UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

IURI ALBANDES CUNHA GOMES

Uso de Redundância Modular Tripla Aproximada para Tolerância a Falhas em Circuitos Digitais

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Profª Drª Fernanda G. de Lima Kastensmidt
Orientadora

Porto Alegre, fevereiro de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Gomes, Iuri Albandes Cunha

Uso de Redundância Modular Tripla Aproximada para Tolerância a Falhas em Circuitos Digitais / Iuri Albandes Cunha Gomes. – 2014.

Orientador: Fernanda Gusmão de Lima Kastensmidt;

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2014.

1. Lógica Aproximada 2. Redundância Modular Tripla 3. Tolerância a falhas 4. Efeitos de Radiação 5. Falhas transientes I. Kastensmidt, Fernanda Gusmão de Lima. II. Uso de Redundância Modular Tripla Aproximada para Tolerância a Falhas em Circuitos Digitais

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Vladimir Pinheiro do Nascimento
Diretor do Instituto de Informática: Prof. Luís da Cunha Lamb
Coordenador do PPGC: Prof. Luigi Carro
Bibliotecário-Chefe do Instituto de Informática: Alexsander Borges Ribeiro

The use of Approximate-TMR for fault-tolerant digital circuits

ABSTRACT

This work consists in the study about the fault tolerance technique TMR in conjunction with approximate computing to mitigate transient faults in digital circuits. The use of Triple Modular Redundancy (TMR) with majority voters can guarantee full single fault masking coverage for a given circuit against transient faults. However, it presents a minimum area overhead of 200% compared to the original circuit. In order to reduce area overhead without compromising significantly the fault coverage, TMR can use approximated circuits approach to generate redundant modules that are optimized in area compared to the original module. Initial studies of this technique have shown that it is possible to reach a good balance between fault coverage and area overhead cost. In this work, we do a further analysis of this approach by using a new method to compute approximate functions and to select the best combinations of approximate circuits targeting the highest fault coverage possible. We use TMR circuits composed exclusively by complex gates and multi-level logic gates. All the tests are done using electrical fault injection, using NGSPICE, and in logical level using the fault injection tool designed specifically for this study. Results show that area overhead can be reduced greatly from 200% to 85%and still reaching fault coverage of more than 95%.

RESUMO

Este trabalho consiste no estudo acerca da técnica de redundância modular tripla usando circuitos aproximados para tolerância a falhas transientes em circuitos digitais. O uso da técnica redundância modular tripla tradicional, conhecida como TMR, garante mascaramento lógico total contra falhas transiente únicas para um dado circuito. No entanto a técnica TMR apresenta um custo extra em área de no mínimo 200% quando comparado com o circuito original. De modo a reduzir o custo extra em área sem comprometer significativamente a cobertura de falhas, a TMR pode usar uma abordagem de circuitos aproximados para gerar módulos redundantes, estes sendo otimizados em área quando comparados com o módulo original. Estudos iniciais desta técnica demonstraram que é possível obter um bom equilíbrio entre custo extra de área e capacidade de mascaramento de falhas. Neste trabalho, aprofundamos a análise desta abordagem utilizando um novo método para computar as funções lógicas aproximadas e a seleção da melhor composição e estrutura dos circuitos aproximados, buscando a maior cobertura a falhas possível. Usamos circuitos TMR compostos por lógica aproximada contendo portas lógicas complexas com lógica aproximada ou com portas lógicas em multi-nível. Todos os testes foram feitos através de injeção de falhas em nível elétrico e em nível lógico. Resultados mostraram que a área pode ser reduzida significativamente, de 200% para próximo de 85%, e ainda sim alcançar um mascaramento de falhas superior a 95%.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

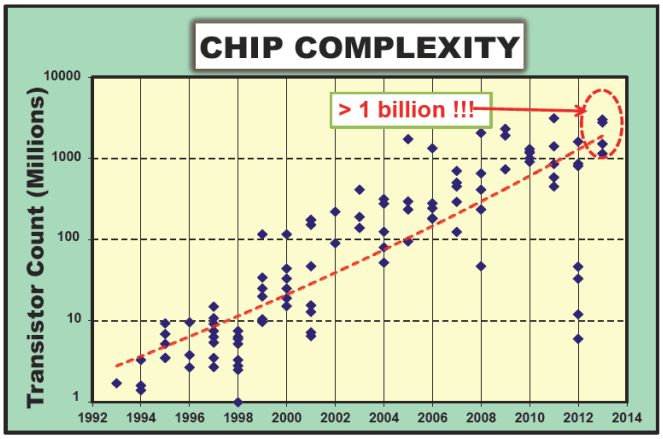
ATMR	Approximate-TMR
BDD	Binary Decision Diagrams
CI	Circuito Integrado
DICE	Dual Intelock Cell
DRAM	Dynamic Random Access Memory
DTMR	Diversity-TMR
DUT	Design Under Test
DWC	Duplication With Comparison
EDAC	Error Detection and Correction
ELT	Enclosed Layout Transistor
FIT	Failure In Time
GAA	Gate All-Around
HDL	Hardware Description Language
LET	Linear Energy Transfer
MBU	Multiple Bit Upset
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
MTTF	Mean Time To Failure
ODC	Observability Don't Cares
RHBD	Radiation Hardening By Design
RMT	Reduntant Multi-Threading
SEE	Single Event Effect
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SOI	Silicon On Insulator
SPICE	Simulated Program with Integrated Circuits Emphasis
SRAM	Static Random Access Memory
STMR	Stateful-TMR
TID	Total Ionising Dose
TMR	Triple Modular Redundancy
VHDL	VHSIC Description Language
VHSIC	Very High Speed Integrated Circuit

Sumário

1. INTRODUÇÃO

A evolução da tecnologia de fabricação de circuitos integrados do tipo CMOS no que diz respeito à miniaturização dos transistores e ao aumento da densidade lógica de transistores por circuito integrado, pode ser vista na Figura 1. O gráfico mostra o aumento na quantidade de transistores usados nos circuitos integrados durantes as ultimas duas décadas. Segundo (International Solid-State Circuits Conference 2013) já são comuns microprocessadores constituídos por mais de dois bilhões de transistores. Outra métrica para se avaliar a evolução da indústria de semicondutores é a freqüência de operação dos circuitos integrados, a Figura 2 mostra o crescimento no desempenho das aplicações durante as ultimas décadas. Pode-se observar que a partir do ano de 2006 a freqüência de operação parou de crescer, segundo (International Solid-State Circuits Conference 2013) a perda no desempenho é causada pela diminuição na tensão de alimentação do circuito, com o objetivo de diminuir o consumo energético do sistema, sendo desta forma possível obter uma troca entre desempenho e consumo de potência, sendo o gasto energético umas das preocupações emergentes na indústria. Mesmo com as tensões de alimentação diminuindo a cada novo nó tecnológico a quantidade cada vez maior de transistores continua tornando o consumo de potência uma séria preocupação para a indústria. A Figura 3 ilustra o aumento no consumo de potência nos últimos anos.

Figura 1: Evolução da densidade de transistores nos circuitos integrados nas ultimas décadas.

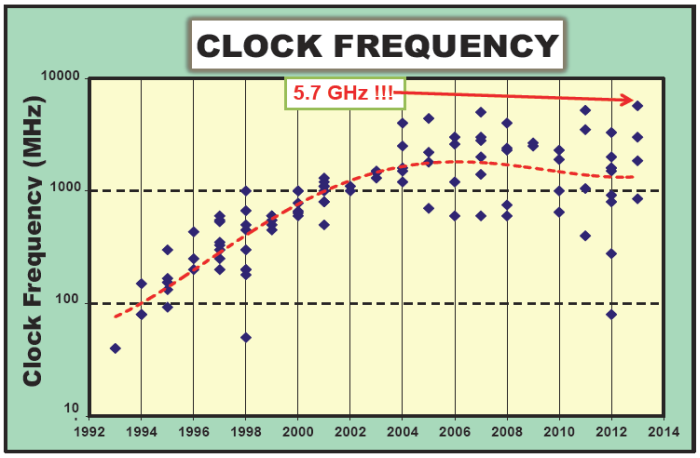


(International Solid-State Circuits Conference 2013)

A miniaturização da tecnologia traz diversos benefícios para os circuitos integrados, entretanto a diminuição das dimensões dos transistores faz com que os dispositivos eletrônicos se tornem mais suscetíveis às falhas causadas pela incidência de

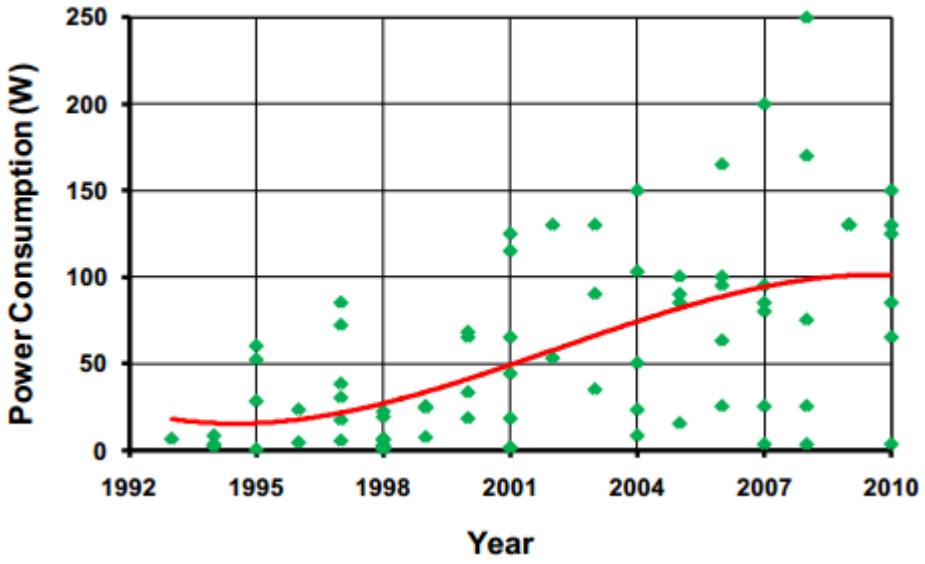
radiação. Atualmente veículos espaciais dependem em larga escala de equipamentos eletrônicos, desta forma a radiação espacial se torna uma questão importante no desenvolvimento das aplicações aeroespaciais. A evolução da tecnologia já faz com que até mesmo aplicações terrestres sejam potencialmente sensíveis a falhas causadas por radiação em nível terrestre (Velazco, Fouillat and Reis 2007).

Figura 2: Evolução da freqüência de operação dos circuitos integrados nas ultimas décadas.



(International Solid-State Circuits Conference 2013)

Figura 3: Evolução do consumo de potência nas ultimas décadas.



(International Solid-State Circuits Conference 2010)

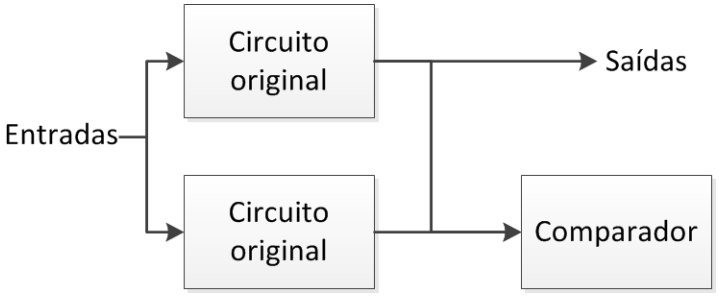
A larga quantidade de aplicações que passaram a utilizar dispositivos eletrônicos faz dos erros causados pela incidência de radiação uma questão não negligenciável já que os mesmos podem ter efeitos graves numa aplicação, afetando diretamente a confiabilidade dos circuitos integrados.

As falhas por radiação são causadas em razão da incidência de partículas energéticas como prótons, elétrons e íons pesados presentes no espaço, e da interação de nêutrons com o material que gera partículas secundarias que podem ionizar o material. (Velazco, Fouillat and Reis 2007). A diminuição dos transistores faz com que os mesmos tornem-se mais sensíveis à ionização causada por partículas de radiação, dessa forma podendo gerar falhas transientes no circuito. Além disso, o aumento da freqüência de operação faz com que mesmo as falhas transientes pequenas sejam capturadas pelas lógicas seqüenciais dos circuitos integrados. Esta falha transiente pode se propagar através da lógica e ser armazenada erradamente por um elemento de memória do circuito. Se a falha é armazenada no elemento de memória, se é observado um erro. De acordo o mascaramento lógico da aplicação uma falha pode causar um erro no funcionamento do sistema, podendo gerar um problema simples ou grave na aplicação.

Para proteger um sistema contra falhas transientes são necessárias técnicas de tolerância à falhas capazes de detectar, mascarar ou corrigir falhas transientes. As técnicas implementadas em hardware são baseadas normalmente em redundância espacial, temporal e de informação para detectar ou corrigir falhas. Uma vez a falha sendo mascarada ou detectada, faz-se necessário corrigi-la caso a arquitetura não consiga corrigi-la sozinha. Conforme as técnicas de hardware implementadas o sistema sofre impactos diferentes como a diminuição na freqüência de operação, aumento em área e potência, além de possuir um alto custo de projeto e fabricação.

Todo o circuito integrado pode ser duplicado, esse método é conhecido como Duplicação com Comparação (DWC, *Duplication With Comparison*), onde há um comparador capaz de detectar falhas que sejam manifestadas como erros nas saídas de um dos módulos, a Figura 4 elucida o DWC. Pode-se também triplicar todo ou parte de um circuito usando a redundância modular tripla (TMR, *Triple Modular Redundancy*), onde um votador de maioria pode mascarar falhas que se manifestem como erro em um dos três módulos (Figura 5). A correção das falhas pode acontecer com a inicialização do sistema, com votadores em laço ou com alguma outra técnica em nível lógico para corrigir a falha.

Figura 4: Técnica Duplicação com Comparação (DWC)

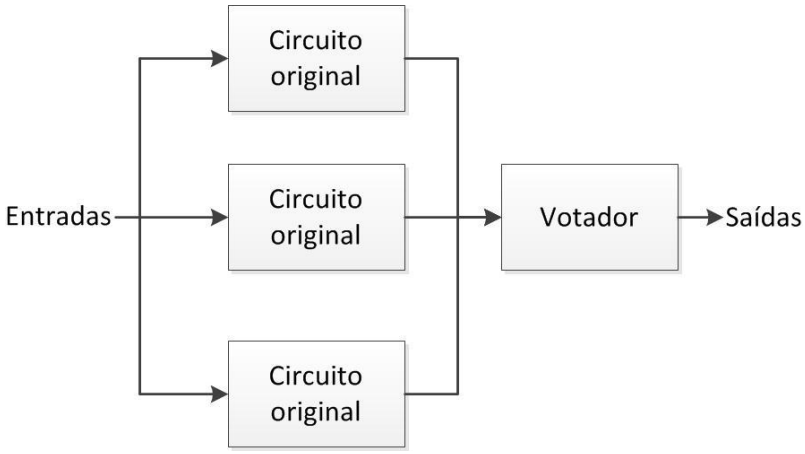


Uma das técnicas de proteção implementada em hardware mais conhecida é a redundância modular tripla (TMR, *triple modular redundancy*). A técnica TMR mais tradicional usa um mascaramento lógico extremo em conjunto com votadores para

mascarar falhas transientes. Essa técnica garante 100% de proteção contra falhas transientes únicas (exceto quando o votador é afetado por uma falha), no entanto possui um custo extra em área de mais de 200%.

A abordagem TMR, descrita anteriormente, pode garantir um mascaramento lógico completo dos erros causados por falhas transientes únicas, no entanto essa técnica possui um custo extra em área de 200% (aumento de área). A fim de reduzir o alto custo de área é possível usar uma abordagem com conceitos da computação aproximada, desta forma os módulos redundantes do circuito teriam uma cobertura menor às falhas, no entanto gerariam um custo extra de área menor. O uso de lógica simplificada nos módulos replicados irá reduzir a cobertura, mas também irá reduzir o custo de área, permitindo uma troca entre ao mascaramento de falhas e o aumento de área para algumas aplicações (Sánchez-Clemente, et al. 2012)(Sierawski, Bhuva and Massengill 2006).

Figura 5: Técnica de Redundância Modular Tripla (TMR).



As técnicas de redundância propiciam um aumento na confiabilidade do circuito integrado, no entanto o aumento no numero de transistores consequentemente aumenta o consumo de energético. Desta forma se faz necessário a busca por técnicas alternativas de tolerância a falha que não venham a aumentar o numero de transistores de forma tão impactante.

Essa dissertação procura aprofundar-se em uma variação do TMR que utiliza conceitos de computação aproximada (*approximate computing*), a técnica foi denominada *approximate-TMR* (ATMR). Esta técnica reduz o tamanho dos blocos replicados do TMR, utilizando circuitos aproximados, de forma a permitir uma troca entre a capacidade do mascaramento de falhas e o aumento de área devido à redundância espacial. Alguns circuitos estudados obtiveram 95% dos nós protegidos, em vez dos 100% do TMR tradicional, com um aumento de área de 100% contra os 200% de aumento do TMR tradicional.

Este trabalho esta organizado da seguinte forma: O Capitulo Dois descreve os efeitos de falhas transientes; O Capitulo Três aborda o estado da arte no que diz respeito a técnicas de tolerância a falhas; O Capitulo Quatro aborda a metodologia desenvolvida

para computação, testes e análises de funções e circuitos aproximados; O Capitulo Cinco apresenta os resultados obtidos para a técnica ATMR; e por fim o Capitulo Seis apresenta as conclusões desta dissertação os trabalho futuros a serem realizados.

2. FALHAS TRANSIENTES

Falhas induzidas por radiação vêm se tornando uma forte ameaça ao correto funcionamento dos dispositivos eletrônicos comerciais. A compreensão dos efeitos da radiação nos dispositivos eletrônicos tornou se importante, especialmente no caso de aplicações espaciais, aviônicas e militares, já que a exposição destes circuitos à partículas energéticas e/ou fótons pode resultar em efeitos graves na aplicação. Diversos casos de mau funcionamento ou falhas terminais de aplicações aeroespaciais já foram relatados (Velazco, Fouillat and Reis 2007), mais recentemente o caso ocorrido em 2008 onde uma aeronave Airbus 330-303 efetuou duas descidas abruptas sucessivamente, uma de 150m e outro de 120m, ferindo seriamente 12 pessoas, a causa do incidente foi uma falha no sistema de computador de bordo, acredita-se que induzida por raios cósmicos. (Microsemi 2011)

Existem dois tipos de interação entre a partícula e o material semicondutor: a ionização direta, gerada pela própria partícula, e a ionizante indireta, gerada por partículas secundarias derivadas da reação entre a partícula primaria e o material colidido. Essa ionização, direta ou indireta, gera um acumulo de carga no que é coletado pelo nó atingido, gerando uma perturbação no nível de tensão do mesmo.

Nas seções seguintes serão elucidados alguns pontos chaves para compreensão dos efeitos causados pela exposição dos circuitos eletrônicos à radiação e por conseqüência à partículas energéticas. Serão revisados os ambientes onde as partículas energéticas são geradas, como as mesmas interagem/colidem com os dispositivos eletrônicos, causando os *single event effects* (SEE), e por fim se entrara de forma mais aprofundada nos SEE, seus efeitos e como mitigá-los.

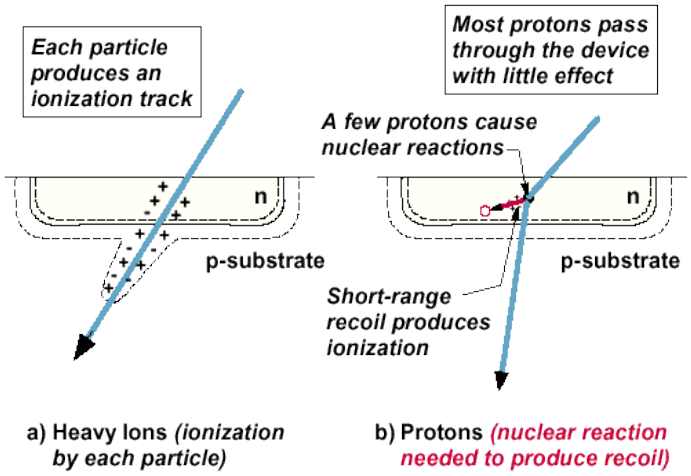
Single Event Effects (SEE) são causados quando partículas energéticas presentes no espaço, como prótons, elétrons e íons pesados (prótons), colidem com uma área sensível do circuito eletrônico, depositando carga na região da junção p-n do transistor. Ou quando nêutrons presentes na atmosfera terrestre colidem com o material semicondutor provocando partículas secundarias (normalmente do tipo alfa) e essas ionizam o material, depositando certa carga na junção p-n. Dependendo de uma série de fatores um SEE pode gerar um efeito não observável, perturbar a operação do circuito de forma transiente, mudar um estado lógico ou até mesmo causar danos permanentes ao dispositivo eletrônico (Dodd and Massengil 2003). Nesta seção iremos examinar os mecanismos físicos básicos que causam SEE nos circuitos eletrônicos. Nosso foco está

limitado aos SEE não destrutivos, examinando os mecanismos e características dos efeitos *single event upsets* (SEU) e *single event transients* (SET).

2.1 Mecanismos físicos da falha transiente de Radiação

Existem duas formas pelas quais uma partícula energética pode depositar cargas em um dispositivo semicondutor: ionização direta pela partícula em si, Figura 6(a), e ionização por partículas secundarias geradas pela colisão entre a partícula incidente e o material atingido, Figura 6(b). Os dois mecanismos podem levar ao mau funcionamento de um circuito devida à carga coletada pelo material atingido.

Figura 6: Mecanismos de deposição de carga de um SEE: (a) ionização direta; (b) ionização indireta.



A. Ionização direta

Quando uma partícula energética colide com um material semicondutor, ela liberta pares de elétron-lacuna pelo caminho conforme perde sua energia. A partícula energética repousa no material ao esgotar toda sua energia, a geração de pares elétron-lacuna e a posterior coleta de carga em decorrência do evento, resulta em um pulso de corrente no dispositivo. O formato desse pulso decorre de dois mecanismos distintos que ocorrem após a passagem da partícula incidente: inicialmente por deriva (*drift*) e, posteriormente, por difusão (*diffusion*). Frequentemente se usa transferência linear de energia (LET, *linear energy transfer*) para descrever o deposito de energia por unidade de distância de uma partícula quando a mesma atravessa um material.

B. Ionização Indireta

Apesar da ionização direta por partículas leves normalmente não depositar carga suficiente para causar SEE isto não significa que essas partículas podem ser ignoradas. Tanto prótons quanto nêutrons podem produzir falhas transientes através da ionização indireta. Conforme nêutrons e prótons, altamente energizados, atravessam o semicondutor eles podem colidir com os núcleos do material gerando algumas reações nucleares.

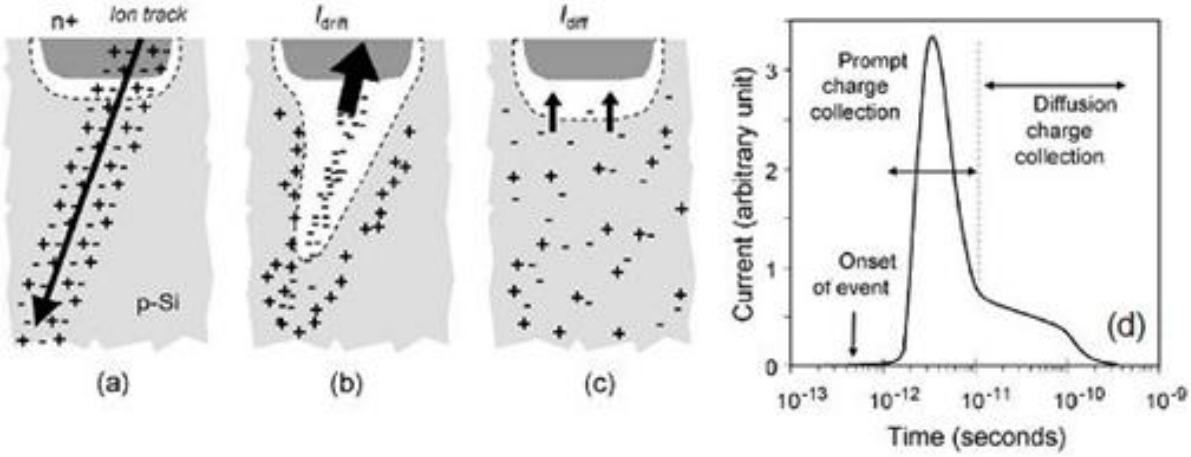
O produto de qualquer uma destas reações nucleares pode depositar carga por onde atravessam, por meio da ionização direta. Essas partículas geradas são muito mais pesadas que o próton ou o nêutron que as gerou, desta forma elas depositam uma densidade de carga maior enquanto viajam sendo assim capazes de gerar SEE.

C. Coleta de carga

Normalmente as junções p-n com polaridade reversa são as áreas mais sensíveis em um circuito integrado quando se diz respeito a falhas causadas por partículas energéticas, principalmente se a junção está flutuando ou sendo carregada de forma fraca (quando o transistor que fornece a corrente, e por consequência a tensão do nó, é pequeno, tornando mais difícil a manutenção do estado do nó). O forte campo presente na zona de depleção da junção p-n, polarizada de forma reversa, pode coletar de forma muito eficiente as cargas induzidas por partículas através do processo de deriva (*drift*) gerando a uma corrente transiente no local.

A Figura 7 ilustra os processos que ocorrem durante a colisão de uma partícula energética com uma junção p-n. Quando o caminho ionizado resultante atravessa ou passa perto da região de depleção os portadores são coletados rapidamente pelo campo elétrico gerando uma perturbação transiente de corrente/tensão no nó (Figura 7a). Uma característica importante é a distorção que ocorre no potencial eletrostático da região de depleção na forma de funil. Esse funil aumenta significativamente a coleta de carga por deriva ao aprofundar o campo elétrico no substrato (Figura 7b). Uma carga extra é coletada conforme os elétrons difundem-se na região de depleção até todos os portadores serem coletados, recombinados ou difundidos (Figura 7c). O gráfico na Figura 7d elucida a diferença na coleta de carga por deriva e por difusão, no caso da coleta por deriva a mesma ocorrem dentro de nanosegundo, já a difusão ocorre numa escala de tempo maior (centenas de nanosegundos).

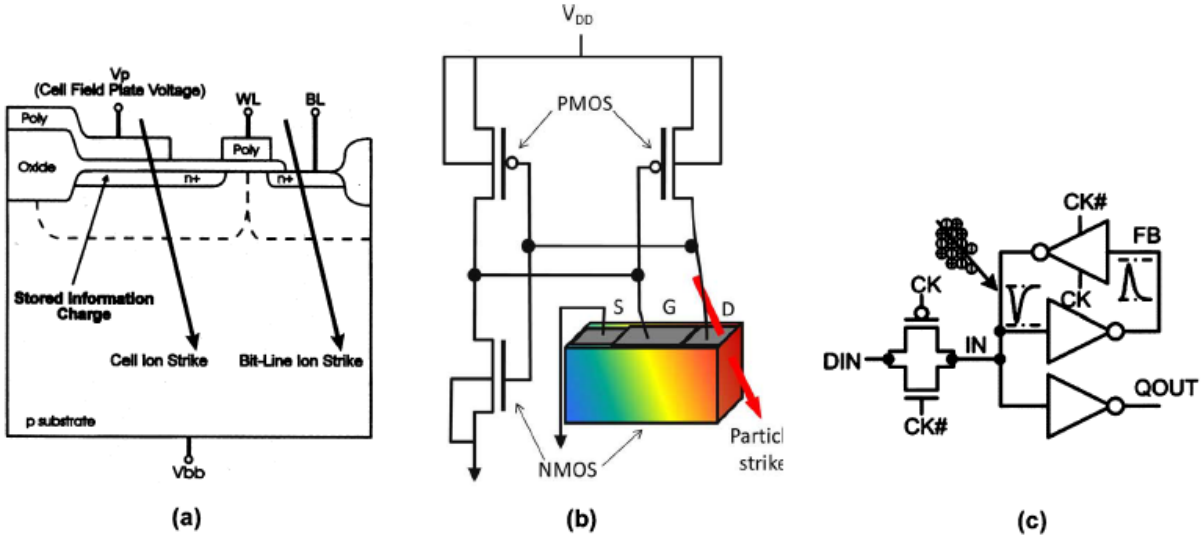
Figura 7: Ilustração da coleta de carga em uma junção p-n de silício (a) imediatamente após a colisão de um íon pesado, (b) durante a coleta de carga por deriva,e o efeito funil (c) durante a coleta de carga por difusão (d) gráfico da corrente induzida em função do tempo (Baumann 2005)



2.2 Definição de Single Event Upset (SEU)

Single event upset (SEU), evento de inversão único em tradução livre, é definido pela inversão do valor armazenado no elemento de memória, essa inversão de valores é comumente chamada de *bit-flip*. Esta falha é temporária, já que pode ser corrigida na próxima captura do elemento de memória, porém esta falha, se propagada, pode gerar erro na execução do circuito. Um SEU ocorre quando uma partícula colide com uma área sensível do elemento de memória e deposita uma carga mínima suficiente no material para ocasionar a inversão do valor armazenado. Esse elemento pode ser uma memória dinâmica (DRAM) (Figura 8a), memória estática (SRAM) (Figura 8b) ou um flip-flop (Figura 8c), e a carga mínima para ocasionar a inversão do valor armazenado é chamada de carga critica (Q_{crit}). O SEU é um fenômeno reversível, o estado da célula pode ser recuperado por uma operação de escrita usual. A taxa de falhas de SEU, a SER (*soft error rate*), normalmente é expressa em falhas no tempo (FIT, *failure in time*), a quantidade de falhas em 10^9 horas (um bilhão de horas).

