if (error2 <= 4)
409         {
410             system_output = module2Data[1];
411             data_invalid_flag = 0;
412         }
413         else if (error3 <= 4)
414         {
415             system_output = module3Data[1];
416             data_invalid_flag = 0;
417         }
418         else
419         {
420             system_output = 0;
421             data_invalid_flag = 1;
422         }
423
424         if ((error1 > 4) ||  (error2 > 4) || (error3 > 4))
425             main_voter_error_det = 1;
426     }
427
428     void buffer()
429     {
430         if (buf_pointer_2 > 100)
431         {
432             buf_pointer_1 = 0;
433             buf_pointer_2 = 0;
434         }
435
436         buf_1[buf_pointer_1] = module1Data[1];
437         buf_pointer_1++;
438         buf_1[buf_pointer_1] = module1Data[2];
439         buf_pointer_1++;
440         buf_1[buf_pointer_1] = module1Data[3];
441         buf_pointer_1++;
442         buf_1[buf_pointer_1] = module1Data[4];
```

```
443         buf_pointer_1++;
444         buf_1[buf_pointer_1] = module1Data[5];
445         buf_pointer_1++;
446         buf_1[buf_pointer_1] = module1Data[6];
447         buf_pointer_1++;
448         buf_1[buf_pointer_1] = module1Data[7];
449         buf_pointer_1++;
450         buf_1[buf_pointer_1] = module1Data[8];
451         buf_pointer_1++;
452         buf_1[buf_pointer_1] = module1Data[9];
453         buf_pointer_1++;
454
455         buf_2[buf_pointer_2] = module2Data[1];
456         buf_3[buf_pointer_2] = module3Data[1];
457         buf_4[buf_pointer_2] = SAR_ADC_voter_data;
458         buf_5[buf_pointer_2] = system_output;
459         buf_6[buf_pointer_2] = cycle_counter;
460         buf_7[buf_pointer_2] = SAR_ADC_voter_error_det || main_voter_error_det;
461         buf_pointer_2++;
462     }
463
464     void uart_print_time()
465     {
466         second = RTC_ReadSecond();
467         minute = RTC_ReadMinute();
468         hour = RTC_ReadHour();
469         day_month = RTC_ReadDayOfMonth();
470         month = RTC_ReadMonth();
471
472         sprintf(second_uart, "%d", second);
473         sprintf(minute_uart, "%d", minute);
474         sprintf(hour_uart, "%d", hour);
475         sprintf(day_month_uart, "%d", day_month);
476         sprintf(month_uart, "%d", month);
477
478         UART_1_PutString(&hour_uart);
479         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
480         UART_1_PutString(":");
481         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
482         UART_1_PutString(&minute_uart);
483         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
484         UART_1_PutString(":");
485         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
486         UART_1_PutString(&second_uart);
487         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
488         UART_1_PutString(" ");
489         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
490         UART_1_PutString(&month_uart);
491         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
492         UART_1_PutString("/");
493         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
494         UART_1_PutString(&day_month_uart);
495         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
496     }
497
498     void uart_print_values()
499     {
500         /* Disable interruption to fault injection */
501         Control_Reg_6_Write(1); //Used only in SRAM fault injection test
502
503         UART_1_PutString("**************************");
504         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
505         UART_1_PutString("\n\r");
506         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
507
508         sprintf(prev_faults_uart, "%d", prev_faults);
509
510         UART_1_PutString("Voter Prev. faults: ");
511         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
512
513         UART_1_PutString(&prev_faults_uart);
514         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
515
516         UART_1_PutString("\n\r");
517         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
518
519         UART_1_PutString("----------------------");
520         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
521
522         UART_1_PutString("\n\r");
```

