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

GISELL BORGES MOURA

Síntese Automática do Leiaute usando o ASTRAN

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Microeletrônica

Orientador: Prof. Dr. Ricardo Augusto da Luz Reis

Porto Alegre
2017

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Moura, Gisell Borges

Síntese Automática do Leiaute usando o ASTRAN/Gisell Borges Moura. – Porto Alegre: PGMICRO da UFRGS, 2017.

101 f.:il.

Orientador: Ricardo Augusto da Luz Reis

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

1. Geração automática. 2. Leiaute3. Netlist otimizada 4. Síntese física 5. Microeletrônica. I. Reis, Ricardo Augusto da Luz. II. Título.

AGRADECIMENTOS

Agradeço aos meus pais, principalmente minha mãe **Dalva** que me deu todo o apoio e suporte em todos os momentos, sempre desejando meu sucesso e felicidade. Sou grata aos meus amigos que sempre me fizeram sorrir nos momentos em que estava esgotada.

Agradeço ao meu orientador **Ricardo Reis** pela oportunidade que me foi dada. Agradeço muito ao **Adriel Ziesemer, Calebe Conceição, Gracieli Posser, Cícero Nunes e Alonso Schmidt e Filipe Peracchi Pisoni** por todas as suas contribuições e orientações técnicas. Aos meus colegas de laboratório que sempre me deram força e incentivaram a ir em frente em todos os momentos: **Cristina Meinhardt, Walter Calienes, Jéssica Hoepers Muller, Ygor Quadros De Aguiar, Geancarlo Abich, Alexandra Lackmann Zimpeck, Jucemar Monteiro, Carolina Metzler, Julia Casarin Puget e Luciana Mendes.**

“*A tarefa não é tanto ver aquilo que ninguém viu,*
mas pensar o que ninguém ainda pensou
sobre aquilo que todo mundo vê.”
— Arthur Schopenhauer

RESUMO

O trabalho usa a síntese do leiaute através do ASTRAN em circuitos que foram otimizados através da técnica de SCCG (*Static CMOS Complex Gates*) visando alcançar reduções em número de transistores. A metodologia apresentada permite a flexibilidade de utilizar células de quaisquer tamanho ou redes de transistores nos circuito otimizados. O trabalho compara estes circuitos otimizados pela método do ASTRAN e circuitos utilizando a metodologia *standard cell*. O fluxo de síntese é composto pelas etapas de otimização da *netlist*, verificação/extração e caracterização da células.

O trabalho adaptou as tecnologias de fabricação CMOS de 600*nm* e 180*nm* para a ferramenta ASTRAN a partir das informações dos *design kits* das bibliotecas *stantard cell* XC06 e XC018 da XFAB. A síntese do leiaute das células complexas geradas é realizada pela ferramenta ASTRAN.

Os experimentos foram realizados nas tecnologias de 180*nm* e 600*nm* para um conjunto de circuitos de *bechmarks* do ITC'99. As comparações foram realizadas entre a *netlist* otimizada e duas *netlists* geradas para cada biblioteca da XFAB. Uma *netlist* abrange todas as células da biblioteca e a outra tem uma restrição de células que são consideradas complexas (somadores, multiplexadores, XOR/XNOR, AOI e OAI). A *netlist* com restrições foi elaborada com a motivação de verificar se uma *netlist* com células complexas geradas exclusivamente para o circuito alvo se tornaria mais benéfico em termos de redução do número de transistores.

Os resultados para 180*nm* apresentaram reduções nos melhores casos em número de transistores com até 15%, em potência dinâmica com até 24% e em potência de *leakage* com até 22%. Os resultados para 600*nm* apresentaram reduções nos melhores casos em número de transistores com até 17%, em área com até 14%, em potência dinâmica com até 22%, em potência de *leakage* com até 29%. Os experimentos mostraram que é possível alcançar reduções em número de transistores ao combinar o uso do ASTRAN com a técnica de otimização pelo uso de SCCG.

Palavras-chave: Síntese automática do leiaute, *netlist* otimizada, CAD, SCCG, Microeletrônica.

ABSTRACT

The work uses the synthesis of the layout through ASTRAN in circuits that have been optimized through the SCCG technique (Static CMOS Complex Gates) in order to achieve reductions in the number of transistors. The presented methodology allows the flexibility of using cells of any size or transistor networks in the optimized circuits. The work compares these circuits optimized by the ASTRAN method and circuits using the standard cell methodology. The synthesis flow is composed by the netlist optimization, verification / extraction and cell characterization steps.

The work adapted 600nm and 180nm CMOS fabrication technologies for the ASTRAN tool from the design information of the XFAB standard cell XC06 and XC018 libraries. The synthesis of the complex cells generated is performed by the ASTRAN tool.

The experiments were performed on the 180nm and 600nm technologies for a set of ITC'99 bechmarks circuits. Comparisons were made between the optimized netlist and two netlists generated for each XFAB library. A netlist covers all cells in the library and the other netlist has a restriction of cells that are considered complex (adders, multiplexers, XOR / XNOR, AOI, and OAI). The netlist with restrictions was designed with the motivation to check if a netlist with complex cells generated exclusively for the target circuit would become more beneficial in terms of reducing the number of transistors.

The results for 180nm showed reductions in the best cases in the number of transistors with up to 15%, in dynamic power up to 24% and in leakage power with up to 22%. The results for 600nm showed reductions in the best cases in the number of transistors with up to 17%, in an area up to 14%, in dynamic power with up to 22%, in leakage power with up to 29%.

The experiments showed that it is possible to achieve reductions in the number of transistors by combining the use of ASTRAN with the optimization technique using SCCG.

Keywords: Automatic Layout Synthesis, optimized netlist, SCCG , CAD, Microelectronics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AOI	Porta Lógica complexa composta pelas portas AND, OR e inversor
CAD	*Computer-Aided Design*
CCS	*Composite Current Source*
CI	Circuito Integrado
CMOS	*Complementary Metal-Oxide Semi-conductor*
DRC	*Design Rule Checker*
ECSM	*Effective Current Source Model*
EDA	*Electronic design automation*
GDSII	*Graphic Data System*
GME	Grupo de Microeletrônica
GND	*Ground*
INV	Porta lógica que inverte o sinal de recebe em sua entrada
LVS	*Layout vs Schematic*
NAND	Células lógica que representa a função booleana (A+B)
NLDM	*Non-Linear Delay Models*
NMOS	Canal N de Semicondutor Metal-Óxido
NOR	Porta lógica que representa a função boobleana
SPICE	*Simulation Program with Integrated Circuit Emphasis*
VDD	Suprimento de energia positivo
TCL	*Tool Command Language*
SCCG	*Static CMOS Complex Gates*

SUMÁRIO

1 INTRODUÇÃO

Atualmente existem circuitos comerciais compostos por bilhões de transistores em um único circuito integrado. O desenvolvimento de novas ferramentas de EDA (*Electronic Design Automation*) tem crescido consideravelmente devido à demanda por melhorias em desempenho, qualidade e redução de consumo de energia em projetos de circuitos. O uso de ferramentas em áreas de computação se tornou essencial para o aumento da produtividade em função do aumento da complexidade dos dispositivos e a agilizar o lançamento produtos no mercado (ZIESEMER, 2014).

1.1 Motivação

Projetos de circuitos baseado em *standard cells* são amplamente utilizados na industria e na academia pela sua confiabilidade e previsibilidade. No entanto, a qualidade do projeto torna-se limitado pelo conjunto de funções lógicas das células disponíveis em bibliotecas tradicionais (POSSER et al., 2010). Um fluxo livre de biblioteca utiliza funções lógicas de qualquer tamanho tornando possível a aplicação de otimizações em projetos de circuitos visando reduções de consumo de energia ou área, por exemplo (REIS et al., 2011). O mapeamento otimizado resultante torna possível reduzir o número de transistores consideravelmente e assim contribuir para a redução de consumo de energia, atraso e área do circuito. A construção do leiaute físico das células a partir das funções lógicas definidas usando o fluxo livre de biblioteca pode ser realizado por geração manual ou geração automática através do uso de uma ferramenta de geração automática de leiaute.

A limitação do número de células impede a otimização de um circuito em número de transistores. A partir da Tabela 1.2 (DETJENS et al., 1987) podemos verificar o número de funções disponíveis para implementação usando portas CMOS considerando um número limitado de transistores em série NMOS e PMOS. Conforme observado na Tabela 1.1, se utilizarmos 4 transistores em série NMOS e 4 transistores em série PMOS resulta em 3503 funções lógicas possíveis. No entanto, implementar todas estas funções acabaria sendo inviável pois as ferramentas de síntese teriam problemas em processar e realizar o mapeamento diante do enorme volume de funções disponíveis. Por isso, ao utilizar uma ferramenta de geração automática de leiautes torna possível aumentar de forma não excessiva

o número de funções lógicas, adicionando apenas as funções necessárias geradas pela aplicação de uma técnica tipo de otimização.

Tabela 1.1 – Número de funções possíveis usando um número máximo de transistores PMOS e NMOS em série.

		Número de transistores PMOS em série				
		1	**2**	**3**	**4**	**5**
Número de transistores NMOS em série	**1**	1	2	3	4	5
	2	2	7	18	42	90
	3	3	18	87	396	1677
	4	4	42	396	3503	28435
	5	5	90	1677	28435	125803

Fonte: DETJENS et al (1987).

A síntese automática do leiaute proporciona flexibilidade para a construção do leiaute físico das células a partir de redes de transistores. O ASTRAN (*Automatic Synthesis of Transistor Networks*) (ASTRAN, 2014) é uma ferramenta de síntese para geração automática de leiaute de redes de transistores com suporte a células com diferentes redes e tamanhos de transistores (ZIESEMER, 2014). Neste trabalho, foi aplicado um método de otimização em circuitos visando reduzir o número de transistores usando a ferramenta ASTRAN para gerar o leiautes dessas células otimizadas através de um fluxo de síntese física.

1.2 Objetivos

Este trabalho tem como objetivo usar a síntese do leiaute através do ASTRAN aplicada em circuitos que foram submetidos a uma técnica de otimização que proporcione a redução do número de transistores. Células de quaisquer tamanho ou redes de transistores podem ser utilizadas no circuito otimizado através da aplicação da metodologia apresentada.

Ao final do fluxo de síntese, o circuito otimizado é elaborado por uma abordagem mista de células: as novas funções lógicas adicionadas na etapa de otimização em que o leiautes das mesmas foram geradas pelo ASTRAN e as outras funções lógicas existentes do circuito que utilizam as células da biblioteca standard cell alvo. O trabalho compara estes circuitos otimizados pela método do ASTRAN e circuitos utilizando a metodologia *standard cell*.

1.3 Organização da Dissertação

Este trabalho está organizado da seguinte maneira:

O Capítulo 2 trata sobre a geração automática do leiaute, apresentando a ferrramenta ASTRAN utilizada neste trabalho e diversas outras ferramentas desenvolvidas pelo grupo de pesquisa GME e de outros trabalhos. O Capítulo 3 apresenta a metodologia que torna viável a otimização em redes de transistores usando o ASTRAN, incluindo a adaptação do ASTRAN para as tecnologias de 180*nm* e 600*nm*. O Capítulo 4 apresenta os resultados dos experimentos da aplicação de SCCG nos circuitos otimizados em termos de número de transistores, área, *timing* e potência dinâmica e de *leakage* para um conjunto de *bechmarks* do ITC'99 nas tecnologias de 180*nm* e 600*nm*. O Capítulo 5 apresenta as conclusões e trabalhos futuros.

2 GERAÇÃO AUTOMÁTICA DO LEIAUTE

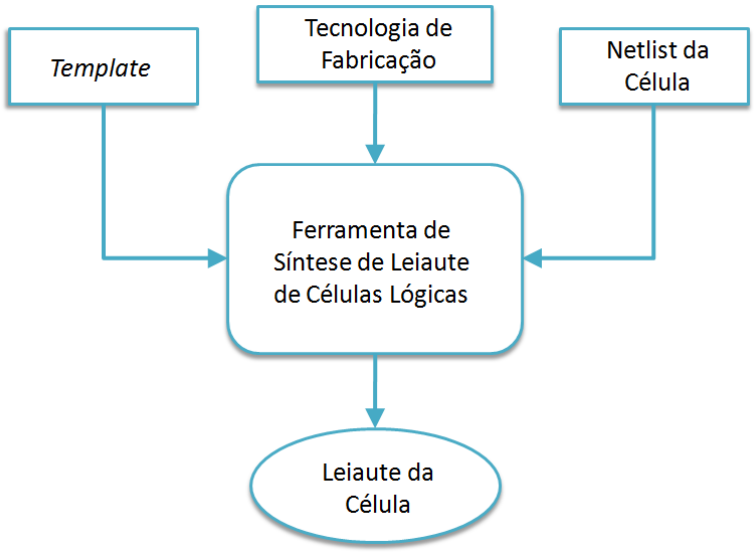
Neste capítulo são apresentadas ferramentas de síntese física desenvolvidas para a geração automática de leiutes, tanto as que foram desenvolvidas pelo grupo de pesquisa GME, quanto a de outros trabalhos. Este trabalho utilizou a ferramenta ASTRAN para a construção do leiaute físico a partir de células com diferentes redes e tamanhos de transistores.

2.1 Introdução

Diversas bibliotecas de células, comerciais ou acadêmicas, são criadas de forma manual por projetistas experientes. Entretanto esse processo consome um tempo de projeto considerável. Por isso, diversos trabalhos buscam a criação de ferramentas de EDA que possam automarizar a geraçãodos leiautes.

A síntese de células é a geração do leiaute a partir de um netlist que contém a descrição das redes de transistores de cada célula correspondente conforme a metodologia apresentanda na na Figura 2.1.

Figura 2.1 – Fluxo padrão para síntese de leiaute de células lógicas.



Fonte: Adaptado de (ZIESEMER et al., 2014).

A partir da Figura 2.1 podemos observar que a ferramenta de síntese de leiaute utiliza três entradas principais (ZIESEMER, 2014):

- A *netlist* com a descrição do circuito, listando todos os transistores com seus respectivos dimensionamentos, interconexões e terminais de entrada e saída.
- O *template* que consiste na descrição nas medidas que formam o leiaute físico que variam conforme a tecnologia utilizada. O arquivo abrange dados como altura ou largura das linhas de alimentação, por exemplo.
- A tecnologia de fabricação a ser usada, que abrange todas as regras de projeto da tecnologia alvo.

A saída do fluxo de síntese é o leiaute resultante da célula, onde os mesmos são analisados pelo projetista afim de verificar a qualidade dos leiautes. Normalmente pequenos ajustes são efetuados de forma manual, se necessário, para posteriormente as células serem incluídas na biblioteca.

2.2 Ferramentas de Geração do Leiaute Segundo a Metodologia TRANCA/UFRGS

O grupo de pesquisa GME (Grupo de Micro Eletrônica) vem realizando o desenvolvimento de diversas ferramentas para a síntese física de Circuitos Integrados desde 1981 (SUSIN, 1981; CARRO, 1989; LUBASZEWSKI, 1990; MORAES, 1990) segundo a metodologia TRANCA (*TRANsparent-Cell Approach*) (REIS,1987). Um resumo sobre as ferramentas desenvolvidas foi apresentado por (LAZZARI, 2003), (ZIESEMER, 2007) e (ZIESEMER, 2014).

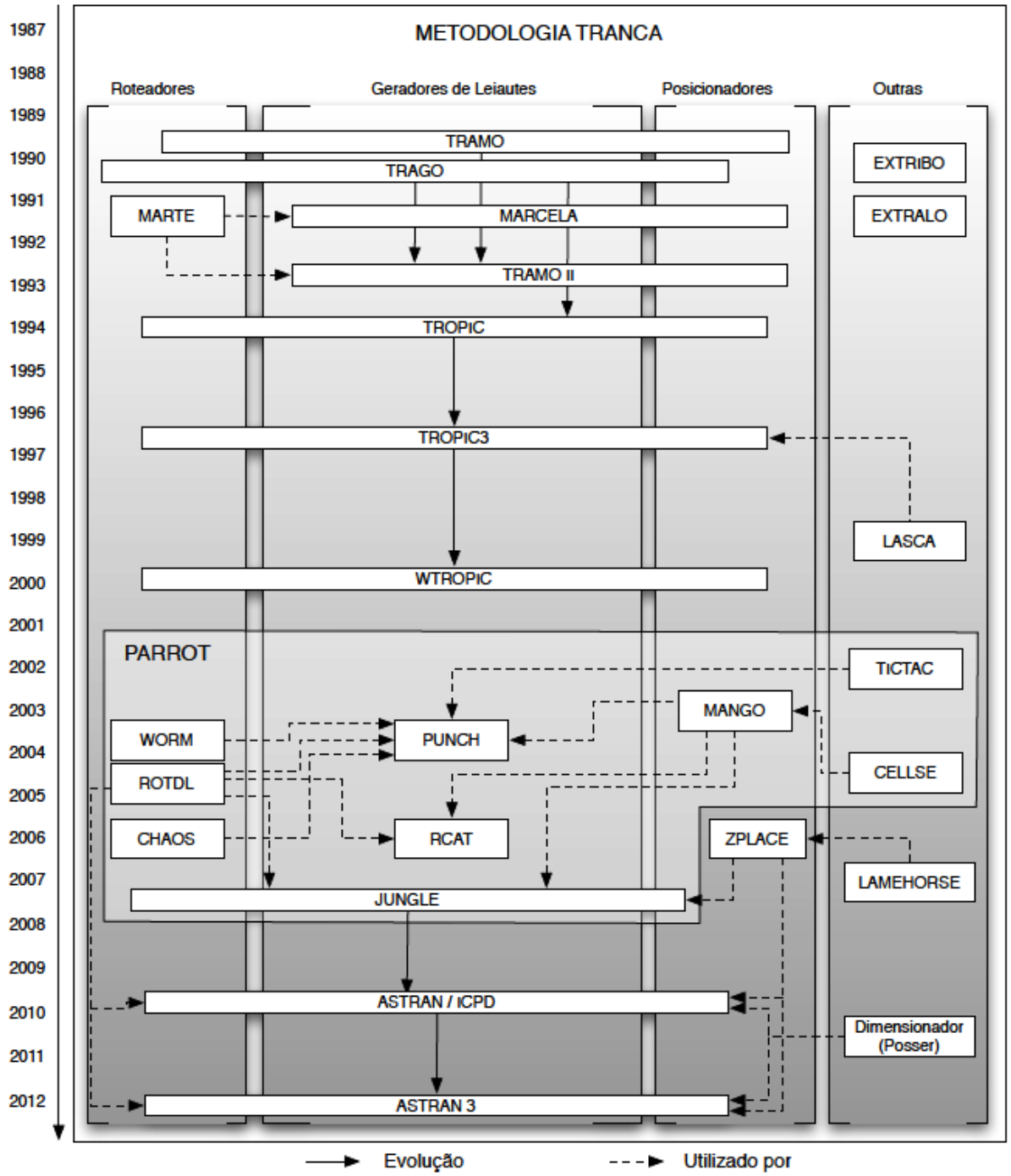
Uma linha do tempo sobre as ferramentas de síntese correspondentes (ZIESEMER, 2014) pode ser vista conforme a Figura 2.2. Porém, várias destas ferramentas já não suportam mais as regras de projeto exigidas atualmente pelas *foundrys* para fabricação de projetos de circuitos integrados.

A ferramenta TRAMO (TRANca Module) (REIS et al., 1988), (LUBASZEWSKI, 1990) realizava a geração de leiautes baseados em uma biblioteca de células usando somente uma camada de metal para as conexões. Já a ferramenta TRAGO (*Tranca Automatic GeneratOr*) (MORAES, 1990) fazia a geração automática do leiaute sem uma biblioteca de células.

O projeto MARCELA (GÜNTZEL; RIBAS; REIS, 1991), (GÜNTZEL, 1993), (GÜNTZEL et al., 1995) constrói uma matriz de uso genérico composta por células básicas distribuídas dentro de unidades básicas repedidas ao longo da matriz para realizar a geração

do leiaute. Para cada unidade básica existe a delimitação de duas linhas de GND, sendo cruzada por uma linha de VCC no centro.

Figura 2.2 – Linha do tempo das ferramentas de síntese física segundo a metodologia TRANCA.



Fonte: ZIESEMER et al (2014).

Baseando-se na abordagem de geração de leiaute do MARCELA temos MARTE (JOHANN; REIS, 1993) que realiza o roteamento das células. Em TRAMOII (REIS et al.,

1991), (REIS, 1993) as células dos circuitos gerados eram transparentes em relação as conexões de metal 1 e metal 2.

A ferramenta TROPIC (*Transparent Reconfigurable Optimized Parametrizable Integrated Circuit generator*) (MORAES, 1994) realiza a geração dos leiautes baseada na *netlist* de transistores e nas regras de projeto usando somente duas camadas de metal. No TROPIC3 (MORAESet al., 1997), o roteamento utiliza três camadas de metal. Para fazer a extração elétrica dos circuitos gerados pelo TROPIC foi utilizado a ferramenta chamada LASCA (FERREIRAet al., 2000). O WTROPIC apresentada por (FRAGOSO, 2001) é uma ferramenta de geração de leiautes para ser utilizada via Internet.