Figura 8: *single event upset* (SEU) em (a) memória DRAM; (b) memória SRAM; (c) Flip-flop.(Dodd and Massengil 2003) (Munteanu and Autran 2008)



2.3 Definição de Single Event Transient (SET)

Single event transient (SET), evento transiente único ou falha transiente única, em tradução livre, constitui uma perturbação de tensão/corrente transiente gerada quando uma partícula energética atinge um nó contido numa parte combinacional do circuito integrado. Com a miniaturização constante do tamanho da tecnologia CMOS já ficou claro que o SET tornou-se um mecanismo significativo nas taxas de erro. O escalonamento da tecnologia veio acompanhando de maiores frequências de operação, menores tensões de alimentação e margens de ruídos menores, tornando maior a sensibilidade do circuito a SET.

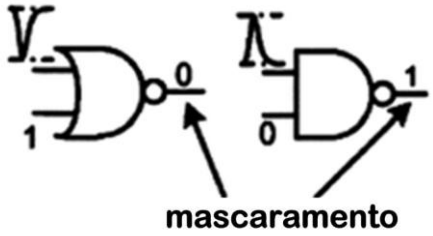
Qualquer nó do circuito combinacional pode ser afetado por uma falha e gerar uma perturbação transiente na tensão de amplitude e duração suficiente opera ser

propagada ao longo do circuito combinacional até ser capturada por um elemento de memória. No entanto somente alguns transientes são capturados, a chance de um SET ser capturado envolve três questões: a probabilidade de existir um caminho funcionalmente sensível ao SET entre o nó e o elemento sequencial; a taxa com a qual o SET perde força a cada nível lógico que atravessa até alcançar o elemento sequencial; e a chance do pulso transiente gerado ser efetivamente capturado e armazenado no elemento sequencial. Essas três questões levam a três fenômenos de mascaramento:

A. Mascaramento Lógico

O mascaramento lógico ocorre quando uma falha afeta um nó que não é capaz modificar a saída da porta lógica seguinte. Por exemplo, quando um SET propaga até a entrada da porta lógica NOR (NAND), mas uma das outras entradas é quem controla o estado da saída, no caso de uma NOR (NAND) a entrada esta em 1(0), o SET será completamente mascarado e a saída permanecera intocada. A Figura 9 elucida a questão.

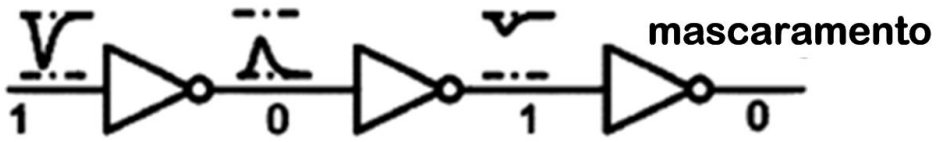
Figura 9: Mascaramento lógico.(Munteanu and Autran 2008)



B. Mascaramento Elétrico

Esse tipo de mascaramento quando o pulso elétrico gerado pelo SET é atenuado conforme ele se propaga por outras portas lógicas. Esse efeito de atenuação ocorre com os transientes que possuem uma largura de banda maior que a frequência de corte da porta lógica (Munteanu and Autran 2008). Nestes casos a amplitude do transiente pode ser reduzida e seus tempos de subida e descida aumentados e, eventualmente, o pulso pode desaparecer. No entanto, em alguns casos pulsos de baixa frequência com grandes amplitudes podem acabar sendo amplificados. A Figura 10 elucida o efeito de mascaramento elétrico em uma cadeia de inversores.

Figura 10: Mascaramento elétrico em uma cadeia de inversores. (Munteanu and Autran 2008)

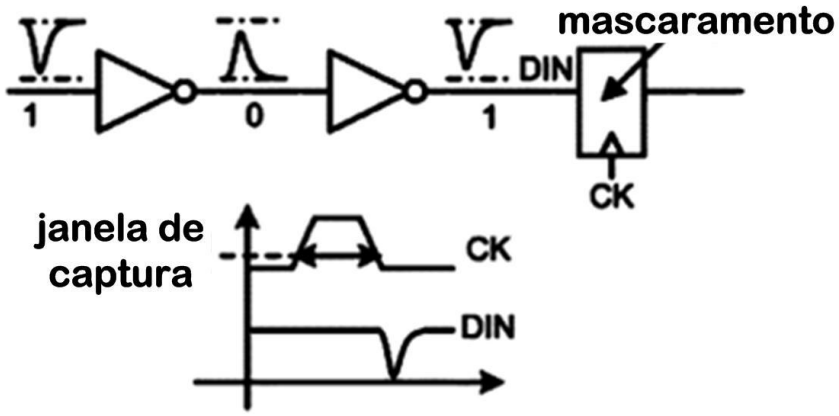


C. Mascaramento Temporal

O mascaramento temporal, ou mascaramento por janela de captura (*latching-window masking*), ocorre quando o SET, resultante da colisão de uma partícula energética, propaga até um circuito seqüencial porém não atende os requisitos temporais (bordas relógio e tempos de *setup* e *hold*) necessários para ser capturado. A Figura 11

elucida um caso de mascaramento temporal no qual o SET chega atrasado em relação a borda de relógio (*CK*), não sendo capturado.

Figura 11: Mascaramento temporal. (Munteanu and Autran 2008)



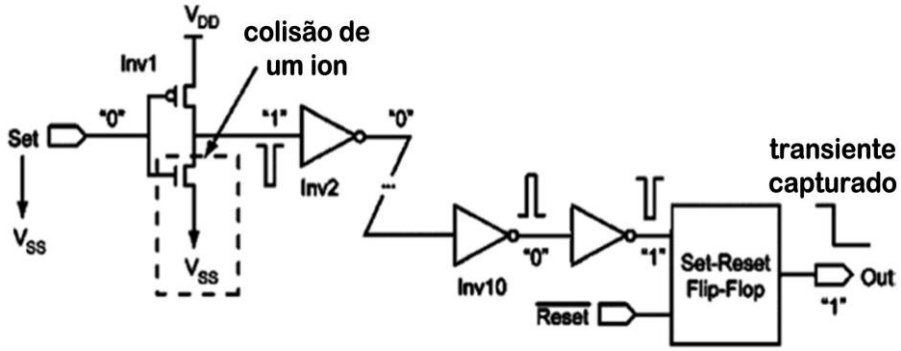
2.4 Modelagem computacional de SEE

Para poder propor e avaliar as técnicas de tolerância a falhas, e melhor compreender o funcionamento dos SEE, são necessários métodos de simulação computacional capazes de modelar os mecanismos e efeitos de uma partícula energética quando a mesma atinge uma área sensível do circuito elétrico. Podemos dividir os métodos de simulação conforme a abstração de seus mecanismos e a precisão dos seus resultados: simulação em nível lógico, em nível elétrico, simulação numérica total em 3D e simulação de modo-misto.

A. Simulação em nível lógico

A simulação de nível lógico permite a análise funcional do circuito e como os estados dos nós da aplicação simulada e suas saídas quando o mesmo sofre com um SEU ou SET. Neste nível de abstração a simulação é feita através de uma descrição comportamental em RTL (*register transfer level*), descrição das portas lógicas ou através de uma descrição dos transistores que constituem o circuito. Linguagens de descrição de hardware (HDL, *hardware description language*) como VHDL e Verilog são comuns para este tipo de simulação. Em alguns casos é possível simular o tempo de atraso das portas lógicas e desta forma obter informações dos sobre questões temporais do design em teste (DUT, *design under test*) além dos estados lógicos. Neste tipo de simulação um SET ou SEU é modelado forçando um valor lógico (verdadeiro, falso ou desconhecido), em um sinal ou nó do DUT, de forma explicita, sendo possível definir somente a largura do transiente. A **Erro! Auto-referência de indicador não válida.** elucida o *testbench* em nível lógico de uma aplicação, neste caso o SET foi propagado e acabou capturado pelo flip-flop.

Figura 12: Simulação de um SET em nível lógico.

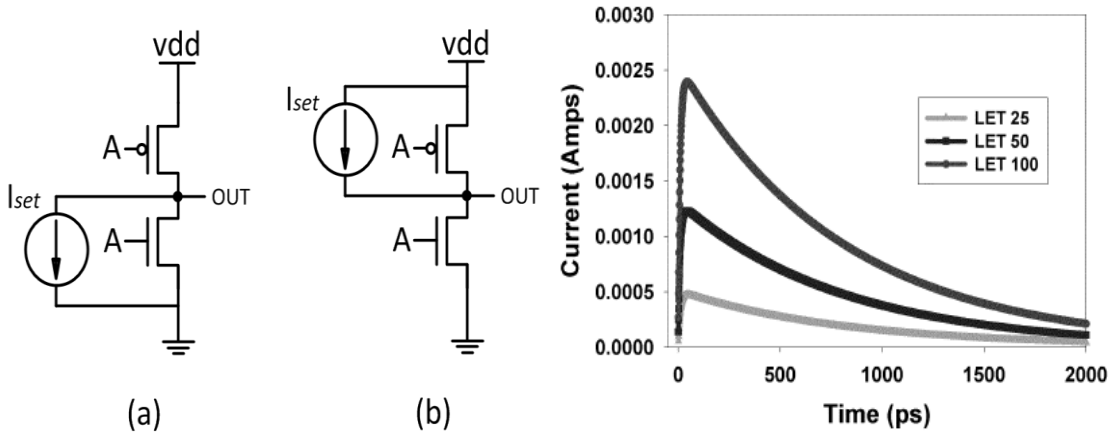


B. Simulação em nível de circuito

A simulação em nível de circuito, também chamada de simulação em nível elétrico, é realizada utilizando simuladores elétricos. Tais softwares resolvem sistemas de equações que descrevem o comportamento de circuitos elétricos. Os modelos de comportamento elétrico estático/dinâmico de diversos dispositivos (transistores, diodos, resistores, capacitores, etc) são os componentes básicos usados por estes simuladores para descrever o circuito. Esses modelos são geralmente baseados em formulas analíticas, os modelos avançados provêm uma alta precisão com um baixo esforço computacional. Essas descrições prévias dos comportamentos dos dispositivos também são denominadas de modelos compactos.

A modelagem dos SEE normalmente é feita através de uma fonte de corrente (Figura 13). Essa abordagem é adequada para muitos casos, mas possui algumas limitações. A precisão da corrente transiente usada pode afetar consideravelmente a precisão do circuito simulado. O gráfico da Figura 13 exemplifica três curvas geradas por fontes de corrente, o SET é modelado com uma exponencial dupla, a carga depositada pela partícula simulada é obtida integrando a curva da corrente em relação ao tempo(Gadlage, et al. 2004).

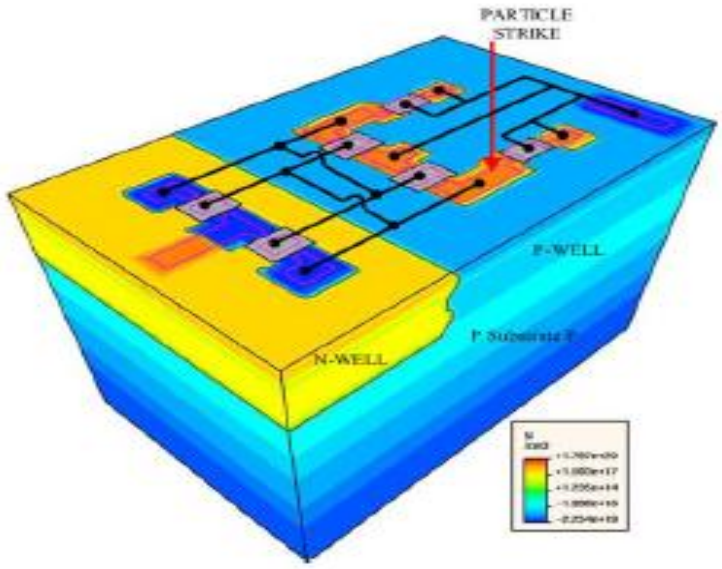
Figura 13: Modelagem de SEE utilizando fontes de corrente. (a) SET 1→0→1; (b) SET 0→1→0. (c) Curvas exponenciais duplas modeladas por uma fonte de corrente.



C. Simulação numérica total em 3D

Este tipo de simulação apresenta os resultados mais precisos para o estudo de SEU, essa simulação modela numericamente o dispositivo por inteiro, como é de se esperar o custo computacional é enorme e somente recentemente (ultima década) se tornou possível a simulação de uma célula de memória SRAM por inteiro (Figura 14) (Munteanu and Autran 2008). Diversos fatores contribuíram para a melhora nos simuladores 3D, melhoras no paralelismo, CPUs mais rápidas, recursos de memória aprimorados e solucionadores lineares com estratégias novas e mais eficientes. Apesar da redução substantiva no tempo necessário para simulação de uma célula inteira, ainda sim o tempo é considerável se comparado com simuladores SPICE ou até mesmo os por modo-misto. O surgimento recente de clusters com centenas de processadores e a melhora dos recursos de memória é uma promessa de que num futuro próximo seja possível simular porções de circuito totalmente em 3D.(Munteanu and Autran 2008)

Figura 14: Simulação numérica total 3D de uma SEU em uma célula de SRAM. (Munteanu and Autran 2008)



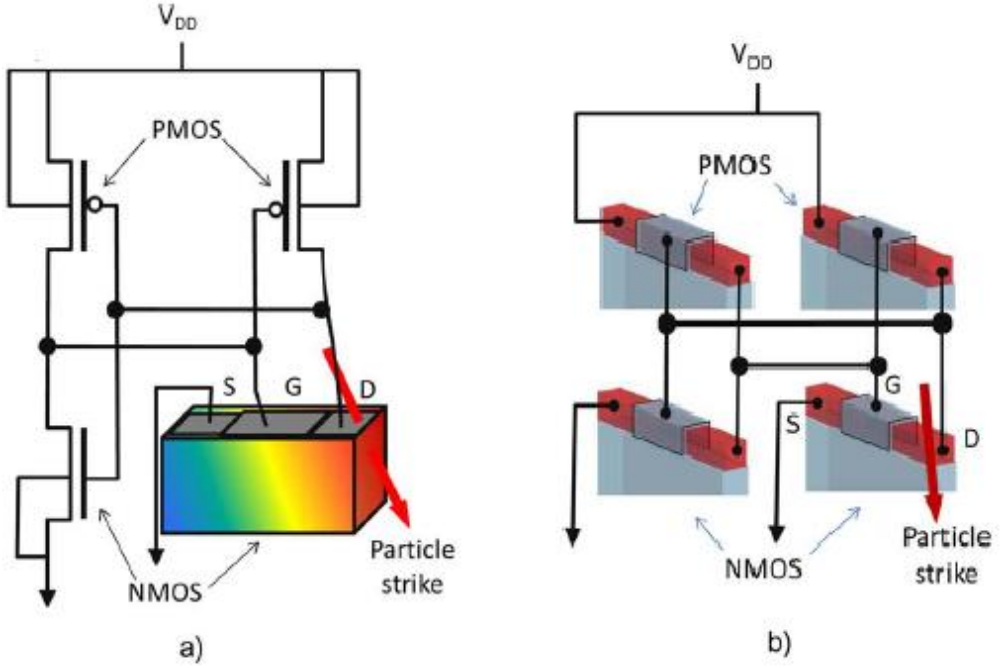
D. Simulação em modo-misto

As limitações da simulação em nível elétrico podem ser sobrepujadas utilizando uma simulação baseada na física do dispositivo para predizer o efeito da inserção de carga no material colidido. Essa técnica é chamada de simulação de modo-misto ou nível-misto, já que os efeitos no dispositivo colidido são simulados utilizando uma modelagem em 3D do transistor, e o resto do circuito é simulado utilizando os mesmo modelos compactos usados nas simulações elétricas (Figura 15a). A corrente transiente gerada da colisão entre a partícula e o dispositivo é computada diretamente no domínio do 3D, não precisando assim simular o SET através de uma fonte de corrente como ocorria na simulação elétrica do circuito.

A simulação por modo-misto traz diversas vantagens e é implementada nos maiores simuladores comerciais, sendo utilizada amplamente em circuitos com poucos

transistores, como uma SRAM (Munteanu and Autran 2008). A modelagem 3D evita os erros de aproximação intrínsecos dos modelos compactos. Neste tipo de modelo também é possível acessar questões como densidade de portadores, campo elétrico e potencial elétrico durante qualquer momento da simulação. Este tipo de técnica de modelagem também permite a simulação do impacto de partículas ionizadas em novos tipos de dispositivos (dispositivos com *gates* múltiplos e nanofios de silício) e a adição de tipos emergentes de fenômenos físicos (como confinamento quântico e transporte quase-balístico) para os quais os modelos compactos não existem ou não são satisfatórios. Por exemplo, o circuito na Figura 15b é usado no estudo da sensibilidade de uma célula de memória SRAM que utiliza dispositivos MOSFET (*metal-oxide semiconductor Field effect transistor*) do tipo *gate all around* (GAA MOSFET).

Figura 15: Simulação de modo-misto, (a) modelagem somente do transistor atingido; (b) modelagem de todos transistores do tipo GAA contidos em uma SRAM.(Munteanu and Autran 2008)



O principal inconveniente da simulação por modo-misto é o aumento do tempo de CPU usado quando comparado com uma simulação em nível de circuito. A simulação por modo-misto é inviável para circuitos complexos, no entanto para casos como células de SRAM o tempo computacional é reduzido significativamente quando comparado com uma simulação numérica em 3D de uma célula completa. (Munteanu and Autran 2008)

3. TÉCNICAS DE MASCARAMENTO DE FALHAS TRANSIENTES

Há diversas técnicas de tolerância a falhas. Algumas visam a detecção de falhas, outras o mascaramento e outras a correção. As técnicas podem ser aplicadas em diversos níveis de projeto de circuito integrado, desde o leiaute, transistor, nível lógico, nível arquitetural e sistêmico. Aqui nesta dissertação iremos focar nas técnicas de mascaramento à falhas transientes que podem ser aplicadas no nível de transistores, no nível lógico e no nível de arquitetura de transferência de registradores (RTL).

As técnicas de mascaramento de falhas no nível de transistor, de portas lógicas e arquitetura são baseadas em redundância espacial ou redundância temporal. Elas necessitam de um votador capaz de mascarar a falha, esse votador é chamado de votador de maioria. Faz-se necessário ter um número mínimo de redundância para que o votador de maioria atue no mascaramento da falha. Normalmente as técnicas são capazes de mascarar falhas únicas. Algumas podem sinalizar (detectar) falhas duplas. Elas apresentam um aumento na área, uma diminuição no desempenho do circuito e aumento da potência. A escolha de uma técnica ou outra se resume no comprometimento entre o custo extra em área, desempenho, potência e cobertura no mascaramento das falhas.

A tabela 2 mostra um resumo das técnicas que serão apresentadas e discutidas em mais detalhes nesta sessão.

Tabela 1: Exemplo de Técnicas de Mascaramento à falhas transientes

Técnicas	Aplicação no nível
Dimensionamento de transistores (Cazeaux, et al. 2005)	**Transistor**
Redundancia Temporal como filtro de SET (ANGHEL, ALEXANDRESCU e NICOLAIDIS 2000)	**Portas Lógicas**
Redundancia Espacial como TMR(Neumann 1956),, DTMR (Tambara, et al. 2013), STMR (Matsumoto, M.Uehara and H.Mori 2011), ATMR (Gomes, et al. 2014)	**Transistor, Porta Lógica e Arquitetura**

3.1 Técnicas de Dimensionamento de Transistores

Técnicas que lidam com redimensionamento de transistores e a razão da largura W entre os transistores do tipo P e do tipo N são um dos meios de produzir circuitos mais

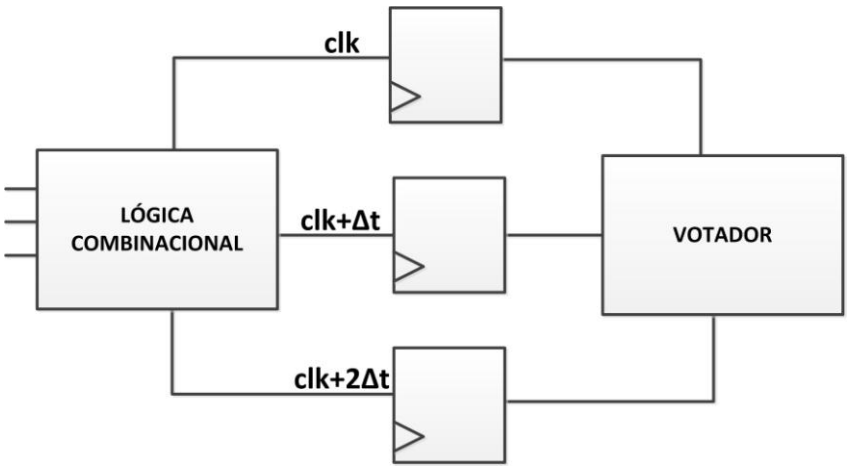
tolerantes a radiação no ambito de falhas transientes. Estudos em (Wirth, Kastensmidt and Ribeiro 2008) indicaram que um aumento do tamanho dos transistores aumenta a carga mínima necessária para que ocorra um SET (carga critica, Q_{crit}), isso ocorre devido o aumento da capacitância no nó redimensionado, tornando Q_{crit} maior, no entanto se ainda sim Q_{crit} é ultrapassada a tendência é que a largura do pulso SET fique maior, fazendo com que o SET se prolongue por mais tempo. Estudos em (Cazeaux, et al. 2005) demonstraram que Q_{crit} de um nó depende mais da força (condutância) da porta lógica que carrega o nó do que da capacitância total do nó em si. Os modelos criados em (Cazeaux, et al. 2005) indicaram que a forma mais efetiva de aumentar Q_{crit} é aumentar o tamanho das portas lógicas que carregam o nó. Já em (Resgui, McCollum and Won 2010) considerações foram feitas quanto a razão entre os transistores PMOS e NMOS de uma porta lógica, este estudo demonstrou que a falta de balanceamento entre os tamanhos dos dois tipo de transistores pode filtrar ou amplificar SETs.

3.2 Técnicas de Redundância Temporal

No caso da redundância temporal, o objetivo da técnica é fazer uso das características do pulso transiente gerado pela colisão de uma partícula para comparar o sinal de saída de um circuito em momentos distintos e assim detectar ou mitigar falhas. Ou seja, a saída de um circuito de lógica combinacional é capturada em dois ou mais momentos diferentes, nos quais a borda de um sinal de relógio é deslocada por um tempo Δt. Ao final, um comparador realiza a detecção de eventuais erros, podendo indicar a ocorrência de um pulso transiente e em qual módulo redundante do circuito houve a manifestação. (ANGHEL, ALEXANDRESCU e NICOLAIDIS 2000)

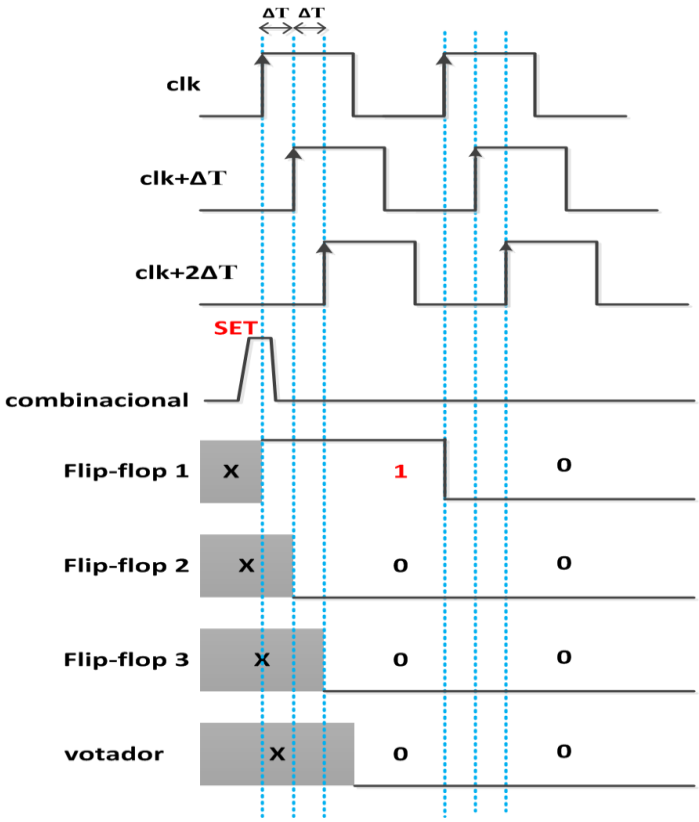
Um esquema baseado na triplicação de blocos constitui a abordagem mais utilizada na redundância temporal. Referida técnica consiste na captura do valor de saída da lógica combinacional em três momentos diferentes, em que a borda do sinal de relógio do segundo armazenamento é deslocado por um tempo Δt e a borda do sinal do terceiro armazenamento é deslocado por um tempo 2Δt. Ao final da triplicação, um votador escolhe a saída correta. Este conceito é ilustrado na Figura 16 e na Figura 17.

Figura 16: Ilustração do conceito de redundância temporal tripla. (Kastensmidt, Carro e Reis 2006)



As penalidades da redundância temporal em um circuito que implementa esta técnica estão relacionadas ao pequeno aumento de área em função dos *latches* extras e à redução de performance, que é dada por $clk+2\Delta t+tp$, onde Δt depende da duração do pulso transiente e *tp* é o atraso do votador de maioria (Kastensmidt, Carro e Reis 2006).

Figura 17: Ilustração dos tempos de atraso e do mascaramento temporal de um SET.



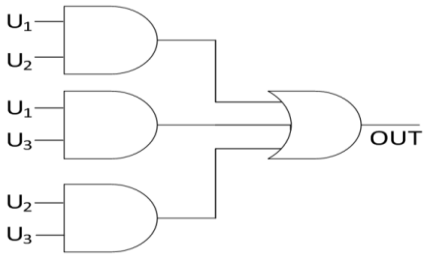
3.3 Técnicas de Redundância Espacial

Uma das primeiras técnicas propostas para o desenvolvimento de projetos de circuitos integrados mais confiáveis foi o NMR (*N-tuple Modular Redundancy*) proposto em (Neumann 1956). Este conjunto de técnicas utiliza a redundância de componentes como meio para mascarar efeitos de falhas. Diversas abordagens variantes do NMR foram propostas, sendo algumas delas elucidadas nas sub-seções seguintes.

3.3.1 Redundância Modular Tripla – TMR

A redundância modular tripla (Figura 5) é uma forma de tolerância a falhas do tipo NMR, na qual três módulos executam uma mesma operação e o resultado é processado por um sistema de votador de maioria (Figura 18), gerando desta forma uma única saída baseada nos três módulos operantes.

Figura 18: Votador de maioria



O TMR é capaz de mascarar 100% falhas transientes únicas (exceto quando a falha ocorre no votador de maioria). Digamos que um erro ocorra em U_1, como na Figura 19a, mesmo sendo propagado o erro será mascarado logicamente pelo votador majoritário e a saída continuara a mesma, a Figura 19b elucida a situação.

Agora vamos supor que outra falha ocorreu desta vez em U_3 antes da falha gerada em U_1 ter sido corrigida. Desta vez o votador majoritário não conseguira mascarar as falhas acabando por gerar um erro na sua saída (Figura 20).

Como podemos ver o TMR é uma técnica de mitigação eficaz contra falhas unicas, no entanto possui um alto custo de área (200% de aumento em área), já que para obter um mascaramento lógico eficaz se faz necessário triplicar o circuito que se deseja proteger. Uma das saídas para contornar este problema é a utilização de circuitos aproximados para diminuir o espaço extra ocupado pelos módulos redundantes do TMR tradicional.

Figura 19: Mascaramento de um TMR sobre uma única falha.