```
523         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
524
525         UART_1_PutString("Faults in ");
526         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
527         uart_print_time();
528         UART_1_PutString(" :");
529         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
530         UART_1_PutString("\n\r");
531         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
532
533         UART_1_PutString("----------------------");
534         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
535         UART_1_PutString("\n\r");
536         do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
537         uart_pointer_1 = 0;
538
539         for (uart_pointer_2=0; uart_pointer_2 < 101; uart_pointer_2++)
540         {
541             sprintf(module1Data_1, "%d", buf_1[uart_pointer_1]);
542             uart_pointer_1++;
543             sprintf(module1Data_2, "%d", buf_1[uart_pointer_1]);
544             uart_pointer_1++;
545             sprintf(module1Data_3, "%d", buf_1[uart_pointer_1]);
546             uart_pointer_1++;
547             sprintf(module1Data_4, "%d", buf_1[uart_pointer_1]);
548             uart_pointer_1++;
549             sprintf(module1Data_5, "%d", buf_1[uart_pointer_1]);
550             uart_pointer_1++;
551             sprintf(module1Data_6, "%d", buf_1[uart_pointer_1]);
552             uart_pointer_1++;
553             sprintf(module1Data_7, "%d", buf_1[uart_pointer_1]);
554             uart_pointer_1++;
555             sprintf(module1Data_8, "%d", buf_1[uart_pointer_1]);
556             uart_pointer_1++;
557             sprintf(module1Data_9, "%d", buf_1[uart_pointer_1]);
558             uart_pointer_1++;
559
560             sprintf(module2Data_1, "%d", buf_2[uart_pointer_2]);
561             sprintf(module3Data_1, "%d", buf_3[uart_pointer_2]);
562             sprintf(SAR_ADC_voter_data_uart, "%d", buf_4[uart_pointer_2]);
563             sprintf(system_output_uart, "%d", buf_5[uart_pointer_2]);
564             sprintf(cycle_counter_uart, "%d", buf_6[uart_pointer_2]);
565             sprintf(cycle_error_uart, "%d", buf_7[uart_pointer_2]);
566
567             UART_1_PutString("Mod_1_[01]: ");
568             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
569             UART_1_PutString(&module1Data_1);
570             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
571             UART_1_PutString("\n\r");
572             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
573
574             UART_1_PutString("Mod_1_[02]: ");
575             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
576             UART_1_PutString(&module1Data_2);
577             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
578             UART_1_PutString("\n\r");
579             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
580
581             UART_1_PutString("Mod_1_[03]: ");
582             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
583             UART_1_PutString(&module1Data_3);
584             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
585             UART_1_PutString("\n\r");
586             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
587
588             UART_1_PutString("Mod_1_[04]: ");
589             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
590             UART_1_PutString(&module1Data_4);
591             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
592             UART_1_PutString("\n\r");
593             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
594
595             UART_1_PutString("Mod_1_[05]: ");
596             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
597             UART_1_PutString(&module1Data_5);
598             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
599             UART_1_PutString("\n\r");
600             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
601
602             UART_1_PutString("Mod_1_[06]: ");
```

```
603             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
604             UART_1_PutString(&module1Data_6);
605             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
606             UART_1_PutString("\n\r");
607             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
608 
609             UART_1_PutString("Mod_1_[07]: ");
610             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
611             UART_1_PutString(&module1Data_7);
612             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
613             UART_1_PutString("\n\r");
614             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
615 
616             UART_1_PutString("Mod_1_[08]: ");
617             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
618             UART_1_PutString(&module1Data_8);
619             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
620             UART_1_PutString("\n\r");
621             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
622 
623             UART_1_PutString("Mod_1_[09]: ");
624             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
625             UART_1_PutString(&module1Data_9);
626             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
627             UART_1_PutString("\n\r");
628             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
629 
630             UART_1_PutString("Mod_2: ");
631             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
632             UART_1_PutString(&module2Data_1);
633             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
634             UART_1_PutString("\n\r");
635             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
636 
637             UART_1_PutString("Mod_3: ");
638             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
639             UART_1_PutString(&module3Data_1);
640             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
641             UART_1_PutString("\n\r");
642             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
643 
644             UART_1_PutString("SAR_ADC_voter: ");
645             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
646             UART_1_PutString(&SAR_ADC_voter_data_uart);
647             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
648             UART_1_PutString("\n\r");
649             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
650 
651             UART_1_PutString("System_output: ");
652             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
653             UART_1_PutString(&system_output_uart);
654             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
655             UART_1_PutString("\n\r");
656             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
657 
658             UART_1_PutString("Cycle_counter: ");
659             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
660             UART_1_PutString(&cycle_counter_uart);
661             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
662             UART_1_PutString("\n\r");
663             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
664 
665             UART_1_PutString("Cycle_error: ");
666             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
667             UART_1_PutString(&cycle_error_uart);
668             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
669             UART_1_PutString("\n\r");
670             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
671 
672             UART_1_PutString("----------------------");
673             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
674             UART_1_PutString("\n\r");
675             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
676         }
677 
678         /* Calculate total faults */
679         calculate_total_faults();
680 
681         /* Reinitialize_faults_counter_eeprom */
682         /* Write eeprom (from 32 to 8) */
```