TICTAC foi o nome dado a ferramenta desenvolvida por (WILKE et al., 2002) e aos demais trabalhos desenvolvidos sobre o tema de verificação temporal e identificação do atraso crítico de circuitos combinacionais. MANGO PARROT foi apresentado em (HENTSCHKE, 2002) e é uma ferramenta que realiza um posicionamento relativo das células ao longo das bandas do circuito com o objetivo de reduzir o comprimento total das conexões.

PARROT é o fluxo de geração de leiaute criado com o uso destas ferramentas e diversas ferramentas foram desenvolvidas e integradas à este fluxo (ZIESEMER, 2007). A ferramenta PARROT PUNCH apresentada por (LAZZARI, 2003) tem a principal característica de gerar leiautes sob demanda, sem a necessidade de uma biblioteca de células. O leiaute é gerado considerando o circuito inteiro como uma célula gigante sem hierarquia (ZIESEMER, 2007). WORM é um roteador desenvolvido em conjunto com a ferramenta PUNCH para realizar as interconexões do circuito.

Um roteador mais robusto que WORM chamado ROTDL foi desenvolvido por (FLACH; HENTSCHKE; REIS, 2004). Um gerador de estimativas de tamanhos de células chamado CELLSE foi apresentado por (ZIESEMER, 2006), possibilitando estimar de forma mais automatizada e precisa as dimensões das células do que o estimador usado anteriormente chamado MANGO.

CHAOS (SANTOSet al., 2006) é um roteador implementado com algoritmos de baixa complexidade e surgiu como uma alternativa para roteamento convergente e ágil. A convergência é garantida pela inserção de espaços estre as bandas enquanto a garantia da agilidade é dada por um pré-planejamento global baseado em árvores de expansão e um roteamento detalhado baseado no *Greedy Channel Router*.

RCAT é um gerador de leiautes regulares baseado em matrizes de células desenvolvido por (MEINHARDT, 2006). A ferramenta tem como principal característica a previsibilidade das suas características elétricas e foi integrada ao fluxo PARROT.

Para a área de circuitos 3D, a ferramenta ZPLACE (HENTSCHKE, 2007) é um posicionador quadrático com refinamento local interativo utilizando o método de *Threshold Accepting* (DUECKet al., 1990) para redução de *wirelength* (comprimento das conexões) com observância do caminho crítico. Ele realiza também posicionamento de circuitos 2D e suporta circuitos com centenas de vezes mais células do que o MANGO PARROT. LAMEHORSE (SAWICKI et al., 2007) é uma ferramenta que realiza o particionamento e posicionamento de PADs e pinos de entrada e saída para circuitos 3D.

JUNGLE PARROT desenvolvido por (ZIESEMER, 2007) é um compilador de partes operativas e gerador automático de células lógicas. A ferramenta realizava o posicionamento dos módulos (somadores, multiplicadores, etc.) de forma regular seguindo um *template* previamente estabelecido e o roteamento era feito utilizando o ROTDL. Além disso, foi definido um formato de arquivo para a descrição estrutural dos circuitos de parte operativa.

O ASTRAN (ZIESEMER, 2014) é uma ferramenta de síntese automática de redes de transistores. O fluxo de síntese do leiaute suporta dimensionamento de redes de transistores, planejamento de planta baixa, posicionamento das interfaces dos circuitos, roteamento detalhado em metal 1, posicionamento incremental e interface gráfica com o usuário. A versão mais atual do ASTRAN suporta compactação em 2D e geração para tecnologias de 350*nm* até 45*nm* (FREEPDK45 DESIGN KIT, 2014).

2.3 Síntese Automática de Redes de Transistores ASTRAN

A ferramenta acadêmica ASTRAN (ZIESEMER, 2014) foi desenvolvida com a proposta de automatizar o projeto do leiaute de células lógicas. O ASTRAN é uma ferramenta de síntese automática de leiaute com suporte a células com quaisquer tamanhos ou redes de transistores, incluindo lógica não-complementar. Ele se baseia em um conjunto de regras de projeto que são comuns a diversos processos de fabricação. Ele tem suporte a nodos de tecnologia entre 350*nm* e 45*nm*.

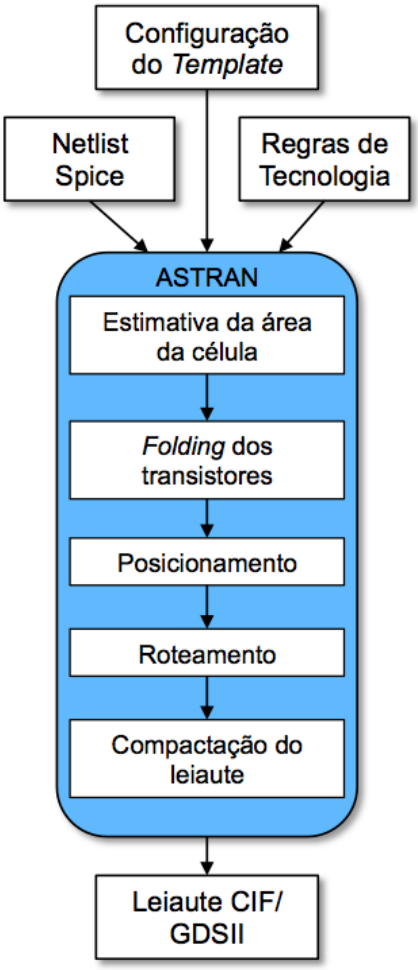
O ASTRAN utiliza uma metodologia para compactação de leiaute com programação linear mista com inteiros (MILP) simultânea em duas dimensões (2D) e realiza o *folding* automático dos transistores estabelecendo assim uma altura uniforme das células. O formato

do estilo de leiaute permite o uso das células geradas em um fluxo de síntese compatível ao fluxo *standard cell*.

O fluxo de síntese para a geração dos leiautes usa como parâmetros de entrada:

- Um *netlist* no formato SPICE;
- Um arquivo de tecnologia com descrição das regras de projeto;
- Um *template* de configuração que define a topologia do leiaute e os parâmetros de controle para o gerador.

Figura 2.3 – Fluxo de síntese física da ferramenta ASTRAN.



Fonte: ZIESEMER et al (2014).

O formato SPICE fornece a lista de todos os transistores com as respectivas informações sobre o comprimento e a largura de todos os *gates*, as interfaces de entrada e saída e as redes que os conectam. O *template* contém a descrições da forma física do leiaute das células como altura, largura das linhas de alimentação ou grade de roteamento, por exemplo. E o arquivo de tecnologia descreve todas as regras de projeto referentes à tecnologia

alvo (ZIESEMER, 2014 ; POSSER, 2011). A partir dos arquivos de entrada, o ASTRAN gera o leiaute de cada célula seguindo as regras de tecnologia especificada e modelo com os parâmetros de geração conforme o fluxo da Figura 2.3. No final do fluxo, o leiaute resultante pode ser exportado para o formato CIF ou GDSII.

O fluxo de síntese do ASTRAN inicia com a leitura dos arquivos de entrada, tendo em seguida a etapa de estimativa da área da célula e a etapa de *folding* dos transistores. A próxima etapa executada é de posicionamento dos transistores e o roteamento interno das células, finalizando com a etapa de compactação do leiaute. As funcionalidades mais detalhadas de cada etapa do fluxo são descritas abaixo (ZIESEMER, 2014):

- Estimativa da área da célula: nesta etapa realiza-se a estimativa da largura (W) máxima dos transistores para as regiões P e N assim como os recursos de roteamento que estarão disponíveis para compor uma representação abstrata da célula que possa ser compactada posteriormente.

- *Folding*: esta etapa consiste na quebra de transistores que excedem a altura máximo da difusão de acordo com as regras de projeto em transistores menores conectados em paralelo. Assim, a altura de cada célula é padronizada para toda a biblioteca. A metodologia utilizada pelo ASTRAN é de posicionamento dinâmico com *folding* estático, onde a partir dos limites dados para o *folding* realiza-se a quebra dos transistores e determina-se a sua posição e orientação de forma a minimizar a área da célula.

- Posicionamento: esta etapa tem por função encontrar um arranjo de transistores dentro da área da célula de forma que atenda a um determinado objetivo. O posicionador da ferramenta usa como heurística tanto o algoritmo de caminho de Euler que foca em encontrar de forma rápida e exata o posicionamento com menor número de quebras de difusões quanto o algoritmo *Threshold Accepting* (HENTSCHKE, 2007) que posiciona transistores com uma estrutura de rede mais complexa de forma.

- Roteamento: durante esta etapa é necessário converter o posicionamento fornecido em uma estrutura de dados que contenha uma representação das partes do circuito que precisam ser conectadas e os lugares onde possam passar as conexões

(conforme a estimativa da área da primeira etapa e o estilo de leiaute definido). O roteamento interno da célula do ASTRAN é realizado usando uma versão modificada do algoritmo de roteamento baseado em negociação de congestionamento chamado *PathFinder* (MCMURCHIE; EBELING, 1995).

- Compactação do leiaute: após o posicionamento e roteamento, o leiaute é compactado. Compactação é o processo de geração do leuite da célula a partir de uma especificação inicial de acordo com as regras de desenho especificadas (CHEN,2009). A recompactação é o termo utilizado para a transformação de um leiaute antigo em um novo com diferentes regras de projeto. A ferramenta utiliza programação linear mista com inteiros (MILP) para resolver o problema de compactação de leiaute, resultando um leiaute final de acordo com os resultados fornecidos pelas etapas de posicionamento e roteamento seguindo as regras de projeto da tecnologia usada.

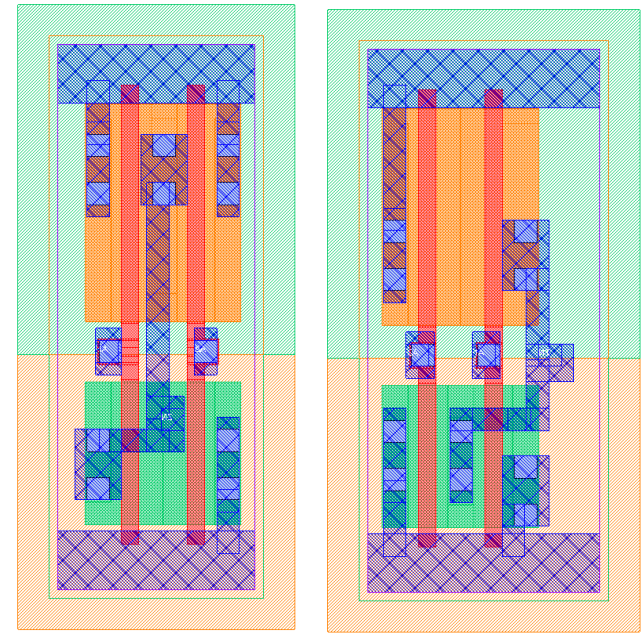
Um exemplo de duas células geradas pelo ASTRAN na tecnologia de 45*nm* pode ser visto na Figura 2.4 (a) que corresponde a uma célula XOR2 e Figura1.3 (b) que corresponde a uma célula NAND2.

Cada leiaute gerado pelo ASTRAN é construído considerando algumas características básicas referentes ao estilo de leiaute (ZIESEMER, 2014):

- Trilhas sobre as difusões P e N para roteamento em metal;
- Linhas de alimentação nas regiões superiores e inferiores da célula para conexão por justaposição com células vizinhas em metal 1;
- Roteamento interno da célula em polisilício, metal 1 e difusão;
- Contatos entre polisilício e metal 1 em qualquer região da célula, exceto transistores;
- Pinos de entrada e saída da célula alinhadas à grade de roteamento do circuito;
- Altura da célula e posição do poço pré-determinados pelo modelo;
- Transistores podem ser posicionados em qualquer lugar dentro das regiões PMOS e NMOS;
- Suporte sem restrições a quaisquer redes de transistores e tamanhos destes;
- Inserção automática de conexões em *L* para roteamento em metal 1 e poly; Uma conexão em *L* é uma conexão de metal ou de *poly* com muitas dobras.

- Posicionamento dos Contatos de Poço e substrato colocados em baixo das linhas de alimentação, conforme definido pelo modelo escolhido pelo projetista;
- Transistores posicionados em duas bandas distintas para transistores PMOS e NMOS.

Figura 2.4 – Leiaute das células NAND2 (a) e XOR2(b) geradas automaticamente pelo ASTRAN para a tecnologia de 45*nm*.



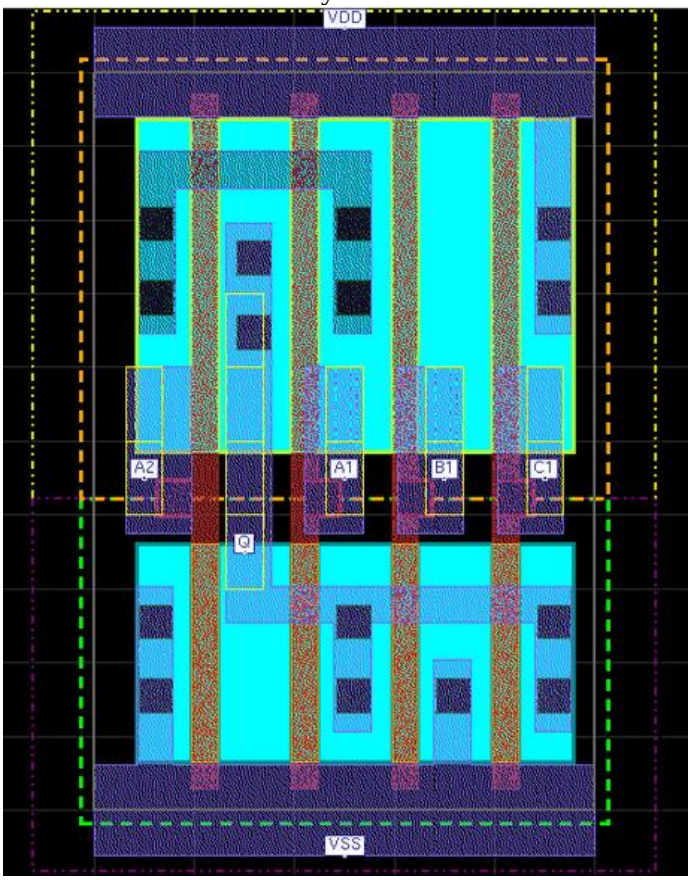
(a) (b)

Fonte: figura elaborada pelo autor.

2.4 Ferramentas relacionadas à geração de células

Em relação a outros trabalhos relacionados à geração automática de células, temos a ferramenta comercial da Nangate (NANGATE LIBRARY CREATOR, 2017) que é voltada para a geração automática de bibliotecas *standard cells*. Atualmente a Nangate acrescentou melhorias na ferramenta que abrangem suporte de nodes de 350*nm* até 22*nm* na geração automática da biblioteca de células, dimensionamento discreto de transistores para FinFET e outras tecnologias não planares, etapas integradas de verificações de DRC e LVS, extração e caracterização das células. Além do leiaute em formato GDSII, a Nangate Library Creator fornece como saída o arquivo *liberty*, o esquemático da célula, o arquivo LEF e a *netlist* em verilog ou SPICE. Um exemplo de um leiaute elaborado pela ferramenta para a células lógica AOI211_X1 na tecnologia de 45*nm* é apresentado na Figura 2.5.

Figura 2.5 – Leiaute da célula AOI211_X1para a tecnologia de 45*nm*a partir da ferramenta Nangate Library Creator.



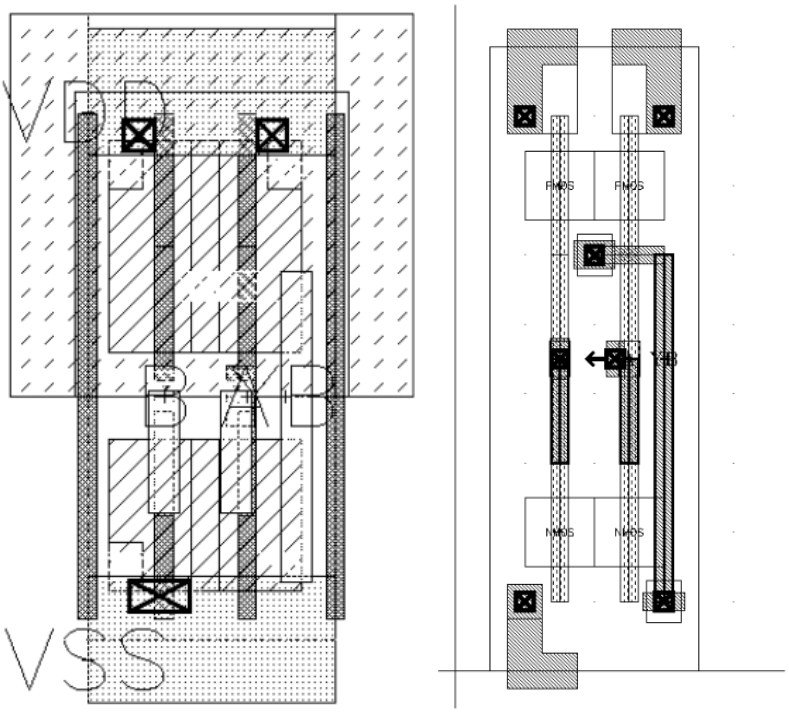
Fonte: (NANGATE LIBRARY CREATOR, 2017).

Outra ferramenta que também foca na geração automática de leiautes de bibliotecas *standard cells* desenvolvida por (NISHIZAWA, 2015) realiza uma abordagem diferente do ASTRAN utilizando como entrada um leiaute simbólico de grade virtual com trilhas fléxiveisem vez de utilizar uma *netlist* SPICE com a descrição do circuito. O Leiaute simbólico descreve a topologia do circuito de cada célula alvo, ou seja contém os dados sobre os componentes do circuito chamados de objetos simbólicos como transistores, contatos e fios de metal.

Nessa metodologia de projeto, bibliotecas de células dispõem de informações das estruturas de suas células como um arquivo de *template*. O *template* de cada célula fornece dados sobre a localização de transistores assim como os fios de sinais de interconexões e entrada/saída em uma grade virtual independente do processo de fabricação. A máscara do leiaute de cada célula é gerada a partir do gerador de leiaute usando as informações da estrutura da célula presente no *template* e os dados do processo de fabricação alvo para o

mapeamento tecnológico. Um exemplo de um leiaute gerado para a célula lógica NAND2 é apresentado na Figura 2.6 (a) e seu leiaute simbólico correspondente na Figura 2.6 (b).

Figura 2.6 – Leiaute da célula AOI211_X1 para a tecnologia de 45nm usando o Nangate Library Creator



(a) (b)

Fonte: (NISHIZAWA, 2015)

2.5 Conclusão

Neste capítulo foram apresentadas ferramentas de EDA, voltadas para a geração automática do leiaute de células lógicas para a tecnologia CMOS. Algumas ferramentas são voltadas para a geração de automática de bibliotecas *standard cell* NANGATE LIBRARY CREATOR e outras para um fluxo livre de biblioteca como o ASTRAN. A sessão 2.4 abordou poucas ferramentas de geração automática pois a maioria delas encontradas durante as revisões bibliográficas eram voltadas para circuitos analógicos. A ferramenta ASTRAN apresentada nesse capítulos que foi utilizada neste trabalho para a síntese do leiaute de células geradas a partir de uma técnica de otimização. Ela proporciona a flexibilidade para gerar funções lógicas de células de quaisquer tamanhos ou redes de transistores.