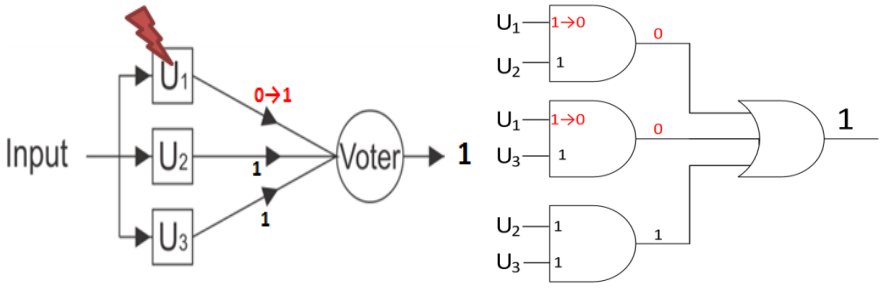
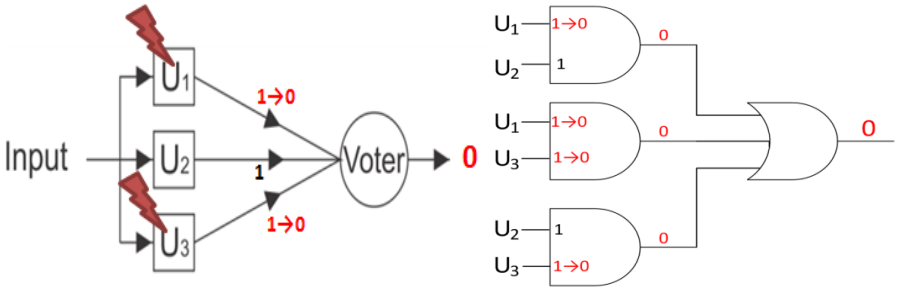


Figura 20: Erro no esquema TMR devido a falhas múltiplas.



3.3.2 TMR Diversificado – DTMR

O *Diversity-TMR* (DTMR), TMR diversificado, utiliza a idéia do TMR tradicional e conceitos redundância por projeto lógico diversificado (*Design Diversity Redundancy*,

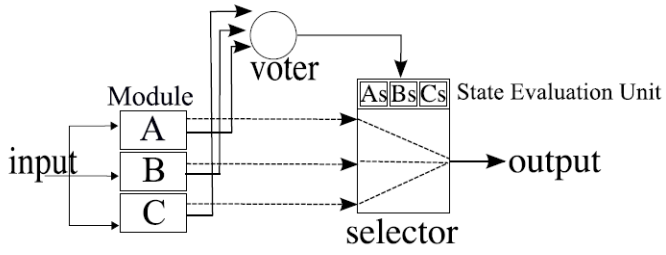
DDR). No DDR cada módulo redundante de um projeto é implementado de forma distinta. Isso pode acontecer no nível de portas lógicas, por exemplo usando somente portas lógicas complexas em um módulo, portas lógicas básicas do tipo NAND e inversores em outro módulo e portas lógicas mistas de uma determinada biblioteca no terceiro módulo. No nível de arquitetura, pode-se pensar em diversas implementações de uma determinada aplicação ou algoritmo em hardware e assim projetar um módulo puramente combinacionais, um módulo baseado em maquinas de estados, e um módulo sendo um processador embarcado, por exemplo, como em (Tambara, et al. 2013). Essa arquitetura DDR é associada à votadores de maioria para criar o DTMR.

Estudos recentes (Tambara, et al. 2013)(O.Hiari, Sadeh and O.Rawashdeh 2012) apontam que o DTMR melhora o mascaramento de falha quando comparado ao TMR tradicional para falhas de modo comum, onde mais de um módulo estaria com falha, mas a manifestação da falha se dará de maneira distinta em cada módulo. Em (Tambara, et al. 2013) o DTMR executava uma multiplicação de matriz e possuía um módulo puramente combinacional, um módulo baseado em maquinas de estado e um ultimo módulo em software executado num miniMIPS. Essa arquitetura tolerava três vezes mais a quantidade de SEU acumulados que uma arquitetura TMR tradicional. Já (O.Hiari, Sadeh and O.Rawashdeh 2012) o DTMR possuía uma copia descrita em HDL (Verilog), outra em software e uma ultima copia analógica. Resultados iniciais demonstraram que a confiabilidade do sistema aumentou reduzindo os erros gerados por uma falha em componente externo.

3.3.3 *Resettable Stateful* TMR

Resettable Stateful TMR é uma técnica de redundância que utiliza o estado (normal ou falho) de cada módulo replicado para aprimorar a confiabilidade de uma arquitetura TMR. *Resettable Stateful TMR* é uma extensão do TMR tradicional já que, além da saída do votador majoritário, utiliza uma unidade de avaliação de estados para decidir a saída da arquitetura (Figura 21). A unidade de avaliação de estados guarda (em registradores) e avalia os estados atuais e antigos dos módulos e a partir disto o seletor decide qual a saída da arquitetura.

Figura 21: ***Stateful-TMR***



Resultados em (Matsumoto, M.Uehara and H.Mori 2011) indicaram que na maioria dos casos o tempo médio para falha (*mean time to failure*, MTTF) do *Resettable Stateful TMR* foi maior que o do TMR tradicional.

3.3.4 Computação Aproximada e ATMR

Computação aproximada refere-se a uma classe de técnicas que atenûa a exigência de equivalência exata entre a especificação e implementação de um sistema de computação. O objetivo da computação aproximada é a obtenção de uma vantagem no desempenho do sistema à custa de algum detrimento na aplicação, melhora na velocidade de execução de um programa, menor utilização da CPU e da memória são exemplos quando se falar em computação aproximada em software. Este conceito pode ser utilizado em nível de hardware, onde a idéia é criar circuitos similares ao circuito original que diferem suas saídas para alguns vetores de entrada, obtendo desta forma uma circuito mais rápido, com menor consumo energético ou menor área. (Sierawski, Bhuva and Massengill 2006)(Sánchez-Clemente, et al. 2012)

Abordagens que utilizam redundância espacial criam copias de um mesmo circuito em conjunto de um circuito extra para detectar ou corrigir um erro na aplicação, protegendo de forma eficaz a aplicação. No entanto a adição das copias acaba penalizando a aplicação principalmente no que se diz respeito à área da mesma. A mescla da técnica de redundância modular tripla (TMR) com abordagens de circuitos aproximados permite que se abra mão de parte da proteção gerada pela técnica de redundância em troca de um menor custo em área. A união das duas abordagens foi denominada ATMR.

No decorrer do capitulo iremos elucidar os conceitos necessários para a compreensão da técnica ATMR (*approximate*-TMR), abordando assuntos como funções lógicas aproximadas, circuitos aproximados, e por fim a técnica ATMR.

3.3.4.1 Funções e circuitos aproximados

Um circuito aproximado é um circuito que realiza uma função lógica diferente porem de correlação estreita com a função lógica original, ou seja, a função aproximada converge maioria de seus mintermos/maxtermos com a função original. Uma aproximação forte diverge seus mintermos/maxtermos poucas vezes em relação à função original (denominada G).

Quando a função original tem seu conjunto de mintermos contida na função aproximada, esta aproximação é chamada de *over-approximated*, em tradução livre sobre-aproximada ou aproximada-superior, e usualmente é representada como H. Sendo assim podemos formular esta relação como G ⊆ H, ou seja, para uma função aproximada ser sobre-aproximada ela precisa possuir todos mintermos da função original. Comumente H possui mintermos (maxtermos) a mais (menos) que a função original.

Digamos que a função original seja:

$$G(A, B) = A * B$$

E a função sobre-aproximada:

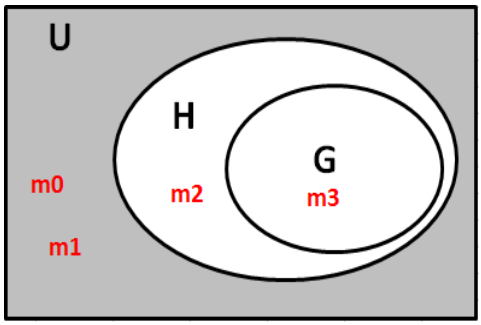
$$H(A, B) = A$$

A Tabela 2 exemplifica a relação H ⊆ G através da tabela verdade de duas funções exemplo, a original (G) e a função sobre-aproximada (H). A Figura 22 ilustra a relação entre G e H mostrando os domínios das funções e os mintermos (sendo U o universo restante de mintermos). Como se pode ver na Tabela 2, a função H converge junto com G no mintermo *m3* e nos maxtermos *m0* e *m1*, a única divergência entre as funções é no mintermo/maxtermo *m2*. A Figura 22 ilustra os domínios de G e H mostrando que de fato o exemplo se encaixa em G ⊆ H. A diferença entre a quantidade de mintermos/maxtermos é conhecida como distância de *Hamming*, e neste caso tem o valor 1 já que G e H divergem em somente um termo. Em contra partida a função H é menor que G na sua quantidade de literais, algo que acarreta vantagem direta quando se fala em área de circuitos.

Tabela 2: Exemplo de uma relação G ⊆ H

Vetores (AB)	*G = A*B*	*H = A*	Mintermo Maxtermo
00	0	0	m0
01	0	0	m1
10	**0**	**1**	**m2**
11	1	1	m3

Figura 22: Relação de mintermos nos conjuntos das funções G e H.



Quando a função aproximada, representada por F, esta contida na função original (G) a mesma é chamada de *under-approximated*, sub-aproximada ou aproximada-inferior e tradução livre. De forma análoga ao caso G ⊆ H, a relação entre a função original e uma função sub-aproximada pode ser formulada por F ⊆ G, ou seja, G deve conter todos mintermos que constituem F, e comumente F possuirá uma quantidade menor de mintermos que G.

Digamos que a função original seja:

$$G(A,B) = A + B$$

E a função sub-aproximada:

$$F(A,B) = A$$

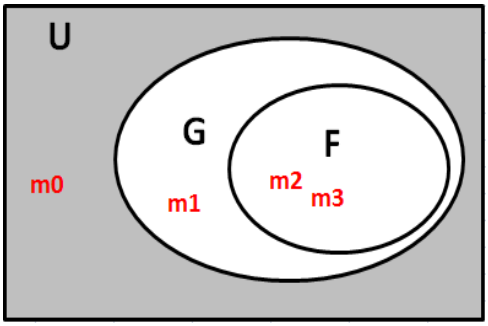
A Tabela 3 exemplifica a relação G ⊆ F através da tabela verdade de duas funções exemplo, a original (G) e a função sobre-aproximada (F). A Figura 23 ilustra a relação

entre G e H mostrando os domínios das funções e os mintermos (sendo U o universo restante de mintermos). Como pode se ver na Tabela 3 a função F converge junto com G nos mintermos *m2* e *m3* no maxtermo *m0*, a única divergência entre as funções é no mintermo/maxtermo *m1*. A Figura 23 ilustra os domínios de G e F mostrando que de fato o exemplo se encaixa em F ⊆ G.A distância de *Hamming* neste caso tem o valor 1 já que G e F divergem em somente um termo. Em contra partida a função F é menor que G na sua quantidade de literais, algo que acarreta vantagem direta quando se fala em área de circuitos.

Tabela 3: Exemplo de uma relação F ⊆ G

Vetores (AB)	*G = A+B*	*F = A*	Mintermo Maxtermo
00	0	0	m0
01	**1**	**0**	**m1**
10	1	1	m2
11	1	1	m3

Figura 23: Relação de mintermos nos conjuntos das funções F e G.



Através dos casos anteriores foi possível compreender e visualizar o que são as funções aproximas, e como as mesmas podem ser classificadas. No entanto derivamos somente uma função de cada tipo para cada um dos exemplos anteriores. O intuito dos exemplos anteriores era demonstrar a troca efetiva entre convergência e tamanho entre uma função e sua aproximada. Porem outros conceitos devem ser elucidados.

Por exemplo, sendo G:

$$G = A * (B + C)$$

Teremos as seguintes possibilidades de funções sobre-aproximadas:

$$H_1 = B + C$$
$$H_2 = A$$
$$H_3 = 1$$

A Tabela 4 representa a tabela verdade da função G e suas funções aproximadas H_1, H_2 e H_3. Como podemos ver todas as funções H respeitam a regra H ⊆ G, no entanto elas divergem de forma diferente de G. A função H_1 possui 2 literais, 1 a menos que G, e tem sua distância de *Hamming* como valor 3, convergindo nos mintermos *m1*, *m2* e *m3*. A função H_2 possui 1 literal, 2 a menos que G, e tem sua distância de

Hamming igual a 1, divergindo somente no mintermo m4. A função H_3 possui 0 literais, 3 a menos que G, e tem sua distância de *Hamming* igual a 5, divergindo nos mintermos *m0*, *m1*, *m2*, *m3* e *m4*. Um fato interessante ocorre neste caso, a função que menos se distância em numero de literais, H_1, não é a função que possui maior semelhança de mintermos. Neste caso a função H_2 é a mais similar das três funções sobre-aproximadas.

Tabela 4: Tabela verdade de G e suas funções sobre-aproximadas (H_1, H_2 e H_3)

Vetores (ABC)	G	H_1	H_2	H_3	Mintermos Maxtermo
000	0	0	0	1	m0
001	0	1	0	1	m1
010	0	1	0	1	m2
011	0	1	0	1	m3
100	0	0	1	1	m4
101	1	1	1	1	m5
110	1	1	1	1	m6
111	1	1	1	1	m7

As mesmas indagações podem ser feitas quanto às funções sub-aproximadas. Ainda, sendo G:

$$G = A * (B + C)$$

Teremos as seguintes possibilidades de funções sobre-aproximadas:

$$F_1 = A * B$$

$$F_2 = A * C$$

$$F_3 = 0$$

A Tabela 5 representa a tabela verdade da função G e suas funções sub-aproximadas F_1, F_2 e F_3. Como podemos ver todas as funções F respeitam a regra $F \subseteq G$, regra que as definem como funções sobre-aproximadas. No entanto cada uma das funções F divergem de forma diferente da função G. Tanto a função F_1 a função F_2 possuem 2 literais, 1 a menos que G, e ambas tem sua distância de *Hamming* em 1, no entanto elas divergem de G em mintermos diferentes, F_1 diverge em *m5* e F_2 em *m6*. A função F_3 possui 0 literais, 3 a menos que G, e tem sua distância de *Hamming* igual a 3, divergindo nos mintermos m5, m6, e m7.

Tabela 5: Tabela verdade de G e suas funções sub-aproximadas (F_1, F_2 e F_3)

Vetores (ABC)	G	F_1	F_2	F_3	Mintermos Maxtermo
000	0	0	0	0	m0
001	0	0	0	0	m1
010	0	0	0	0	m2
011	0	0	0	0	m3
100	0	0	0	0	m4
101	1	0	1	0	m5
110	1	1	0	0	m6
111	1	1	1	0	m7

Por fim a Tabela 6 sintetiza as relações entre a função original ($G = A * (B + C)$) e suas funções sub-aproximas (F_1, F_2 e F_3) e sobre-aproximadas (H_1, H_2 e H_3), podendo analisar a quantidade de literais de cada aproximação lógica e a similaridade entre as funções através da distância de *Hamming*. Assim podemos observar que nem sempre a diferença pequena de literais se traduz em função uma boa de similaridade com a função original, ou seja, por vezes uma função aproximada pequena (baixo numero de literais) vai ser melhor que uma função aproximada maior.

Tabela 6: Funções aproximadas para $G = A * (B + C)$ e suas características de tamanho e convergência

Aproximação	Função	Literais	*Hamming*
Sub-aproximada	$F_1 = A * B$	2	1
	$F_2 = A * C$	2	1
	$F_3 = 0$	0	3
Sobre-aproximada	$H_1 = B + C$	2	3
	$H_2 = A$	1	1
	$H_3 = 1$	0	5

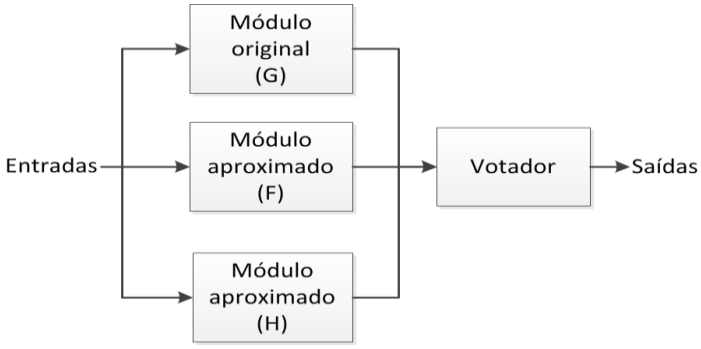
Os casos anteriores eram pequenos, em todos eles a função G não passava de três literais, derivando dessa forma uma baixa quantidade de funções aproximadas. Não raramente circuitos lógicos possuem quantidades maiores de literais. Nestes casos a quantidade de funções aproximadas, derivadas de uma função original, aumenta rapidamente conforme a quantidade de entradas da função aumenta (funções de 6 entradas ficam nas dezenas de aproximações, já as de 8 entradas nas centenas).

Na seção anterior vimos o funcionamento de uma abordagem TMR, e nesta seção investigamos como os conceitos de computação aproximada podem ser aplicados em nível de hardware para se obter circuitos menores. Na próxima seção trataremos da fusão das duas abordagens à fim de diminuir o impacto negativo na área de um esquema TMR. Chamamos a mescla destes dois conceitos de TMR aproximado.

3.3.4.2 ATMR – TMR aproximado

A abordagem do TMR pode garantir um mascaramento lógico completo dos erros causados por SET únicos (exceto em casos onde a falha ocorre no votador de maioria), no entanto essa técnica possui um custo extra em área de 200% (aumento de área). A fim de reduzir o alto custo de área é possível usar uma abordagem com conceitos da computação aproximada, desta forma os módulos redundantes do circuito seriam formas aproximadas do módulo original (Figura 24), está idéia foi proposta primeiramente em (Sierawski, Bhuva and Massengill 2006). O uso de lógica simplificada nos módulos replicados irá reduzir a cobertura, mas também ira reduzir o custo geral de área, permitindo uma troca entre a cobertura de falhas e o aumento de área. (Sierawski, Bhuva and Massengill 2006)(Sánchez-Clemente, et al. 2012)(Gomes and Kastensmidt 2013).

Figura 24: Esquema ATMR: modulo original e dois módulos aproximados (F e H)



Quando os circuitos aproximados são usados em um TMR um dos três módulos poderá apresentar um resultado diferente mesmo na ausência de falha. Isto impõe uma restrição na abordagem do ATMR, somente um dos módulos aproximados pode divergir da saída do circuito original, durante um dado vetor de entrada, permitindo desta forma que o votador escolha o valor correto na ausência de falha. A Tabela 7 elucida a relação dos circuitos aproximados e o original em um esquema ATMR, demonstrando que na ausência de falhas o votador continuaria gerando o resultado correto.

Para um esquema ATMR funcionar corretamente, ou seja, para ele manter seu votador indicando o valor correto na ausência de falhas, é necessário seguir a seguinte regra: $F \subseteq G \subseteq H$, isto é, todo mintermo de F deve ser um mintermo de G, e todo mintermo de G deve ser um mintermo de H. Essa regra assegura que H só ira avaliar em 0 quando G avaliar em 0, e que F só ira ter sua saída em 1 quando G tiver sua saída em 1. Olhando novamente para a Tabela 7 podemos ver que três relações diferentes são criadas entre as funções pela restrição, ilustrada na Figura 25, $F \subseteq G \subseteq H$:

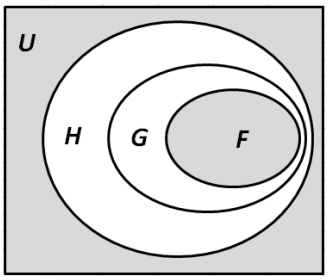
- $G = F = H$, onde a abordagem funciona com um TMR tradicional, três módulos convergindo para um mesmo valor.
- $G = F \neq H$, quando G e F convergem para 0 mas H avalia em 1
- $G = H \neq F$, quando G e H convergem para 1 mas F avalia em 0

Importante ressaltar que a regra $F \subseteq G \subseteq H$ visa impedir casos onde ambos módulos aproximados (F e H) venham a divergir simultaneamente do módulo original caso onde $G \neq F=H$. Outro fato é que um ATMR pode ser composto por duas funções G e uma aproximada (F ou H), podendo criar arquiteturas G,G e F ou G,G e H.

Tabela 7: Tabela verdade de um esquema ATMR

Vetores (ABC)	G = $A * (B + C)$	F = $A * B$	H = A	Saída do votador	Relação entre funções
000	0	0	0	0	G = F = H
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	0	0	**1**	0	**G = F ≠ H**
101	1	**0**	1	1	**G = H ≠ F**
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

Figura 25: Representação ilustrativa da relação entre G, a função original, H, função sobre-aproximada e F, função sub-aproximada. Áreas em cinza representam os mintermos protegidos pelo esquema, quando todas funções convergem para um mesmo valor.



A análise anterior foi feita sob a condição em que não existe falha no ATMR, visando elucidar um conceito inicial de relacionamento entre a função original e suas funções aproximadas, no entanto o objetivo de um ATMR é a mitigação de falhas por isso é necessário analisar o comportamento do mesmo sobre situações adversas. Supondo a seguinte configuração da um ATMR:

Circuito original:

$$\mathbf{G} = \boldsymbol{A} * (\boldsymbol{B} + \boldsymbol{C})$$

Circuito sub-aproximado:

$$\mathbf{F} = \boldsymbol{A} * \boldsymbol{B}$$

Circuito sobre-aproximado:

$$\mathbf{H} = \boldsymbol{A}$$

A. Caso 1: Falha durante uma situação G = F= H:

O primeiro exemplo simula uma falha no circuito sub-aproximado (H) do ATMR quando o vetor de entradas esta em 000. Nesta situação a relação entre as funções é G = F = H. Mesmo com a mudança na relação entre os módulos o votador mantém o valor correto em sua saída, mascarando o erro. Sempre que a relação entre funções, inicialmente, seja G = F = H o ATMR se comportara como um TMR tradicional, garantindo 100% mascaramento lógico contra falhas transientes únicas propagadas na saída de um dos módulos.

Tabela 8: Tabela verdade de um esquema ATMR, falha em um módulo durante um vetor protegido.

Vetores (ABC)	$\mathbf{G} = \boldsymbol{A} * (\boldsymbol{B} + \boldsymbol{C})$	$\mathbf{F} = \boldsymbol{A} * \boldsymbol{B}$	H = A	Saída do votador	Relação entre funções
000	0	**0→1**	0	**0**	**G = F = H→ G = H ≠ F**
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	0	0	1	0	G = F ≠ H
101	1	0	1	1	G = H ≠ F
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

B. Caso 2: Falha durante uma situação G = F ≠ H:

No primeiro momento deste exemplo um erro ocorre no módulo H, durante o vetor 100, fazendo com que sua saída mude de 1 para 0 (Tabela 9). A falha 1→0 do módulo H muda a relação entre as funções de G=F≠H→ G=F=H, porém o votador continua com o valor correto, mascarando o erro. Interessante também ressaltar que nesta situação de falha o esquema estaria protegido até mesmo contra uma segunda falha, já que a relação voltaria a ser G=F≠H ou mudaria para G=H≠F.

Em um segundo momento um erro ocorre no módulo G, durante o vetor 100, fazendo com que sua saída mude de 0 para 1 (Tabela 10). Desta vez a falha 0→1 do módulo G não é mascarada pelo votador já que a mudança da relação entre as funções, G=F≠H→G=H≠F, faz com que G e H concordem com um mesmo valor, neste caso o valor incorreto 1. O vetor 100 é chamado de vetor desprotegido. Se um segundo erro ocorresse, no módulo H ou novamente módulo G, a saída do votador seria corrigida.

Através destes dois exemplos é possível constatar que na situação G = F ≠ H um erro só pode ser gerado quando a falha ocorre no módulo G ou no módulo F, fazendo com que dois módulos convirjam para o valor incorreto. Interessante comentar que as falhas propagadas pelo votador sempre serão do tipo 0→1 pela associação dos módulos.

Tabela 9: Tabela verdade de um esquema ATMR, falha no módulo H durante um caso G = F ≠ H.

Vetores (ABC)	$G = A * (B + C)$	$F = A * B$	H = A	Saída do votador	Relação entre funções
000	0	0	0	0	G = F = H
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	0	0	**1→0**	**0**	**G=F≠H→G=F=H**
101	1	0	1	1	G = H ≠ F
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

Tabela 10: Tabela verdade de um esquema ATMR, falha no módulo G durante um caso G = F ≠ H.

Vetores (ABC)	$G = A * (B + C)$	$F = A * B$	H = A	Saída do votador	Relação entre funções
000	0	0	0	0	G = F = H
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	**0→1**	0	**1**	**0→1**	**G=F≠H→G=H≠F**
101	1	0	1	1	G = H ≠ F
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

C. Caso 3: Falha durante uma situação G = H ≠ F:

Este exemplo é similar ao anterior, um erro ocorre no módulo F, vetor 101, fazendo com que sua saída mude de 0 para 1 (Tabela 11). A falha 0→1 do módulo F muda a relação entre as funções de G=H≠F→ G=F=H, porém o votador continua com o valor correto, mascarando o erro. Interessante também ressaltar que nesta situação de falha o esquema estaria protegido até mesmo contra uma segunda falha, já que a relação voltaria a ser G=H≠F ou mudaria para G=F≠H.

Em um segundo momento um erro ocorre no módulo G, durante o vetor 101, fazendo com que sua saída mude de 1 para 0 (Tabela 12). Desta vez a falha, 0→1, do módulo G não é mascarada pelo votador já que a mudança da relação entre as funções, G=H≠F→G=F≠H, faz com que G e F concordem com um mesmo valor, neste caso é o valor incorreto 0. O vetor 101 é chamado de vetor desprotegido. Se um segundo erro ocorresse, no módulo F ou novamente módulo G, a saída do votador seria corrigida.

Através destes dois exemplos é possível constatar que na situação G = H ≠ F um erro só pode ser gerado quando a falha ocorre no módulo G ou no módulo H, fazendo com que dois módulos convirjam para o valor incorreto. Interessante comentar que as falhas propagadas pelo votador sempre serão do tipo 1→0 pela associação dos módulos.

Tabela 11: Tabela verdade de um esquema ATMR, falha no módulo F durante um caso G = H ≠ F.

Vetores (ABC)	G = $A * (B + C)$	F = $A * B$	H = A	Saída do votador	Relação entre funções
000	0	0	0	0	G = F = H
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	0	0	1	0	G = F ≠ H
101	1	**0→1**	1	**1**	**G=H≠F→G=F=H**
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

Tabela 12: Tabela verdade de um esquema ATMR, falha no módulo G durante um caso G = H ≠ F.

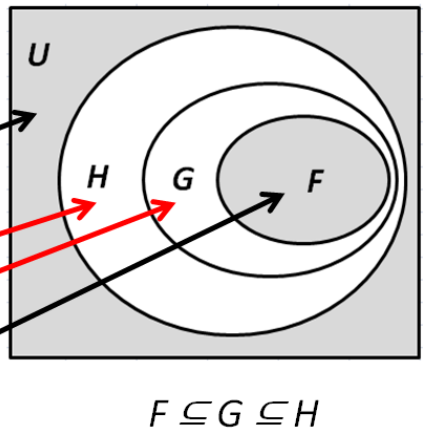
Vetores (ABC)	G = $A * (B + C)$	F = $A * B$	H = A	Saída do votador	Relação entre funções
000	0	0	0	0	G = F = H
001	0	0	0	0	G = F = H
010	0	0	0	0	G = F = H
011	0	0	0	0	G = F = H
100	0	0	1	0	G = F ≠ H
101	**1→0**	0	1	**1→0**	**G=H≠F→G=F≠H**
110	1	1	1	1	G = F = H
111	1	1	1	1	G = F = H

Através dos casos exemplificados anteriormente é possível fazer algumas considerações quanto ao esquema ATMR. Um ATMR pode estar em três situações em um dado tempo, uma situação onde ambos módulos aproximados concordam com o original (G=F=H), e outras duas situações onde F e H divergem de valor entre si (G=F≠H ou G=H≠F). Os módulos aproximados convergem para o mesmo valor do módulo original, quando o vetor de entrada (mintermo) se encontra na área cinza da Figura 26, nesta situação o esquema ATMR estará protegido contra falhas únicas.

Quando o vetor de entrada se encontra na área branca da Figura 26 os módulos aproximados (F e H) estarão em divergência (F≠H), e o ATMR não estará 100% protegido contra falhas únicas. Nesta situação de divergência entre F e H a saída do será desempatada e definida pelo módulo original. Erros em G não são mascarados quando o estado do esquema é F≠H, mas se o erro ocorrer em F ou H ele pode vir a ser mascarado. Erros no módulo F serão mascarados sempre que H avaliar em 1 sua saída. Erros em H serão mascarados sempre que F avaliar em 0 sua saída.

Figura 26: Análise dos vetores desprotegidos.