```
683               data_write[3] = 0;
684               data_write[2] = 0;
685               data_write[1] = 0;
686               data_write[0] = 0;
687               if(CySetTemp() == CYRET_SUCCESS)
688               {
689                         EEPROM_1_Write(data_write, 1);
690               }
691           prev_faults = 0;
692
693           /* Enable interruption to fault injection */
694           Control_Reg_6_Write(0); //Used only in SRAM fault injection test
695
696           UART_1_PutString("***** Software rst *****");
697           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
698           UART_1_PutString("----------------------");
699           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
700
701           CySoftwareReset(); //Used only in SRAM fault injection test
702       }
703
704       void uart_print_status()
705       {
706           /* Disable interruption to fault injection */
707           Control_Reg_6_Write(1); //Used only in SRAM fault injection test
708
709           sprintf(fault_counter_uart, "%d", fault_counter);
710           sprintf(target_addr_uart, "%d", target_addr);
711
712           UART_1_PutString("Tot faults: ");
713           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
714           UART_1_PutString(&fault_counter_uart);
715           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
716           UART_1_PutString(" - ");
717           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
718
719           uart_print_time();
720
721           UART_1_PutString(" ");
722           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
723           UART_1_PutString(" - ");
724           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
725           UART_1_PutString(" ");
726           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
727           UART_1_PutString(&target_addr_uart);
728           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
729           UART_1_PutString("\n\r");
730           do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
731
732           /* Enable interruption to fault injection */
733           Control_Reg_6_Write(0); //Used only in SRAM fault injection test
734       }
735
736       void uart_control_write()
737       {
738           if (error_capture == 1)
739           {
740               buffer_counter++;
741               if (buffer_counter == 50)
742               {
743                   uart_print_values();
744                   error_capture = 0;
745                   buffer_counter = 0;
746               }
747           }
748           if ((SAR_ADC_voter_error_det == 1) || (main_voter_error_det == 1))
749           {
750               error_capture = 1;
751               fault_counter++;
752               SAR_ADC_voter_error_det = 0;
753               main_voter_error_det = 0;
754               count_alive = 0;
755           }
756           else
757           {
758               count_alive++;
759               if (count_alive == 245000)
760               {
761                   uart_print_status();
762                   count_alive = 0;
```