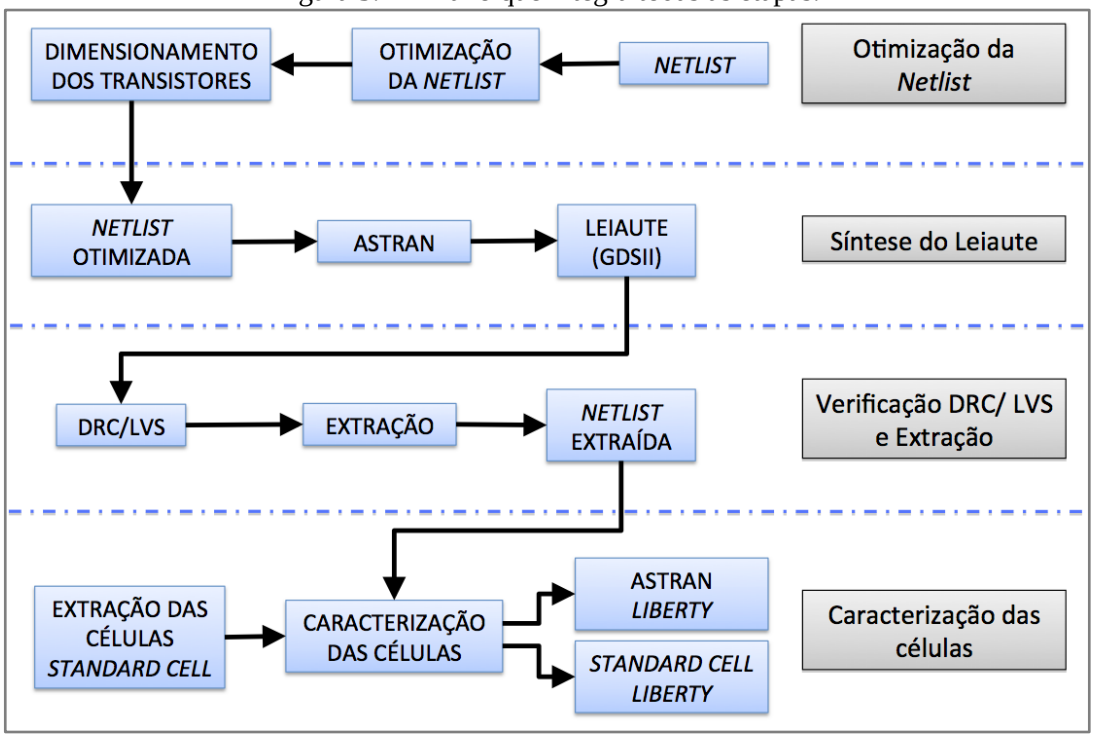
3 OTIMIZAÇAO DE REDES DE TRANSISTORES USANDO O ASTRAN

Neste Capítulo, é apresentada a metodologia que torna viável a otimização de *netlists* através do uso da técnica de SCCG (*Static CMOS Complex Gates*), usando a ferramenta ASTRAN para realizar a síntese do leiaute das células complexas geradas na etapa de otimização. O capítulo aborda as etapas de otimização da *netlist*, verificação/extração e caracterização das células assim como a adaptação do ASTRAN para as tecnologias de 180*nm* e 600*nm*.

3.1 Metodologia

A partir de um *netlist* no formato SPICE, o leiaute de células de quaisquer tamanhos ou redes de transistores podem ser geradas com a ferramenta ASTRAN. O trabalho utiliza um fluxo que torna viável a aplicação de técnicas de otimizações em *netlists* de circuito, onde a síntese é realizada pelo ASTRAN para as novas células geradas na etapa de otimização.

Figura 3.1 – Fluxo que integra todas as etapas.



Fonte: Adaptado de (POSSER, 2011).

Entretanto, realizar comparações entre circuitos gerados pelo ASTRAN e circuitos de uma biblioteca *standard cell* sem estabelecer a padronização de configurações de algumas etapas não seria justo, pois as configurações de extração e caracterização das células não

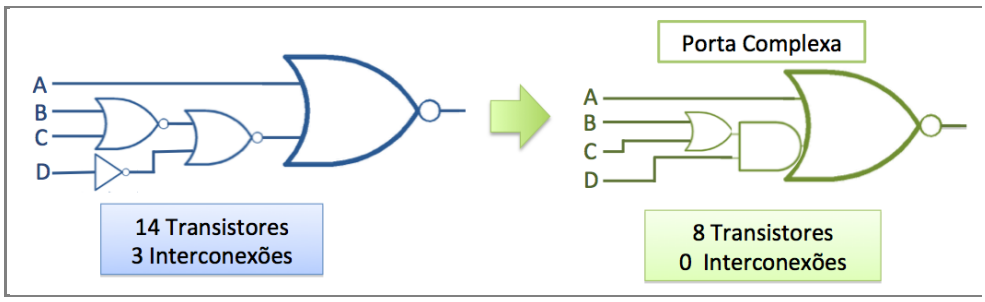
seriam as mesmas. O fluxo apresentando na Figura 3.1 propõe ajustar essas diferenças a partir das etapas de extração e caracterização. As etapas que compõem o fluxo são: otimização da netlist, síntese do leiaute, verificação/extração e caracterização das células.

3.1.1 Otimização da *Netlist*

A etapa inicial é composta de duas partes principais que são realizadas pela mesma ferramenta : a otimização da *netlist* que consiste na aplicação de uma técnica de otimização em uma *netlist* de circuito alvo e o dimensionamento dos transistores nos circuitos otimizados resultantes.

A técnica de otimização aplicada neste trabalho foi pelo uso de SCCG (*Static CMOS Complex Gates*) usando a ferramenta desenvolvida por (CONCEIÇÃO et al., 2016). Uma estratégia para a redução do número de transistores em um circuito é o uso de SCCG (*Static CMOS Complex Gates*), onde as funções lógicas de diversas entradas podem ser implementadas usando uma única porta ao invés de várias (CONCEIÇÃO et al., 2016 ; POSSANI et al., 2013; REIS, 2011). É nomeado de SCCG as portas que implementam uma função lógica que tenha mais de dois níveis de lógica como as portas AOI e OAI, por exemplo. Na Figura 3.2, um circuito com um determinado conjunto de portas totalizando 14 transistores e 3 interconexões tem uma representação ao lado por uma porta complexa de equivalente lógica composta por 8 transistores e 0 interconexões. Além da redução do número de transistores, a SCCG também teve redução do número de conexões.

Figura 3.2 – Aplicando o uso de portas complexas.

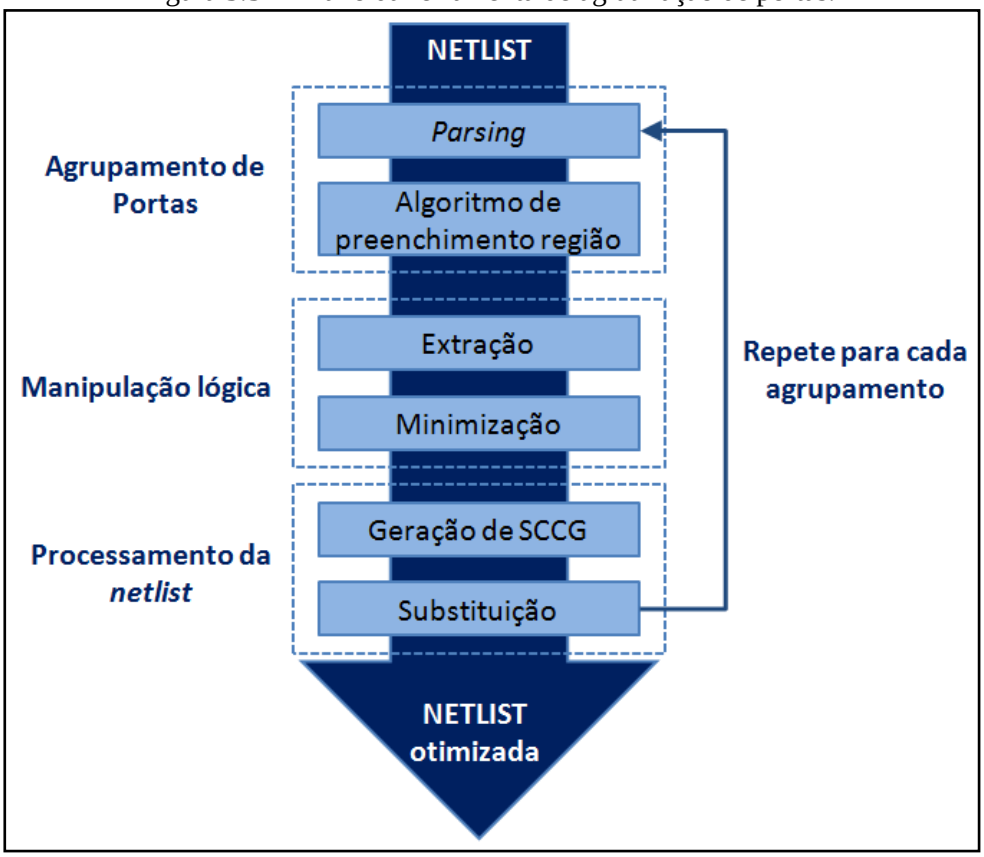


Fonte: Adaptado de (REIS, 2011).

O funcionamento da ferramenta consiste na aglutinação de portas lógicas com *fanout* 1 a partir de uma *netlit* gerada por uma ferramenta de síntese lógica. Assim, um conjunto de portas lógicas é subistituido por portas complexas que mantém a mesma função lógica. O fluxo da ferramenta é dividido em três etapas principais conforme a Figura 3.3: agrupamento de portas, manipulação lógica e processamento da *netlist*. A etapa inicial chamada de

agrupamento de portas se divide em duas partes. A primeira identifica os agrupamentos de portas conectadas de *fanout* 1 com uma auxílio de um *parser* para *netlist* no formato verilog que marca todas as portas combinacionais de apenas uma saída cujo *fanout* seja igual a 1.

Figura 3.3 – Fluxo da ferramenta de aglutinação de portas.



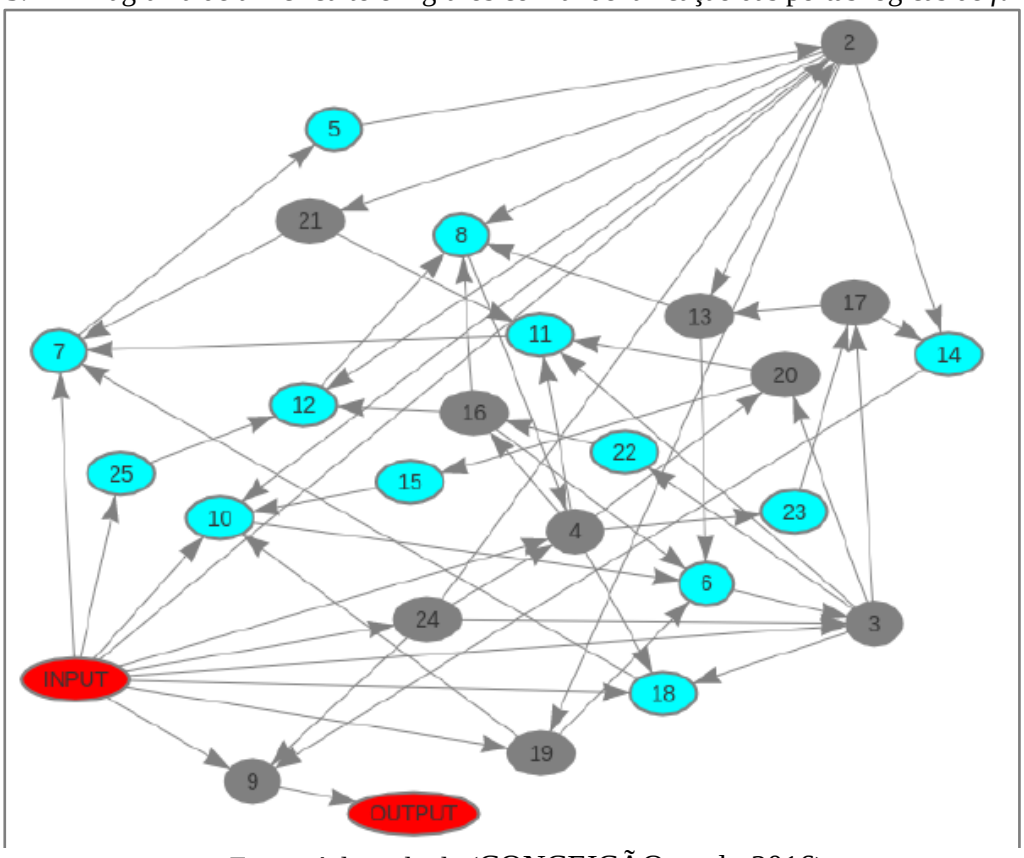
Fonte: Adaptado de (CONCEIÇÃO et al., 2016).

A segunda parte agrupa estas células que são conectadas pela aplicação de um algoritmo de preenchimento de região, onde as entradas e a saída do conjunto de portas conectadas são as entradas e saída do agrupamento que será subistituido posteriormente. Um exemplo sobre a identificação executada pelo *parser* para um dado circuito é apresentado na Figura 3.4 em formato de grafos, onde as portas de *fanout* 1 são marcadas em azul. A próxima etapa de manipulação lógica realiza a extração das expressões lógicas e a minimização das mesmas. A composição do agrupamento é realizada a partir da expressão de cada porta e entrada e assim a função lógica resultante pode ser minimizada utilizando a ferramenta Berkeley SIS (SENTOVICHet al., 2016).

A última etapa é o processamento da netlist em que a função lógica minimizada é usada para gerar a SCCG correspondente e realizar a subistituição pelo conjunto de portas equivalentes. No final do fluxo tem-se a geração de uma nova *netlist* otimizada com a

inclusão das portas complexas geradas no processo. Um exemplo de uma *netlist* original segue no Apêndice E.3 e a mesma *netlist* com a otimização aplicada pela ferramenta é apresentada no Apêndice E.4. Além da *netlist* otimizada, a ferramenta fornece os circuitos de cada porta complexa gerada no processo no formato SPICE e sua expressão lógica. No final do fluxo, a ferramenta realiza o dimensionamento dos transistores de cada célula gerada usando o método de esforço lógico.

Figura 3.4 – Diagrama de um circuito em grafos com a identificação das portas lógicas de *fanout* 1.



Fonte: Adaptado de (CONCEIÇÃO et al., 2016).

A partir do novo *netlist* que foi otimizado, as novas células complexas têm seus transistores dimensionados pelo método de esforço lógico com o auxílio da ferramenta desenvolvida por (CONCEIÇÃO et al., 2016). A ferramenta que aplica a técnica do uso de SCCG também realiza o dimensionamento dos transistores usando o método de esforço lógico no final do fluxo para as células geradas.

3.1.2 Síntese do Leiaute

Após o dimensionamento dos transistoresdas células complexas geradas na etapa anterior, os circuitos em formato SPICE das mesmas são encaminhados para a próxima etapa

de síntese do leiaute. Entretanto, é necessário realizar a aplicação do *folding* nos transistores destas células antes de executar a síntese do leiaute. O *folding* aplicado consiste em adicionar transistores em paralelo dividindo o valores em duas ou três vezes para a maioria dos casos.

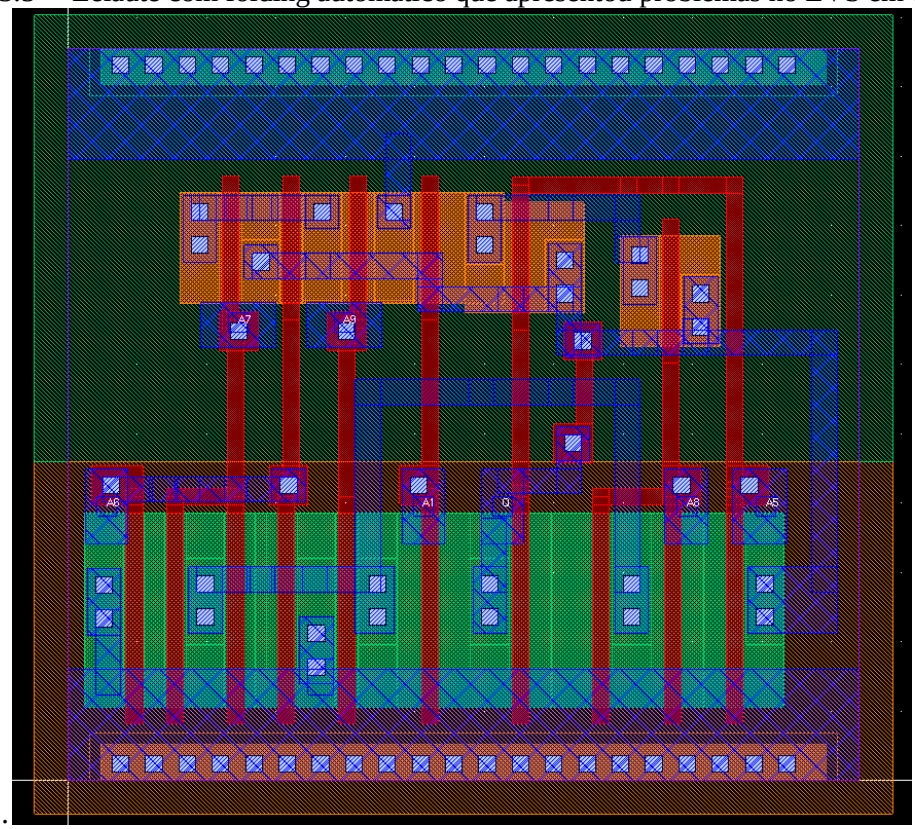
O ASTRAN tem a funcionalidade de *folding* automático dos transistores, entretanto dependendo da complexidade do circuito alguns casos apresentaram problemas pois a ferramenta não conseguiu finalizar a geração do leiaute por dificuldades em encontrar uma solução para o roteamento *intra-cell*. Outro problema surgiu na etapa de verificação LVS, pois os valores de W dos transistores presentes no leiaute e na descrição do circuito em SPICE se tornavam diferentes pela aplicação do *folding* automático. O relatório de erros do LVS fornecia as alterações a serem ajustadas no circuito SPICE. Entretanto, após realizar essas alterações a ferramenta continuava a reportar os mesmo erros. Ao realizar previamente a aplicação da quebra dos transistores nos circuitos das células, o LVS não apresentava mais os erros reportados.

Além disso, o posicionamento do *poly* de quebras automaticas de transistores também geravam reportes de erro na etapa de LVS. A ferramenta sempre reportava quando a quebra não era posicionada na sequencia lateral e sim sendo um pouco mais afastado usando uma trilha de metal 1 para conectar ambos. A frequência de ocorrência desse problema era proporcional ao número de entradas do circuito, ou seja, um maior o número de entradas incrementa ainda mais a complexidade do roteamanto *intra-cell*.

O leiaute da Figura 3.5 em 600*nm* tem aplicado *folding* automático e demonstra um exemplo sobre o problema de posicionamento do *poly* dessas quebras de transistores para a célula complexa de expressão lógica !((A1+(A5*A8))*(A9+(A7*A6))). Podemos observar que a quebra do transistor da entrada A6 é um pouco afastada usando a trilha de metal 1 para se conectar a outra parte.

O mesmo circuito com *folding* aplicado no circuito SPICE é apresentando na Figura 3.6, onde está marcado em laranja as linhas com as respectivas quebras dos transistores. O teste inicial realizou a quebra do valor de W em duas partes apenas no transistor mn_5 da entrada A6, mas a ferramenta não finalizou o leiaute. Por isso, todos os transistores que tinha o mesmo valor de W de 14U foram quebrados seguindo o mesmo padrão e assim o leiaute foi finalizado. O leiaute do SPICE da Figura 3.7 é apresentando na Figura 3.8, onde os erros de verificação de LVS não ocorreram novamente.

Figura 3.5 – Leiaute com folding automático que apresentou problemas no LVS em 600*nm*.



Fonte: figura elaborada pelo autor.

Figura 3.6 – Spice com folding manual aplicado em laranja na tecnologia de 600*nm*.