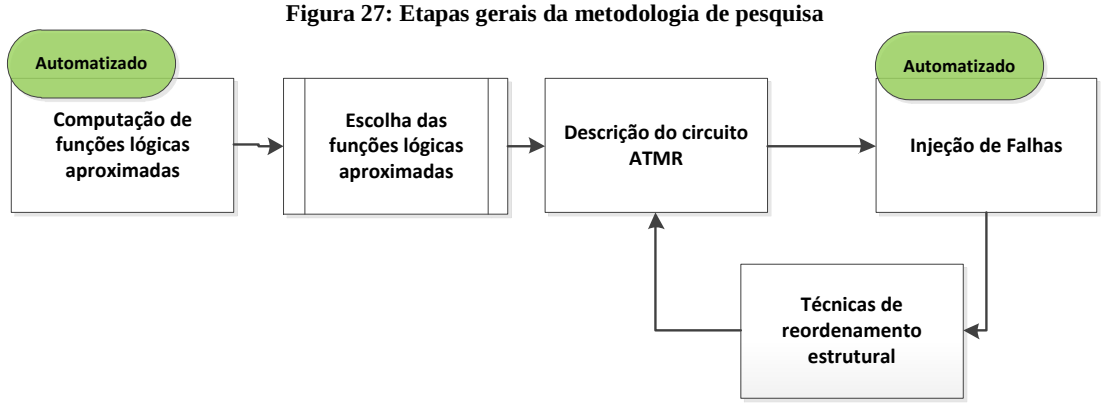
Vetores (ABC)	G = A*(B+C)	F = A*B	H = A	Saida do votador
000	0	0	0	0
001	0	0	0	0
010	0	0	0	0
011	0	0	0	0
100	**0**	**0**	**1**	**0**
101	**1**	**0**	**1**	**1**
110	1	1	1	1
111	1	1	1	1



4. METODOLOGIA PROPOSTA PARA ATMR

Durante o processo de pesquisa se desenvolveu uma metodologia para gerar os circuitos aproximados e avaliar a técnica de tolerância a falhas por redundância ATMR (Gomes, et al. 2014). A metodologia pode ser dividida de forma geral em cinco etapas (Figura 28): computação das funções lógicas aproximadas, escolha das funções lógicas aproximadas, descrição do circuito ATMR, teste dos circuitos através de injeção de falhas e a aplicação de técnicas de reordenamento estrutural.

Figura 27: Etapas gerais da metodologia de pesquisa



A etapa de computação das funções lógicas aproximadas deriva funções F e H a partir de uma função G. A computação das funções lógicas aproximadas pode ser baseada em diversos conceitos como, diagramas de decisão binária (BDD, *binary decision diargam*)(Sierawski, Bhuva and Massengill 2006), fixação de nó (*stuck-at*) com regras baseadas em paridade e funções *unate* (Sánchez-Clemente, et al. 2012) e usando o conceito de ordem parcial (*order concept*)(Gomes, et al. 2014). Na metodologia utilizada, as funções aproximadas foram computadas usando inicialmente o método por fixação de nó (Sánchez-Clemente, et al. 2012). Posteriormente se adaptou ou algoritmo de fatoração, baseado no conceito de ordem parcial (Martins, et al. 2010)(Martins, Ribas and Reis 2012), para as necessidades da técnica ATMR, automatizando esta etapa da metodologia (Gomes, et al. 2014).

O segundo processo tem como objetivo selecionar as funções que irão compor o esquema ATMR. Para tal é necessário analisar manualmente cada função aproximada computada, e definir qual possivelmente teria o melhor custo-benefício em questão de área e mascaramento lógico para o circuito ATMR como um todo. É possível automatizar parcialmente a escolha das funções F e H utilizando uma função de custo que envolveria a redução do tamanho da função (número de literais) e a quantidade de

vetores que diferem na saída entre a função G e a respectiva aproximada (distância de *Hamming*).

A terceira etapa do processo é a descrição do circuito ATMR para posteriormente injetar falhas. Inicialmente os circuitos foram descritos em nível elétrico em SPICE. Posteriormente se optou por uma descrição em nível lógico de transistores utilizando a linguagem Verilog tendo em vista que este nível de abstração atenderia as necessidades para os testes.

A quarta etapa é a injeção de falhas nos circuitos ATMR. A injeção foi inicialmente feita em nível elétrico, quando o circuito ATMR esta descrito em SPICE, usando o software NGSPICE (Paolo Nenzi 2010). Em outro momento se avaliou que injeções de falha em nível lógico de transistores cobririam as necessidades de testes para a pesquisa, também tornaria esta etapa mais rápida que a elétrica. A injeção em nível lógico foi feita utilizando um aplicativo desenvolvido em C++ especificamente para tal função, utilizando juntamente o software ModelSim e linguagem Verilog. Este tipo de injeção de falha é toda automatizada.

A quinta etapa se refere a aplicação das técnicas de reordenamento estrutural. Nesta etapa são feitas modificações no circuito do ATMR de forma que se obtenham todas possíveis variações estruturais do circuito. Posterior a esta reestruturação do circuito se realiza novamente a etapa de injeção de falha. Esta etapa é feita de forma manual, ou seja, cada circuito ATMR é descrito novamente com a devida modificação estrutural, tornado essa etapa a mais demorada. A Figura 28 ilustra, de forma mais detalhada, a metodologia elucidada nos parágrafos anteriores. As seções seguintes aprofundaram cada parte da metodologia de trabalho, explicando decisões de projeto (nível de modelagem dos testes), implementação de técnicas (reordenamento estrutural) e desenvolvimento das ferramentas utilizadas (computação de funções e injeção de falhas).

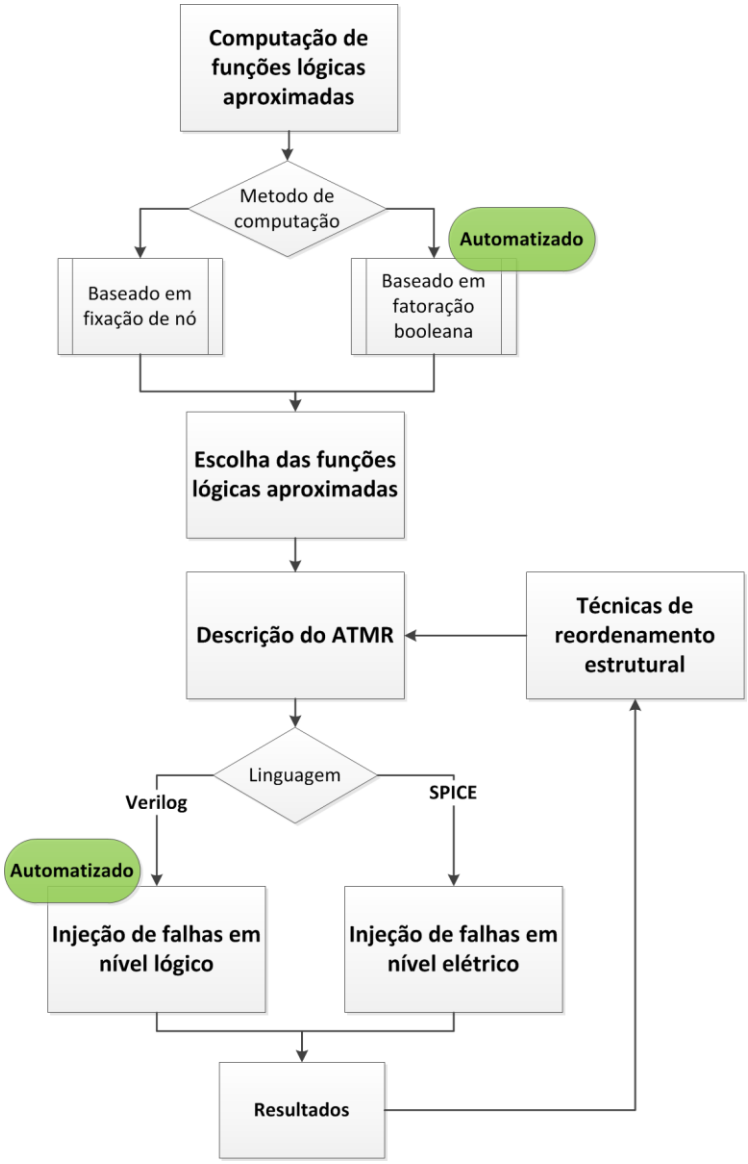
4.1 Computação de funções lógicas aproximadas

Os conceitos sobre as funções aproximadas, com seus tipos e subtipos já foram abordados na seção 3.3.4.1, no entanto não se abordou a criação das mesmas. A computação das funções lógicas aproximadas é o primeiro passo para a criação de um circuito ATMR. Foram usados dois métodos de computação de funções e circuitos aproximados: fixação de nó (Sánchez-Clemente, et al. 2012) e fatoração por ordem parcial (Gomes, et al. 2014).

4.1.1 Computação de funções aproximadas por fixação de nó

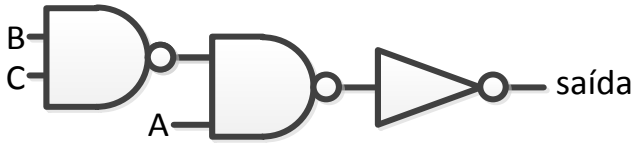
Este método computa os circuitos aproximados baseando se em conceitos de funções *unate*. Este método fixa valores em nós do circuito, forçando 0 ou 1 no nó (*stuck-at*), seguindo regras baseadas em conceitos de funções *unate* e paridade de nós para garantir a computação correta das funções sub-aproximadas e sobre-aproximadas.

Figura 28: Metodologia completa utilizada para o desenvolvimento e análise da técnica de circuitos ATMR.



Classificamos a paridade de um nó da seguinte forma: sendo *y* a saída do circuito e *x* um nó do circuito, *x* é dito um nó par se todos os caminhos entre *x* e *y* possuem um numero par de inversões, na Figura 29 *A* é um nó par em relação à *saída*. Quando todos os caminhos entre *x* e *y* possuem um numero impar de inversões dizemos que *x* é um nó impar. Quando *x* possui dois ou mais caminhos, sendo um deles par e outro impar dizemos que *x* é uma variável *binate*, não possuindo paridade desta forma.

Figura 29: Paridade: A é um nó pare em relação a saída, B e C são nós impar.



As seguintes regras determinam como a partir de um nó par computar de forma correta uma função lógica sub-aproximada (F) ou e uma função sobre-aproximada (H):

- Tendo o nó uma paridade par, uma fixação em 1 (*stuck-at 1*), gera uma função sobre-aproximada (H). A Figura 30 elucida esta situação, a fixação em 1 da entrada *A* faz com que o circuito original se transforme em um circuito sobre-aproximado.
- Tendo o nó uma paridade par, uma fixação em 0 (*stuck-at 0*), gera uma função sub-aproximada (F). A Figura 31 ilustra a situação, o fixado em 0 da entrada *A* faz com que o circuito original se torne um circuito sub-aproximado.

Figura 30: Criação de uma função sobre-aproximada (H) através de fixação 1 em um nó par

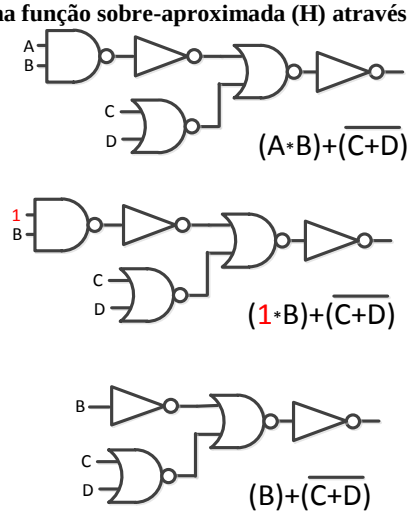
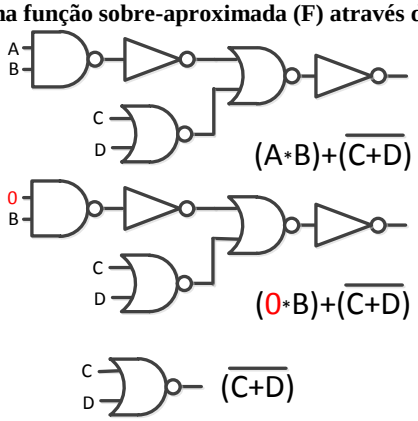


Figura 31: Criação de uma função sobre-aproximada (F) através de fixação 0 em um nó par



As seguintes regras determinam como a partir de um nó impar computar de forma correta uma função lógica sub-aproximada (F) ou e uma função sobre-aproximada (H):

- Tendo o nó uma paridade impar, uma fixação em 1 (*stuck-at 1*), gera uma função sub-aproximada (F). A Figura 32 elucida esta situação, o fixação em 1 da entrada *D* faz com que o circuito original se transforme em um circuito sub-aproximado.
- Tendo o nó uma paridade impar, um fixação em 0 (*stuck-at 0*), gera uma função sobre-aproximada (H) A Figura 33 ilustra a situação, a fixação em 0 da entrada *D* faz com que o circuito original se torne um circuito sobre-aproximado.

Figura 32: Criação de uma função sub-aproximada (F) através de fixação 1 em um nó impar

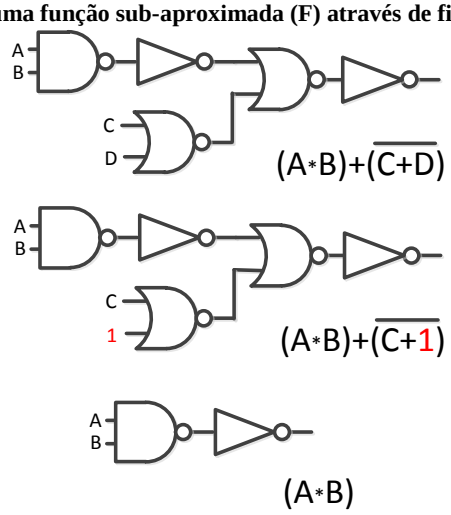
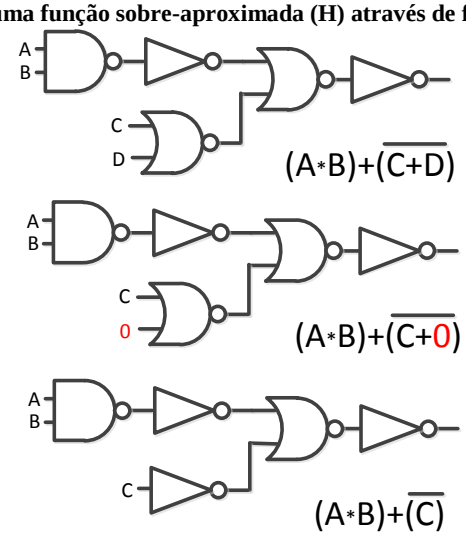
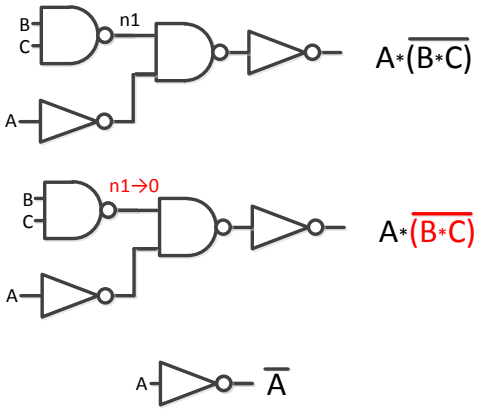


Figura 33: Criação de uma função sobre-aproximada (H) através de fixação 0 em um nó impar



Importante ressaltar que a fixação não é restrita às entradas do circuito, podendo ser aplicado em qualquer nó. Na Figura 34 o nó *n1* é fixado em 0, *n1* é um nó impar portanto a função gerada pela técnica é uma sobre-aproximada.

Figura 34: Fixação de um nó interno do circuito.



4.1.2 Computação de funções aproximadas por fatoração booleana

Esta abordagem para a computação de funções aproximadas é uma versão modificada do algoritmo de fatoração booleana apresentado em (Martins, et al. 2010). O ponto inicial do algoritmo é a função booleana original (G). Essa função é fatorada em funções menores (sub-funções) e estas são combinadas usando o conceito de ordem parcial. O conceito de ordem parcial classifica três funções, onde uma função pode ser menor (possui uma quantidade menor de mintermos que a função original), maior (possui um numero maior de mintermos que a função original) ou não comparável. Como já foi abordado no capitulo 3.3.4.1, a função menor do conceito de ordem parcial representa a função F (sub-aproximada), já a função maior do conceito de ordem parcial é a função H (sobre-aproximada) abordada nos capítulos passados. Uma função não comparável converge alguns mintermos de G e alguns mintermos de U (Figura 35). Os cofatores e cofatores cúbicos são combinados para criar o conjunto de funções permitidas. Este conjunto é usado para cortar as funções que não serão uma boa solução, diminuindo assim o tempo de execução do algoritmo significativamente. (Gomes, et al. 2014)

Figura 35: Z é uma função não comparável.

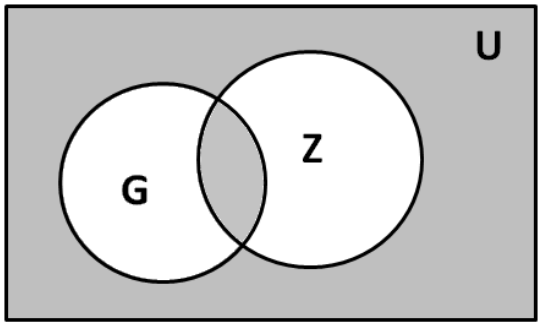


Tabela 13: Exemplo de uma relação não comparável.

Vetores (AB)	$A + B$	$\overline{A} + \overline{B}$
00	0	1
01	1	1
10	1	1
11	1	0

O próximo passo é guardar as variáveis na polaridade correta em um conjunto chamado de balde. Essas variáveis são guardadas como pares limitados (*bounded pairs*), tuplas do tipo {função; expressão}. Novos pares limitados são criados a partir de pares limitados mais simples (pares com menos literais) que foram computados em interações anteriores.

Além de fatorar a função G, o algoritmo proposto também separa todas as funções, sub-aproximadas e sobre-aproximadas (funções F e H respectivamente), que possuem um número de literais mais baixo que a função original (função G). A distância de *Hamming* é computada por uma operação XOR entre a função G e cada

uma das funções sub-aproximadas e sobre-aproximadas. A saída do algoritmo retorna uma lista de funções F e H, com a distância de *Hamming* de cada uma delas.

Figura 36: Pseudo código do algoritmo de fatoração booleana.

```
createSmallerAndLargerEquations(Function target) {
    Equation solution = factorize(target);
    int targetLiterals = countLiterals(solution);
    vector<Function> smaller = computeSmallerFunctions();
    vector<Function> larger = computeLargerFunctions();


    for(Function f : smaller) {
        Equation solutionSmaller = factorize(f);
        int literals = targetLiterals - countLiterals(solutionSmaller);
        int differentBits = countDiffBits(f,target);
        storeSmaller(f,solutionSmaller, literals, differentBits);
    }

    for(Function f : smaller) {
        Equation solutionLarger = factorize(f);
        int literals = targetLiterals - countLiterals(solutionSmaller);
        int differentBits = countDiffBits(f,target);
        storeSmaller(f,solutionSmaller, literals, differentBits);
    }
}
```

Utilizando a seguinte função como exemplo:

$$G(A,B,C) = A * B + A * C$$

O primeiro passo é a computação das funções permitidas. A computação dos cofatores cúbicos (*cube cofactors*) ira resultar em funções diferentes que serão usadas como o conjunto de funções permitidas. Próximo passo é a criação das representações dos literais. Isso ira criar os pares limitados e inseri-los no balde de formulas de 1-literal. Assim que o balde de 1-literal está cheio a combinação das partes começa a produzir as combinações do balde de 2-literais. Somente as sub-funções que estão no conjunto das funções permitidas são aceitas como funções intermediarias. A combinação continua até o balde de 3-literais, onde a função G é remontada. Estes passos estão ilustrados na Figura 37. Depois desta etapa estar completa, as sub-funções são inseridas nos conjuntos F e H conforme ilustrado na Figura 38. As características destas funções são retornadas em um arquivo tabelado, neste caso similar à Tabela 14.

Figura 37: Funções necessárias para compor a função G

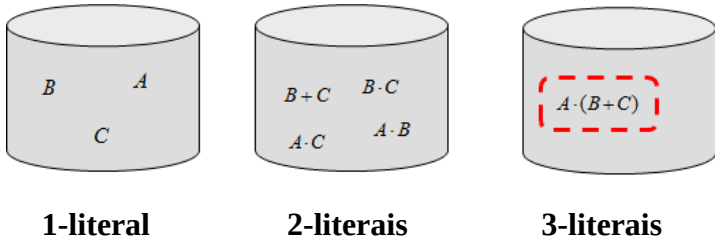


Figura 38: Funções candidatas para compor as funções F e H.

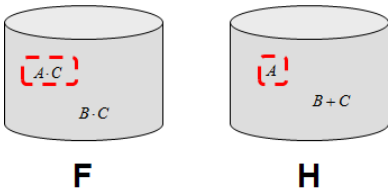


Tabela 14: Características das funções candidatas

Aproximação	Função	Literais	*Hamming*
F	$A * B$	2	1
	$B * C$	2	1
H	$B + C$	2	3
	A	1	1

4.2 Escolha das funções lógicas aproximadas

Como vimos nos capítulos anteriores a computação das funções aproximadas retorna um conjunto de funções candidatas a compor o circuito ATMR final. A escolha destas funções é essencial para a obtenção de um circuito ATMR com um bom custo-benefício entre mascaramento lógico e o custo extra de área (*overhead*). Sabemos que a capacidade de mascaramento está diretamente relacionada à quantidade total de vetores desprotegidos. Por exemplo, supondo as seguintes composições de circuitos ATMR ilustradas na Tabela 15.

Tabela 15: Exemplo de composições ATMR e suas características

Composição	G	F	H	Vetores desprotegidos	Aumento de área
S1	$A * (B + C)$	$A * B$	$B + C$	4	133%
S2	$A * (B + C)$	$A * B$	A	2	100%
S3	$A * (B + C)$	$A * B$	$A * (B + C)$	1	167%
S4	$A * (B + C)$	$A * (B + C)$	A	1	133%

As composições de circuito ATMR da Tabela 15 possuem cada uma delas 8 vetores, já que G possui 3 entradas. Se analisarmos as composições levando em conta os o total de vetores do circuito e os vetores desprotegidos, podemos chegar a seguinte conclusão preliminar quanto ao mascaramento lógico:

- O mascaramento da composição S1 ficara entre 50-100%
- O mascaramento da composição S2 ficara entre 75-100%
- O mascaramento da composição S3 ficara entre 87,5-100%
- O mascaramento da composição S4 ficara entre 87,5-100%

Se levarmos em consideração o aumento de área veremos que S2 e S4 são composições com um custo beneficio interessante. S1 pode ser excluída a primeira vista já que tem a pior proteção, em análise parcial, e tanto S4 (com o mesmo aumento)

quanto S2 (com um aumento menor) possuem um mascaramento potencialmente maior. Já S4 tem um aumento de área menor que S3, tendo, em análise parcial, uma proteção igual à S3, desta forma S3 se torna obsoleto. Já a comparação S2 versus S4 apresenta uma troca interessante entre mascaramento e área, desta forma, ambas se tornam opções viáveis de composição.

Já vimos em exemplos anteriores que por vezes as funções de menor tamanho, poucos literais, possuem uma convergência maior com G que funções com maior numero de literais. Um exemplo claro pode ser visto na Tabela 14 no conjunto das funções sobre-aproximadas (H). Neste conjunto de funções candidatas temos uma função com 2 literais, $H = B + C$, com uma divergência de 3 mintermos, e a função $H = A$, com somente 1 literal e 1 mintermo divergente. No entanto este é um caso simples onde a função original possui somente 3 literais. Tendo o caso mais complexo onde a função original é:

$$Gz = \left(\left(\left(\overline{A} + \overline{B}\right) * \overline{C}\right) + \overline{D}\right) * \left(\overline{E} + \overline{F}\right)$$

O algoritmo retornara a Tabela 16 para as funções sobre-aproximadas:

Tabela 16: Funções H candidatas para Gz

Cod	Expressão	Literais	*Hamming*
H1	(!D + !C)	2	15
H2	(!E + !F)	2	15
H3	(!B + (!D + !A))	3	23
H4	((!B + !D) + !C)	3	23
H5	(!C + (!D + !A))	3	23
H6	((!D + !C) * (!E + !F))	4	3
H7	(!D + (!C * (!B + !A)))	4	11
H8	((!D * (!E + !F)) + !C)	4	11
H9	((!C * (!E + !F)) + !D)	4	11
H10	((!F * (!D + !C)) + !E)	4	11
H11	((!E * (!D + !C)) + !F)	4	11
H12	(!A + (!D + (!B * !C)))	4	19
H13	((!B + !D) + (!A * !C))	4	19
H14	((!E + !F) * (!B + (!D + !A)))	5	9
H15	(!F + (!E * (!B + (!D + !A))))	5	13
H16	(!E + (!F * (!B + (!D + !A))))	5	13
H17	(((!B + !D) * (!E + !F)) + !C)	5	17
H18	(!B + ((!D + !A) * (!E + !F)))	5	17
H19	(!A + ((!B + !D) * (!E + !F)))	5	17
H20	(((!B + !A) * (!E + !F)) + !D)	5	17
H21	(!C + ((!D + !A) * (!E + !F)))	5	17
H22	((!B * (!E + !F)) + (!D + !A))	5	21
H23	((!D * (!E + !F)) + (!B + !A))	5	21
H24	((!C * (!E + !F)) + (!B + !D))	5	21
H25	((!B + !D) + (!A * (!E + !F)))	5	21
H26	((!C * (!E + !F)) + (!D + !A))	5	21

Como podemos ver a quantidade de funções candidatas H aumentou significativamente, agora são 26 funções. Também foram computadas as funções candidatas sub-aproximadas, as quais chegaram a uma quantidade de 66 funções. Ou seja, foram computadas 92 funções candidatas. Essas análises de custo-benefício entre funções se tornam impraticáveis para casos grandes, como de 8 literais por exemplo, onde a quantidade de funções sobre-aproximadas (H) e sub-aproximadas (F) ficam na casa das centenas de candidatas. Dessa forma se faz necessário o uso de abordagens a fim de diminuir a quantia inicial de funções candidatas fazendo uma filtragem inicialmente automatizada e uma escolha manual das funções que irão compor o circuito ATMR.

Uma análise rápida na Tabela 16 revela que situações onde funções maiores têm um custo-benefício ruim quando comparada a outra função de igual tamanho ou menor são comuns. Por exemplo, a H25 (5 literais, 21 divergências) se comparada à H18 (5 literais e 17 divergências) apresenta um custo-benefício ruim. Esta situação pode ser facilmente contornada selecionando somente os melhores casos para cada tamanho de função.

Tabela 17: Filtragem funções H candidatas através dos melhores casos

Cod	Expressão	Literais	*Hamming*
H1	(!D + !C)	2	15
H2	(!E + !F)	2	15
H3	(!B + (!D + !A))	3	23
H4	((!B + !D) + !C)	3	23
H5	(!C + (!D + !A))	3	23
H6	((!D + !C) * (!E + !F))	4	3
H14	((!E + !F) * (!B + (!D + !A)))	5	9

Já se compararmos H14 (5 literais, 9 divergências), melhor caso de 5 literais, com a H3 (3 literais, 23 divergências) teremos uma situação explicita de troca entre área e mascaramento de falhas, onde H3 terá um aumento em área menor que H14, porém terá um mascaramento lógico menor, desta forma em primeira análise essa funções são interessantes. Porém se compararmos H14 (5 literais, 9 divergências) com H6 (4 literais, 3 divergências) voltaremos para um caso onde H14 é claramente uma escolha ruim de função. O mesmo pode ser dito de H3 (3 literais, 23) se comparada à H1 (2 literais, 15 divergências). Esta questão pode ser tratada através da criação de uma variável de custo para cada função candidata. Esta variável é criada através de uma simples operação aritmética entre as características da função (quantidade de literais e vetores divergentes). O valor da variável de custo é o produto entre a quantidade de literais e a distância de *Hamming*. Através desta variável simples de custo é possível filtrar as funções explicitamente ruins (H14, H3, H4 e H5) do conjunto de funções que possuem algum custo-benefício entre si classificando a tabela em ordem crescente de custo. Essa

variável, por consequência, também funciona para separar os melhores casos para cada tamanho de função candidata.