```
763             }
764         }
765     }
766
767     void resets_counter_eeprom()
768     {
769         /* Read eeprom (from 8 to 16) */
770         data_read_A = CY_GET_REG8(CYDEV_EE_BASE);
771         data_read_A = data_read_A << 8;
772         data_read_B = CY_GET_REG8(CYDEV_EE_BASE + 1);
773         num_resets = data_read_A + data_read_B;
774
775         /* Increase reset counter */
776         num_resets++;
777
778         /* Write eeprom (from 16 to 8) */
779         data_write[1] = num_resets & 0xff;
780         data_write[0] = (num_resets >> 8) & 0xff;
781         if(CySetTemp() == CYRET_SUCCESS)
782         {
783             EEPROM_1_Write(data_write, 0);
784         }
785     }
786
787     void reinitialize_faults_counter_eeprom()
788     {
789         /* Read eeprom (from 8 to 32) */
790         data_read_A = CY_GET_REG8(CYDEV_EE_BASE + 16);
791         data_read_A = data_read_A << 24;
792         data_read_B = CY_GET_REG8(CYDEV_EE_BASE + 17);
793         data_read_B = data_read_B << 16;
794         data_read_C = CY_GET_REG8(CYDEV_EE_BASE + 18);
795         data_read_C = data_read_C << 8;
796         data_read_D = CY_GET_REG8(CYDEV_EE_BASE + 19);
797         prev_faults = data_read_A + data_read_B + data_read_C + data_read_D;
798
799         /* Write eeprom (from 32 to 8) */
800         data_write[3] = 0;
801         data_write[2] = 0;
802         data_write[1] = 0;
803         data_write[0] = 0;
804         if(CySetTemp() == CYRET_SUCCESS)
805         {
806             EEPROM_1_Write(data_write, 1);
807         }
808     }
809
810     void faults_counter_eeprom()
811     {
812         /* Read eeprom (from 8 to 32) */
813         data_read_A = CY_GET_REG8(CYDEV_EE_BASE + 16);
814         data_read_A = data_read_A << 24;
815         data_read_B = CY_GET_REG8(CYDEV_EE_BASE + 17);
816         data_read_B = data_read_B << 16;
817         data_read_C = CY_GET_REG8(CYDEV_EE_BASE + 18);
818         data_read_C = data_read_C << 8;
819         data_read_D = CY_GET_REG8(CYDEV_EE_BASE + 19);
820         prev_faults = data_read_A + data_read_B + data_read_C + data_read_D;
821
822         /* Increase fault counter */
823         prev_faults++;
824
825         /* Write eeprom (from 32 to 8) */
826         data_write[3] = prev_faults & 0xff;
827         data_write[2] = (prev_faults >> 8) & 0xff;
828         data_write[1] = (prev_faults >> 16) & 0xff;
829         data_write[0] = (prev_faults >> 24) & 0xff;
830         if(CySetTemp() == CYRET_SUCCESS)
831         {
832             EEPROM_1_Write(data_write, 1);
833         }
834     }
835
836     CY_ISR(Interrupt_1)
837     {
838         flag_1 = 1;
839     }
840
841     CY_ISR(Interrupt_4)
842     {
```

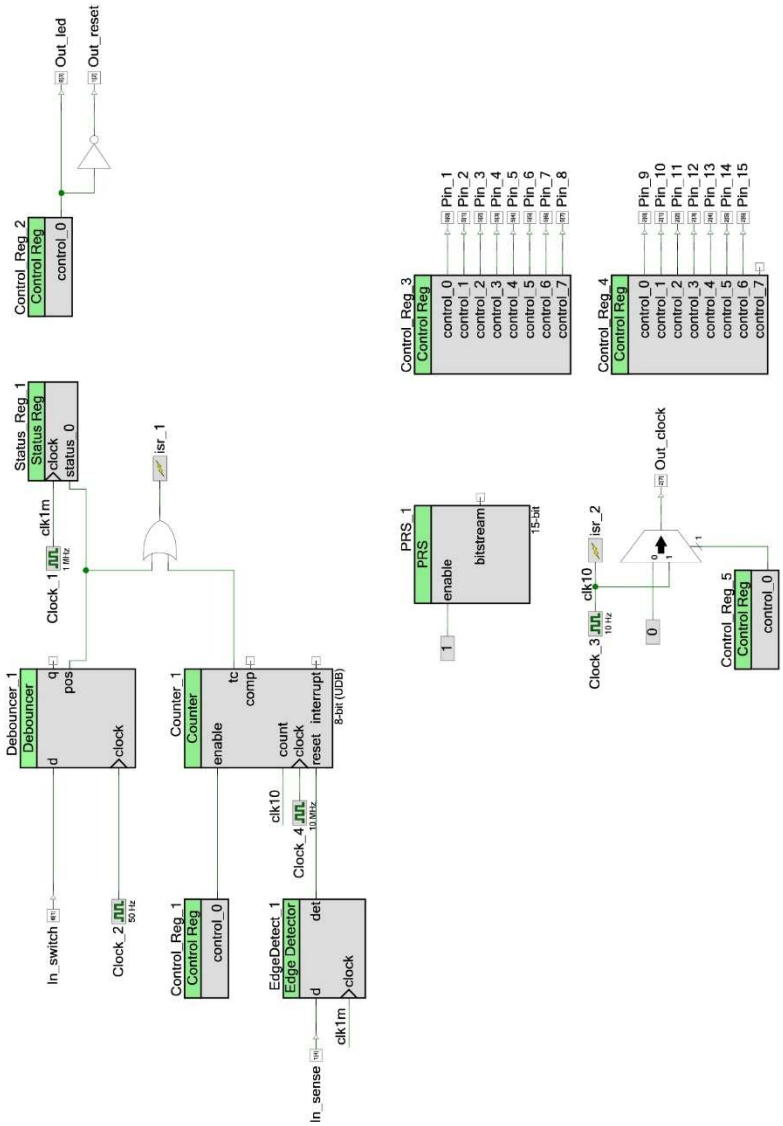
```
843         flag_2 = 1;
844     }
845 
846     int main()
847     {
848         /* Enable led of reset */
849         Control_Reg_4_Write(1);
850 
851         /* Disable led of fault injection */
852         Control_Reg_5_Write(led_fault_inject);
853 
854         /* Stop starting to stop fault injection teste */
855         while (Status_Reg_5_Read() == 0)
856         {
857         }
858 
859         /* Initialize interruptions */
860         isr_1_StartEx(Interrupt_1);
861         isr_3_Start();
862         isr_4_StartEx(Interrupt_4);
863         CyDelay(100);
864         isr_1_ClearPending();
865         isr_3_ClearPending();
866         isr_4_ClearPending();
867         CyGlobalIntEnable;
868 
869         /* Initialize EEPROM */
870         EEPROM_1_Start();
871 
872         /* Initialize UART */
873         UART_1_Start();
874 
875         /* Initialize DMA */
876               DMA_1_Config();
877               DMA_2_Config();
878               DMA_3_Config();
879 
880         /* Initialize Sample_hold */
881         Sample_Hold_1_Start();
882         Sample_Hold_1_Enable();
883         Sample_Hold_2_Start();
884         Sample_Hold_2_Enable();
885         Sample_Hold_3_Start();
886         Sample_Hold_3_Enable();
887 
888         /* Initialize modules */
889               Module_1_Start();
890               Module_2_Start();
891               Module_3_Start();
892 
893         /* Initialize counters */
894         Counter_1_Start();
895         Counter_2_Start();
896 
897         /* Initialize clock_1 */
898         Clock_1_Start();
899 
900         /* Delay to stabilize operation */
901         CyDelay(5000);
902 
903         /* First write in UART */
904 
905             /* Increase number resets in eeprom */
906             resets_counter_eeprom();
907 
908             /* Read preview fault injection in eeprom and reinitialize counter */
909             reinitialize_faults_counter_eeprom();
910 
911             /* Calculate total faults */
912             calculate_total_faults();
913 
914         sprintf(num_resets_uart, "%d", num_resets);
915         sprintf(prev_faults_uart, "%d", prev_faults);
916         sprintf(total_faults_uart, "%d", total_faults);
917 
918         if (num_resets < 10000)
919         {
920             UART_1_PutString("\n\r");
921             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
922 
```