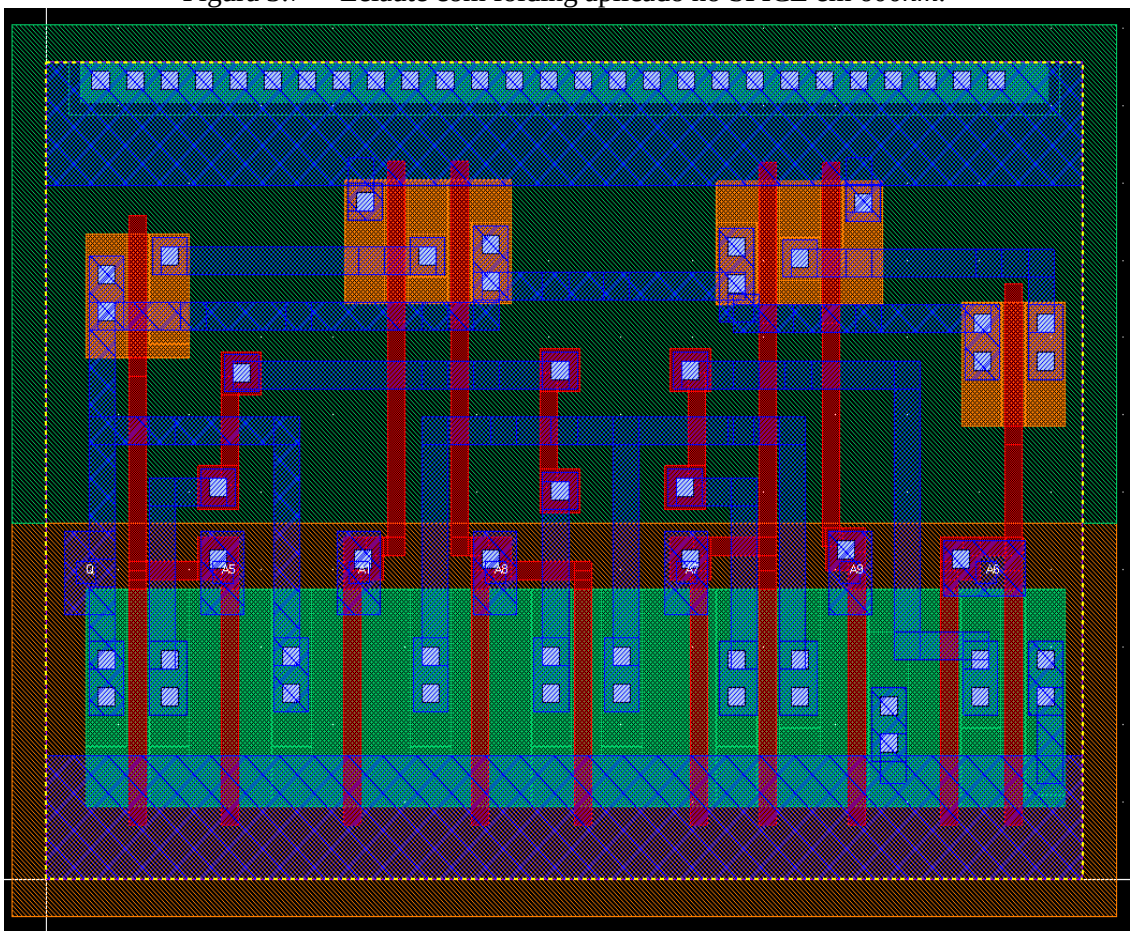
```
.subckt gate_2_v1 A1 A5 A6 A7 A8 A9 Q vdd! gnd!
mp_0 vdd! A1 n_12 vdd! pmos4 W=4U L=0.6U
mp_1 n_12 A5 Q vdd! pmos4 W=4U L=0.6U
mp_2 n_12 A8 Q vdd! pmos4 W=4U L=0.6U
mp_3 vdd! A9 n_21 vdd! pmos4 W=4U L=0.6U
mp_4 n_21 A7 Q vdd! pmos4 W=4U L=0.6U
mp_5 n_21 A6 Q vdd! pmos4 W=4U L=0.6U

mn_0 Q A1 n_42 gnd! nmos4 W=7U L=0.6U
mn_1 Q A5 n_36 gnd! nmos4 W=7U L=0.6U
mn_12 Q A5 n_36 gnd! nmos4 W=7U L=0.6U
mn_2 n_36 A8 n_42 gnd! nmos4 W=7U L=0.6U
mn_22 n_36 A8 n_42 gnd! nmos4 W=7U L=0.6U
mn_3 n_42 A9 gnd! gnd! nmos4 W=7U L=0.6U
mn_4 n_42 A7 n_45 gnd! nmos4 W=7U L=0.6U
mn_42 n_42 A7 n_45 gnd! nmos4 W=7U L=0.6U
mn_5 n_45 A6 gnd! gnd! nmos4 W=7U L=0.6U
mn_52 n_45 A6 gnd! gnd! nmos4 W=7U L=0.6U

.ends
```

Fonte: figura elaborada pelo autor.

Figura 3.7 – Leiaute com folding aplicado no SPICE em 600*nm*.

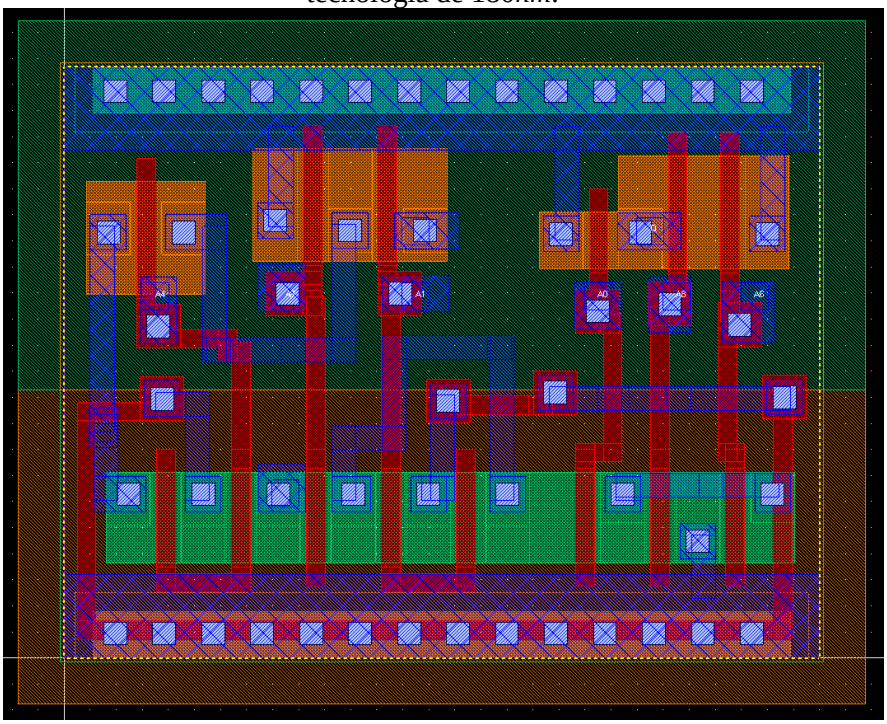


Fonte: figura elaborada pelo autor.

Após os ajustes de *folding* dos transistores, a síntese do leiaute das células complexas da nova *netlist* é realizada através da ferramenta ASTRAN e importação dos mesmos para o formato GDSII. Durante a etapa de síntese do leiaute a ferramenta realiza o roteamento *intra-cell*usando a camada de metal 1. No entanto, se a ferramenta não consegue completar todo o roteamento com metal 1, são deixadas indicações como contatos sem conexões e nomes de redes distribuídos na área do leiaute a serem ajustados com a camada de metal 2 durante a etapa de DRC e assim finalizar a solução de roteamento encontrada.

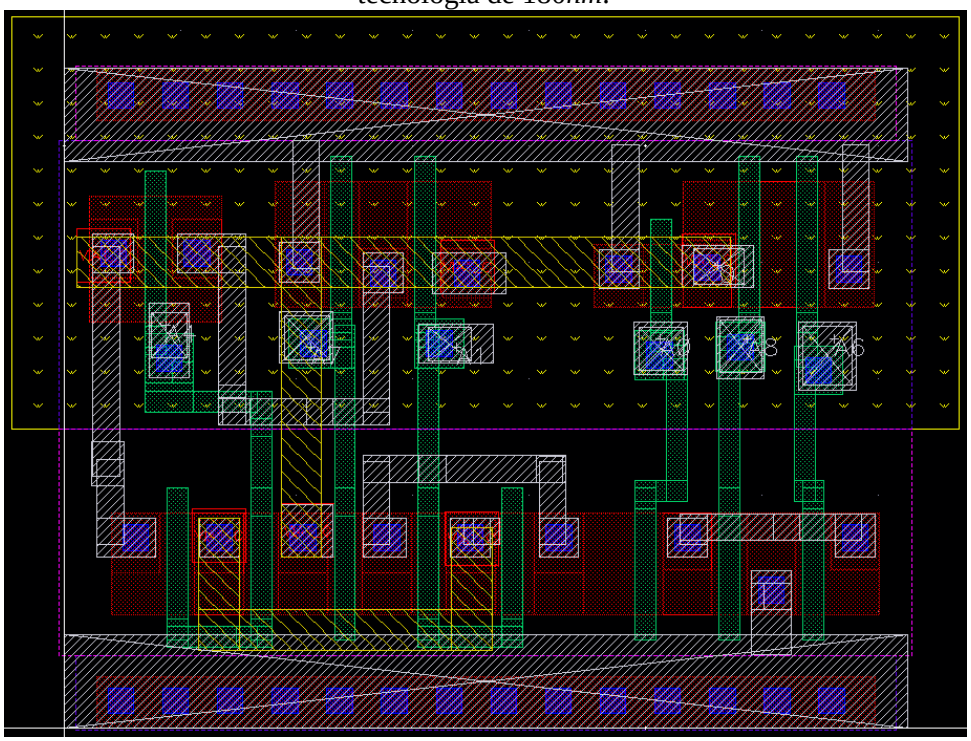
Os ajustes com a camada de metal 2 ocorreram com muita frequência na tecnologia de 180*nm.* Para a tecnologia de 600*nm* houveram poucos casos pois a diferença de altura da célula em relação à 180*nm* proporcionou um espaço maior para o roteamento *intra-cell.* Um exemplo de leiaute que necessitava de ajustes com metal 2 é mostrado na Figura 3.8 e o mesmo leiaute após os ajustes da camada de metal 2 é apresentado na Figura 3.9.

Figura 3.8 – Leiaute de uma célula complexa do circuito B04 sem o acréscimo de metal 2 na tecnologia de 180*nm*.



Fonte: figura elaborada pelo autor.

Figura 3.9 – Leiaute de uma célula complexa do circuito B04 com osajutes de metal 2 aplicado na tecnologia de 180*nm*.



Fonte: figura elaborada pelo autor.

Os leiautes em formato GDSII são importados para a ferramenta Virtuoso Layout Suite (CADENCE, 2014) para realizar a execução das etapas de verificação e extração. Para auxiliar a importação dos leiautes, o arquivo de *layermap* é usando como parâmetro de entrada, pois o mesmo informa a lista de todas as camadas usadas para uma tecnologia alvo e sua numeração correspondente associada. O ASTRAN gera um arquivo de *layermap* após finalizar cada leiaute. Para este trabalho não foi necessário utilizá-lo, pois os arquivos de *layermap* do *design kit* das tecnologia de 180*nm* e 600*nm* da XFAB são carregados automáticamente ao iniciar a ferramenta. O arquivo de *layermap* utilizado neste trabalho é apresentado no Apêndice C.

3.1.3 Verificação e Extração

As etapas de verificação de DRC e LVS são executadas apenas para os leiautes gerados pelo ASTRAN, pois os leiautes das células de cada biblioteca da XFAB não necessitam dessas verificações. É nessa etapa que são ajustadas, quando houver, correções para que o leiaute esteja conforme os padrões das regras de projeto da tecnologia utilizada. A etapa de extração das capacitâncias parasitas é executada tanto para os leiautes gerados pelo ASTRAN quanto para os leiautes pertencentes às bibliotecas *standard cell* XC06 e XC018 da XFAB. O mesmo tipo de configuração de extração, RC (Resistor e Capacitor), é aplicada para ambos os fluxos. Em seguida, a etapa de caracterização utiliza as *netlists* extraídas de cada célula como entrada em conjunto com os outros arquivos de configuração para gerar o arquivo *liberty* que contém as informações dos valores estimados de potência, atraso e ruído dessas células.

3.1.4 Caracterização de Células

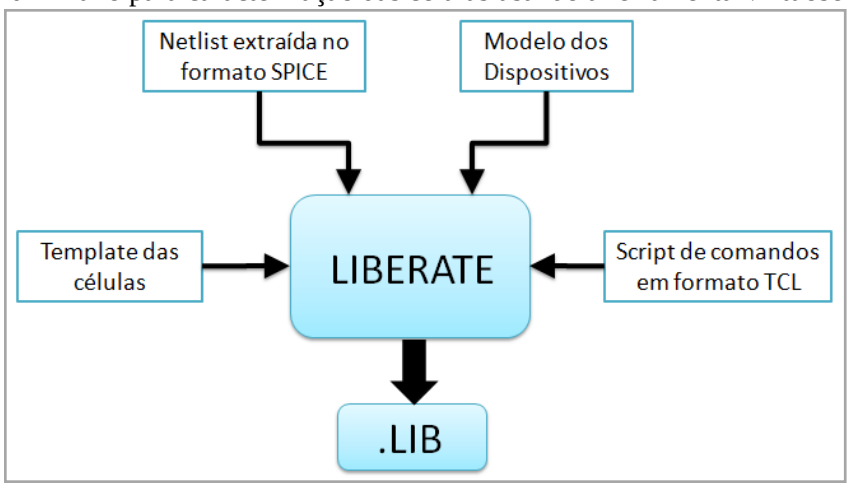
A caracterização é a etapa que gera o arquivo *liberty* (.lib) que contém os valores estimados de atraso, potência consumida e ruído que cada célula irá ter após a manufatura. Para realizar a caracterização das células usou-se a ferramenta Virtuoso Liberate (CADENCE, 2016). Conforme a Figura 3.10, a ferramenta de caracterização utiliza como entrada os seguintes arquivos:

- A *netlist* extraída no formato SPICE com as informações das capacitâncias parasitas das células.

- O arquivo com o modelo dos dispositivos (transistor, resistor e capacitor).
- O arquivo de *template*.
- O arquivo em formato TCL com as diretrizes para executar a caracterização.

A *netlist* extraída é a *netlist* gerada na etapa de extração com ajustes nos nomes de "vdd" e "gnd" retirando o caracter "!". Esse ajuste é realizado pois a ferramenta de caracterização gera mensagens de alerta solicitação a remoção do respectivo caracter durante a execução. Um exemplo de *netlit* extraída para a célula INX1 na tecnologia de 180*nm* é apresentada no Apêndice B1.

Figura 3.10 – Fluxo para caracterização das células usando a ferramenta Virtuoso Liberate.



Fonte: figura elaborada pelo autor.

O arquivo de modelo dos dispositivos contém os modelos dos transitores, resistores e capacitores da tecnologia utilizada. Por exemplo, a tecnologia de 600*nm* da biblioteca XC06 da XFAB utiliza o modelo de transistor nmos4 e pmos4.

O arquivo *template* define os valores referentes a atraso e potência e configura as informações sobre as células a serem caracterizadas conforme os itens a seguir:

- Configuração do *slew threshold* para 10% e 90% seguindo o padrão de configuração da ferramenta.
- Definição dos valores de porcentagem para o atraso das entradas e da saída seguindo o padrão de configuração da ferramenta.
- Definição dos valores de transição mímino e máximo.
- Definição dos *templates* de atraso (*delay_template*) e de potência, onde os mesmos são elaborados usando os valores de *slew* e capacitância de *load*. (*power_template*) para cada dimensionamento declarado.
- Configuração dos valores de *slew* e capacitância de *load*de cada célula.

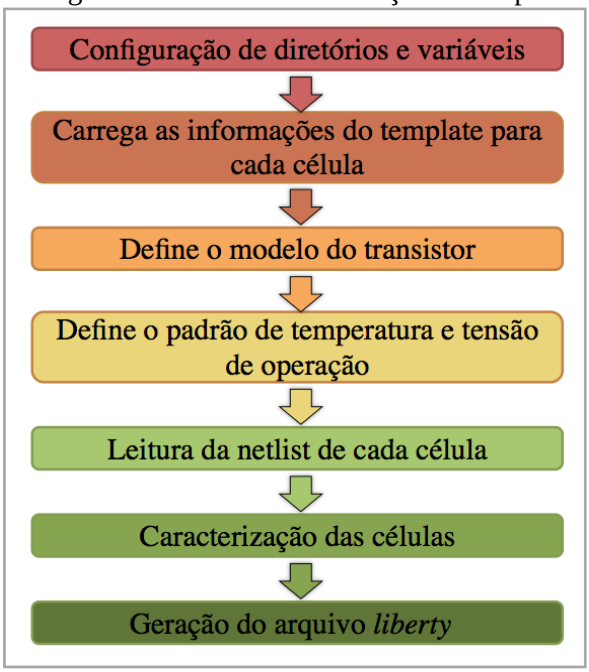
- Definição de cada célula, listando suas entradas, saída e *templates* de atraso e potência correspondentes.

Os valores dos *templates* de atraso e potência para cada dimensionamento (X1, X2, X3) foram definidos utilizando os valores correspondentes dos inversores da biblioteca *standard cell* da tecnologia alvo. Os valores de *slew* são sempre os mesmos para todas as células enquanto a capacitância de load tem valores correspondentes para cada dimensionamento declarado. Assim, o dimensionamento X1 foi definido para corresponder aos dados de *load capacitance* do inversor INX1, o dimensionamento X2 corresponde aos dados do inversor INX2 e assim seguindo o mesmo padrão para os outros dimensionamentos disponíveis.

Além dessas configurações, a ferramenta Virtuoso Liberate permite a definição de arcos de atraso e potência. Os arcos são configurações condicionais associados a uma condição lógica específica das entradas ou da saída. Assim, quando houver incidência dessa determinada condição lógica, o *script* não utilizará o *template* padrão para atraso ou potência, mas sim os valores associados nas configurações condicionais (LLOPART, 2016). Um exemplo de *template* para a tecnologia de 180*nm* para um conjunto de células é apresentado no Apêndice D1.

O *script* em formato TCL executa a sequencia de ações pararealizar caracterização. O fluxo completo de execução do script segue conforme a Figura 3.11.

Figura 3.11 – Fluxo de execução do script.



Fonte: Adaptado de (LLOPART, 2016).

A parte inicial consiste na configuração de variáveis e diretórios com a inclusão da *netlist* SPICE de todas as células, os modelos dos dispositivos e o arquivo de *template*. Em seguida, o modelo dos transistores assim como os modelos dos resistores e capacitores são definidos. Em seguida, é definido o "*corner*" para a simulação que corresponde às condições de operação referentes à temperatura e tensão (NARESH, 2010 ; POSSER, 2011):

- Melhor (*Best*): considera que o circuito trabalha nas melhores condições de ambiente, com temperatura de 0ºC e valor de tensão 10% acima do normal.
- Típico (*Typical*): considera que o circuito trabalha em condições normais do ambiente, tendo como temperatura ambiente 25ºC e tensão padrão.
- Pior (*Worst*): considera que o circuito trabalha nas piores condições de ambiente, com uma temperatura de 125ºC e valor de tensão 10% abaixo do normal.

Este trabalho utilizou a configuração típico (*typical*) para todos os experimentos, com uma tensão de operação de 3,3V para a tecnologia de 600*nm* e tensão de operação de 1,8V para a tecnologia de 180*nm*.

Após a leiutura da netlist das células, o *script* define o modelo de atraso e executa a caracterização das células. A ferramenta Virtuoso Liberate suporta os seguintes modelos de atraso (SHARMA, 2015):

- NLDM (*Non-Linear Delay Models*): o modelo caracteriza o atraso da entrada para saída e os tempos de transição inferior e superior da porta da saída com sensibilidade ao tempo de transição de entrada, carga de saída e estados da entrada. Os parâmetros de tempo de atraso e transição são utilizados para sintetizer o modelo.
- CCS (*Current Source Models*): o modelo é caracterizado pela captura da forma de onda da corrente no capacitor de carga da célulae tem sensibilidade ao tempo de transição de entrada, carga de saída e estados da entrada. O modelo força o mecanismo de caracterização a ter uma capacitância diferente de zero conectada à saída da célula.
- ECSM (*Effective Current Source Model*): captura a forma de onda da tensão na saída da célula durante sua transição. Esse modelo pode capturar a sensibilidade de carga a partir de uma capacitância de carga zero.

Este trabalho utilizou o modelo de configuração padrão NLDM para a caracterização usando o simulador elétrico SPECTRE (CADENCE, 2016) O modelo de atraso NLDM foi escolhido por suportar nodos de tecnologia maiores que 130*nm* e este trabalho utilizou os

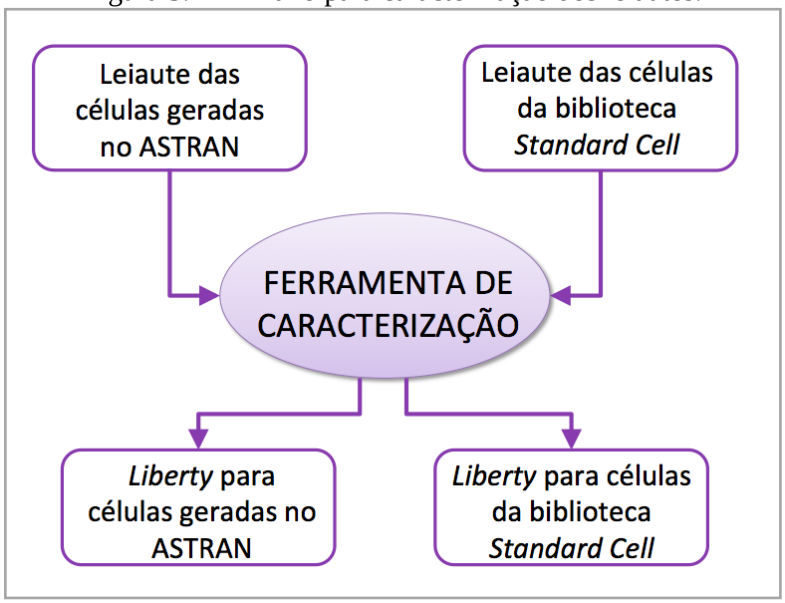
nodos de tecnologia de 180*nm* e 600*nm*. A última parte executa os comandos para a geração do arquivo *liberty* no final do fluxo.

3.2 Avaliação das células

A etapa de caracterização resultou em dois arquivos *liberty*: um arquivo com as informações das células complexas geradas pelo ASTRAN e outro arquivo referente às células recaracterizadas da biblioteca *standard cell* conforme a Figura 3.12. As mesmas configurações nas etapas de extração e caracterização aplicadas para as células geradas pelo ASTRAN foram aplicadas nas células da biblioteca *standard cell* alvo. Desse modo, os ajustes executados através do fluxo tornaram possível que comparações entre as células possam ser efetuadas (POSSER; ZIESEMER; GUIMARES; WILKE; REIS, 2010).