Tabela 18: Filtragem funções H candidatas através de uma variável de custo

Altura	Cod	Expressão	Literais	*Hamming*	*Custo*
0	**H6**	((!D + !C) * (!E + !F))	4	3	12
1	**H1**	(!D + !C)	2	15	30
2	**H2**	(!E + !F)	2	15	30
3	**H14**	((!E + !F) * (!B + (!D + !A)))	5	9	45
4	**H3**	(!B + (!D + !A))	3	23	69
5	**H4**	((!B + !D) + !C)	3	23	69
6	**H5**	(!C + (!D + !A))	3	23	69

O ponto de corte define a posição da tabela que separa o conjunto de funções explicitamente ruins do conjunto de funções com custo-benefício entre si na tabela de funções candidatas (candidatas-finais). Estando a tabela de funções candidatas organizada de forma crescente através da variável de custo, podemos definir o ponto de podagem da seguinte forma:

- Sendo p o ponto de corte.
- n a altura da função x_n na tabela.
- C_n a variável de custo da função x_n na posição n
- Lit_n a quantidade de literais da função x_n na posição n,

O ponto de corte é definido pela seguinte condição:

$$p = n \; quando \; C_n < C_{n+1} \; e \; Lit_n \leq Lit_{n+1} \qquad (1)$$

Utilizando a Tabela 18 como exemplo, começando da posição $n = 0$, teremos $C_0 = 12$ e $C_1 = 30$, satisfazendo $C_n < C_{n+1}$, no entanto a condição $Lit_n \leq Lit_{n+1}$ não é satisfeita já que $Lit_0 = 4$ e $Lit_1 = 2$. Para $n = 1$ somente uma das condições de (1) é satisfeita, já que $C_1 = C_2$ e $Lit_1 = Lit_2$.

Somente quando $n = 2$, caso onde $C_2 = 30$ e $C_3 = 45$ e $Lit_2 = 2$ e $Lit_3 = 3$, as condições em (1) são satisfeitas e o ponto de podagem é encontrado sendo $p = 2$. Com o valor de p podemos separar os conjuntos de funções onde:

- Uma função x_n é considerada explicitamente ruim se sua altura n é maior que o ponto de corte p, ou seja, quando $n > p$. Sendo desta forma descartada da tabela de funções candidatas.
- Funções com $n \leq p$ continuam como candidatas para compor o circuito ATMR. Pertencentes então ao conjunto de candidatas-finais.

A Figura 39 ilustra a filtragem através de uma variável de custo. A quantidade de funções candidatas foi diminuída de 26 para 3, uma quantia tratável manualmente. Fazendo uma análise final das funções restantes podemos ver que a comparação de H6

com H1 ou H2 sinaliza uma situação de custo-benefício onde H6 é uma opção mais segura (apenas 3 divergências) no entanto mais custosa em área (4 literais) para a a composição do circuito ATMR. O oposto pode ser afirmado de H1 e H2, sendo elas menores em área, porém possuem um menor fator de mascaramento. A escolha entre as funções candidatas-finais é uma decisão de projeto, e varia conforme a necessidade da aplicação ATMR. A Figura 40 ilustra outro exemplo de funções candidatas, desta vez para *Gx*, sendo a tabela referente a funções sub-aproximadas (F).

$$Gx = \left(\left(\left(\overline{A} * \overline{B}\right) + \left(\overline{C} + \overline{D}\right)\right) * \left(\overline{E} * \overline{F}\right)\right)$$

Figura 39: Filtragem da tabela de funções H candidatas através de uma variável de custo para Gz

Cod	Expressão	Literais	*Hamming*	Custo
H1	(!D + !C)	2	15	30
H2	(!E + !F)	2	15	30
H3	(!B + (!D + !A))	3	23	69
H4	((!B + !D) + !C)	3	23	69
H5	(!C + (!D + !A))	3	23	69
H6	((!D + !C) * (!E + !F))	4	3	12
H7	(!D + (!C * (!B + !A)))	4	11	44
H8	((!D * (!E + !F)) + !C)	4	11	44
H9	((!C * (!E + !F)) + !D)	4	11	44
H10	((!F * (!D + !C)) + !E)	4	11	44
H11	((!E * (!D + !C)) + !F)	4	11	44
H12	(!A + (!D + (!B * !C)))	4	19	76
H13	((!B + !D) + (!A * !C))	4	19	76
H14	((!E + !F) * (!B + (!D + !A)))	5	9	45
H15	(!F + (!E * (!B + (!D + !A))))	5	13	65
H16	(!E + (!F * (!B + (!D + !A))))	5	13	65
H17	(((!B + !D) * (!E + !F)) + !C)	5	17	85
H18	(!B + ((!D + !A) * (!E + !F)))	5	17	85
H19	(!A + ((!B + !D) * (!E + !F)))	5	17	85
H20	((((!B + !A) * (!E + !F)) + !D)	5	17	85
H21	(!C + ((!D + !A) * (!E + !F)))	5	17	85
H22	((!B * (!E + !F)) + (!D + !A))	5	21	105
H23	((!D * (!E + !F)) + (!B + !A))	5	21	105
H24	((!C * (!E + !F)) + (!B + !D))	5	21	105
H25	((!B + !D) + (!A * (!E + !F)))	5	21	105
H26	((!C * (!E + !F)) + (!D + !A))	5	21	105

Cod	Expressão	Literais	*Hamming*	Custo
H6	((!D + !C) * (!E + !F))	4	3	12
H1	(!D + !C)	2	15	30
H2	(!E + !F)	2	15	30
H7	(!D + (!C * (!B + !A)))	4	11	44
H8	((!D * (!E + !F)) + !C)	4	11	44
H9	((!C * (!E + !F)) + !D)	4	11	44
H10	((!F * (!D + !C)) + !E)	4	11	44
H11	((!E * (!D + !C)) + !F)	4	11	44
H14	((!E + !F) * (!B + (!D + !A)))	5	9	45
H15	(!F + (!E * (!B + (!D + !A))))	5	13	65
H16	(!E + (!F * (!B + (!D + !A))))	5	13	65
H3	(!B + (!D + !A))	3	23	69
H4	((!B + !D) + !C)	3	23	69
H5	(!C + (!D + !A))	3	23	69
H12	(!A + (!D + (!B * !C)))	4	19	76
H13	((!B + !D) + (!A * !C))	4	19	76
H17	(((!B + !D) * (!E + !F)) + !C)	5	17	85
H18	(!B + ((!D + !A) * (!E + !F)))	5	17	85
H19	(!A + ((!B + !D) * (!E + !F)))	5	17	85
H20	((((!B + !A) * (!E + !F)) + !D)	5	17	85
H21	(!C + ((!D + !A) * (!E + !F)))	5	17	85
H22	((!B * (!E + !F)) + (!D + !A))	5	21	105
H23	((!D * (!E + !F)) + (!B + !A))	5	21	105
H24	((!C * (!E + !F)) + (!B + !D))	5	21	105
H25	((!B + !D) + (!A * (!E + !F)))	5	21	105
H26	((!C * (!E + !F)) + (!D + !A))	5	21	105

Figura 40: Filtragem da tabela de funções F candidatas através de uma variável de custo para Gx

Cod	Expressão	Literais	*Hamming*	*Custo*
F1	(!E * (!D * !F))	3	5	15
F2	(!E * (!F * !C))	3	5	15
F3	((!E * !F) * (!D + !C))	4	1	4
F4	((!E * !A) * (!F * !C))	4	9	36
F5	((!E * !B) * (!F * !A))	4	9	36
F6	((!E * !A) * (!D * !F))	4	9	36
F7	((!D * !F) * (!E * !C))	4	9	36
F8	((!E * (!B * !F)) * !C)	4	9	36
F9	((!E * (!B * !F)) * !D)	4	9	36
F10	((!E * !F) * (!D + (!B * !A)))	5	3	15
F11	((!E * !F) * (!C + (!B * !A)))	5	3	15
F12	((!E * (!A + !C)) * (!D * !F))	5	7	35
F13	(((!E * !F) * (!B + !D)) * !C)	5	7	35
F14	((!E * !A) * (!F * (!B + !D)))	5	7	35
F15	((!E * (!A + !C)) * (!B * !F))	5	7	35
F16	((!E * !A) * (!F * (!B + !C)))	5	7	35
F17	((!F * (!D + !A)) * (!E * !C))	5	7	35
F18	((!F * (!D + !A)) * (!E * !B))	5	7	35
F19	((!E * !A) * (!F * (!D + !C)))	5	7	35
F20	(((!E * !F) * (!B + !C)) * !D)	5	7	35
F21	((!E * (!B * !F)) * (!D + !C))	5	7	35
F4	((!E * !A) * (!F * !C))	4	9	36
F5	((!E * !B) * (!F * !A))	4	9	36
F6	((!E * !A) * (!D * !F))	4	9	36
F7	((!D * !F) * (!E * !C))	4	9	36
F8	((!E * (!B * !F)) * !C)	4	9	36
F9	((!E * (!B * !F)) * !D)	4	9	36
F22	((!E * (!B * !A)) * (!D * !F))	5	11	55
F23	((!E * (!B * !A)) * (!F * !C))	5	11	55

Cod	Expressão	Literais	*Hamming*	*Custo*
F3	((!E * !F) * (!D + !C))	4	1	4
F1	(!E * (!D * !F))	3	5	15
F2	(!E * (!F * !C))	3	5	15
F10	((!E * !F) * (!D + (!B * !A)))	5	3	15
F11	((!E * !F) * (!C + (!B * !A)))	5	3	15
F12	((!E * (!A + !C)) * (!D * !F))	5	7	35
F13	(((!E * !F) * (!B + !D)) * !C)	5	7	35
F14	((!E * !A) * (!F * (!B + !D)))	5	7	35
F15	((!E * (!A + !C)) * (!B * !F))	5	7	35
F16	((!E * !A) * (!F * (!B + !C)))	5	7	35
F17	((!F * (!D + !A)) * (!E * !C))	5	7	35
F18	((!F * (!D + !A)) * (!E * !B))	5	7	35
F19	((!E * !A) * (!F * (!D + !C)))	5	7	35
F20	(((!E * !F) * (!B + !C)) * !D)	5	7	35
F21	((!E * (!B * !F)) * (!D + !C))	5	7	35
F4	((!E * !A) * (!F * !C))	4	9	36
F5	((!E * !B) * (!F * !A))	4	9	36
F6	((!E * !A) * (!D * !F))	4	9	36
F7	((!D * !F) * (!E * !C))	4	9	36
F8	((!E * (!B * !F)) * !C)	4	9	36
F9	((!E * (!B * !F)) * !D)	4	9	36
F22	((!E * (!B * !A)) * (!D * !F))	5	11	55
F23	((!E * (!B * !A)) * (!F * !C))	5	11	55

4.3 Descrição de circuitos e injeção de falhas

Posterior à escolha das funções lógicas que irão compor o circuito ATMR, a função origina G e suas funções aproximadas F e H, vem a etapa de descrição da arquitetura elaborada e a injeção de falhas (SET). A etapa de injeção de SET esta diretamente relacionada com a forma escolhida para descrever o circuito ATMR. Uma descrição na linguagem SPICE incorre na injeção de falhas em nível elétrico. Já para a injeção de falhas em nível lógico a necessidade é de uma descrição em uma linguagem "pseudo-Verilog", um tipo de descrição simples baseada nas diretivas de Verilog que posteriormente será transformada, pela ferramenta de injeção de falhas, em uma descrição completa em Verilog.

As linguagens SPICE e Verilog são amplamente conhecidas e utilizadas quando se diz respeito à descrição de circuitos lógicos. No caso da linguagem SPICE serão elucidados somente os pontos principais para a descrição do circuito e injeção de falhas usando o simulador elétrico NGSPICE. A descrição em pseudo-Verilog será tratada mais profundamente sendo um ponto chave para o funcionamento correto da ferramenta de injeção de SET desenvolvida para os estudos.

4.3.1 Injeção de falhas em nível elétrico

A utilização de simuladores elétricos para modelagem de falhas foi abordado conceitualmente no sub-capitulo 2.4. Este método de injeção de falhas, especificamente injeção de SET, foi utilizado nas etapas iniciais do estudo desta dissertação, utilizando o simulador elétrico NGSPICE.

A simulação SET é feita através da inserção de uma fonte de corrente (dupla exponencial) no nó que se deseja emular a falha (Wirth, et al. 2005)(Wirth, Kastensmidt and Ribeiro 2008)(Simionovski and Wirth 2012). A corrente transiente gerada por um SEE normalmente é um pulso curto com um tempo de subida rápido e o tempo de descida dependente do tipo de partícula energética e das características da junção p-n atingida. Este efeito pode ser modelado através de seis constantes (Tabela 19). Para tempos t menores que T_R a corrente I é igual a I_1. Para tempos t entre T_R e T_F, a corrente I é dada por (3). E para tempos t maiores que T_F a corrente I é expressa por (4).

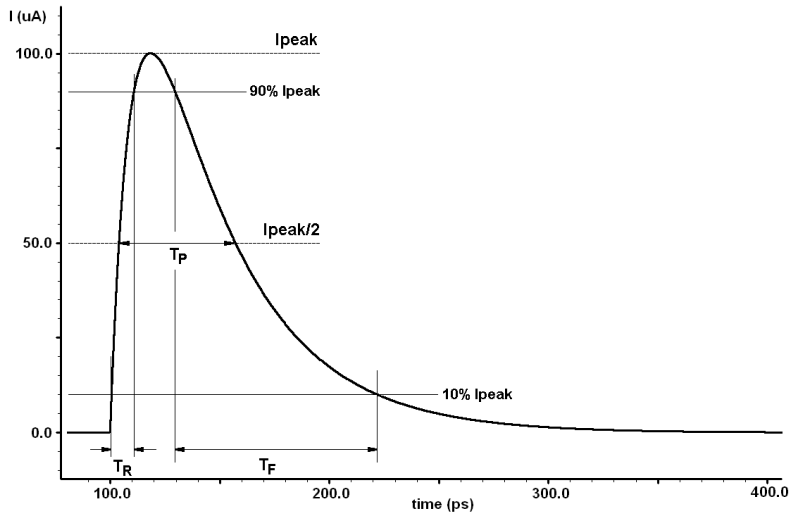
Tabela 19: Constantes de tempo usadas em uma exponencial dupla

Constantes	Parâmetro
I_1	Corrente inicial
I_2	Amplitude do pulso
T_R	Atraso do tempo de subida
τ_R	Constante de subida
T_F	Atraso do tempo de descida
τ_F	Constante de descida

$I(t) = I_1$	$0 < t \leq T_R$	(2)
$I(t) = I_1 + (I_2 - I_1) \cdot \left(1 - e^{-\frac{t-T_R}{\tau_R}}\right)$	$T_R < t \leq T_F$	(3)
$I_p(t) = I_1 + (I_2 - I_1) \cdot \left(1 - e^{-\frac{t-T_R}{\tau_R}}\right) + (I_1 - I_2) \cdot \left(1 - e^{-\frac{t-T_F}{\tau_F}}\right)$	$T_R < t \leq T_{final}$	(4)

A Figura 41 demonstra uma curva de pulso de corrente típica. O tempo de subida do pulso (T_R) é moldado por τ_R e normalmente fica em torno de picosegundos. O tempo de descida do pulso (T_F) é moldado por τ_F e é dependente da tecnologia e das características da partícula energéticas. A corrente I_2 é definida de forma a se alcançar o pico de corrente (I_{peak}) desejado (Simionovski and Wirth 2012)

Figura 41: Modelagem de uma curva dupla exponencial de corrente.



(Simionovski and Wirth 2012)

A declaração de uma fonte de corrente dupla exponencial para o simulador NGSPICE é feita na seguinte forma:

I{*nome*} Nó1 Nó2 EXP(I_1 I_2 T_R τ_R T_F τ_F)

Através desta declaração é possível modelar dois tipos de SET com uma fonte de corrente dupla exponencial:

- **SET_{101}**: este tipo de SET é gerado quando uma partícula colide com uma junção p-n de um transistor NMOS. Esse SET tem como característica um pulso do tipo 1→0→1. A Figura 42 ilustra a modelagem em nível de transistores, a declaração e a curva deste tipo de SET.
- **SET_{010}**: este tipo de SET é gerado quando uma partícula colide com uma junção p-n de um transistor PMOS. Esse SET tem como característica um pulso do tipo 0→1→0 . A Figura 43 ilustra a modelagem em nível de transistores, a declaração e a curva deste tipo de SET.

Figura 42: Declaração e simulação de um SET 1→0→1 no NGSPICE através de uma fonte de corrente.

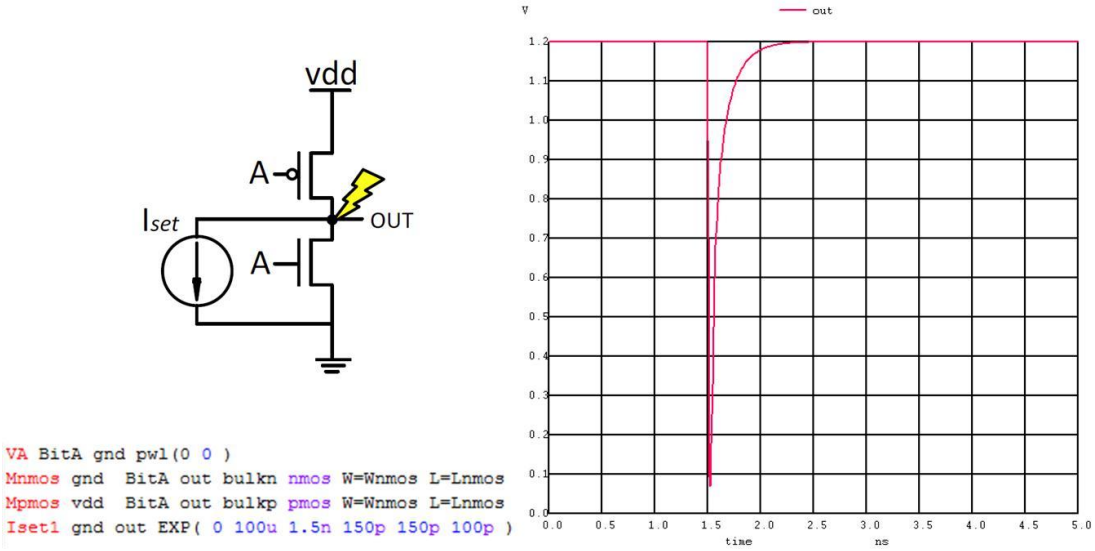
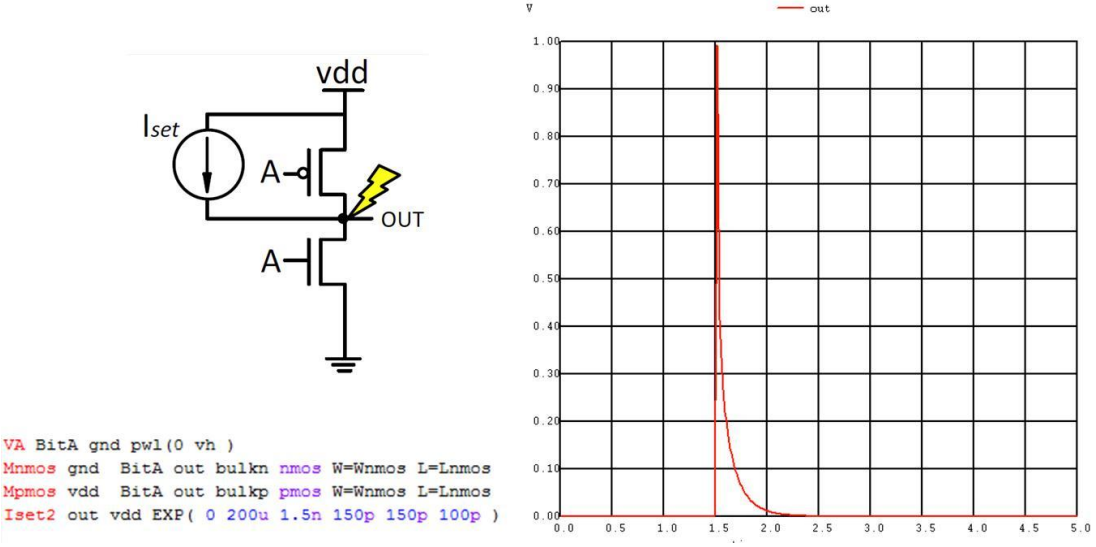


Figura 43: Declaração e simulação de um SET 0→1→0 no NGSPICE através de uma fonte de corrente.



A injeção de SET no circuito ATMR foi realizada de forma separada, testando um módulo do esquema por vez durante cada um dos vetores desprotegidos. A injeção de falhas elétricas funciona de formas diferentes entre um circuito que usa somente porta lógica complexa e um circuito multi-nível.

A. Circuito ATMR de portas Complexas

Como já abordado no sub-capítulo 3.3.4.2, a saída e os módulos G, F e H do circuito ATMR mascaram de modo diferente a uma falha dependendo do tipo de relação de estado entre eles em um dado vetor de entrada. Brevemente:

- **G=F=H:** falhas únicas são mascaradas.
- **G=F≠H:** falhas ocorrem quando a saída de G ou F passa de 0→1.
- **G=H≠F:** falhas ocorrem quando a saída de G ou H passa de 1→0.

sendo assim G está suscetível a falhas do tipo SET_{101}, vindas de sua rede *pull-down* de transistores NMOS, e falhas do tipo SET_{010}, vindas da sua rede *pull-up* de transistores

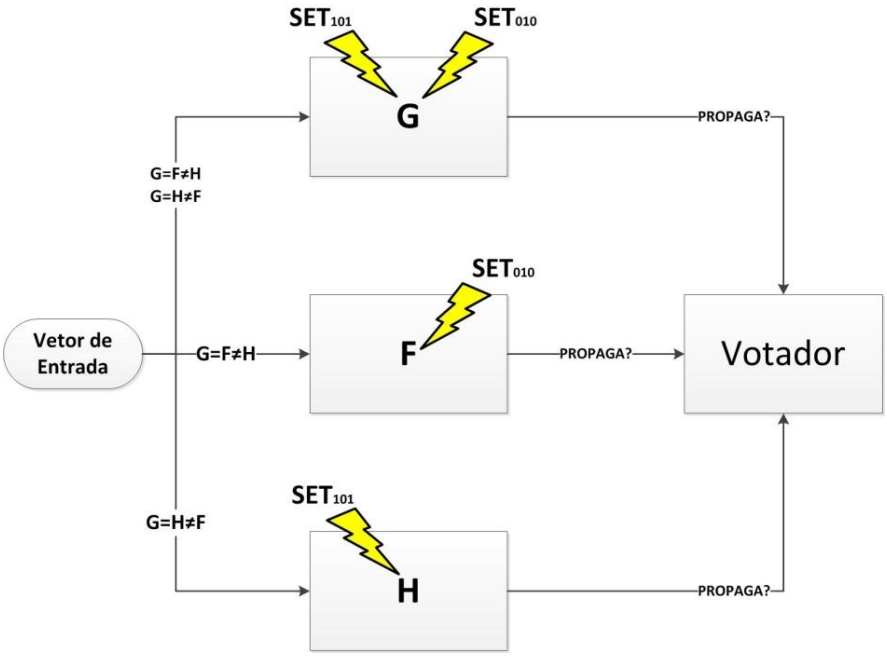
PMOS. O módulo H é suscetível somente à falhas do tipo SET_{101},geradas pela rede *pull-down* quando uma partícula colide com uma junção p-n de um transistor NMOS. As falhas SET_{010} na saída dos módulos H podem ocorrer, porém são mascaradas pelo votador. No caso do módulo F as falhas SET_{010} em sua saída não são mascaradas pelo votador, sendo assim propagadas para a saída. Já as falhas SET_{101}não propagam além do votador. A Tabela 20 sintetiza essas questões.

Tabela 20: Propagação e mascaramento de SET, tipos de vetores desprotegidos e rede de origem

Módulo	SET_{101}	SET_{010}	Estado dos módulos	Rede
G	Propaga	Propaga	Ambos	Ambas
F	Mascara	Propaga	G=F≠H	Pull-Up
H	Propaga	Mascara	G=H≠F	Pull-Down

A inserção de falhas é feita de forma individual em cada um dos módulos, inserindo falhas nos nós do circuito conforme as características do módulo e do estado do circuito (definido pelo vetor de entrada). Digamos que, por exemplo, o teste esteja sendo feito em um módulo F. Olhando para a Tabela 20 sabemos que devemos modelar falhas do tipo SET_{010}, inserindo fontes de corrente nos nós da rede *pull-up*, avaliando os circuitos somente quando o vetor de entrada leva a um estado G=H≠F entre os módulos, situação onde o votador não ira mascarar o SET. A Figura 44 ilustra parte da metodologia de teste em nível elétrico, ilustrando a questão dos vetores desprotegidos (quais estados desprotegidos para cada módulo) e que tipos de falhas são inseridas em cada módulo. Além disto a Figura 44 levanta uma questão não abordada ainda, a propagação de um SET de um nó interno do módulo até sua saída.

Figura 44: Inserção de falhas SET



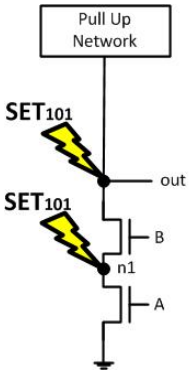
Além do mascaramento, proporcionado pelas características lógicas do esquema ATMR, é necessário analisar o mascaramento proporcionado pela estrutura dos circuitos que compõe cada módulo. Ou seja, é necessário analisar se a perturbação gerada no nó do circuito ira se propagar para a saída do módulo testado. Peguemos com exemplo a seguinte situação:

- Um circuito sobre-aproximado funcionalmente $H = \overline{A * B}$
- Sendo A=0 e B=0 um vetor desprotegido da arquitetura
- E o estado do circuito durante tal vetor sendo G=H≠F

teremos que emular falhas do tipo SET_{101} inserindo fontes de corrente em todos os nós da rede *pull-down* do módulo H. A Figura 45 ilustra a descrição do circuito e a inserção de falhas no módulo em questão. Já a Figura 46 elucida o resultado da injeção de SET. Como pode-se ver a injeção de falha no nó *out*, podendo ela vir somente a junção p-n do transistor B, gera um SET. Já a falha no nó *n1*, podendo ser gerada pelo *source* do transistor B ou pelo *drain* do transistor A, é mascarada, esse mascaramento ocorre pelo fato do transistor B estar desligado, desta forma impedido a propagação da perturbação. Dizemos então que esse módulo tem 1 junção p-n desprotegida de suas 8.

Um exemplo final é elucidado na Figura 47. Neste caso são injetadas três falhas do tipo SET_{101} em um circuito um pouco mais complexo. Novamente é possível ver que somente um dos SETs é mascarado, no caso o Iset41. Os outros dois, Iset42 e Iset43, propagam para a saída.