```
923             UART_1_PutString("***** System started *****");
924             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
925 
926             UART_1_PutString("\n\r");
927             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
928 
929             UART_1_PutString("Resets: ");
930             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
931 
932             UART_1_PutString(&num_resets_uart);
933             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
934 
935             UART_1_PutString("\n\r");
936             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
937 
938             UART_1_PutString("Total faults: ");
939             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
940 
941             UART_1_PutString(&total_faults_uart);
942             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
943 
944             UART_1_PutString("\n\r");
945             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
946 
947             UART_1_PutString("Prev. faults: ");
948             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
949 
950             UART_1_PutString(&prev_faults_uart);
951             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
952 
953             UART_1_PutString("\n\r");
954             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
955 
956             UART_1_PutString("**************************");
957             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
958 
959             UART_1_PutString("\n\r");
960             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
961         }
962         else
963         {
964             UART_1_PutString("***** System started *****");
965             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
966 
967             UART_1_PutString("\n\r");
968             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
969 
970             UART_1_PutString("Prev. faults: ");
971             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
972 
973             UART_1_PutString(&prev_faults_uart);
974             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
975 
976             UART_1_PutString("\n\r");
977             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
978 
979             UART_1_PutString("***End of 10.000 resets***");
980             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
981 
982             UART_1_PutString("\n\r");
983             do {tmpStat = UART_1_ReadTxStatus();} while(~tmpStat & UART_1_TX_STS_COMPLETE);
984 
985             for (;;){}
986         }
987 
988         /* Disable led of reset */
989         Control_Reg_4_Write(0);
990 
991         /* Initialize Real time Clock */
992         RTC_TIME_DATE Start;
993 
994         Start.Sec = 0u;
995         Start.Min = 0u;
996         Start.Hour = 0u;
997         Start.DayOfMonth = 1u;
998         Start.Month = 1u;
999         Start.Year = 2014u;
1000
1001        RTC_WriteTime(&Start);
1002        RTC_Start();
```