O *liberty* de cada netlist otimizada é elaborado a partir das informações desses arquivos de caracterização gerados. O *liberty* utiliza uma abordagem mista, ou seja, ele é formado pelas células complexas do circuito a partir do *liberty* ASTRAN e das outras células lógicas do *liberty* recaracterizado da biblioteca *standard cell*.

Figura 3.12 – Fluxo para caracterização dos leiautes.



Fonte: figura elaborada pelo autor.

A fim de validar e avaliar as etapas de verificação, extração e caracterização antes de utilizá-las no fluxo, foram gerados leiautes de células lógicas com o ASTRAN na tecnologia de 45*nm* usando a biblioteca da NangateFreePDK45 (NANGATE, 2014) como referência para comparações célula a célula em termos de área, potência dinâmica e de *leakage*,

capacitância de entrada e atraso. A *netlist* SPICE das células lógicas geradas pelo ASTRAN é a *netlist* original da biblioteca da NANGATE pois o objetivo não é aplicar técnicas de otimização e sim apenas validar as configurações definidas para essas etapas.

Os resultados foram gerados para um conjunto de células combinacionais com funções lógicas mais simples e muito utilizadas nos menores *bechmarks* do ITC'99 como o B02 e B06. As seguintes células e seus respectivos dimensionamentos foram utilizadas: AOI21 (X1, X2, X4), AOI22 (X1, X2, X4), INV (X1, X2, X4, X8), NOR2 (X1, X2, X4) e OAI21 (X1, X2, X4).

Tabela 3.1 – Comparação célula a célula entre os leiautes gerados pelo ASTRAN e a biblioteca Nangate FreePDK45.

	Área	Potência de *Leakage*	Potência Dinâmica	Capacitância de entrada	Atraso
Células	**Redução(%)**	**Redução(%)**	**Redução(%)**	**Redução(%)**	**Redução(%)**
AOI21_X1	0,00%	-4,41%	0,00%	1,51%	0,14%
AOI21_X2	0,00%	0,00%	0,07%	0,29%	0,02%
AOI21_X4	0,00%	3,90%	0,17%	0,45%	0,08%
AOI22_X1	0,00%	-0,02%	0,52%	2,16%	0,07%
AOI22_X2	-5,56%	0,02%	2,63%	1,19%	2,22%
AOI22_X4	-2,94%	-0,04%	0,40%	0,97%	0,19%
INV_X1	0,00%	0,00%	0,16%	1,86%	0,27%
INV_X2	0,00%	0,00%	0,01%	0,75%	0,04%
INV_X4	0,00%	0,00%	0,05%	1,08%	0,11%
INV_X8	0,00%	0,00%	0,05%	0,67%	-0,59%
NOR2_X1	0,00%	0,00%	0,17%	0,93%	0,16%
NOR2_X2	-10,00%	0,00%	0,00%	0,13%	-0,01%
NOR2_X4	0,00%	0,00%	-0,05%	0,49%	-0,16%
OAI21_X1	-12,50%	3,49%	-0,32%	2,16%	0,09%
OAI21_X2	0,00%	-0,01%	0,57%	1,43%	0,40%
OAI21_X4	-3,85%	-3,53%	-0,34%	0,86%	0,19%
Média	-2,18%	-0,04%	0,26%	1,06%	0,20%

Fonte: tabela elaborada pelo autor.

A Tabela 3.1 realiza a comparação célula a célula entre as células geradas pelo ASTRAN e as células da biblioteca da NANGATE. Em relação aos dados sobre a área, a maior parte das células geradas alcançaram o mesmo valor de área. Para alguns casos como a célula NOR2_X2 e a célula OAI21_X1, o *folding* dos transistores afetou a largura da célula resultando uma área maior que a original. Os valores referentes à potência dinâmica e de *leakage*, capacitância de entrada e atraso seguem o mesmo padrão, com variações mínimas em redução pois os circuitos e o dimensionamento são os mesmos da biblioteca original. As

variações míminas de todas as células para os termos avaliados validaram as configurações aplicadas nas etapas de verificação, extração e caracterização.

3.3 Configuração do ASTRAN para uma tecnologia

O ASTRAN utiliza um conjunto de regras de projeto que é comum a diversos processos de fabricação e o leiaute das células geradas tem compatibilidade com os principais modelos usados por bibliotecas comerciais *standard cells*. A ferramenta tem suporte atual para os nodos de tecnologia de 350*nm*, 65*nm* e 45*nm* (ZIESEMER, 2014).

Neste trabalho, o ASTRAN foi adaptado para as tecnologias de 180*nm* e 600*nm* usando como referênciaas informações das bibliotecas XC018 (180*nm*) e XC06 (600*nm*) da XFAB (X-FAB, 2017). Essas duas tecnologias foram escolhidas visando deixar encaminhado para trabalhos futuros que queiram realizar a fabricação de circuitos integrados, pois estas tecnologias já tiveram outros projetos fabricados em outros trabalhos do grupo de pesquisa GME.

O ASTRAN utiliza três arquivos como parâmetros de entrada: um *netlist* no formato SPICE, o arquivo de tecnologia com as regras de projeto e o template de configuração. Os arquivos de tecnologia foram elaborados com base nos arquivos de regras de projeto da ferramenta Virtuoso Layout Suite (CADENCE, 2014) disponível nos *design kits* correspondentes. O arquivo de tecnologia ajustado para 180*nm* utilizado neste trabalho é apresentando no Apêndice A.1.

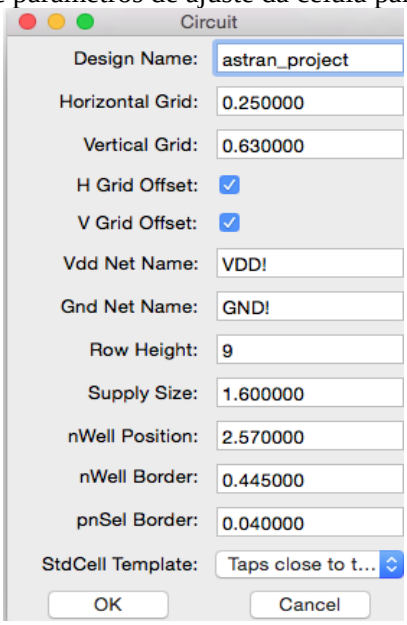
A netlist de entrada para qualquer tecnologia tem a restrição que o modelo do transistor precisa ser sempre NMOS e PMOS, por isso os modelos ne/pe (180*nm*) e nmos4/pmos4 (600*nm*) foram ajustados conforme o padrão para a síntese do leiaute. Ao finalizar a síntese, as *netlists* são ajustadas para os modelos da tecnologia correspondente para serem utilizadas nas etapas posteriores de verificação de DRC e LVS.

Para o *templante* de configuração são configurados os parâmetros da célula que mudam conforme a tecnologia alvo. Os parâmetros das células seguem o padrão da maioria das bibliotecas *standard cells*: altura fixa das células, posicionamento em bandas, alimentação nas extremidades superiores e inferiores e pinos alinhados a uma grade de roteamento. E alguns dos parâmetros são flexibilizados com o objetivo de permitir a configuração de células com características diferentes: largura horizontal e vertical da grade de roteamento, nomes das

redes de alimentação, largura das redes de alimentação em metal 1, altura da banda em passos da grade de roteamento e posição do poço N dentro das células.

Para a configuração de cada tecnologia foram utilizadas como referência os parâmetros de células inversora INX1 de cada biblioteca da X-FAB para o ajuste dos parâmetros de uma célula INX1 gerada pelo ASTRAN. Até a conclusão de todos os ajustes, cada leiaute gerado pelo ASTRAN foi importado na ferramenta Virtuoso Layout Suite (CADENCE, 2014) para a comparação das medidas com a célula de referência da biblioteca. Os parâmetros da célula são configurados seguindo o padrão apresentado na Figura 3.13. A adaptação das configurações abrage também o arquivo de *layermap* usado na importação dos leiautes para a ferramenta Virtuoso Layout Suite, onde os mesmos não precisaram ser ajustados pois cada *design kit* da tecnologia correspondente da XFAB carregava automaticamente um arquivo de *layermap* que era compatível.

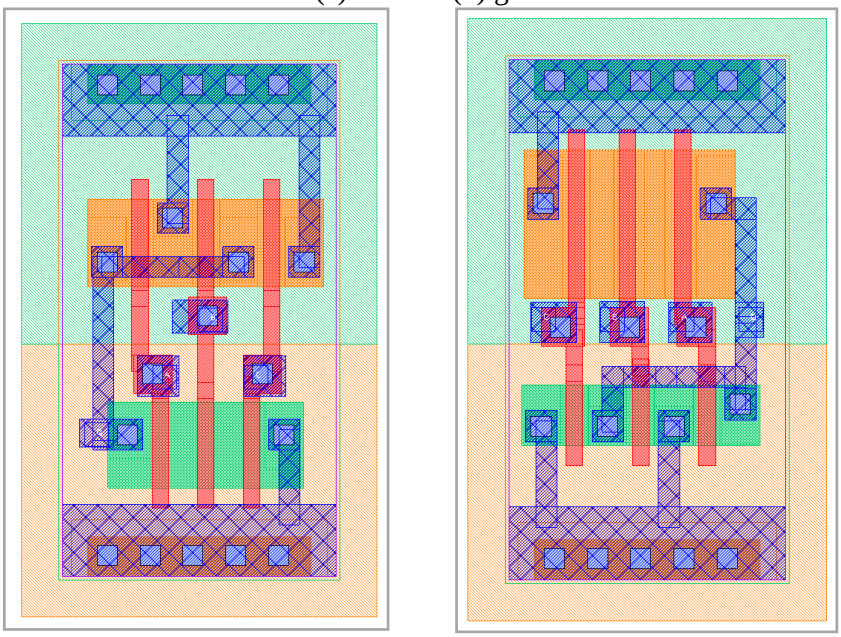
Figura 3.13 – Exemplo de parâmetros de ajuste da célula para a tecnologia de 180*nm*.



Fonte: figura elaborada pelo autor.

Após o ajuste da célula INX1, leiautes de diversas células lógicas foram gerados e testados para verificar e validar os parâmetros das tecnologias de 180*nm* e 600*nm*. Algumas células geradas pelo ASTRAN para a tecnologia de 180*nm* podem ser vistas na Figura 3.14 (a) que corresponde a uma célula NAND3 e Figura 3.14 (b) que corresponde a uma célula NOR3. E outros exemplos são apresentados na Figura 3.15 para uma célula de um meio somador HAX1 e na Figura 3.16 de um mux MU2X1.

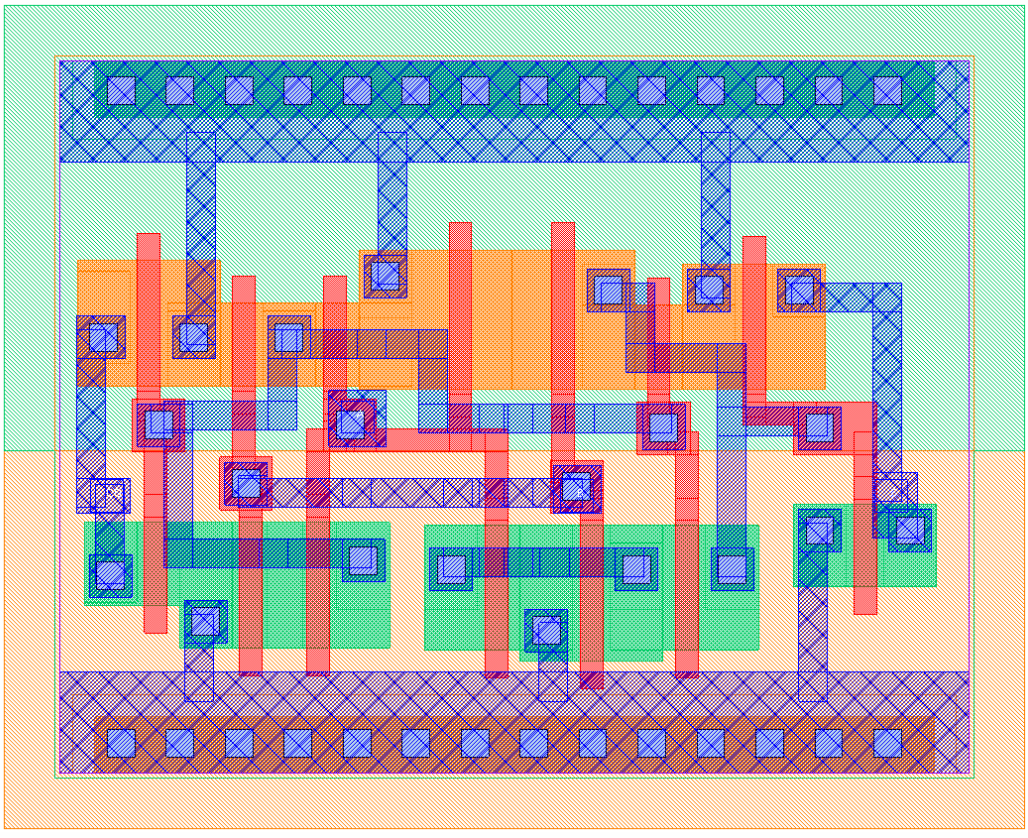
Figura 3.14 – Leiaute das células NAND3 (a) e NOR3(b) geradas ASTRAN em 180*nm*.



(a) (b)

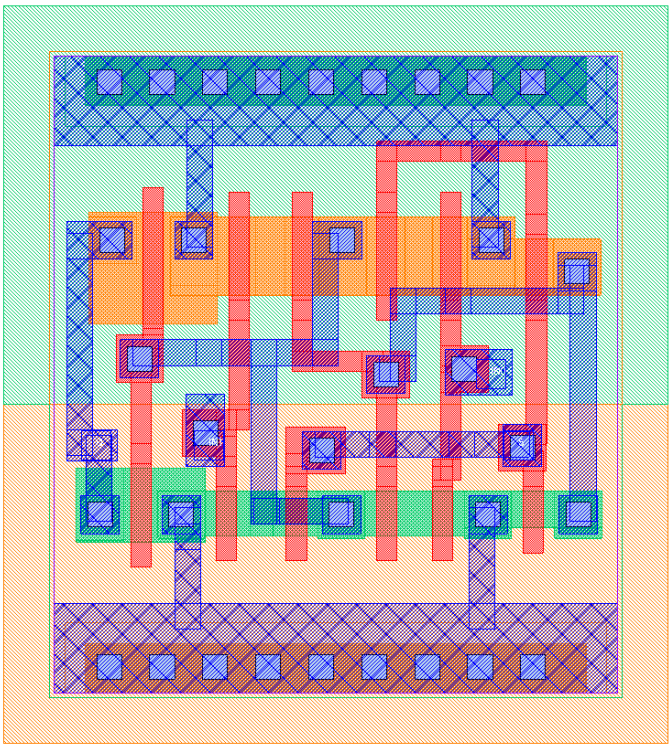
Fonte: figuras elaboradas pelo autor.

Figura 3.15 – Leiaute de uma célula de um meio somador HAX1 gerada pelo ASTRAN em 180*nm*.



Fonte: figura elaboradas pelo autor.

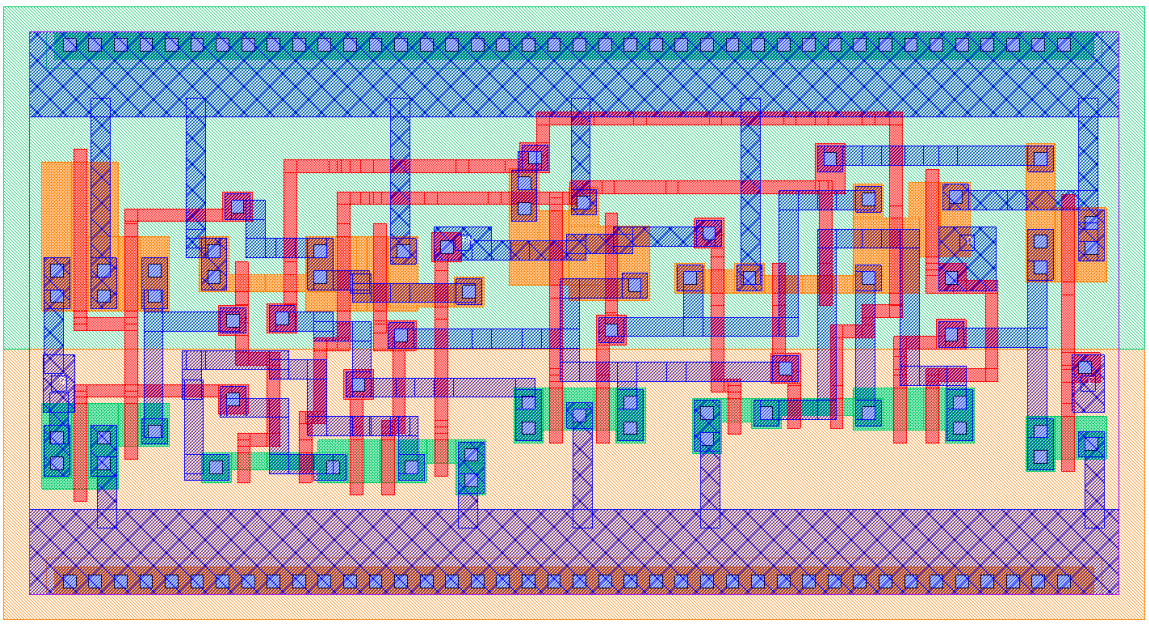
Figura 3.16 – Leiaute de uma célula de um mux MU2X1 gerada pelo ASTRAN em 180*nm*.



Fonte: figura elaboradas pelo autor.

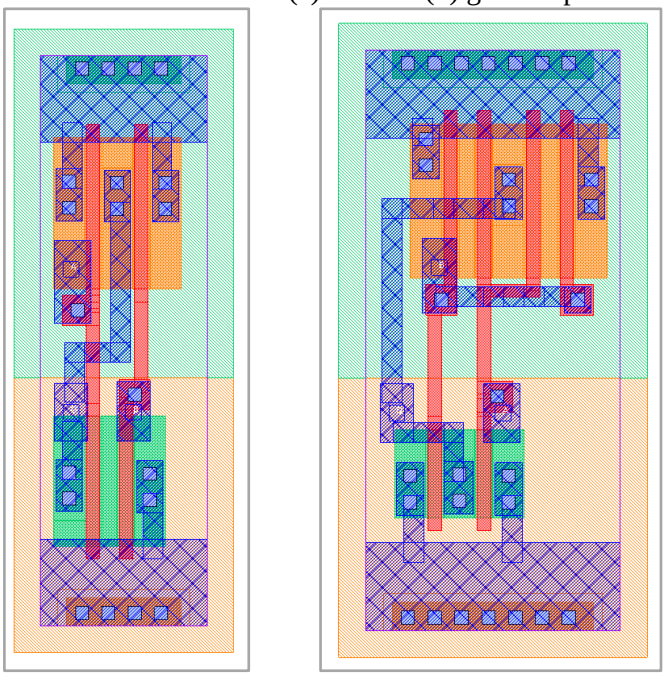
Para a tecnologia de 600*nm*, os exemplos são apresentados na Figura 3.17 com um flip-flop tipo D DFRRQX1 na Figura 3.18 (a) para a porta NAND2, Figura 3.18 (b) para a porta NOR2 e na Figura 3.19 com a célula AO32X1. No Capítulo 4 são apresentados exemplos de células complexas geradas para cada uma destas estas tecnologias.

Figura 3.17 – Leiaute de um flip-flop tipo D DFRRQX1 gerado pelo ASTRAN em 600*nm*.



Fonte: figura elaboradas pelo autor.

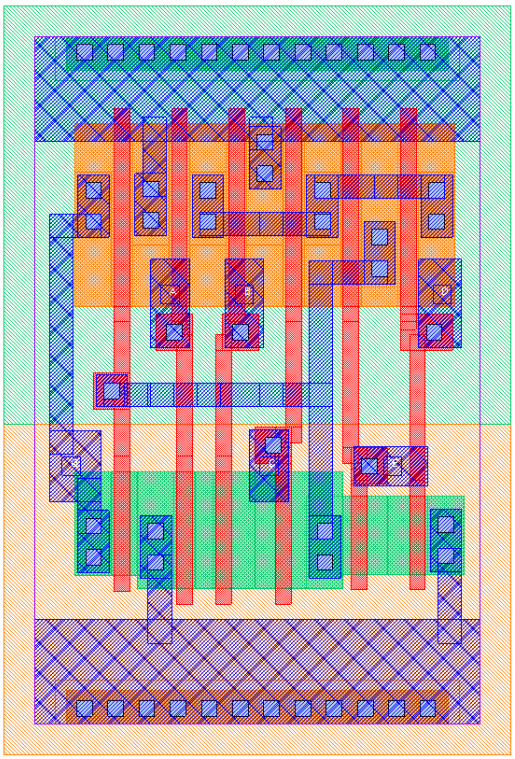
Figura 3.18 – Leiaute das células NAND2 (a) e NOR2(b) geradas pelo ASTRAN em 600nm.