Figura 45: Exemplo de descrição de circuito e inserção de falha.




```
*ENTRADAS
VA BitA gnd pwl(0 0 )
VB BitB gnd pwl(0 0 )

*REDE PULL-UP
Mpmos1 vdd  BitA out bulkp pmos W=Wnmos L=Lnmos
Mpmos2 vdd  BitB out bulkp pmos W=Wnmos L=Lnmos

*REDE PULL-DOWN
Mnmos1 gnd  BitA n1 bulkn nmos W=Wnmos L=Lnmos
Mnmos2 n1   BitB out bulkn nmos W=Wnmos L=Lnmos

*INJEÇÃO DE FALHAS SET101
Iset1 gnd out EXP( 0 200u 1.5n 150p 150p 100p )
Iset2 gnd n1 EXP( 0 200u 3.5n 150p 150p 100p )
```

Figura 46: Análise de mascaramento do circuito

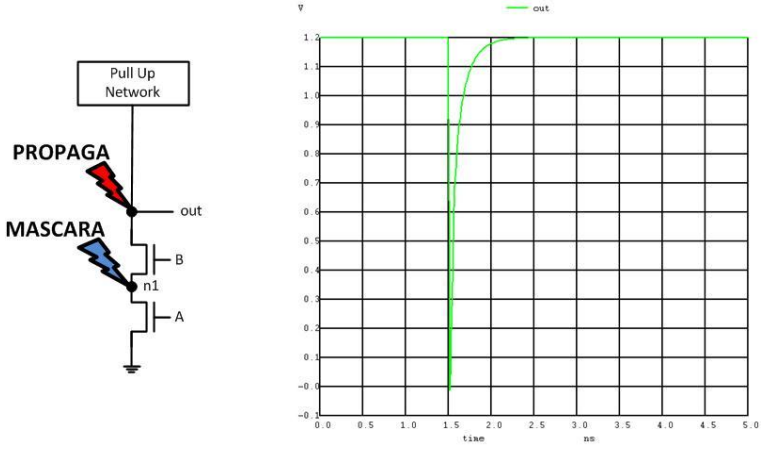
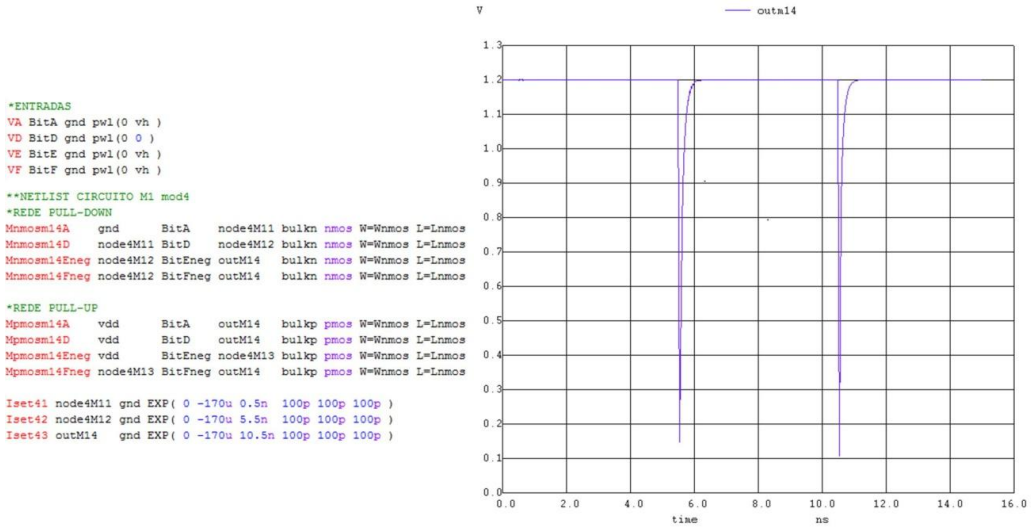


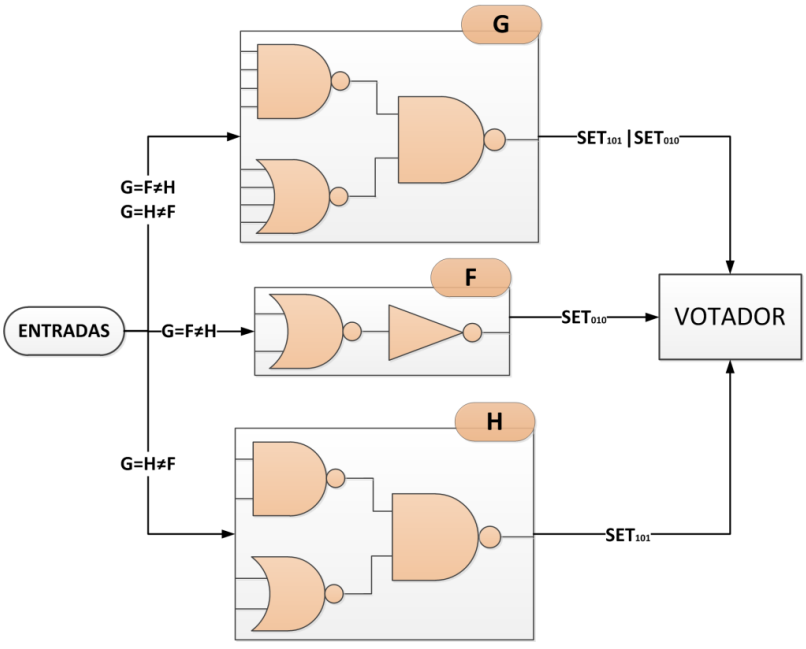
Figura 47: Exemplo de descrição e injeção de falhas. Somente a falha do nó 4M13 é mascarada (Iset41). Os outros dois SETs são propagado para a saída do circuito.



B. Circuito ATMR em multi-nível

A injeção de falhas em circuitos ATMR configurado em mais de um nível lógico modifica algumas questões de modelagem de SET devido à característica inversora intrínseca da lógica CMOS. Como visto no sub-capitulo 4.1.1, a definição de uma função aproximada como F ou H depende diretamente da paridade entre nó e saída, o encadeamento de portas lógicas CMOS implica diretamente numa mudança de paridade de um nível da lógica a outro, sendo assim necessário ajustar o tipo de SET injetado. Um exemplo de ATMR multi-nível é ilustrado na Figura 48, as regras de estado continuam as mesmas, assim como as condições de suscetibilidade de cada módulo e o mascaramento do votador.

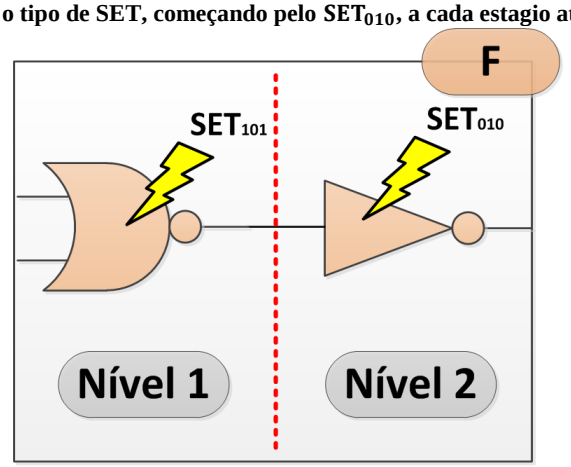
Figura 48: Exemplo de ATMR em multi-nível.



O diferencial no tratamento dos circuitos ATMR em multi-nível se refere aos níveis lógicos que causam uma inversão no pulso de propagação entre um estagio e outro. Por exemplo, ainda na Figura 48, para o módulo F gerar uma falha não mascarável pelo votador é necessário que sua saída propague um SET_{010} durante um vetor que forme um estado G=F≠H entre os módulos. Para que esta falha ocorra é necessário que:

- Um SET_{010} seja gerado pela rede *pull-up* da porta lógica do nível 2 (Inversor; Figura 49) sem que ocorra um mascaramento, desta forma o SET propaga até a saída do módulo.
- A entrada do segundo nível lógico receba um SET_{101} propagado pela saída do primeiro nível lógico (NOR; Figura 49), ocasionando a troca do estado dos transistores do inversor, gerando por conseqüência uma troca de valor na saída do módulo.

Figura 49: Injeção de falhas em um módulo F de um ATMR em multi-nível (2 níveis). Começando do nível mais alto se inverte o tipo de SET, começando pelo SET_{010}, a cada estagio até chegar à entrada.



Para o caso do módulo H a situação é similar, a Figura 50 ilustra a situação, para que a saída da porta NAND no nível 2 venha a propagar um SET_{101} é necessário que:

- Um SET_{101} seja gerado em um dos nós da porta lógica NAND no nível 2 sem que o transiente seja mascarado, gerando um SET_{101} na saída do módulo H.
- Uma de suas entradas receba um SET_{010} propagado por uma das portas lógicas do nível 1 (NAND ou NOR), no entanto, mesmo que o SET se propague até a entrada é necessário que a troca de estados dos transistores afetados pelo SET venha a gerar um caminho entre a alimentação da rede *pull-down* e a saída.

O caso do módulo G é mais simples, devido às características de propagação de transiente este módulo continua sendo testado com injeções do tipo SET_{101} e SET_{010}. A Figura 51 ilustra a situação. O único cuidado neste caso é que a inversão dos ocorre em relação ao vetor desprotegido. Por exemplo, sendo *x* é um vetor e o estado gerado por ele G=F≠H, as portas lógicas do nível 2 receberão uma falha SET_{010} durante *x* e as portas lógicas do nível 1 sofreram um transiente SET_{101}.

Figura 50: Injeção de falhas em um módulo H de um ATMR em multi-nível (2 níveis). Começando do nível mais alto se inverte o tipo de SET, começando pelo SET_{101}, a cada estagio até chegar à entrada.

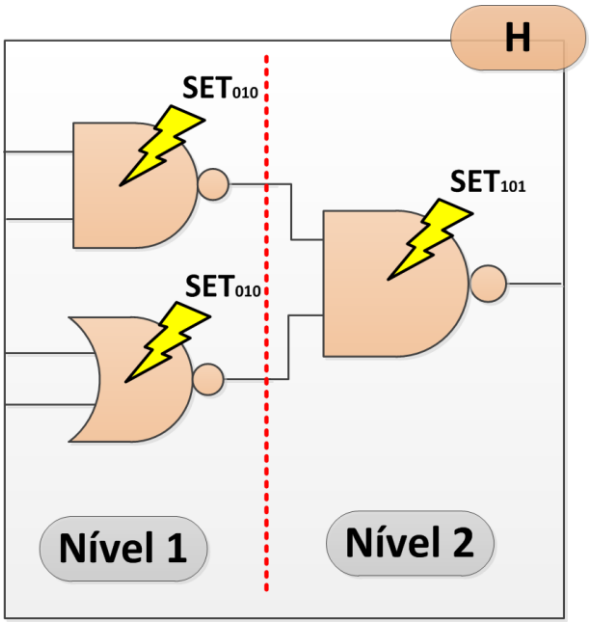
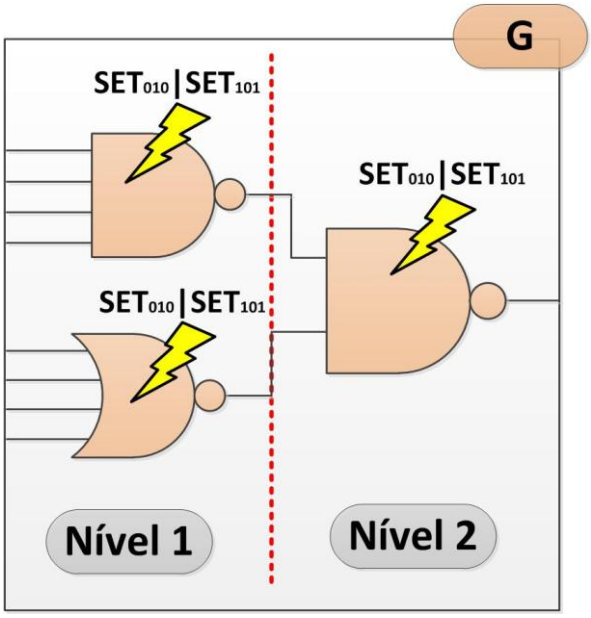


Figura 51: Injeção de falhas em um módulo H de um ATMR em multi-nível (2 níveis).



4.3.2 Injeção de falhas em nível lógico de transistores

A injeção de falha é feita em nível lógico de transistor, é similar à injeção descrita no sub-capitulo anterior, porém neste nível de abstração a tensão é uma variável discreta, ou 0 ou 1. A modelagem da falha transiente, o pulso SET, é feita utilizando comandos específicos de Verilog, tal comando gera a troca de estado de um nó do circuito. Para uma análise mais eficaz da técnica ATMR se fez necessário o desenvolvimento de uma ferramenta de injeção de falhas especifica para tal propósito. A injeção é feita utilizando um aplicativo desenvolvido em C++, utilizando juntamente o software ModelSim e a linguagem de descrição de hardware Verilog.

Um SET pode ser emulado em nível lógico, usando o software ModelSim, forçando a troca de estado em um nó. Em Verilog isto pode ser feito usando o comando *force/release* (Figura 52). O comando *force* faz com que o estado de um nó seja forçado a assumir um valor lógico desejado.O comando *release* faz com que o nó declarado volte ao seu estado original. Desta forma podemos simular um SET_{101} forçando 0 através do comando e posteriormente devolver o nó ao estado original usando o comando *release*.

```
Modelsim commands
force   wireName = forcedValue;
release wireName;
```

Figura 52: Declaração do comando force e do comando release.

O processo de descrição do circuito, teste e análise dos resultados pode ser bem demorado se feito manualmente (nó por nó). Para testar um circuito ATMR é necessário declarar o comando *force/release* em cada um dos nós para cada vetor do DUT. Além disso é necessário analisar visualmente , através do *testbench*, cada um dos sinais de estado dos nós e definir onde ocorreram erros. Para circuitos pequenos isto estes passos se tornam uma tarefa longa, para circuitos grandes se torna completamente inviável. Desta forma surgiu a necessidade do automatizar esse processo.

A automatização do processo de testes se deu inicialmente pela criação de *tasks* e circuitos extras dentro da própria descrição do DUT. Para que se automatizassem os testes, se criou uma *task* para injeção de falhas (forçando valores nos nós) e um circuito para detecção de erros (facilitando a visualização de erros) (Figura 53). No entanto a adição de novas descrições em Verilog, *task* para injeção de falhas (Figura 54) e circuito de detecção, aumentou a complexidade dos códigos, forçando assim o desenvolvimento de uma ferramenta com o propósito de gerar automaticamente os arquivos de códigos necessários.

Figura 53: Diagrama em bloco do arquivo Verilog com ATMR, *tasks* de injeção de falhas e circuito para detecção de erros.

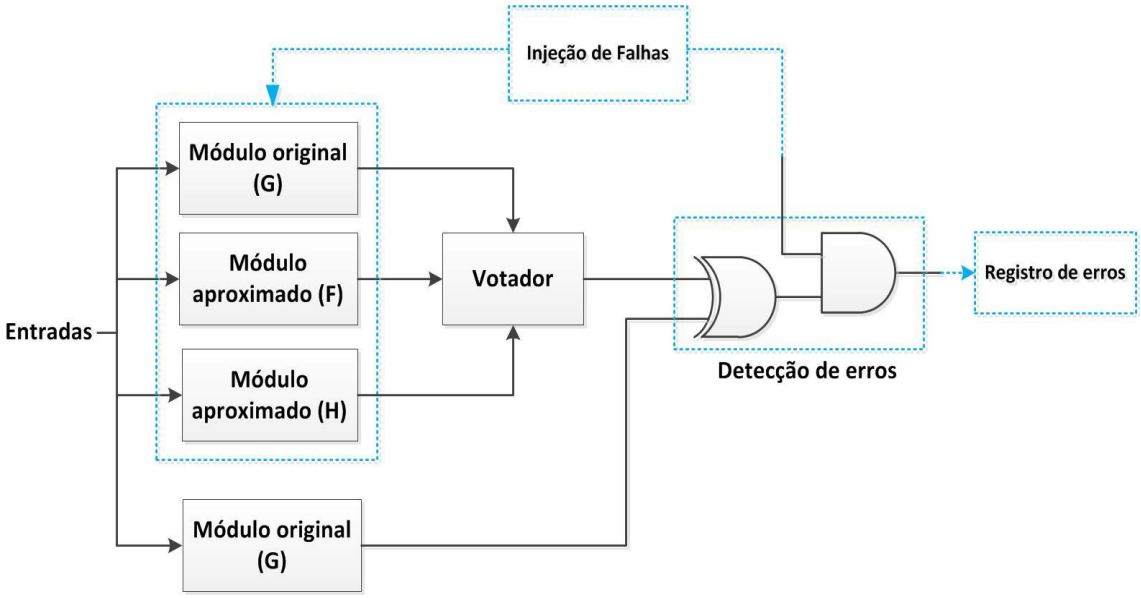


Figura 54: Task de injeção de falhas através dos comandos force e release

```
//Error injection task - Verilog
task ErrorInjection;
  input [2:0] nodeID;
  input forceValue;
  input [2:0] nodeWeight;
  begin
    #1;
    case (nodeID)
      0: force N0 = forceValue;
      1: force N1 = forceValue;
      [...]
    endcase
    force forcedNode = nodeID;
    force weight = nodeWeight;
    force injection = 1;
    #1;
    force injection = 0;
    case (nodeID)
      0: release N0;
      1: release N1;
      [...]
    endcase
  end
endtask
```

O ponto inicial, para o funcionamento da ferramenta, é uma descrição simples do circuito ATMR numa linguagem similar a Verilog (Figura 55). Esta descrição define as conexões dos transistores de cada módulo do ATMR bem como os nós de saída de cada módulo. A partir desta descrição a ferramenta gera dois tipos de arquivos necessários para a simulação e avaliação:

- **.Do file (TCL + comandos do ModelSim)**: Responsável pela geração dos vetores de entrada, execução das diretivas de simulação, análises dos resultados simulados e escrita do relatório em um arquivo de texto.
- **Verilog file (.v)**: Responsável por Injetar falhas (SET), detectar erros e atualizar os registradores de dados.

Figura 55: Descrição de entrada da ferramenta de injeção de falhas.

```
input A B C D
output outG
output outF
output outH
//Gold G
pmos (outG, vdd, A);
pmos (outG, N1, D);
pmos (N1, N2, C);
pmos (N2, vdd, B);
nmos (outG, N3, B);
nmos (outG, N3, C);
nmos (outG, N3, D);
nmos (N3, gnd, A);
//1-approximation F
pmos (outF, vdd, A);
nmos (outF, gnd, A);
//0-approximation H
pmos (outH, vdd, A);
pmos (outH, vdd, B);
nmos (outH, N0, B);
nmos (N0, gnd, A);
```

Depois da criação dos arquivos citados é possível executar a simulação diretamente pelo ModelSim (usando os arquivos *.v* e *.do* criados) ou pode ser executado

direto da ferramenta, esta rodando o ModelSim em *background* (por linha de comando). Após a simulação ter sido executado se é gerado um arquivo de texto possuindo relatórios diversos sobre o ATMR testado, um exemplo de relatório é ilustrado na Figura 56. O relatório de resultado é dividido em três análises principais, vetores desprotegidos, características dos nós e resultados globais.

A análise de vetores desprotegidos indica primeiramente para quais vetores o ATMR não satisfaz o estado G=F=H. A parte da tabela *nodes* indica individualmente quais nós do circuito propagam SET. A parte definida como *junctions* da tabela separa as falhas propagas por rede (*pull-up* e *pull-down*) referente a cada vetor desprotegido.

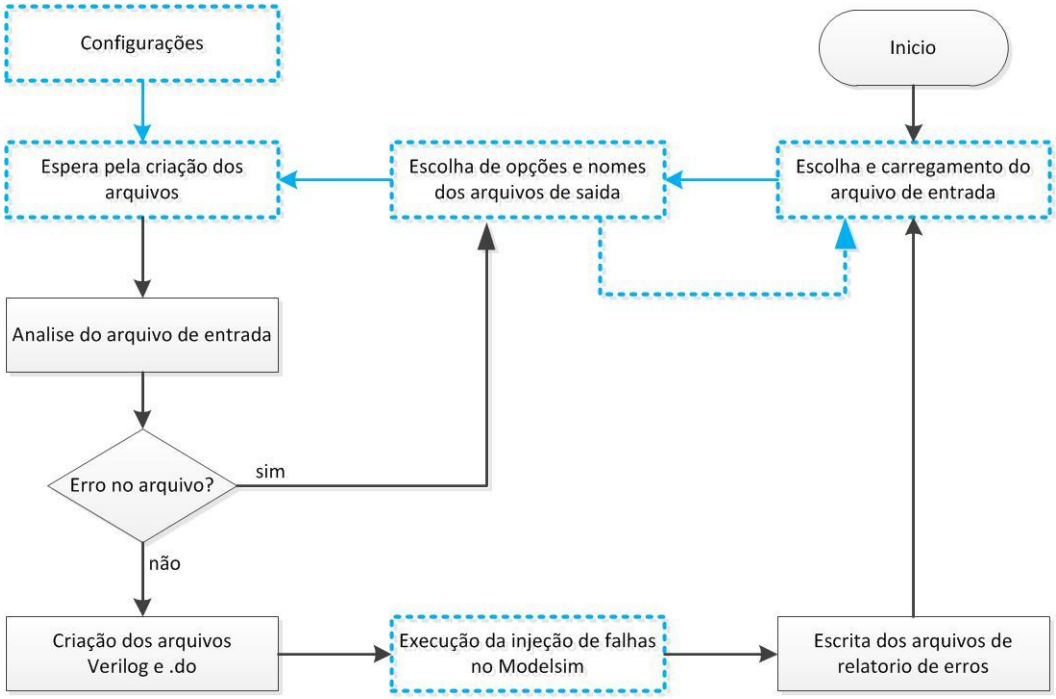
A análise dos nós primeiramente identifica qual a ID de cada nó adquirido na descrição inicial do circuito. Em seguida ele indica qual o peso de cada nó. O peso de um nó é determinado pela quantidade de junções p-n ligadas a ele, ou seja, quantas junções podem gerar um SET no nó em questão. O outro argumento é a quantidade de vetores que deixam o nó desprotegido, ou seja, quantos SETs o nó propaga. O ultimo argumento se refere ao total de junções desprotegidas geradas pelo nó.

Os resultados globais são a parte mais crucial do relatório. Nesta parte as junções desprotegidas são quantificadas e separadas por cada módulo e suas redes. Mais abaixo é dado o impacto negativo das redes de cada módulo no mascaramento lógico do ATMR testado. Um fluxograma simples do funcionamento da ferramenta é ilustrado na Figura 57.

```
########## UNPROTECTED VECTORS ##########

              Nodes                |           Junctions
       00 01 02 03 04 05 06 | Gup Gdw Fup Fdwn Hup Hdw Total (details)
0001 : .  .  .  .  x  .  x  |  .   x   .   .   .   x   3 (0+2+0+0+0+1)
0011 : .  x  x  .  x  .  x  |  .   x   .   .   .   x   7 (0+6+0+0+0+1)
0101 : .  .  .  .  x  .  x  |  .   x   .   .   .   x   3 (0+2+0+0+0+1)
0111 : .  .  .  x  x  x  .  |  x   .   x   .   .   .   6 (3+0+3+0+0+0)

################# NODES #################

ID : name, weight, unprotected vectors, unprotected junctions
00 : N1, 4, 0, 0
01 : N2, 2, 1, 2
02 : N3, 2, 1, 2
03 : N4, 2, 1, 2
04 : outG1, 3, 1, 3 (up),  2, 3, 6 (down)
05 : outF1, 1, 1, 1 (up),  2, 0, 0 (down)
06 : outH1, 1, 0, 0 (up),  1, 3, 3 (down)


############# GLOBAL RESULTS ############

Protected p-n junction

G (up)  : 3 unprotected
G (down): 10 unprotected
F (up)  : 3 unprotected
F (down): 100% protected
H (up)  : 100% protected
H (down): 3 unprotected |

Impact on TMR

G (up)  : 0.67%
G (down): 2.2%
F (up)  : 0.67%
F (down): 0.0%
H (up)  : 0.0%
H (down): 0.67%
```

Figura 56: Relatório de resultados gerado pela ferramenta.

Figura 57: Fluxograma da ferramenta de injeção de falhas desenvolvida.



4.4 Técnicas de reordenamento estrutural

A área sensível a falhas de um circuito lógico pode ser definida como o conjunto de nós sensíveis (junções p-n dos transistores). Desta forma, para qualquer porta lógica CMOS contendo *n* junções p-n nas redes *pull-up* e *pull-down*, irão existir *m* nós sensíveis, das *n* junções do circuito, que poderão ser afetados por um SET e propagar o erro para a saída da porta lógica para cada vetor de entrada. Podemos usar a razão *m/n*para descrever a área do circuito sensível a falhas. Como já visto antes, o objetivo de um TMR tradicional é reduzir a quantia *m* de nós sensíveis para zero, já que no TMR todas as falhas únicas são mascaradas pelo votador.

Quando se usa o ATMR, não se faz mais possível a redução de *m* para zero na presença de falhas, como já foi descrito anteriormente, para alguns vetores de entrada, a saída dos três módulos redundantes esta incompatível. Porem o objetivo de diminuir ao maximo o valor de *m* continua existindo. Quando utilizamos o ATMR é possível tirar proveito do conhecimento prévio de quais vetores de entrada podem gerar falhas, sabendo quais vetores estão desprotegidos permite que se personalize de forma especifica o circuito ATMR de forma que os vetores sensíveis venham a ter suas que chances de produzir falhas diminuídas. Existem dois tipos de reordenamento que podemos fazer em um circuito ATMR para aprimorar o mascaramento lógico do mesmo, o reordenamento de entradas e o reordenamento de transistores.

4.4.1 Reordenamento de entradas

A primeira abordagem para personalizar um circuito aproximado é o reordenamento de entradas. O reordenamento de entrada consiste basicamente na troca dos sinais de entrada dos transistores do circuito de forma que tal permutação não modifique a função lógica ou a topologia do circuito.

Na Figura 58 temos dois circuitos que representam a mesma função lógica (NAND), a única diferença entre eles é a entrada que cada um dos transistores recebe. Vamos assumir que o único vetor desprotegido para esta função lógica é A=1 e B=0, essas entradas irão fazer o transistor *T2a* e *T1b* estarem desligados, portanto não existe corrente significativa, desta forma, o valor lógico contido na fonte do transistor não é transmitido ao dreno do mesmo. Antes de aprofundar mais o exemplo, é importante lembrar que com essa configuração de entradas não é possível um erro ser gerado a partir da rede *pull-up* já que a saída do circuito esta em 1 e os dispositivos PMOS geram apenas falhas SET_{010}, por essa razão os transistores da rede estão omitidos na Figura 58.

Agora supúnhamos que um SET_{101} ocorra no nó *n1a* (Figura 58a), isso pode ocorrer quando uma partícula atinge uma das junções p-n conectadas ao nó (dreno de *T1a* ou fonte de *T2b*), esta perturbação transiente na tensão ira possuir um caminho direto para a saída (*out*) já que *T2a* esta ligado, desta forma permitindo que o SET_{010} propague até a saída (*out*). Porém caso um SET_{101} ocorra no nó *n1b* (Figura 58b), o mesmo não será propagado para saída (*out*) simplesmente porque o transistor *T2b* esta desligado, desta forma impedindo que qualquer corrente passe por ele. No entanto nenhum dos circuitos ira mascarar um SET_{101} gerado no seu respectivo nó *n2*, ambos SET_{101} ocorrendo quando uma partícula energética atinge o dreno do transistor *T2a* ou *T2b*. Assumindo que o circuito tem somente 4 vetores, temos para o circuito da Figura 59(a) um total de 36 junções (8 para cada vetor), sendo 3 delas desprotegidas, como vimos acima, sendo assim temos uma razão de 3/36 junções sensíveis. Já o circuito da Figura 59(b) possui uma razão *m/n* igual a 1/36. A Figura 59 ilustra o exemplo.

Figura 58: Técnica de reordenamento de entradas.

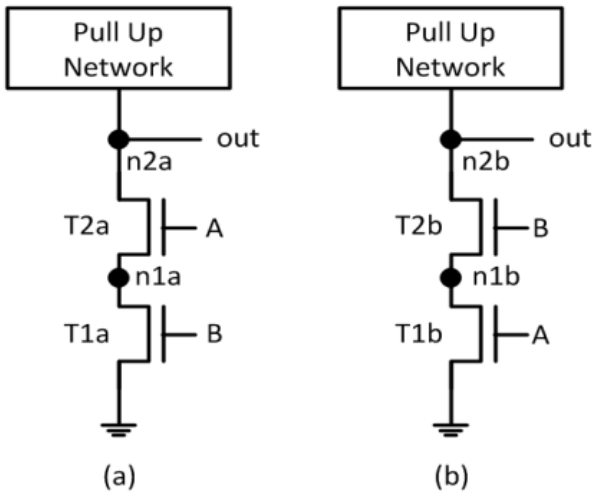
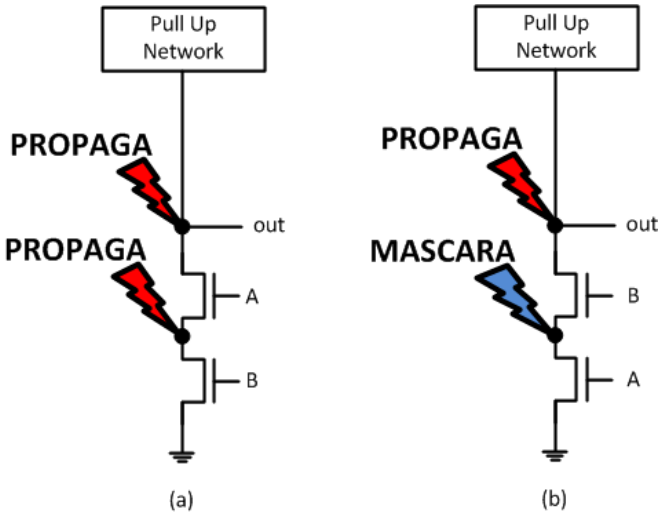


Figura 59: Melhora do mascaramento através da técnica de reordenamento de entradas.



Por fim pode-se ver que, além do impacto no mascaramento lógico, não seria necessário modificar o leiaute do circuito para efetuar a troca das entradas dos transistores, sendo possível dessa forma a utilização da técnica em uma biblioteca de células pré-definida.

4.4.2 Reordenamento de transistores

A segunda abordagem para criar circuitos ATMR mais confiáveis é a análise de topologias diferentes, a idéia é similar à técnica de reordenamento de entradas descrita anteriormente, porem a permutação é feita na disposição dos transistores. O reordenamento de transistores é constituído na modificação do arranjo dos transistores do circuito sem que ocorra uma modificação na função lógica que o mesmo representa.