```
1003
1004        for(;;)
1005        {
1006            /* Reset of flip-flops */
1007            Control_Reg_2_Write(1);
1008            CyDelayCycles(5);
1009            Control_Reg_2_Write(0);
1010
1011            /* Wait for sampling control */
1012            while (Status_Reg_1_Read() == 0)
1013            {
1014            }
1015
1016            /* Wait for the end of the sampling */
1017            CyDelayCycles(660);
1018
1019            /* Start conversions */
1020            Control_Reg_1_Write(1);
1021            CyDelayCycles(5);
1022            Control_Reg_1_Write(0);
1023
1024            /* Processing */
1025            SAR_ADC_voter();
1026            main_voter();
1027            buffer();
1028            uart_control_write();
1029
1030            /* Wait for the end of the conversions */
1031            while (Status_Reg_2_Read() == 0)
1032            {
1033            }
1034
1035            /* Fault injection through button */
1036            if (flag_1 == 1)
1037            {
1038                flag_1 = 0;
1039                DMA_module1Data[3] = DMA_module1Data[3] ^ 128;
1040            }
1041
1042            /* Fault injection in peripheral
1043            if (flag_2 == 1)
1044            {
1045                flag_2 = 0;
1046                if (first_fault == 1)
1047                {
1048                    updated_value = *pointer_addr;
1049                    *pointer_addr = updated_value ^ target_bit[state_bit];
1050                }
1051                prs_number = (Status_Reg_4_Read() << 8) & 0x1f00;
1052                prs_number = prs_number + Status_Reg_3_Read();
1053                if (prs_number <= 7892)
1054                {
1055                    state_bit++;
1056                    if (state_bit == 8)
1057                    {
1058                        state_bit = 0;
1059                    }
1060
1061                    first_fault = 1;
1062                    faults_counter_eeprom();
1063                    target_addr = periph_reg[prs_number];
1064                    pointer_addr = (reg8 *) periph_reg[prs_number];
1065                    *pointer_addr = *pointer_addr ^ target_bit[state_bit];
1066
1067                    led_fault_inject = !led_fault_inject;
1068                    Control_Reg_5_Write(led_fault_inject);
1069                }
1070            } */
1071
1072            /* Fault injection in SRAM */
1073            if (flag_2 == 1)
1074            {
1075                flag_2 = 0;
1076
1077                prs_number = (Status_Reg_4_Read() << 8) & 0xff00;
1078                prs_number = prs_number + Status_Reg_3_Read();
1079
1080                if (low_high_addr == 0)
1081                {
1082                    low_high_addr = 1;
```