(a) (b)

Fonte: figuras elaboradas pelo autor.

Figura 3.19 – Leiaute da célula AO32X1 geradas pelo ASTRAN em 600nm.



Fonte: figura elaboradas pelo autor.

4 RESULTADOS

A partir da meteodologia apresentada no Capítulo 3, otimizações aplicadas em circuitos a nível SPICE se tornaram possíveis com o auxílio da ferramenta ASTRAN. Este capitulo apresenta as avaliações dos resultados para circuitos que foram otimizados usando células complexas e a síntese do leiaute dos mesmos gerada pela ferramenta ASTRAN. Os resultados são avaliados para um conjunto de *bechmarks* do ITC'99 nas tecnologias de *180nm* e 600*nm* em termos de consumo de energia, área, atraso e número de transistores. Comparações são realizadas entre as *netlists* otimizadas seguindo o método ASTRAN e circuitos utilizando a metodologia *standard cell*.

Os resultados são avaliados em termos de número de transistores, área, potência dinâmica e de *leakage* e *timing* para o conjunto de *benchmarks* do ITC'99 (ITC'99, 2016) nas tecnologias de 180*nm* e 600*nm*. Esses *bechmarks* foram escolhidos por abrangerem circuitos com tamanhos variados em relação ao de número de transistores. Circuitos pequenos como o B01 e B02 que contém menos de 400 transistores proporcionaram uma rápida avaliação durantes os ajustes de cada etapa da metologia, além de verificar o impacto daotimização da *netlist* nos mesmos. Em circuitos como o B05 e B07 com até 2700 transistores, é possível avaliar o impacto da otimização aoaumentar o número de transistores. A técnica de uso de SCCG (*Static CMOS Complex Gates*) foi aplicada para o conjunto de circuitos de *benchmarks* usando a ferramenta desenvolvida por (CONCEIÇÃO et al., 2016), resultando um *netlist* otimizado para cada circuito conforme a metodologia do Capítulo 3. Os leiautes das células complexas do *netlist* otimizado foram gerados pelo ASTRAN. Para realizar a avaliação dos resultados usou-se a ferramenta comercial Encounter RTL Compiler (CADENCE, 2014), sintetizando todos os circuitos para um *clock* de 400MHz. Os valores de área foram medidos manualmente através da ferramenta Virtuoso e inseridos nos respectivos arquivos de *liberty* utilizados como entrada na ferramenta RTL Compiler.

Para cada circuito de *bechmark* são geradas três *netlists*:

- *Netlist* sem restrições da biblioteca da XAFB que abrange todas as células.
- *Netlist* com restrições da biblioteca da XFAB que filtra células que são consideradas complexas (somadores, multiplexadores, XOR/XNOR, AOI e OAI). Também chamada de limitada.

- *Netlist* otimizada que tem o leiaute das células complexas geradas pelo ASTRAN.

A *netlist* com restrições foi elaborada com a motivação deverificar se uma *netlist* com células complexas geradas exclusivamente para o circuito alvo do zero pode ser mais benéfico em termos de redução do número de transistores em relação a versão que utilize o conjunto original de complexas pré-existentes na biblioteca *standard cell*. Essa *netlist* com restrições é usada como entrada na etapa de otimização para a criação da *netlist* otimizada. Nos testes iniciais a netlist sem restrições tinha sido utilizada como entrada, entretanto se observou que uma parte das células complexas geradas correspondiam a uma função lógica já existente na biblioteca da XFAB. O filtro executado para separar quais funções lógicas já existiam ou não na biblioteca tornou o processo mais complicado, por isso a *netlist* com restrições foi utilizada como entrada.

4.1 Aplicação de SCCG sem restrição de transistores em série

A ferramenta de aglutinação de portas permite restringir o número de transistores em série. Assim, o primeiro experimento não restringe o número de transistores em série nas células complexas geradas com o objetivo de verificar a viabilidade da síntese do leiaute e as células complexas geradas em relação à número de entradas e complexidade do circuito.

O experimento foi realizado para os *bechmarks* B01, B02, B03, B06, B08 e B09 que eram circuitos pequenos em termos de portas lógicas em relação aos demais para facilitar as avaliações sobre as *netlists* otimizadas geradas. O impacto no aumento no dimensionamento antes da verificação dos resultados era esperando por não aplicar o controle dos transistores em série.

Apenas os circuitos B03 e B09 tiveram a maior parte dos leiautes gerados com sucesso neste experimento. As poucas células em que os leiautes não foram finalizados foram subistituidas pelas portas equivalentes conforme a expressão lógica da porta complexa do *netlist*. As células não finalizadas com um grande dimensionamento tiveram diversas variações de *folding* aplicadas nos transistores em relação à valores de W. Entretanto o roteamento *intra-cell* do ASTRAN apresentou dificuldades para ser finalizado, pois o *folding* dos transistores tornou-se excessivo nesses casos.

4.3 Aplicação de SCCG com restrição de transistores em série

A segunda parte dos experimentos limitou para no máximo quatro transistores em série as células complexas geradas da *netlist* otimizada e foi aplicada em todos os *bechmarks*. Os resultados mostraram que dimensionamento dos transistores reduziu um pouco os valores e o número de entradas por porta complexa se tornou um pouco mais distribuído.

Através da Tabela 4.1 é realizada uma comparação para o circuito B09 em termos de funções lógicas e entradas correspondentes para a versão com restrição e sem restrição. O circuito B03 apresentou pouca diferença de uma versão dos experimentos para outra. O circuito B08 apresentou problemas e não foi finalizado no experimento anterior, mas após as alterações do experimento atual todas as células complexas foram geradas pelo ASTRAN. A Tabela 4.2 mostra os dados da versão com e sem restrição do B08 em termos de funções lógicas e entradas correspondentes.

Tabela 4.1 – Número de entradas e expressões lógicas das complexas geradas para o circuito B09 entre a versão com e sem controle de transistores em série.

B09 - Com Restrição de 4 transistores em série		
Célula	**Expressão Lógica**	**N° entradas**
gate_0	!(((A6+A10)*(A7+A9))*(A8+A11))	6
gate_2	!(((A3+(A6*A7))*(A2+A0))*(A1+A5))	7
gate_3	!(((A10+(A8*A7))*A12)*(A7+((A9+A11)*(A5+A1))))	9
gate_4	!((A11*(A10+A7))*((A8+A9)+A6))	6
gate_7	!((A5+A7)*(A4+A6))	4
gate_14	!((A7*((A6*A3)+(A1*A8)))*((A5*A4)+(A0*A9)))	9

B09 - Sem Restrição de transistores em série		
Célula	**Expressão Lógica**	**N° entradas**
gate_0	!(((A6+A10)*(A7+A9))*(A8+A11))	6
gate_7	!((A5+A7)*(A4+A6))	4
gate_8	!((A11*(A10+A7))*((A8+A9)+A6))	6
gate_9	!(((A9+(A8*A7))*A11)*(A7+((A12+A10)*(A4+A1))))	9
gate_10	!(((A2+(A6*A7))*(A5+A0))*(A1+A4))	7
gate_12	!((((((A2+A4)*(A8+A10))*(A1+A3))*(A7+A9))*(A0+A5))*(A6+A11))	12
gate_13	!((A5+A7)*(A4+A6))	4

Fonte: tabela elaborada pelo autor.

O circuito B09 apresentando na Tabela 4.2 teve um balanceamento do número de entradas mais equilibrado, pois uma porta de 12 entradas foi eliminada na versão com restrição. O circuito B08 apresentou uma boa distribuição de entradas na versão com restrição, eliminando portas com mais de 10 entradas. As portas complexas apresentadas nas Tabela 4.1 e Tabela 4.2 representam as novas funções lógicas inseridas após a etapa de otimização da *netlist*. A lista de *bechmarks* com leiautes finalizados alterou dos circuitos B03

e B09 para os circuitos B01, B03, B06, B08 e B09, ou seja, a redução nos valores de dimensionamento e número de entradas das portas complexas melhorou a viabilidade da geração dos leiautes.

Tabela 4.2 – Número de entradas e expressões lógicas das complexas geradas para o circuito B08 entre a versão com e sem controle de transistores em série..

B08 - Com Restrição de 4 transistores em série		
Célula	**Expressão Lógica**	**N° entradas**
gate_0	!(((A6+A10)*(A7+A9))*(A8+A11))	6
gate_2	!(((A3+(A6*A7))*(A2+A0))*(A1+A5))	7
gate_3	!(((A10+(A8*A7))*A12)*(A7+((A9+A11)*(A5+A1))))	9
gate_4	!((A11*(A10+A7))*((A8+A9)+A6))	6
gate_7	!((A5+A7)*(A4+A6))	4
gate_14	!((A7*((A6*A3)+(A1*A8)))*((A5*A4)+(A0*A9)))	9

B08 - Sem Restrição de transistores em série		
Célula	**Expressão Lógica**	**N° entradas**
gate_0	!((A17*(A18+A10))*((((((A1+A13)+A12)+A15)+A14)+A16)+A11))	10
gate_1	!((A11*(A12+A13))*(((A7+A8)+A9)+A10))	7
gate_2	!(((A14+(((A10+A2)+A6)*(A13+A12)))*((A1+A3)+A15))*(A16+A11))	11
gate_3	!(A3*(A4+(A2*A1)))	4
gate_4	!((A3+(A0*(A4+A2)))*((A1+A6)+A5))	7
gate_5	!((((A11*A4)*A12)*((A2*A9)+(((A6*A7)*A120)*A14)))+((((A8*A3)*A5)*A10)*A1))	14
gate_6	!((A5+A0)*((A3+A1)+A2))	5

Fonte: tabela elaborada pelo autor.

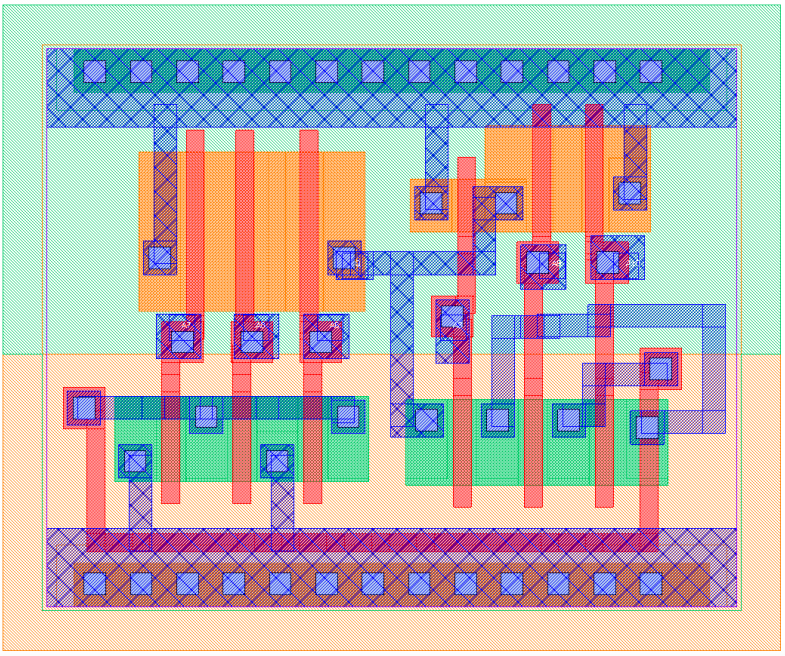
Entretanto, os *bechmarks* B02, B04, B05, B07 e B10 apresentaram problemas na geração do leiaute de diversas células complexas. A partir dessas células, foram efetuados testes utilizando um dimensionamento mínimo para os valores de Wp e Wn dos transistores com o objetivo de verificar quais detalhes ainda estavam afetando a finalização dos leiautes dessas células complexas. Os leiautes resultantes de algumas células conseguiram ser finalizados, mostrando que o número de entradas excessivo tornava mais complexo o roteamento *intra-cell.* Além do aumento da complexidade do roteamento, o dimencionamento por esforço lógico também é influenciado pelo número de entradas.

4.4 Aplicação de SCCG com restrição de transistores em série e número de entradas

A terceira parte dos experimentos limitou as células complexas geradas da *netlist* otimizada para no máximo quatro transistores em série e 6 entradas e foi aplicada em todos os *bechmarks*. A limitação de entradas foi escolhida para 6 pois nos experimentos anteriores todas as células com 6 entradas ou menos não apresentaram problemas na etapa de síntese do leiaute. Os resultados não apresentaram problemas na finalização dos leiautes das complexas

em todos *bechmark* testados. Um exemplo de leiaute de uma célula complexa gerada com essas restrições aplicadas na tecnologia de 180*nm* é apresentada na Figura 4.1, onde a complexa tem respectivamente 6 entradas, 12 transistores e sua expressão lógica correspondente é !((A3*(A10+A9))*((A7+A8)+A6)). Outros exemplos de células complexas geradas pela ferramenta ASTRAN para 180*nm* são apresentados no Apêndice G.

Figura 4.1 – Célula complexa de 6 entradas e 12 transistores gerada pela ferramenta ASTRAN para a tecnologia de 180*nm*.



Fonte: figura elaborada pelo autor.

4.4.1 Resultados para 180nm

A *netlits* otimizada é comparada com duas *netlists* geradas a partir da biblioteca XC018 da XFAB: a netlist sem restrições (total de 828 células) e a netlist com restrições que abrage um filtro de 232 células (somadores, multiplexadores, XOR/XNOR, AOI e OAI).

A Tabela 4.3 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação ao número de transistores de cada circuito de *benchmark*. Os dados da versão otimizada em relação à versão com restrições da XFAB mostraram reduções em número de transistores acima de 10% nos circuitos B02, B05 e B10. Cada um destes circuitos tem uma quantidade de complexas inclusas na *netlist*com valores variados,onde o B02 tem 3 complexas e B10 com 25 complexas. Por isso, quanto maior for o número de complexas na *netlist* não significa que o circuito sempre vai alcançar melhores reduções em número de

transistores. Os *bechmarks* B04, B05 e B07 são os maiores circuitos em termos de número de transistores e também são os que tem as maiores concentrações de células complexas nas *netlists* otimizadas, por exemplo. E apenas o circuito B05 alcançou reduções superiores a 10%.

Ao realizar a comparação com os dados da *netlist* sem restrições da XFAB, podemos observar que poucos circuitos alcançaram uma otimização que é considerada mímina por não ultrapassar 5%. O B04 é o pior caso com uma porcentagem excedente de 47,85%, entretanto é importante salientar que a base da *netlist* otimizada é a versão com restrição da XFAB. Por isso, há casos como o B04 e outros circuitos que a base da versão com restrição da XFAB tem resultados com valores muito excedentes em relação a sua versão sem restrição. E isso influencia diretamente na comparação dos valores com a versão sem restrição da XFAB.

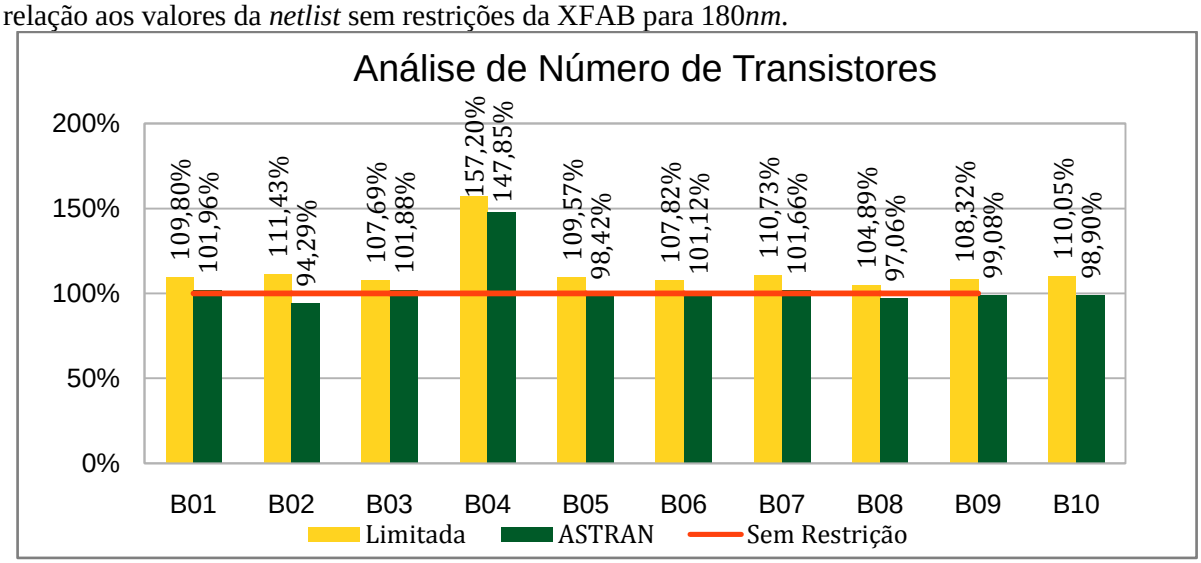
Tabela 4.3 – Comparaçãoentre a *netlist* otimizada e as outras *netlists* da XFAB em relação ao número de transistores em 180*nm*.

Número de Transistores						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B01	336	312	7,14%	306	312	-1,96%
B02	234	198	15,38%	210	198	5,71%
B03	1372	1298	5,39%	1274	1298	-1,88%
B04	3798	3572	5,95%	2416	3572	-47,85%
B05	3046	2736	10,18%	2780	2736	1,58%
B06	386	362	6,22%	358	362	-1,12%
B07	2808	2578	8,19%	2536	2578	-1,66%
B08	1072	992	7,46%	1022	992	2,94%
B09	1406	1286	8,53%	1298	1286	0,92%
B10	1204	1082	10,13%	1094	1082	1,10%
Média	**1566,20**	**1441,60**	**8,46%**	**1329,40**	**1441,60**	**-4,22%**

Fonte: tabela elaborada pelo autor.

O gráfico da Figura 4.2 reúne os dados da Tabela 4.3, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. Assim, os valores da versão sem restrição são usados como uma linha vermelha de 100% e qualquer valor que alcançar redução é representando abaixo dessa linha vermelha e vice-versa. A diferença de valores entre a versão com restrição e a versão sem restrição da XFAB fica mais nítida na Figura 4.2, onde os resultados da *netlist* limitada excedem em 10% para a maioria dos circuitos e tendo o caso mais extremo do B04 com 57,20%. O B04 é dos maiores circuitos em termos de número de transistores, por isso a versão limitada com a restrição das 232 céulas acabou proporcionando esse impacto.

Figura 4.2 – Análise do número de transistores da *netlist* limitada XFAB e *netlist* otimizada em relação aos valores da *netlist* sem restrições da XFAB para 180*nm*.



Fonte: figura elaborada pelo autor.

A Tabela 4.4 mostra as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação à área de cada circuito de *benchmark*. Os dados da versão otimizada em relação à versão com restrições da XFAB valores apresentaram valores muitos próximos que não ultrapassam 2% para todos os circuitos. Entretanto, ao comparar com a versão sem restrições, os circuitos B02, B05, B07 e B10 excedem em torno de 10% em área. O impacto é relacionado ao tipo de circuito em que a otimização foi aplicada, já que metade dos circuitos não pertence ao grupo de maior volume de transistores.

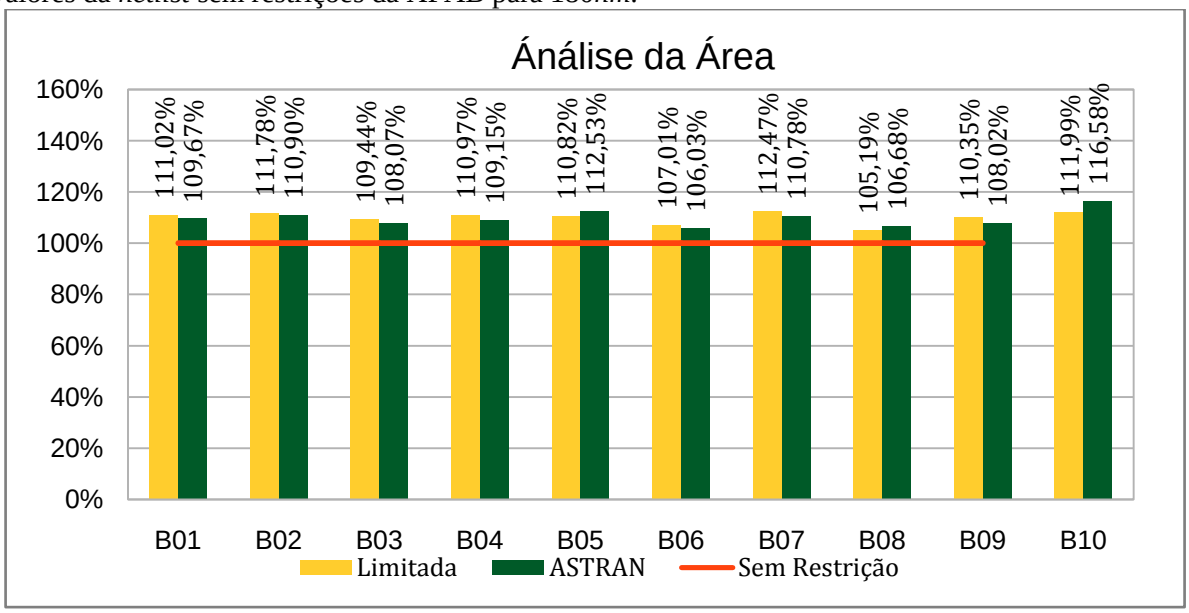
Tabela 4.4 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à área em 180*nm*.