A Figura 60 ilustra duas topologias possíveis para a função $\overline{A * (B + C)}$. Assumindo dessa vez que o único vetor desprotegido do esquema ATMR é A = 0, B = 0 e C = 1. Se um SET_{101}ocorre no nó *n1a* (Figura 60a) a perturbação não vai ser propagada já que não existe um caminho direto entre o nó e a saída (o transistor *T3a* está desligado). A única forma para o circuito na Figura 60a propagar um erro é se um SET_{101} ocorrer no nó *n2a* (colisão com o dreno de *T3a*). No caso do circuito da Figura 60b um SET_{101} no nó *n1b* (colisão com dreno de *T3b* ou fonte de *T1b e* T2b) irá propagar para a saída do módulo através do transistor *T2b* já que o mesmo está ligado. O nó *n2b* também ira gerar um erro caso um SET_{101} venha a ocorrer nele.

Assumindo que o circuito tem somente 8 vetores, temos para o circuito da Figura 60(a) um total de 96 junções (12 para cada vetor), sendo somente 1 delas desprotegida, como vimos acima, sendo assim temos uma razão de 1/96 junções sensíveis. Já o circuito da Figura 60(b) possui uma razão *m/n* igual a 5/96. A Figura 61 ilustra o exemplo. Para esta técnica de reordenamento seria necessário modificar o *layout* do circuito para efetuar a modificação topológica dos transistores, sendo dessa forma dificultada a utilização da técnica em uma biblioteca de células.

Figura 60: Reordenamento de transistores

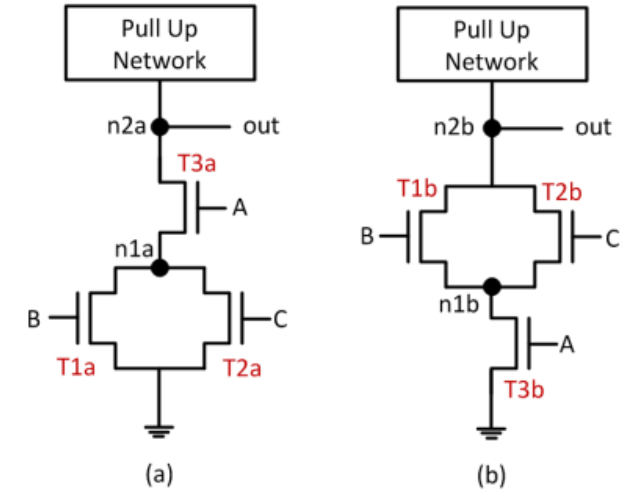
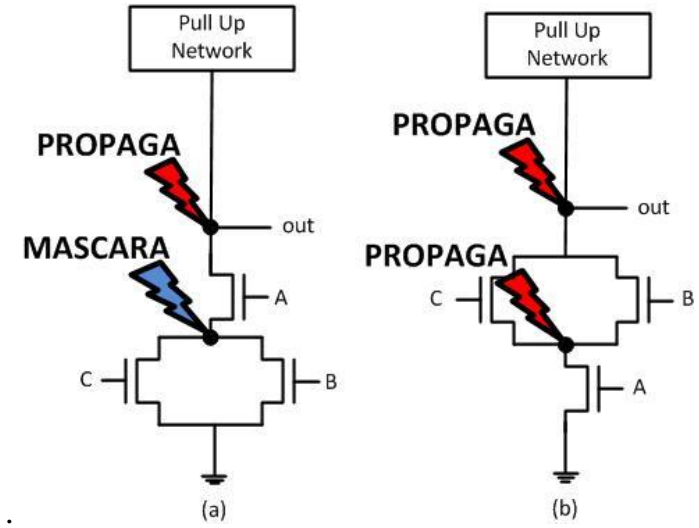


Figura 61: Melhora do mascaramento através da técnica de reordenamento de transistores



5. RESULTADOS

O estudo da técnica de tolerância à falha ATMR foi realizado utilizando esquemas variados, tanto no âmbito das funções lógicas originais, quanto na questão da composição dos módulos originais (G), sub-aproximados (F) e sobre-aproximados (H). Podemos dividir os resultados conforme a característica da técnica ATMR que se buscava avaliar: reordenamento estrutural ou granularidade.

5.1 Resultados: técnicas de reordenamento estrutural

Primeiramente a análise foi feita com intenção de avaliar a viabilidade das técnicas de reordenamento estrutural. Esta avaliação foi feita utilizando injeção de falha em nível elétrico. Foram testados ao todo seis esquemas ATMR diferentes, cada um com uma função original, dos quais quatro esquemas de portas complexas e dois em multi-nível para avaliar a técnica nestes tipos de circuitos.

A. Estudo de caso: Circuito 1

O primeiro circuito testado é uma função lógica implementada como porta complexa. A composição do ATMR foi feita da seguinte forma:

Função original:

$$G = \overline{(A * B) + (C * D)}$$

Função sub-aproximada:

$$F = \overline{B + (C * D)}$$

Função sobre-aproximada:

$$H = \overline{C * D}$$

Esta composição de circuito ATMR possui um aumento de área de 125%, possuindo ao todo 18 transistores. Desta forma o ATMR possui 36 junções p-n para cada um de seus 16 vetores de entrada, totalizando 576 junções. Este esquema tem ao todo 6 vetores desprotegidos, isto é, seis situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original. A Tabela 21 lista os vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 22 corresponde ao resultado de junções

desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 96,18% de suas junções protegidas
- Pior circuito com 90,97% de suas junções protegidas
- Diferença entre melhor e pior é de 5,21%

Tabela 21: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 1 no âmbito do reordenamento estrutural.

Entradas (ABCD)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
0100	G=H≠F	5	3
0101	G=H≠F	5	7
0110	G=H≠F	3	9
1100	G=F≠H	3	11
1101	G=F≠H	3	11
1110	G=F≠H	3	11
		22 junções	**52 junções**

Tabela 22: Quantificação das junções desprotegidas de cada módulo do estudo de caso 1 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	14	32
F	3	5
H	5	15
	22 junções	**52 junções**

B. Estudo de caso: Circuito 2

O segundo circuito testado é uma função lógica implementada como porta complexa. A composição do ATMR foi feita da seguinte forma:

- Função original:
$$G = \overline{A * (B + C + D)}$$
- Função sub-aproximada:
$$F = \overline{A}$$
- Função sobre-aproximada:
$$H = \overline{A * D}$$

Esta composição de circuito ATMR possui um aumento de área de 75%, possuindo ao todo 14 transistores. Desta forma o ATMR possui 28 junções p-n para cada um de seus 16 vetores de entrada, totalizando 448 junções. Este esquema tem ao todo 4 vetores desprotegidos, isto é, quatro situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original. A Tabela 23 lista os

vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 24 corresponde ao resultado de junções desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 97,40% de suas junções protegidas
- Pior circuito com 94,97% de suas junções protegidas
- Diferença entre melhor e pior é de 2,43%

Tabela 23: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 2 no âmbito do reordenamento estrutural.

Entradas (ABCD)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
1000	G=H≠F	4	6
1010	G=H≠F	5	7
1100	G=H≠F	3	9
1110	G=F≠H	3	7
		15 junções	**29 junções**

Tabela 24: Quantificação das junções desprotegidas de cada módulo do estudo de caso 2 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	11	19
F	1	1
H	3	10
	15 junções	**29 junções**

C. Estudo de caso: Circuito 3

O terceiro circuito testado é uma função lógica implementada como porta complexa. A composição do ATMR foi feita da seguinte forma:

- Função original:
$$G = \overline{A + (B * C * D)}$$
- Função sub-aproximada:
$$F = \overline{A + D}$$
- Função sobre-aproximada:
$$H = \overline{A}$$

Esta composição de circuito ATMR possui um aumento de área de 75%, possuindo ao todo 14 transistores. Desta forma o ATMR possui 28 junções p-n para cada um de seus 16 vetores de entrada, totalizando 448 junções. Este esquema tem ao

todo 4 vetores desprotegidos, isto é, quatro situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original. A Tabela 25 lista os vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 26 corresponde ao resultado de junções desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 97,74% de suas junções protegidas
- Pior circuito com 95,66% de suas junções protegidas
- Diferença entre melhor e pior é de 2,08%

Tabela 25: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 3 no âmbito do reordenamento estrutural.

Entradas (ABCD)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
0001	G=H≠F	3	5
0011	G=H≠F	3	5
0101	G=H≠F	5	7
0111	G=F≠H	2	8
		13 junções	**25 junções**

Tabela 26: Quantificação das junções desprotegidas de cada módulo do estudo de caso 3 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	9	19
F	1	3
H	3	3
	13 junções	**25 junções**

D. Estudo de caso: Circuito 4

O quarto circuito testado é uma função lógica implementada como porta complexa. A composição do ATMR foi feita da seguinte forma:

- Função original:

$$G = \overline{A * \left((C * B) + \left(D * \left(\overline{E} + \overline{F}\right)\right)\right)}$$

- Função sub-aproximada:

$$F = \overline{A * \left(C + \left(D * \left(\overline{E} + \overline{F}\right)\right)\right)}$$

- Função sobre-aproximada:

$$H = \overline{A * D * \left(\overline{E} + \overline{F}\right)}$$

Esta composição de circuito ATMR possui um aumento de área de 150%, possuindo ao todo 30 transistores. Desta forma o ATMR possui 60 junções p-n para cada um de seus 64 vetores de entrada, totalizando 3840 junções. Este esquema tem ao todo 10 vetores desprotegidos, isto é, dez situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original. A Tabela 27 lista os vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 28 corresponde ao resultado de junções desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 98,80% de suas junções protegidas
- Pior circuito com 96,04% de suas junções protegidas
- Diferença entre melhor e pior é de 2,74%

Tabela 27: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 4 no âmbito do reordenamento estrutural.

Entradas (ABCDEF)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
101000	G=H≠F	3	17
101001	G=H≠F	3	17
101010	G=H≠F	3	17
101011	G=H≠F	3	10
101111	G=H≠F	9	10
111000	G=F≠H	5	13
111001	G=F≠H	5	17
111010	G=F≠H	5	17
111011	G=F≠H	5	17
111111	G=F≠H	5	17
		46 junções	**152 junções**

Tabela 28: Quantificação das junções desprotegidas de cada módulo do estudo de caso 4 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	28	85
F	10	38
H	8	29
	46 junções	**152 junções**

E. Estudo de caso: Circuito 5

O quinto circuito testado é uma função lógica implementada em multi-nível. A composição do ATMR foi feita da seguinte forma:

- Função original:

$$G = \overline{\overline{((A * B) + (C * D))} * \overline{(E + F)}}$$

- Função sub-aproximada:

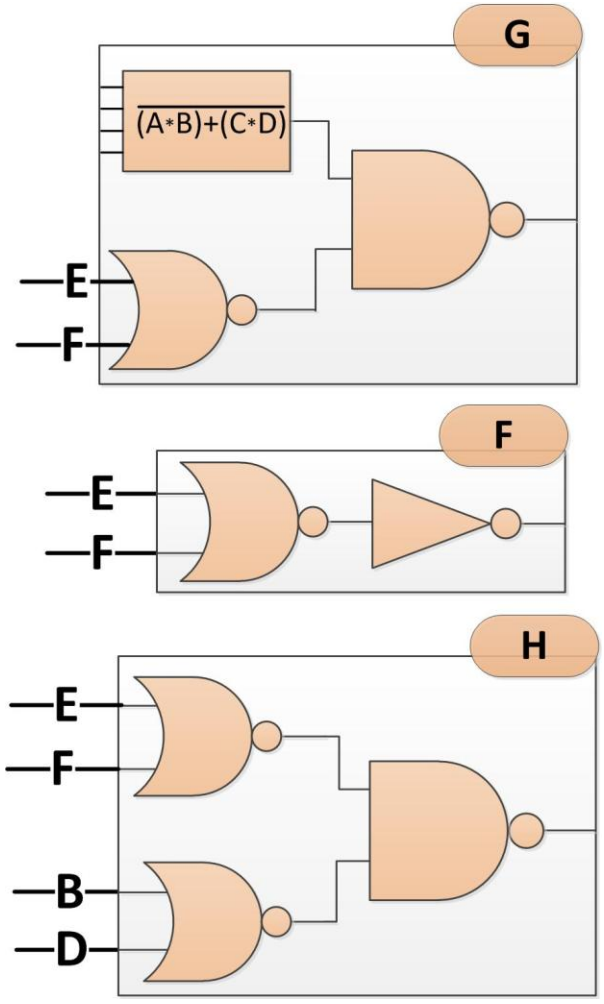
$$F = \overline{\overline{E + F}}$$

- Função sobre-aproximada:

$$H = \overline{\overline{(B + D)} * \overline{(E + F)}}$$

Esta composição de circuito ATMR, ilustrada na Figura 62, possui um aumento de área de 112,5%, possuindo ao todo 36 transistores. Desta forma o ATMR possui 72 junções p-n para cada um de seus 64 vetores de entrada, totalizando 4824 junções. Este esquema tem ao todo 11 vetores desprotegidos, isto é, onze situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original.

Figura 62: Composição multi-nível do circuito 5.



A Tabela 29 lista os vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 30 corresponde ao resultado de junções desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 97,95% de suas junções protegidas
- Pior circuito com 96,53% de suas junções protegidas
- Diferença entre melhor e pior é de 1,42%

Tabela 29: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 5 no âmbito do reordenamento estrutural.

Entradas (ABCDEF)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
000100	G=F≠H	9	11
001100	G=H≠F	11	13
010000	G=F≠H	9	11
010100	G=F≠H	9	13
011000	G=F≠H	11	11
011100	G=H≠F	9	13
100100	G=F≠H	11	11
110000	G=H≠F	5	19
110100	G=H≠F	5	17
111000	G=H≠F	5	19
111100	G=H≠F	5	13
		89 junções	**151 junções**

Tabela 30: Quantificação das junções desprotegidas de cada módulo do estudo de caso 5 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	60	96
F	15	15
H	14	40
	89 junções	**151 junções**

F. Estudo de caso: Circuito 6

O quinto circuito testado é uma função lógica implementada em multi-nível. A composição do ATMR foi feita da seguinte forma:

- Função original:

$$G = \overline{\left((A * B * (C + D)\right)} * \overline{(E + F)}$$

- Função sub-aproximada:

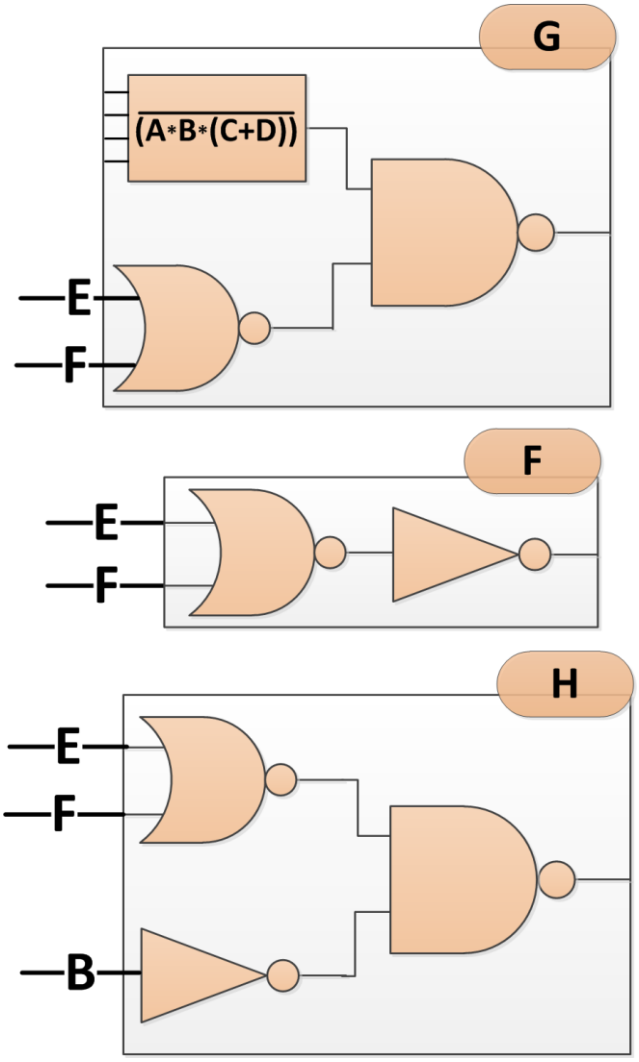
$$F = \overline{\overline{E + F}}$$

- Função sobre-aproximada:

$$H = \overline{(B)} * \left(\overline{E + F}\right)$$

Esta composição de circuito ATMR, ilustrada na Figura 63, possui um aumento de área de 100%, possuindo ao todo 32 transistores. Desta forma o ATMR possui 64 junções p-n para cada um de seus 64 vetores de entrada, totalizando 4096 junções. Este esquema tem ao todo 8 vetores desprotegidos, isto é, oito situações onde um dos circuitos aproximados não converge par ao mesmo valor do circuito original.

Figura 63: Composição multi-nível do circuito 6.



A Tabela 31 lista os vetores de entrada que deixam o circuito ATMR desprotegido, e o estado do esquema conforme a entrada. A tabela também quantifica a quantidade de nós desprotegidos por cada vetor, separando entre dois casos, o de melhor reordenamento estrutural e o de pior reordenamento estrutural. A Tabela 32 corresponde ao resultado de junções desprotegidas para cada módulo que compõe o ATMR, novamente os resultados são separados entre melhor e o pior reordenamento estrutural. Em ultima análise temos:

- Melhor circuito com 98,51% de suas junções protegidas
- Pior circuito com 97,44% de suas junções protegidas
- Diferença entre melhor e pior é de 1,07%

Tabela 31: Estado do ATMR conforme o vetor de entrada, e quantia de junções desprotegidas do estudo de caso 6 no âmbito do reordenamento estrutural.

Entradas (ABCDEF)	Estado do ATMR	Junções desprotegidas	
		Melhor caso	Pior caso
010000	G=H≠F	8	11
010100	G=H≠F	8	13
011000	G=H≠F	8	13
011100	G=H≠F	8	13
110000	G=H≠F	9	11
110100	G=F≠H	8	16
111000	G=F≠H	6	14
111100	G=F≠H	6	14
		61 junções	**105 junções**

Tabela 32: Quantificação das junções desprotegidas de cada módulo do estudo de caso 6 no âmbito do reordenamento estrutural.

Módulo	Junções desprotegidas	
	Melhor caso	Pior caso
G	40	72
F	15	15
H	6	18
	61 junções	**105 junções**

Por fim a Tabela 33 sintetiza os resultados obtidos utilizando a as técnicas de reordenamento estrutural.

Tabela 33: Resultados dos testes das técnicas de reordenamento estrutural

Circuito	Aumento de área	Vetores desprotegidos	Melhor caso	Pior caso	Diferença
C1 - porta complexa	125%	6	96,18%	90,97%	5,21%
C2 - porta complexa	75%	4	97,40%	94,97%	2,42%
C3 - porta complexa	75%	4	97,74%	95,66%	2,08%
C4 - porta complexa	150%	10	98,80%	96,04%	2,74%
C5 - multi-nível	112,5%	11	97,95%	96,53%	1,42%
C6 - multi-nível	100%	8	98,51%	97,44%	1,07%

5.2 Resultados: granularidade da composição ATMR

A análise da granularidade visa avaliar a utilização de composições diferentes para um mesmo ATMR, isto é, qual o comportamento de um ATMR baseado em uma função G variando somente suas funções aproximadas F e H. Esta avaliação foi realizada através de injeção de falhas em nível lógico, se utilizou a ferramenta de testes de ATMR desenvolvida durante o estudo. Foram testados ao todo 4 tipos de função G diferentes, todas as funções do ATMR sendo descritas como portas complexas. Cada uma das funções originais (G) derivou de cinco a sete composições diferentes de ATMR (conjunto G, F e H) através da variação das funções aproximadas. Três destes conjuntos de composições foram descritos com exclusivamente portas complexas e mais um utilizando portas lógicas em multi-nível.

A. Estudo de caso 1: granularidade da função Gx

O primeiro estudo de caso foi criado utilizando se a seguinte função original de 6 entradas com portas complexas:

$$Gx = \overline{\big((A + B) * C\big) + D) * (E + F)}$$

Baseando se na função *Gx*, foram computadas 93 funções aproximadas, das quais 26 era sub-aproximadas (*Fx*) e 67 eram sobre-aproximadas (*Hx*). Algumas destas funções foram selecionadas (Tabela 34) para compor sete esquemas diferentes (Tabela 35).

Tabela 34: Características das funções selecionadas para *Gx*

Aproximação		Função	Hamming
Fx	*Fx1*	$\overline{(C + D) * (E + F)}$	3
	Fx2	$\overline{C + D}$	15
Hx	*Hx1*	$\overline{((B * C) + D) * (E + F)}$	3
	Hx2	$\overline{D * (E + F)}$	9
	Hx3	$\overline{D * E}$	17

Tabela 35: Composições de ATMR para o caso 1: *Gx*

Composição	Aumento de área	Módulo 1	Módulo 2	Módulo 3	Vetores	Transistores
S1	83%	*Gx*	*Fx2*	*Hx2*	24	22
S2	100%	*Gx*	*F1x*	*Hx3*	20	24
S3	117%	*Gx*	*Fx1*	*Hx2*	12	26
S4	133%	*Gx*	*Fx2*	*Gx*	15	28
S5	150%	*Gx*	*Gx*	*Hx2*	9	30
S6	167%	*Gx*	*Fx1*	*Gx*	3	32
S7	183%	*Gx*	*Fx2*	*Gx*	3	34
TMR	200%	*Gx*	*Gx*	*Gx*	0	36

A Figura 64 compara a razão de junções p-n protegidas de cada composição de circuito ATMR de *Gx*. Como podemos ver a diferença entre um TMR tradicional, com 200% de custo extra de área, e a composição *S1* com, 88% de aumento de área, é de 4,7% para o melhor caso e 8,9% para o pior. A média de diferença entre os melhores e piores casos é de 2,2%. Também podemos comparar o numero total de junções desprotegidas para melhores e piores casos. (Figura 65). Como podemos ver, para os melhores casos, é possível notar que algumas composições menores possuem praticamente o mesmo numero de junções desprotegidas, como no caso de *S1* (83% de aumento de área) e *S2* (100%), com 132 e 133 junções desprotegidas respectivamente. Existem casos também onde composições menores possuem um menor numero de junções desprotegidas que uma composição maior, como por exemplo, *S3* (75 junções) versus S4 (90 junções desprotegidas) e *S6* (18 junções) contra *S7* (22 junções).

Figura 64: Taxa de junções p-n protegidas para cada composição de *Gx*

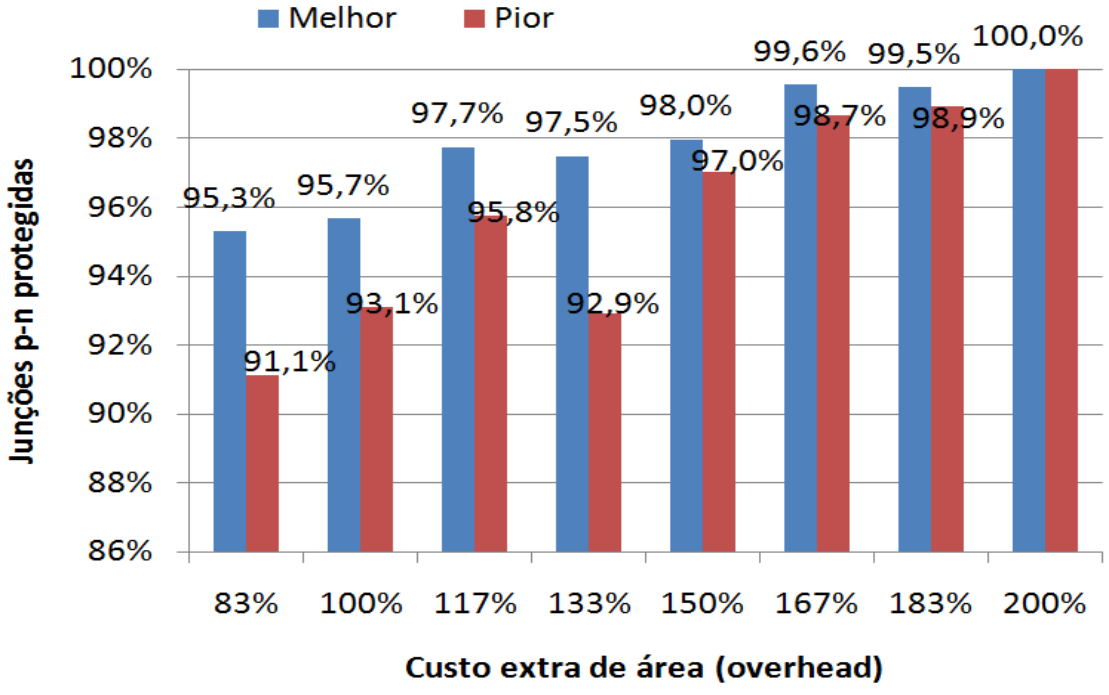
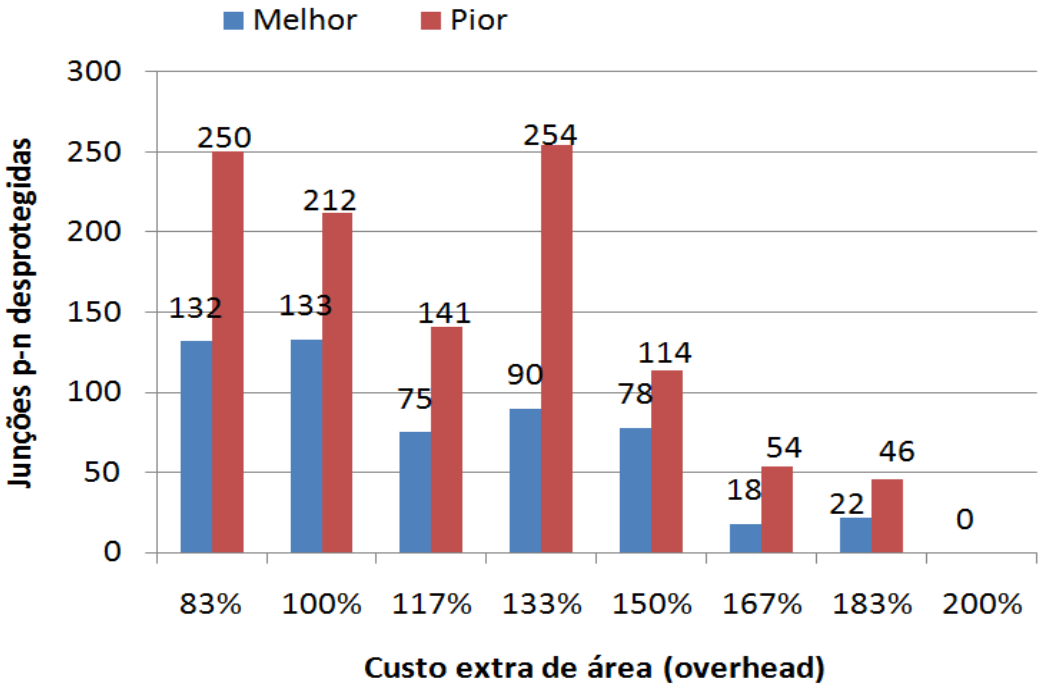


Figura 65: Valor absoluto de junções p-n desprotegidas para cada composição de *Gx*



Por fim a Tabela 36 relaciona a quantidade de vetores desprotegidos com o mascaramento da melhor e da pior estrutura e a diferença entre os casos. Pode-se ver que a taxa de junções protegida aumenta conforme se diminui a quantidade de vetores desprotegidos.

Tabela 36: Resultados de mascaramento lógico para melhor e pior estrutura para as composições de Gx

Composição	Vetores	Estrutura		Diferença
		Melhor	Pior	
S1	24	95,3%	91,1%	4,2%
S2	20	95,7%	93,1%	2,6%
S3	12	97,7%	95,8%	2,0%
S4	15	97,5%	92,9%	4,6%
S5	9	98,0%	97,0%	0,9%
S6	3	99,6%	98,7%	0,9%
S7	3	99,5%	98,9%	0,6%
TMR	0	100,0%	100,0%	0,0%

B. Estudo de caso 2: granularidade da função Gy

O segundo estudo de caso foi criado utilizando se a seguinte função original de 8 entradas utilizando portas complexas:

$$Gy = \overline{(A+B)*(C+D)*(E+F)*(G+H)}$$

Baseando se na função *Gy*, foram computadas 317 funções aproximadas, das quais 86 era sub-aproximadas (*Fy*) e 231 eram sobre-aproximadas (*Hy*). Algumas destas

funções foram selecionadas (Tabela 37) para compor cinco esquemas diferentes (Tabela 38)

Tabela 37: Características das funções selecionadas para *Gy*

Aproximação		Função	Hamming
Fy	*Fy1*	$\overline{(A+B)*(C+D)*(E+F)}$	27
	Fy2	$\overline{(A+B)*(C+D)}$	63
	Fy3	$\overline{A+B}$	111
Hy	*Hy1*	$\overline{((B*C)+D)*(E+F)}$	65

Tabela 38: Composições de ATMR para o caso 2: *Gy*

Composição	Aumento de área	Módulo 1	Módulo 2	Módulo 3	Vetores	Transistores
S1	75%	*Gy*	*Fy3*	*Hy1*	176	28
S2	100%	*Gy*	*Fy2*	*Hy1*	128	32
S3	125%	*Gy*	*Fy3*	*Hy1*	92	36
S4	150%	*Gy*	*Fy2*	*Gy*	63	40
S5	175%	*Gy*	*Fy1*	*Gy*	27	44
TMR	200%	*Gy*	*Gy*	*Gy*	0	48

A Figura 66 compara a razão de junções p-n protegidas de cada composição de circuito ATMR de *Gy*. Como podemos ver a diferença entre um TMR tradicional, com 200% de custo extra área, e a composição *S1* com, 75% de aumento de área, é de 10,2% para o melhor caso e 13,5% para o pior. A média de diferença entre os melhores e piores casos é de 3,9%. Também podemos comparar o numero total de junções desprotegidas para melhores e piores casos. (Figura 67).