```
1083                    target_addr = low_addr + prs_number;
1084                }
1085                else
1086                {
1087                    low_high_addr = 0;
1088                    target_addr = high_addr + prs_number;
1089                }
1090
1091                if ((target_addr != (uint32)&flag_2) && (target_addr != (uint32)&low_addr) && (target_addr != (uint32)&high_addr) && (target_addr != (uint32)&low_high_addr) && (target_addr != (uint32)&state_bit))
1092                {
1093                    faults_counter_eeprom();
1094
1095                    pointer_addr = (reg8 *) target_addr;
1096                    *pointer_addr = *pointer_addr ^ target_bit[state_bit];
1097
1098                    state_bit++;
1099                    if (state_bit == 8)
1100                    {
1101                        state_bit = 0;
1102                    }
1103
1104                    led_fault_inject = !led_fault_inject;
1105                    Control_Reg_5_Write(led_fault_inject);
1106                }
1107            }
1108
1109            /* Move DMA to temporary buffer */
1110            module1Data[1] = DMA_module1Data[1];
1111            module1Data[2] = DMA_module1Data[2];
1112            module1Data[3] = DMA_module1Data[3];
1113            module1Data[4] = DMA_module1Data[4];
1114            module1Data[5] = DMA_module1Data[5];
1115            module1Data[6] = DMA_module1Data[6];
1116            module1Data[7] = DMA_module1Data[7];
1117            module1Data[8] = DMA_module1Data[8];
1118            module1Data[9] = DMA_module1Data[9];
1119            module2Data[1] = DMA_module2Data[1];
1120            module3Data[1] = DMA_module3Data[1];
1121
1122            /* Increments cycle_counter */
1123            cycle_counter++;
1124
1125            /* Sends the running bit */
1126            running_bit = !running_bit;
1127            Control_Reg_3_Write(running_bit);
1128
1129            /* Enable interruption to fault injection */
1130            Control_Reg_6_Write(0);
1131        }
1132    }
```

APÊNDICE G – ESQUEMÁTICO DA IMPLEMENTAÇÃO DO AE DO ESTUDO DA SEÇÃO 5



APÊNDICE H – CÓDIGO QUE IMPLEMENTA O AE DO ESTUDO DA SEÇÃO 5

```
#include <project.h>

uint16 prs_value = 0;
uint8 prs_value_A = 0;
uint8 prs_value_B = 0;

CY_ISR(Interrupt_1)
{
    Control_Reg_2_Write(1);
    Control_Reg_5_Write(0);
    CyDelay(1000);
    Control_Reg_2_Write(0);
    Control_Reg_5_Write(1);
}

CY_ISR(Interrupt_2)
{
    PRS_1_Step();
    CyDelay(2);
    prs_value = PRS_1_Read();

    prs_value_A = prs_value & 0xff;
    prs_value_B = (prs_value >> 8) & 0xff;

    Control_Reg_3_Write(prs_value_A);
    Control_Reg_4_Write(prs_value_B);
}

int main()
{
    CyDelay(100);

    Control_Reg_2_Write(1);
    Control_Reg_5_Write(0);

    while (Status_Reg_1_Read() == 0)
    {
    }

    Control_Reg_2_Write(0);
    Control_Reg_5_Write(1);

    isr_1_StartEx(Interrupt_1);
    isr_2_StartEx(Interrupt_2);
    CyDelay(100);
    isr_1_ClearPending();
    isr_2_ClearPending();
    CyGlobalIntEnable;

    Counter_1_Start();
    PRS_1_Start();

    Control_Reg_1_Write(1);

    for(;;)
    {

    }
}
```

APÊNDICE I – TRABALHOS PUBLICADOS/SUBMETIDOS

CHENET, C. P. et al. Exploring design diversity redundancy to improve resilience in mixed-signal systems. Submetido para **Microelectronics Reliability** em 19 de abril de 2015.

CHENET, C. P.; LANOT, A. J. C.; BALEN, T. R. Design diversity redundancy with spatial-temporal voting applied to data acquisition systems. In: IEEE LATIN AMERICAN TEST WORKSHOP, 15., 2014, Fortaleza. **Proceedings…** Piscataway: IEEE, 2014. p. 1-6.

CHENET, C. P.; LANOT, A. J. C.; BALEN, T. R. Design diversity redundancy applied to a data acquisition system: a case study. In: WORKSHOP ANUAL SOBRE OS EFEITOS DA RADIAÇÃO IONIZANTE EM COMPONENTES ELETRÔNICOS E FOTÔNICOS DE USO AEROESPACIAL, 2013, São José dos Campos. **Proceedings…** São José dos Campos: Instituto de Estudos Avançados, 2013. p. 23-27.