	Área(um²)					
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B01	732,44	723,54	1,22%	659,73	723,54	-9,67%
B02	506,51	502,54	0,78%	453,15	502,54	-10,90%
B03	2875,64	2839,78	1,25%	2627,67	2839,78	-8,07%
B04	8006,68	7875,36	1,64%	7214,94	7875,36	-9,15%
B05	6786,23	6890,88	-1,54%	6123,85	6890,88	-12,53%
B06	824,08	816,57	0,91%	770,12	816,57	-6,03%
B07	6054,13	5962,84	1,51%	5382,68	5962,84	-10,78%
B08	2283,75	2316,10	-1,42%	2171,07	2316,10	-6,68%
B09	3006,22	2942,67	2,11%	2724,20	2942,67	-8,02%
B10	2619,50	2726,93	-4,10%	2339,08	2726,93	-16,58%
Média	**3369,52**	**3359,72**	**0,24%**	**3046,65**	**3359,72**	**-9,84%**

Fonte: tabela elaborada pelo autor.

O gráfico da Figura 4.3 reúne os dados da Tabela 4.4, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. A análise de área nos mostra que todos os *bechmarks* da versão com restrição excederam os valores obtidos em torno de 10% em relação à versão sem restrição, assim como a versão otimizada que usa como base a *netlist* limitada. Os valores próximos entre a versão limitada da XFAB e a *netlist* otimizada ficam bem visíveis na Figura 4.3.

Figura 4.3 – Análise de área da *netlist* limitada XFAB e *netlist* otimizada do ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 180*nm*.



Fonte: figura elaborada pelo autor.

A Tabela 4.5 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação à *timing* de cada circuito de *benchmark*. Os valores obtidos para *timing* são referentes ao *slack* do pior caminho reportado pelo RTL Compiler. Os dados da versão otimizada em relação à versão com restrições da XFAB não alcançam reduções, onde a maioria dos circuitos excede os valores com violações de 23% no máximo. Porém, os piores casos se concentram nos circuitos que apresentam o maior número de transistores e também de células complexas na *netlist.* Os *bechmarks* contém a respectiva quantidade de complexas em suas *netlists*: 23 para o circuito B04, 50 para o circuito B05 e 43 para o circuito B07.

A aplicação da técnica de SCCG nestes circuitos proporcionou impactos significativos em *timing*. A mesma proporção desse impacto também se mostra ao comparar os dados com a versão sem restrições da XFAB, onde os mesmo circuitos são os mais afetados. Entretanto, os circuitos menores em número de transistores B01, B02, B03, B06 e B10 alcançaram valores de redução acima de 10%. A memor quantidade de complexas de suas *netlists* otimizadas que

atinge no máximo 25 células se mostrou benéfico na comparação com a versão sem restrições.

Tabela 4.5 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à área em 180*nm*.

	Timing (ps)					
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B01	1737	1667	-4,03%	1502	1667	10,99%
B02	1766	1628	-7,81%	1469	1628	10,82%
B03	562	538	-4,27%	443	538	21,44%
B04	485	-887	-154,68%	642	-887	-172,38%
B05	8	-861	-100,93%	23	-861	-102,67%
B06	1694	1610	-4,96%	1317	1610	22,25%
B07	506	202	-60,08%	405	202	-50,12%
B08	1302	1216	-6,61%	1200	1216	1,33%
B09	601	459	-23,63%	687	459	-33,19%
B10	1231	1068	-13,24%	725	1068	47,31%
Média	**989,20**	**664,00**	**-38,02%**	**841,30**	**664,00**	**-24,42%**

Fonte: tabela elaborada pelo autor.

Para a potência de leakage, a Tabela 4.6 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB de cada circuito de *benchmark*. Metade dos circuitos do experimento apresentaram reduções acima de 10% na comparação com a *netlist* com restrições, onde o melhor caso alcançou 22,45% no B02. Os circuitos B02, B05 e B10 que obtiveram as maiores reduções em número transistores estão inclusos nesse grupo acima de 10%. Ao realizar a comparação com a versão sem restrições, a maioria dos circuitos não alcançaram reduções tendo como exceção apenas os circuitos B02 e B05 com um mínimo valor que não ultrapassa 6%.

No entanto, mesmo alcançando reduções acima de 10% nas comparações com a *netlist* limitada, os valores obtidos em potência de *leakage* para a tecnologia de 180*nm* são extremamente baixos. Tanto em área como em número de transistores, os dados da versão otimizada apresentam poucas reduções e muitos valores excedentes em relação a *netlist* sem restrições. Isso influenciou diretamente os valores obtidos para a potência de *leakage* nessa comparação.

O gráfico da Figura 4.4 reúne os dados da Tabela 4.6, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. A análise de potência de *leakage* nos mostra que todos os *bechmarks* da versão com restrição excederam os valores obtidos em torno de 10% ou mais em relação à versão sem restrição. A

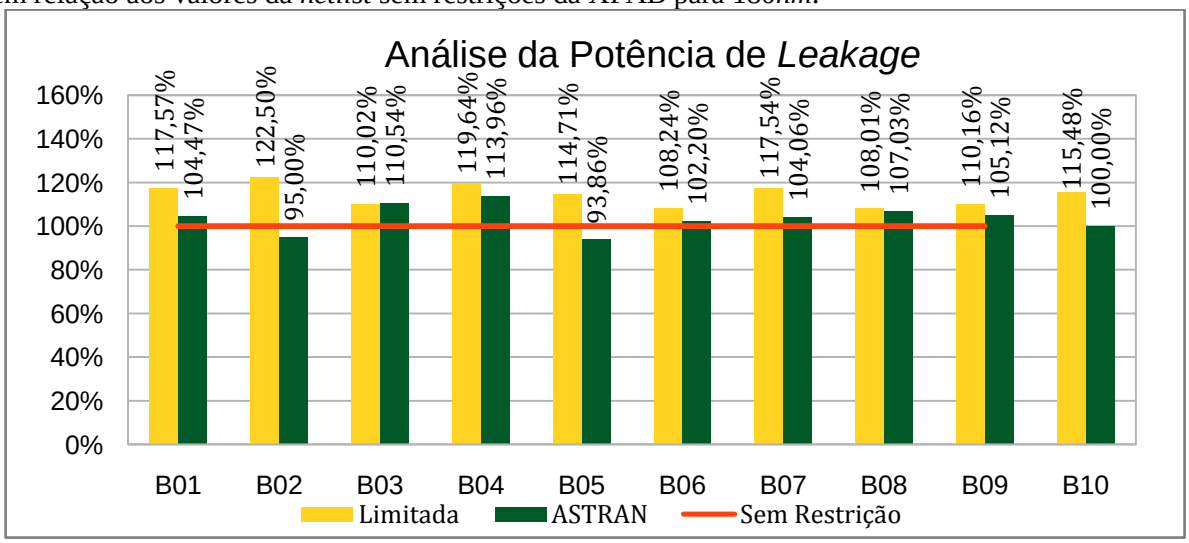
versão otimizada do ASTRAN tem pequenas variações mas a maioria excede os valores da versão sem restrições.

Tabela 4.6 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à potência de *leakage* em 180*nm*.

Potência de *Leakage* (nW)						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B01	0,368	0,327	11,14%	0,313	0,327	-4,47%
B02	0,245	0,190	22,45%	0,200	0,190	5,00%
B03	1,274	1,280	-0,47%	1,158	1,280	-10,54%
B04	3,607	3,436	4,74%	3,015	3,436	-13,96%
B05	3,267	2,673	18,18%	2,848	2,673	6,14%
B06	0,394	0,372	5,58%	0,364	0,372	-2,20%
B07	2,781	2,462	11,47%	2,366	2,462	-4,06%
B08	0,984	0,975	0,91%	0,911	0,975	-7,03%
B09	1,312	1,252	4,57%	1,191	1,252	-5,12%
B10	1,201	1,040	13,41%	1,040	1,040	0,00%
Média	**1,543**	**1,401**	**9,20%**	**1,341**	**1,401**	**-3,62%**

Fonte: tabela elaborada pelo autor.

Figura 4.4 – Análise de potência de *leakage* da *netlist* limitada XFAB e *netlist* otimizada do ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 180*nm*.



Fonte: figura elaborada pelo autor.

Para a potência dinâmica, a Tabela 4.7 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB de cada circuito de *benchmark*. Os dados da versão otimizada em relação à versão com restrições da XFAB mostraram reduções em potência dinâmica acima de 15% nos circuitos B01, B02, B03, B04, B06 e B10 que representam 60% dos circuitos do experimento. Entre os três circuitos com a maior quantidade de transistores, o

B07 foi o único que não alcançou redução com um dado muito próximo do valor comparado. Ao realizar a comparação com a versão sem restrições, apenas os circuitos B01, B02 e B10 alcançam uma porcentagem acima de 15%, onde apenas o B07 excedeu em 5% o valor obtido.

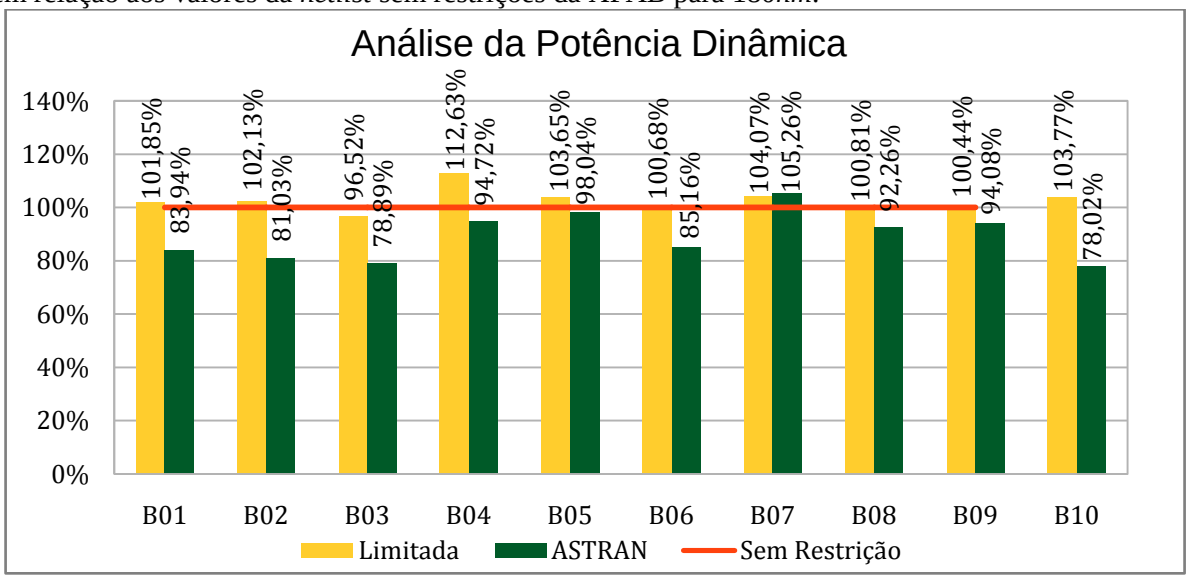
Tabela 4.7 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à potência dinâmica em 180*nm*.

Potência Dinâmica (uW)						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B01	282,95	233,19	17,59%	277,80	233,19	16,06%
B02	191,86	152,22	20,66%	187,86	152,22	18,97%
B03	1148,66	938,80	18,27%	1190,04	938,80	21,11%
B04	3107,20	2612,99	15,91%	2758,72	2612,99	5,28%
B05	2225,49	2105,17	5,41%	2147,16	2105,17	1,96%
B06	328,62	277,97	15,41%	326,39	277,97	14,84%
B07	2003,77	2026,67	-1,14%	1925,33	2026,67	-5,26%
B08	807,06	738,60	8,48%	800,55	738,60	7,74%
B09	1100,82	1031,09	6,33%	1096,03	1031,09	5,92%
B10	884,39	664,97	24,81%	852,30	664,97	21,98%
Média	**1208,08**	**1078,17**	**13,17%**	**1156,22**	**1078,17**	**10,86%**

Fonte: tabela elaborada pelo autor

O gráfico da Figura 4.5 é elaboradao a partirdos dados da Tabela 4.7, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. Para a comparação dos valores obtidos para versão limitada da XFAB, a maior parte dos circuitos obteve valores muito próximos da versão sem restrições.

Figura 4.5 – Análise da potência dinâmica da *netlist* limitada XFAB e *netlist* otimizada do ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 180*nm*.



Fonte: figura elaborada pelo autor.

O *benchmark* B04 foi o circuito que apresentou a maior diferença de valores ultrapassando em 12% e mesmo assim a versão otimizada o ASTRAN alcançou um valor mímino de 5% de redução em potência dinâmica. A versão otimizada do ASTRAN obteve reduções em 90% dos circuitos, com alguns casos com valores muito próximo que variam entre 5% e 8%.

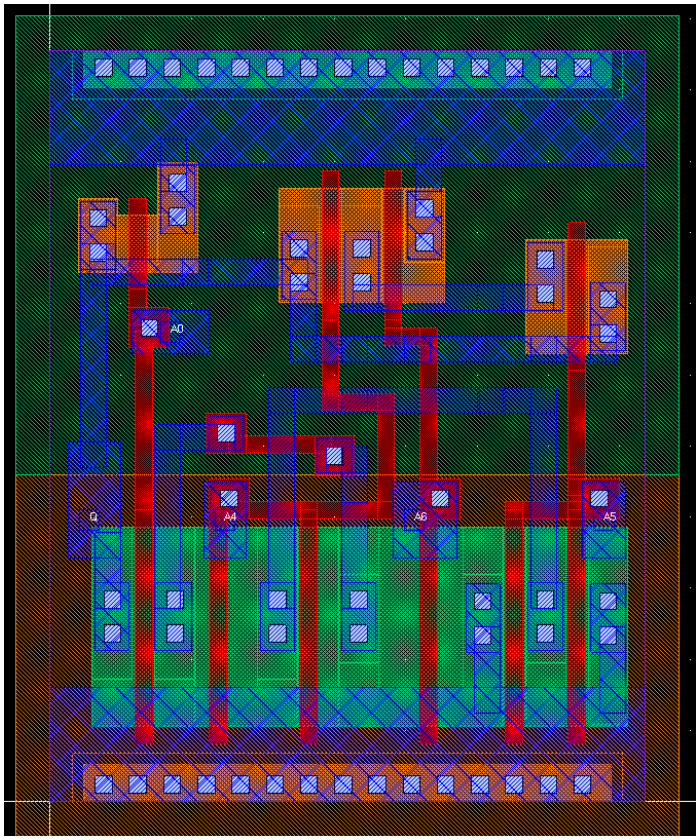
Os experimentos para a tecnologia de 180*nm* realizaram comparações entre a biblioteca *standard cell* XC018 da XFAB e as respectivas *netlists* (com e sem restrição de células) e a *netlist* otimizada. Os resultados obtidos mostraram reduções em número de transistores e em potência dinâmica e de *leakage*. Os valores obtidos em relação à area foram bem próximos dos valores comparados da XFAB. Em relação à *timing*, os dados mostraram que o impacto do uso de SCCG foi muito significativo, principalmente nos circuitos que apresentaram o maior número de transistores. As maiores reduções obtidas da *netlists* otimizada foram em relação à *netlist* com restrições da XFAB.

4.4.2 Resultados para 600nm

Os experimentos para a tecnologia de 600*nm* usaram a biblioteca *standard cell* XC06 da XFAB como base para as comparações de resultados. Durante a síntese do leiaute, a maior parte das células geradas não apresentou a necessidade de acréscimo de metal 2 para complementar o roteamento *intra*-cell do ASTRAN. A diferença de altura e largura das células para 600*nm* proporcionou um espaço maior para o roteamento e amenizou o tempo de geração dos leiaute das células complexas.

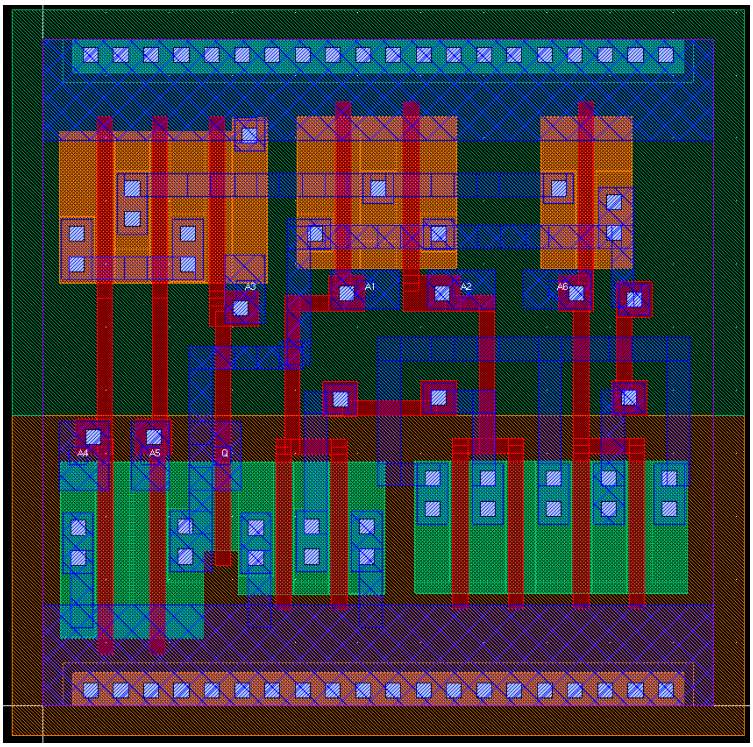
Os experimentos foram realizados para os circuitos B03, B06, B08 e B09 aplicando as mesmas restrições de 6 entradas e 4 transistores em série no máximo para as células complexas geradas na etapa de otimização. Os testes iniciais com a aplicação da técnica do uso de células complexas foram na tecnologia de 600*nm* para esses quatro circuitos, entretanto os valores obtidos para a potência de *leakage* eram extremamente pequenos. Devido a isso, este trabalho gerou o maior volume de resultados na tecnologia de 180*nm* que proporcionou valores de leakage um pouco mais significativos. Um exemplo de leiaute de uma célula complexa de expressão lógica !(A0*(A6+(A4*A5))) gerada para o circuito B01 é mostrada na Figura 4.6. Um outro exemplo de leiute para a expressão lógica !((A3+(A5*A4))+((A6*A2)*A1)) gerada para o circuito B06 é apresentada na Figura 4.7

Figura 4.6 – Leiaute de uma porta complexa gerada pelo ASTRAN para o circuito B01 em 600*nm*.



Fonte: figura elaborada pelo autor.

Figura 4.7 – Leiaute de uma porta complexa gerada pelo ASTRAN para o circuito B06 em 600*nm*.



Fonte: figura elaborada pelo autor.

A *netlits* otimizada é comparada com duas *netlists* geradas a partir da biblioteca XC06 da XFAB: a netlist sem restrições (total de 572 células) e a netlist com restrições que abrage um filtro de 204 células (somadores, multiplexadores, XOR/XNOR, AOI e OAI). A Tabela 4.8 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação ao número de transistores de cada circuito de *benchmark*. Os dados da versão otimizada em relação à versão com restrições da XFAB mostraram valores míminos com pouca variação de um circuito para outro em relação ao número de transistores. Ao comparar os dados com a versão sem restrição, todos os *bechmarks* alcançaram reduções acima de 11%.

O número de complexas atribuídas nas *netlists* otimizadas varia de 5 até 21 células, onde o circuito B09 é o que tem 21 células. Entretanto, o circuito que teve a maior porcentagem de redução foi o B03 com 17%, ou seja, ter o maior número de complexas na *netlist* não significa que o circuito sempre vai alcançar melhores reduções em número de transistores.

Tabela 4.8 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação ao número de transistores em 600*nm*..