Figura 66: Taxa de junções p-n protegidas para cada composição de *Gy*

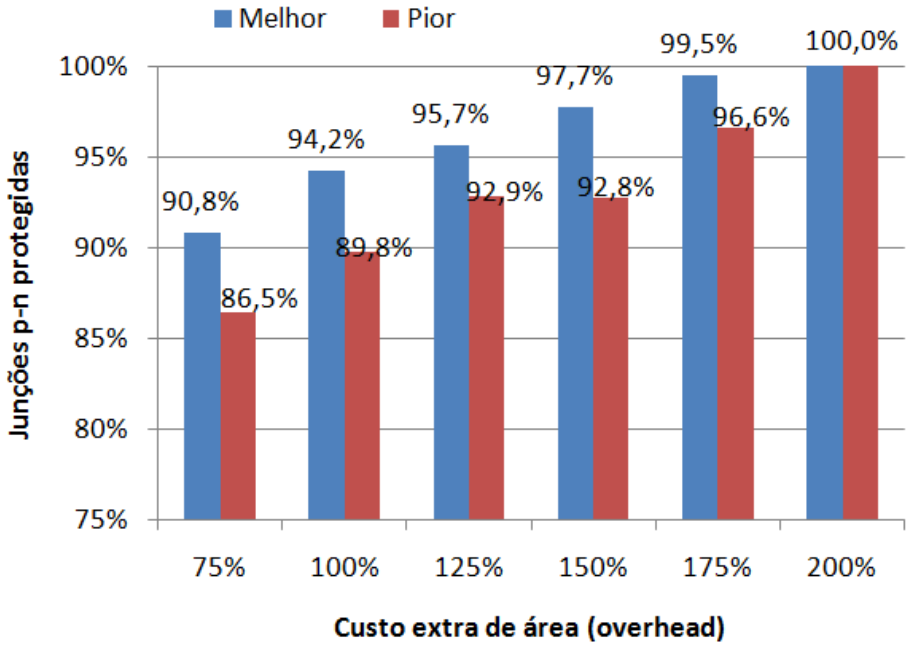
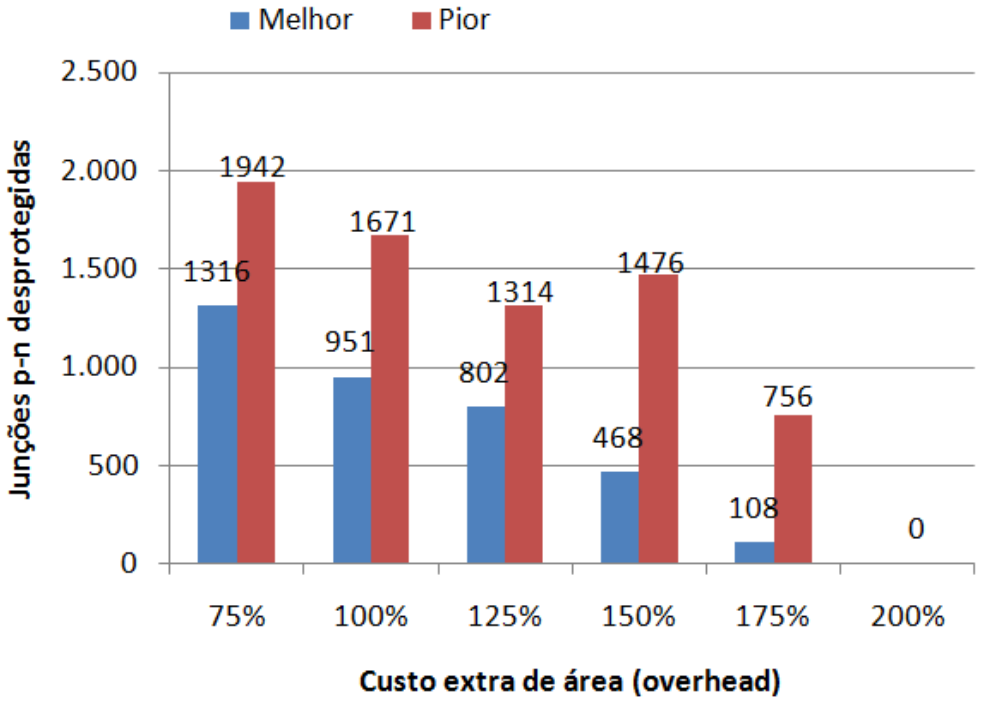


Figura 67: Valor absoluto de junções p-n desprotegidas para cada composição de *Gy*



Por fim a Tabela 39 relaciona a quantidade de vetores desprotegidos com o mascaramento da melhor e da pior estrutura e a diferença entre os casos. Pode-se ver que a taxa de junções protegida aumenta conforme se diminui a quantidade de vetores desprotegidos.

Tabela 39: Resultados de mascaramento lógico para melhor e pior estrutura para as composições de Gy

Composição	Vetores	Estrutura		Diferença
		Melhor	Pior	
S1	176	90,8%	86,5%	4,4%
S2	128	94,2%	89,8%	4,4%
S3	92	95,7%	92,9%	2,8%
S4	63	97,7%	92,8%	4,9%
S5	27	99,5%	96,6%	2,9%
TMR	0	100%	100,0%	0,0%

C. Estudo de caso 3: granularidade da função Gz

O terceiro estudo de caso foi criado utilizando se a seguinte função original de 8 entradas com portas complexas:

$$Gz = \overline{\big(A + B + (C * D)\big) * (E + F + G) * H}$$

Baseando se na função *Gz*, foram computadas 914 funções aproximadas, das quais 136 era sub-aproximadas (*Fz*) e 778 eram sobre-aproximadas (*Hz*). Algumas

destas funções foram selecionadas (Tabela 40) para compor seis esquemas diferentes (Tabela 41).

Tabela 40: Características das funções selecionadas para *Gz*

Aproximação		Função	Hamming
Fy	*Fz1*	$\overline{H * (A + B + (C * D))}$	7
	Fz2	$\overline{\overline{H * (E + G + F)}}$	21
	Fz3	$\overline{H}$	37
Hy	*Hz1*	$\overline{\big(H * (A + B)\big) * (E + G + F)}$	7

Tabela 41: Composições de ATMR para o caso 3: *Gz*

Composição	Aumento de área	Módulo 1	Módulo 2	Módulo 3	Vetores	Transistores
S1	88%	*Gz*	*Fz3*	*Hz1*	44	30
S2	113%	*Gz*	*Fz3*	*Gz*	37	34
S3	138%	*Gz*	*Fz2*	*Hz1*	14	38
S4	150%	*Gz*	*Fz2*	*Gz*	21	40
S5	163%	*Gz*	*Fz1*	*Gz*	7	42
S6	175%	*Gz*	*Gz*	*Hz1*	7	44
TMR	200%	*Gz*	*Gz*	*Gz*	0	48

compara a razão de junções p-n protegidas de cada composição de circuito ATMR de *Gz*. Como podemos ver a diferença entre um TMR tradicional, com 200% de custo extra de área, e a composição *S1* com, 88% de aumento de área, é de 2,6% para o melhor caso e 4,5% para o pior. A média de diferença entre os melhores e piores casos é de 1,1%. Também podemos comparar o numero total de junções desprotegidas para melhores e piores casos. (Figura 69). Existem casos onde composições menores possuem um menor numero de junções desprotegidas que uma composição maior, como por exemplo, *S1* (402 junções) versus S1 (450 junções desprotegidas) e *S3* (135 junções) contra *S4* (154 junções). Por fim a Tabela 42 relaciona a quantidade de vetores desprotegidos com o mascaramento da melhor e da pior estrutura e a diferença entre os casos. Pode-se ver que a taxa de junções protegida aumenta conforme se diminui a quantidade de vetores desprotegidos.

Figura 68: Taxa de junções p-n protegidas para cada composição de *Gz*

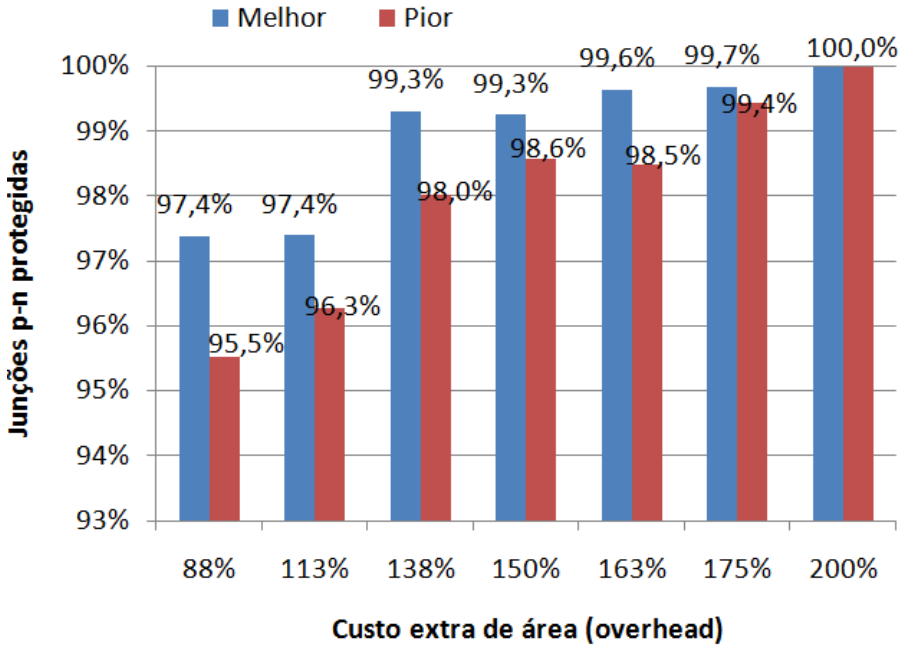
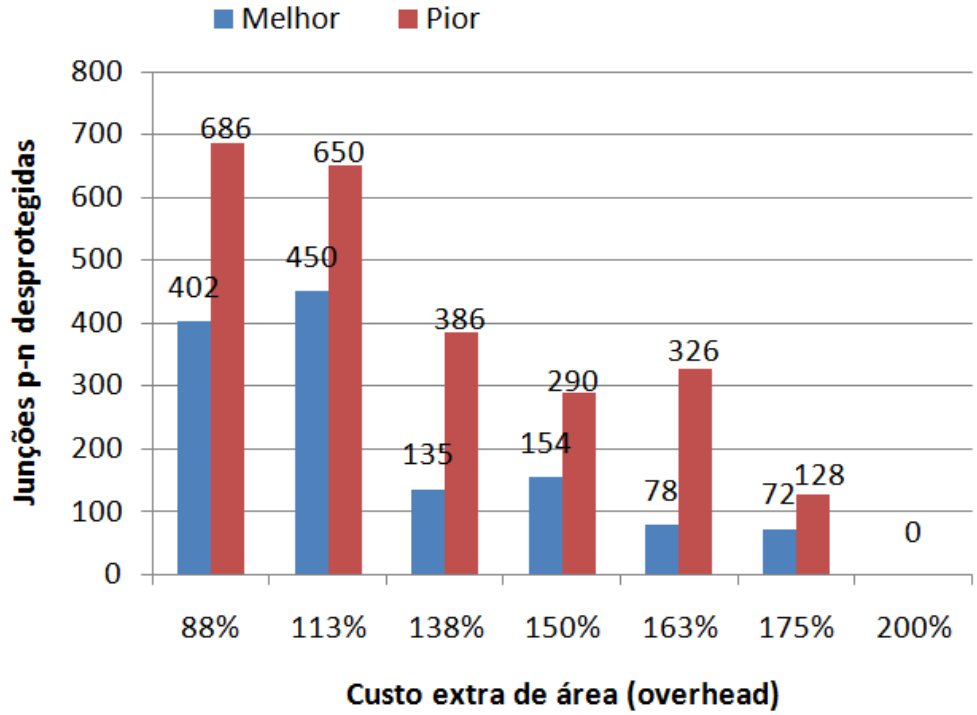


Figura 69: Valor absoluto de junções p-n desprotegidas para cada composição de *Gz*



Por fim a Tabela 42 relaciona a quantidade de vetores desprotegidos com o mascaramento da melhor e da pior estrutura e a diferença entre os casos. Pode-se ver que a taxa de junções protegida aumenta conforme se diminui a quantidade de vetores desprotegidos.

Tabela 42: Resultados de mascaramento lógico para melhor e pior estrutura para as composições de Gz

Composição	Vetores	Estrutura		Diferença
		Melhor	Pior	
S1	44	97,4%	95,5%	1,9%
S2	37	97,4%	96,3%	1,1%
S3	14	99,3%	98,0%	1,3%
S4	21	99,3%	98,6%	0,7%
S5	7	99,6%	98,5%	1,2%
S6	7	99,7%	99,4%	0,3%
TMR	0	100,0%	100,0%	0,0%

D. Estudo de caso 4: granularidade da função Gw

O quarto estudo de caso foi criado utilizando a seguinte função original de 6 entradas utilizando portas lógicas em multi-nível:

$$Gw = \overline{\left(\overline{(A+B)} + \overline{(C+D)}\right) * \overline{(E+F)}}$$

Baseando se na função *Gw*, foram computadas 106 funções aproximadas, das quais 23 era sub-aproximadas (*Fw*) e 86 eram sobre-aproximadas (*Hw*). Algumas destas funções foram selecionadas (Tabela 43) para compor seis esquemas diferentes (Tabela 44)

Tabela 43: Características das funções selecionadas para *Gw*

Aproximação		Função	Hamming
Fw	*Fw1*	$\overline{\overline{(C+D)} * \overline{(E+F)}}$	1
	Fw2	$\overline{\overline{(C+D)} * \overline{(E)}}$	5
Hw	*Hw1*	$\overline{\left(\overline{(B)} + \overline{(C+D)}\right) * \overline{(E)}}$	1
	Hw2	$(E+F)$	3
	Hw3	E	9

Tabela 44: Composições de ATMR para o caso 4: *Gw*

Composição	Aumento de área	Módulo 1	Módulo 2	Módulo 3	Vetores	Transistores
S1	122%	*Gw*	*Gw*	*Hw3*	19	40
S2	133%	*Gw*	*Gw*	*Hw2*	3	42
S3	144%	*Gw*	*Fw1*	*Hw1*	16	44
S4	156%	*Gw*	*Fw2*	*Gw*	5	46
S5	167%	*Gw*	*Fw1*	*Gw*	1	48
S6	178%	*Gw*	*Gw*	*Hw1*	15	50
TMR	200%	*Gw*	*Gw*	*Gw*	0	54

A Figura 70 compara a razão de junções p-n protegidas de cada composição de circuito ATMR de *Gz*. Como podemos ver a diferença entre um TMR tradicional, com 200% de custo extra de área, e a composição *S1* com, 122% de aumento de área, é de 2,01% para o melhor caso e 4,96% para o pior. A média de diferença entre os melhores e piores casos é de 1,37%. Também podemos comparar o numero total de junções desprotegidas para melhores e piores casos. (Figura 71). Existem casos onde composições menores possuem um menor numero de junções desprotegidas que uma composição maior, como por exemplo, *S2* (34 junções) versus S2 (75 junções desprotegidas) e *S5* (12 junções) contra *S6* (68 junções).

Figura 70: Taxa de junções p-n protegidas para cada composição de *Gw*

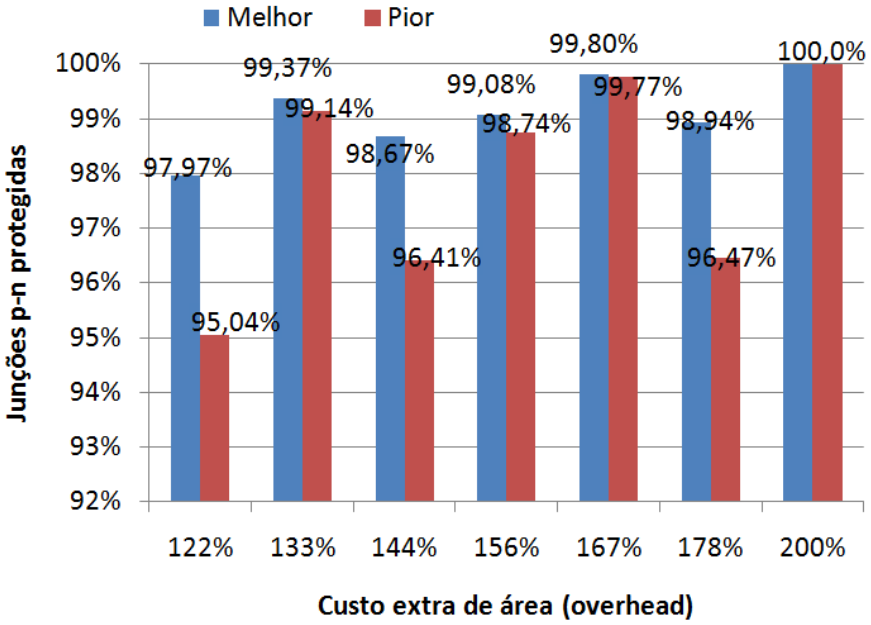
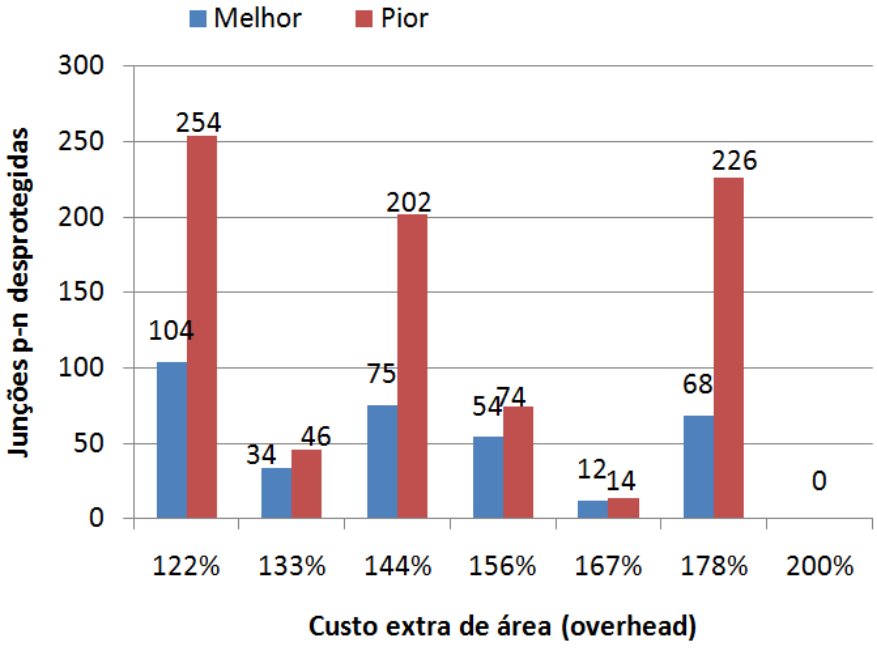


Figura 71: Valor absoluto de junções p-n desprotegidas para cada composição de *Gw*



Por fim a Tabela 45 relaciona a quantidade de vetores desprotegidos com o mascaramento da melhor e da pior estrutura e a diferença entre os casos. Pode-se ver que a taxa de junções protegida aumenta conforme se diminui a quantidade de vetores desprotegidos. Neste caso de circuito multi-nível é possível relacionar a quantidade de vetores desprotegidos com a diferença entre a melhor e a pior estrutura.

Tabela 45: Resultados de mascaramento lógico para melhor e pior estrutura para as composições de Gw

Composição	Vetores	Estrutura		Diferença
		Melhor	Pior	
S1	19	97,97%	95,04%	2,93%
S2	3	99,37%	99,14%	0,22%
S3	16	98,67%	96,41%	2,25%
S4	5	99,08%	98,74%	0,34%
S5	1	99,80%	99,77%	0,03%
S6	15	98,94%	96,47%	2,47%
TMR	0	100,0%	100,0%	0,00%

6. CONCLUSÕES

Com a miniaturização da tecnologia nas dimensões nanométricas as falhas causadas por SET se tornam mais relevantes, desta forma torna se cada vez mais importante a eficiência das técnicas de tolerância à falhas. A utilização de um esquema TMR juntamente com abordagens de circuitos aproximados pode permitir um bom equilíbrio entre aumento de área e mascaramento de falhas.

Este trabalho consiste de um estudo acerca da utilização da técnica ATMR para uma maior tolerância de SET em circuitos digitais. Inicialmente foi realizada uma revisão teórica sobre as origens da radiação espacial e os seus efeitos em circuitos integrados. Posteriormente, o estado da arte no que diz respeito às técnicas de detecção e mitigação de falhas baseadas em redundância aplicáveis a sistemas sujeitos à radiação foi apresentado, e a metodologia de desenvolvimento de uma composição ATMR foi definida.

A injeção de falhas nos circuitos ATMR demonstrou que a estrutura dos circuitos lógicos que compõem a abordagem afeta diretamente o poder de mascaramento lógico da arquitetura. As técnicas de reordenamento demonstraram ser uma boa alternativa para aprimorar a abordagem ATMR, em alguns casos a diferença entre a melhor estrutura e a pior chegou 5,21% de no âmbito das junções p-n protegidas.

Quando comparado com circuitos TMR tradicionais, diversas composições ATMR revelaram ótimos custos benefícios. Dos 32 circuitos analisados, os melhores casos, 30 esquemas ficaram acima de 95% de proteção de suas junções p-n, com uma variação de aumento de área entre 83% até 183%. Além disso, 9 destes alcançaram pelo menos 99% de junções protegidas, variando o custo extra de área entre 133% e 183%.

A análise da quantia total de junções p-n desprotegidas demonstraram que para alguns casos uma composição menor é melhor que uma composição que ocupa mais área, justificando assim a busca pelas melhores combinações de estrutura e composições da abordagem ATMR.

Por fim, a utilização dos vetores desprotegidos como métrica se mostrou uma boa alternativa. Além do impacto obvio que um vetor desprotegido estatisticamente gera, ele também dificulta os ganhos através das técnicas de reordenamento já que a otimização estrutural do circuito envolverá os estados de chaveamento dos transistores durante o vetor extra.

Os trabalhos relacionados a esta dissertação geraram duas publicações em anais de eventos (Gomes and Kastensmidt 2013)(Gomes, et al. 2014).

Alguns pontos devem ser abordados para trabalhos futuros. Primeiramente as ferramentas desenvolvidas, tanto para computação quanto para injeção de falhas, podem ser melhoradas e integradas em uma só ferramenta. Uma segunda situação é o desenvolvimento de outra ferramenta, esta com objetivo de, em primeiro estagio de desenvolvimento, aplicar as técnicas de reordenamento estrutural nos circuitos ATMR, em estágios finais aprimorar a ferramenta para que a mesma utilize de alguma técnica computacional para obter a melhor estrutura possível dado uma composição ATMR. Por fim, mas não menos importante, a formulação de uma função de custo pela qual seja possível não só separar as funções aproximadas explicitamente ruins do conjunto que apresenta custo-benefício entre si, mas que também consiga avaliar de forma eficaz qual melhor a função entre as candidatas dada uma restrição ou de área ou de mascaramento lógico.

BIBLIOGRAFIA

A. Anghel; D. Alexandrescu; M. Nicolaidis. Evaluation of the soft error tolerance technique based on time or hardware redundancy. **IEEE Integrated Circuits and Systems**, **Proceedings**… [S.l.:s.n.], p. 237-242, 2000.

BAUMANN, Robert C. Radiation-induced soft errors in advanced semiconductor technologies. **IEEE Transactions on Device and Materials Reliability**, [S.l.:s.n.], v.5, n.3, p. 305-316, 2005.

CADOLI, Marco et al. k-Approximating circuits. **IEEE Transacations on Computers**, [S.l.:s.n.], v.55 , n.7, p. 913-917, 2006, [.

CAZEAUX, J.M et al. Transistor level gate sizing for increased robustness to transient fault. In: INTERNATIONAL ON-LINE TESTING SYMPOSIUM, 2005. **Proceedings**…[S.l.:s.n.] , p. 23-28, 2005.

DODD, Paul E., MASSENGIL, Lloyd W. Basic mechanism and modeling od single-event upset in digital microelectronics. **IEEE Transactions on Nuclear Science**, [S.l.:s.n.], v.50, n.3, p. 583-602, 2003.

DODD, Paul. E. Impact of technology trends on SEU in CMOS SRAMs. **IEEE Transactions on Nuclear Science**, [S.2797:s.2804.], v.43, n.6, 1996.

GADLAGE, Matthew J. Single event transiente pulsewidths in digital microcircuits. **IEEE Transactions on Nuclear Science**, [S.l.:s.n.], v.51, n.6, p. 3285-3290, 2004.

GOMES, Iuri A.C. et al. Methodology for achieving best trade-off of area and fault masking coverage in ATMR. In: LATIN AMERICAN TEST WORKSHOP, 2014. **Proceedings**…[S.l.:s.n.], p.1-6, 2014.

GOMES, Iuri Albandes Cunha; KASTENSMIDT, Fernanda Gusmão de Lima. Reducing TMR overhead by combining approximate circuit, transistor topology and inputpermutation approaches. In: SYMPOSIUM ON INTEGRAD CIRCUITS AND SYSTEMS DESIGN, 26th, 2013. **Proceedings**…[S.l.:s.n.], p. 1-6, 2013.

International Solid-State Circuits Conference. **ISSCC Trends 2013.**Disponivel em: http//isscc.org/doc/2013/2013_Trends.pdf Acesso em: 28/01/2014.

KASTENSMIDT, Fernanda Gusmão de Lima; CARRO, Luigi; REIS, Ricardo. **Fault-tolerance techniques for SRAM-based FPGAs**. [s.l.]: Springer, 2006.

MARTINS, Mayler G.A. et al. **Boolean Factoring with multi-objetive goals.** [S.l.: s.n.], 2010.

MARTINS, Mayler Gama Alvarenga. **Functional composition and applications.** 13th International Symposium on Quality Electronic Design (ISQED), 2012. **Proceedings…** [S.l.:s.n.], p. 235-242, 2013.

MARTINS, Mayler; RIBAS, Renato Perez; REIS, André I. Functional composition: a new paradigma for performing logic synthesis. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, 2012. **Proceedings…** [S.l.:s.n.], p. 229-234, 2012.

MATSUMOTO, K.; UEHARA, M.; MORI, H. Evaluations of resettable stateful NMR. In: WORKSHOPS ADVANCED INFORMATION NETWORKING AND APPLICATIONS, 2011. **Proceedings**…[S.l.:s.n.], p. 740-745, 2011

MICROSEMI. **Neutron-induced single event upset (SEU) FAQ**. Disponivel em: http://www.actel.com/documents/neutron_seu_faq.pdf. Acesso em: 01/2014.

MOHANRAM, Kartik; TOUBA, Nur A. Partial error masking to reduce soft error failure rate in logic circuits. In: INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 2003. **Proceedings**…[S.l.:s.n.], p. 433-440, 2003.

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

NETO, Egas Henes et al. Using bulk built-in current sensors, to detect soft errors. **IEEE Micro**, [S.l.:s.n.],2006, v.25, n.5. p. 11-18)

NEUMANN, John von. Probabillistic logics and syhthesis of reliable organisms form unreliable components. **Automata Studies,** [S.l.:s.n.], v.34, n.1, p. 43-98, 1956.

HIARI, O.; SADEH, W.; RAWASHDEH, O. Towards single-chip diversity TMR for Automotive applications**. IEEE International Conference on Electro/Information Technology. Proceedings…**[S.l.:s.n.], p. 1-6 2012.

CHEYNET, P. Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors. **IEEE Transactions on Nuclear Science**, [S.l.:s.n.], v.47, n.6, p.2231-2236, 2000.

PAOLO, Nenzi; VOGT, Holger. Ngspice users manual version 22. Disponivel em: http://ngspice.sourceforge.net/.2010 Acesso em: 01/2014