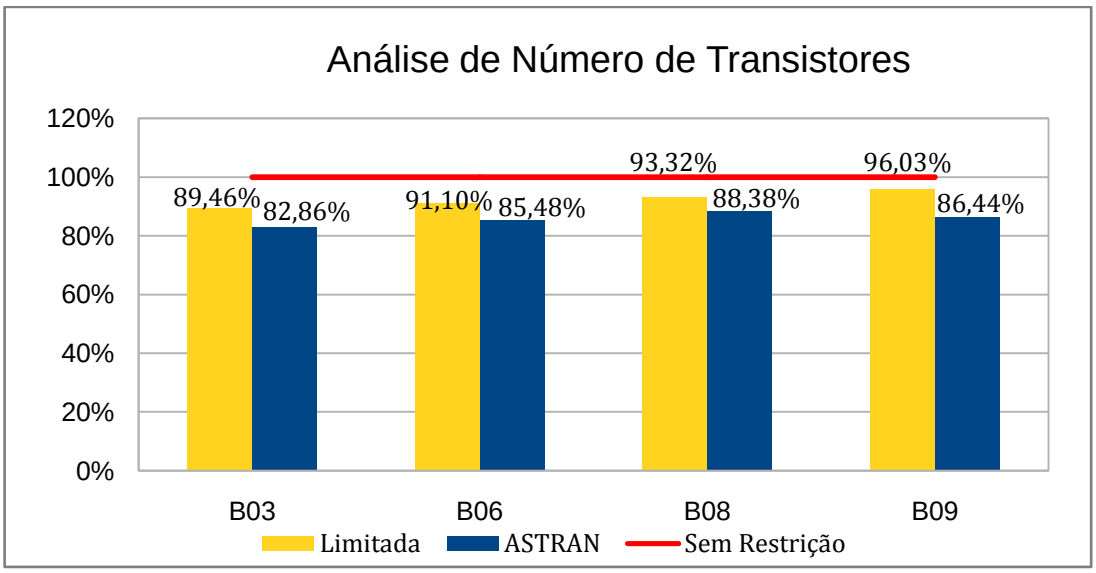
Número de Transistores						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B03	1409	1305	7,38%	1575	1305	17,14%
B06	389	365	6,17%	427	365	14,52%
B08	1132	1072	5,30%	1213	1072	11,62%
B09	1452	1307	9,99%	1512	1307	13,56%
Média	**1095,50**	**1012,25**	**7,21%**	**1181,75**	**1012,25**	**14,21%**

Fonte: tabela elaborada pelo autor.

O gráfico da Figura 4.8 reúne os dados da Tabela 4.8, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. Portanto, os valores da versão sem restrição são referenciados como uma linha vermelha de 100% e qualquer valor que alcançar redução é representando abaixo dessa linha vermelha e vice-versa.

Podemos observar que os resultados de todos os circuitos da versão com restrições alcançaram reduções em relação a versão sem restrições. A base da *netlist* otimizada é a versão com restrição da XFAB, ou seja, a tendência é que os circuitos da versão otimizada do ASTRAN também alcaçassem esses valores reduzidos.

Figura 4.8 – Análise do número de transistores da *netlist* limitada XFAB e *netlist* otimizada do ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 600*nm*.



Fonte: figura elaborada pelo autor.

A Tabela 4.9 mostra as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação à área dos *benchmarks*. Os dados da versão otimizada em relação à versão com restrições da XFAB valores apresentaram valores muitos próximos para todos os circuitos. Ao comparar com a versão sem restrições, a maioria dos circuitos alcançam reduções acima de 10%, exceto no B09 com 5%. O circuito B06 que tem o menor número de transistor foi o que obteve a melhor porcentagem de 14.6%.

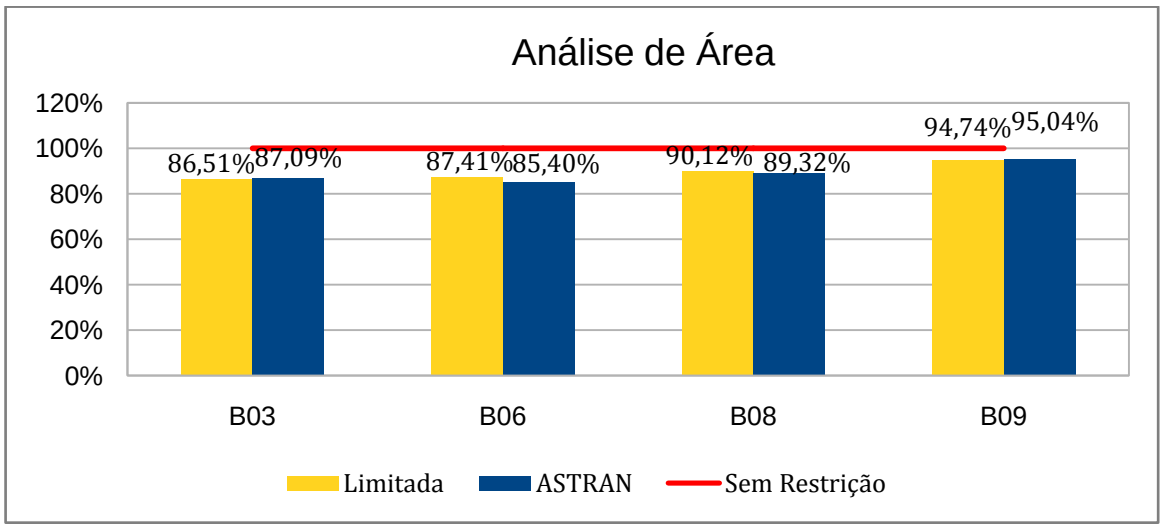
Tabela 4.9 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à área em 600*nm*..

	Area(um²)					
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B03	50640	50980	-0,67%	58536	50980	12,91%
B06	14755	14416	2,30%	16880	14416	14,60%
B08	40986	40624	0,88%	45480	40624	10,68%
B09	54709	54884	-0,32%	57746	54884	4,96%
Average	**40272,50**	**40226,00**	**0,55%**	**44660,50**	**40226,00**	**10,78%**

Fonte: tabela elaborada pelo autor.

O gráfico da Figura 4.9 reúne os dados da Tabela 4.9, usando como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. A análise de área nos mostra que todos os *bechmarks* da versão com restrição obtiveram reduções em relação à versão sem restrição. Podemos observar que a versão otimizada do ASTRAN obteve valores muito próximos da *netlist* limitada, alcançando também reduções na comparação.

Figura 4.9 – Análise para os resultados de área da *netlist* limitada XFAB e *netlist* otimizada com ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 600*nm*.



Fonte: figura elaborada pelo autor.

A Tabela 4.10 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB em relação à *timing* de cada circuito de *benchmark*, onde *timing* é referente ao *slack* do pior caminho reportado pelo RTL Compiler. Os dados da versão otimizada em relação à versão com restrições da XFAB não alcançam reduções, onde todos os circuitos têm violações acima de 31%. O menor circuito em termos de número de transistores, B06, obteve a maior violação de 49%.

Ao comparar com a versão sem restrições, as violações de *timing* dobram em todos os circuitos, com valores acima de 68%. As análises de número de transistores e área alcançaram maiores valores de redução nas comparações com a versão sem restrição. Entretanto, o acréscimo das 204 células que foram filtradas na versão com restrição causando um impacto muito significativo nos valores obtidos. O mapeamento com a inclusão dessas células se mostrou muito mais otimizado do que a *netlist* otimizada com as células complexas.

Tabela 4.10 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação ao *timing* em 600*nm*.

	***Timing* (ps)**					
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B03	-5645	-8240	-31,49%	-2090	-8240	-74,64%
B06	-1848	-3651	-49,38%	-1046	-3651	-71,35%
B08	-3558	-5441	-34,61%	-1728	-5441	-68,24%
B09	-5224	-7885	-33,75%	-2434	-7885	-69,13%
Average	**-4068,75**	**-6304,25**	**-37,31%**	**-1824,50**	**-6304,25**	**-70,84%**

Fonte: tabela elaborada pelo autor.

Para a potência de *leakage*, a Tabela 4.11 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB de cada circuito de *benchmark*. Os valores obtidos para 600*nm* são tão pequenos que podem ser considerados desprezíveis. As comparações com a versão com restrições apresentaram porcentagens em torno de 10% e com a versão sem restrições apresentaram porcentagens em torno de 20%. A proporção das porcentagens obtidas entre os circuitos têm valores muito próximos uns dos outros.

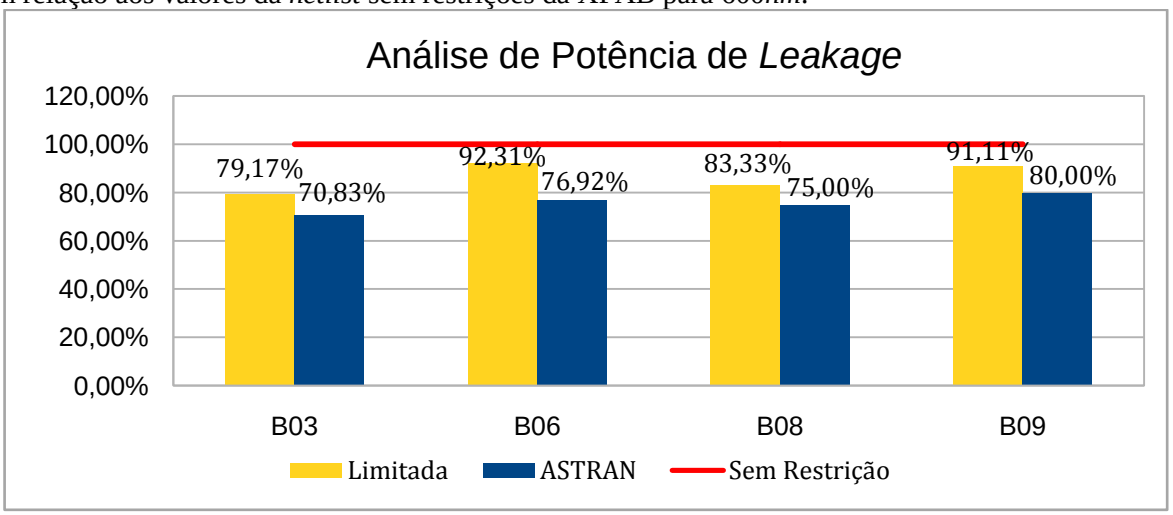
Tabela 4.11 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à potência de *leakage* em 600*nm*..

Potência de *Leakage* (nW)						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B03	0,038	0,034	10,53%	0,048	0,034	29,17%
B06	0,012	0,010	16,67%	0,013	0,010	23,08%
B08	0,030	0,027	10,00%	0,036	0,027	25,00%
B09	0,041	0,036	12,20%	0,045	0,036	20,00%
Average	**0,030**	**0,027**	**12,35%**	**0,036**	**0,027**	**24,31%**

Fonte: tabela elaborada pelo autor.

A Figura 4.10 usa os dados da Tabela 4.11 como base estabelecendo como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. A análise de potência de *leakage* nos mostra que todos os *bechmarks* da versão com restrição alcançam reduções, mesmo que os valores sejam extremamente baixos, assim como os dados da versão otimizada do ASTRAN. Não ocorreram casos em que um circuito obteve valores muito divergentes dos outros. A proporção dos valores obtidos em 600*nm* é bem menor do que os dados gerados na tecnologia de 180*nm*. Por exemplo, o circuito B06 obteve um valor de *leakage* 10 vezes menor do que o que foi obtido em 180*nm*.

Figura 4.10 – Análise potência de *leakage*da *netlist* limitada XFAB e *netlist* otimizada com ASTRAN em relação aos valores da *netlist* sem restrições da XFAB para 600*nm*.



Fonte: figura elaborada pelo autor.

Para a potência dinâmica, a Tabela 4.12 apresenta as comparações entra a *netlist* otimizada em relação às *netlists* da XFAB de cada circuito. Os dados da versão otimizada mostraram valores muito próximos em relação à versão com restrições da XFAB. O circuito B06 com menor quantidade de transistores foi o que apresentou o maior valor de porcentagem de 6,73%. Ao comparar com a versão sem restrições, todos os circuitos apresentaram reduções acima de 10%. Os circuitos B03 e B06 são os que apresentam a menor quantidade de complexas e alcançaram porncentagens acima de 20%.

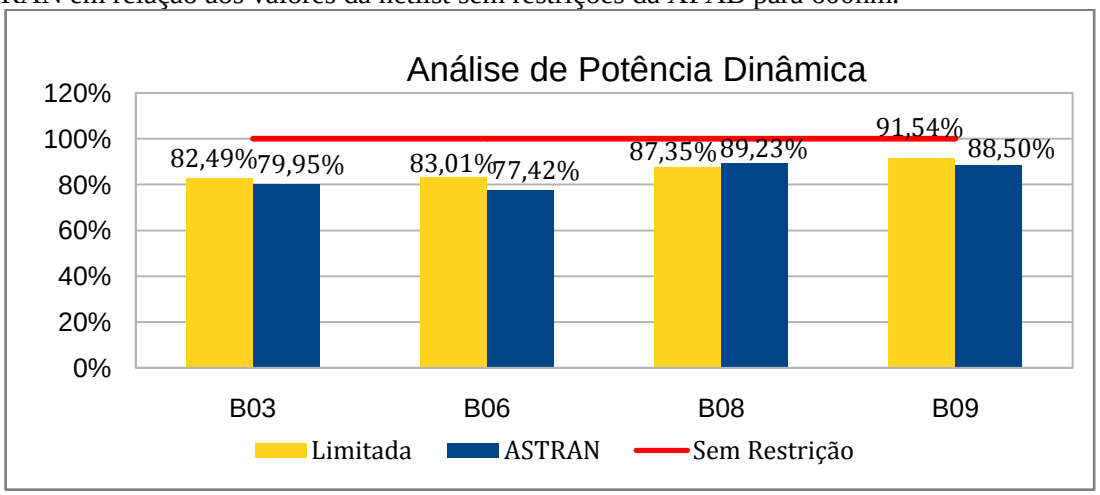
Tabela 4.12 – Comparação entre a *netlist* otimizada e as outras *netlists* da XFAB em relação à potência dinâmica em 600*nm*.

Potência Dinâmica (uW)						
Circuito	**XFAB Com Restrição**	**ASTRAN**	**Redução(%)**	**XFAB Sem Restrição**	**ASTRAN**	**Redução(%)**
B03	21322,02	20666,94	3,07%	25848,61	20666,94	20,05%
B06	6504,18	6066,34	6,73%	7835,89	6066,34	22,58%
B08	16082,94	16428,19	-2,15%	18411,49	16428,19	10,77%
B09	21476,20	20762,35	3,32%	23460,68	20762,35	11,50%
Average	**16346,33**	**15980,96**	**2,75%**	**18889,17**	**15980,96**	**16,23%**

Fonte: tabela elaborada pelo autor.

A partir dos dados da Tabela 4.12, a Figura 4.11 é elaborada tendo como referência a *netlist* sem restrições da XFAB como centro da comparação em vez da *netlist* otimizada. A análise de potência dinâmica nos mostra que todos os *bechmarks* da versão com restrição alcançam reduções assim como os dados da versão otimizada do ASTRAN. A versão com restrição usada como base da versão otimizada proporcionou maiores reduções nas comparações com a versão sem restrições que comtempla todas as células da biblioteca da XFAB.

Figura 4.11 – Análise para potência dinâmica da netlist limitada XFAB e netlist otimizada com ASTRAN em relação aos valores da netlist sem restrições da XFAB para 600nm.



Fonte: figura elaborada pelo autor.

Os experimentos para a tecnologia de 600*nm* usando a biblioteca *standard cell* XC06 da XFAB e as respectivas *netlists*, com e sem restrição de células, como base para comparação resultaram em reduções do número de transistores, área e potência dinâmica e de *leakage*. A avaliação dos dados em relação à *timing* mostraram que o impacto do uso de SCCG foi muito maior nessa tecnologia, abrangendo a mesma proporção em todos os circuitos avaliados. As maiores reduções obtidas da *netlists* otimizada foram em relação à a *netlist* sem restrições da XFAB.

5 CONCLUSÕES

Neste trabalho, realizou-se o uso da síntese automática do leiaute através da ferramenta ASTRAN aplicada em circuitos submetidos à técnica de otimização pelo uso de SCCG visando alcançar reduções em número de transistores. A flexibilidade da ferramenta ASTRAN permite gerar leiautes de quaisquer tamanhos ou redes de transistores.

A metodologia apresentada utiliza um fluxo de síntese composto pelas etapas de otimização da *netlist*, verificação/extração e caracterização da células. A etapa de otimização da *netlist* foi executada utilizando a ferramenta desenvolvida por (CONCEIÇÃO, 2016) que resulta uma *netlist* otimizada com as novas células complexas geradas já dimensionadas pelo método de esforço lógico, onde os leiautes dessas células são geradas pelo ASTRAN. As células complexas geradas pela ferramenta são limitadas em no máximo 4 transistores em série e 6 entradas, onde esses valores foram ajustados em função da viabilidade de geração dos leiautes e do dimensionamento dos mesmos.

Este trabalho realizou comparações em termos de área, número de transistores, consumo de energia e *timing* entre circuitos otimizados gerados pelo método ASTRAN e circuitos usando a metodologia *standard cell*. O trabalho adaptou as tecnologias de fabricação CMOS de 600*nm* e 180*nm* para a ferramenta ASTRAN a partir das informações dos *design kits* das bibliotecas *stantardcell* XC06 e XC018 da XFAB.

Os experimentos foram realizados nas tecnologias de 180*nm*e 600*nm* para um conjunto de circuitos de *bechmarks*do ITC'99. As comparações foram realizadas entre a *netlist* otimizada e duas *netlists* geradas para cada biblioteca da XFAB. Um *netlist* abrange todas as células da biblioteca e a outra tem uma restrição de células que são consideradas complexas (somadores, multiplexadores, XOR/XNOR, AOI e OAI). A *netlist* com restrições foi elaborada com a motivação de verificar se uma *netlist* com células complexas geradas exclusivamente para o circuito alvo do zero se tornaria mais benéfico em termos de redução do número de transistores.

Os experimentos para a tecnologia de 180*nm* realizaram comparações entre a biblioteca *standard cell* XC018 da XFAB e as respectivas *netlists* (com e sem restrição de células) e a *netlist* otimizada para os bechmarks de B01 a B10. Os resultados obtidos em relação à *netlist* com restrições da XFAB mostraram reduções nos melhores casos em número de transistores com até 15%, em potência dinâmica com até 24% e em potência de *leakage* com até 22%. Os valores obtidos em relação à area foram bem próximos dos valores

comparados da XFAB. Em relação à *timing*, os dados mostraram um impacto pelo uso de SCCG significativo, tendo os piores casos nos circuitos B04, B05 e B07 que tinham o maior número de transistores assim como o maior número de células complexas nas respectivas *netlist* otimizadas. A mesma proporção desse impacto também se mostra ao comparar os dados com a versão sem restrições da XFAB, onde os mesmo circuitos são os mais afetados.

Ao comparar com a *netlist* sem restrições, os valores obtidos em área, número de transistores e potencia de *leakage* excederam um pouco ou alcançaram valores muito próximos daqueles comparados. A potencia dinâmica apresentou dados com uma porcentagem de redução para alguns casos de até 20%. O B04 apresentou um valor muito excedente em número de transistores com 47,85%. O impacto no B04 e presente de forma amenizada nos outros circuitos tem relação com a *netlist* utilizada como base para a criação da *netlist* otimizada: a versão com restrição da XFAB. A biblioteca da XC018 da XFAB é composta de 828 células, onde a versão com restrições filtra 232 células e quando comparada com a versão sem restrição apresentam dados excedentes. A partir desses dados, a *netlist* otimizada alcança reduções em cima desses valores excedentes resultando poucos casos que alcancem redução com uma porcentagem bem menor para a versão sem restrições.

Os experimentos para a tecnologia de 600*nm* realizaram comparações entre a biblioteca *standard cell* XC018 da XFAB e as respectivas *netlists* (com e sem restrição de células) e a *netlist* otimizada para os *bechmarks* B03, B06, B08 e B09. Os resultados obtidos em relação à *netlist* sem restrições da XFAB mostraram reduções nos melhores casos em número de transistores com até 17%, em área com até 14%, em potência dinâmica com até 22%, em potência de *leakage* com até 29%. Em relação à *timing*, os dados mostraram um impacto pelo uso de SCCG mais significativo do que aqueles obtidos na tecnologia de 180*nm* na mesma proporção em todos os circuitos alcançando uma média de 70% excedente.

A proporção de impacto na versão com restrição fica em torno de 30% excedente, ou seja, o impacto é significativo mas redução na metade em relação aos obtidos da versão sem restrição. Os resultados da comparação com a versão com restrição mostraram valores muito próximos dos comparados em área e potencia dinâmica. Para numero de transistores e potência de *leakage*, as porcentagens alcançam no máximo 10% em reduções. A biblioteca XC06 da XFAB é composta de 572 células, onde a versão com restrições filtra 204 células e quando comparada com a versão sem restrição já apresenta reduções. A partir desses dados, a *netlist* otimizada alcança porcentagens de reduções maiores poris os valores de entrada já estão com uma parcela redução aplicada.

Os dados obtidos para a potência de leakage tanto para a tecnologia de 180*nm* como para 600*nm* são consideravelmente baixos se comparados aos valores obtidos da potência dinâmica, especialmente em 600*nm* que podem ser considerados desprezíveis. Entretanto, para tecnologias mais recentes o consumo de *leakage* proporciona uma influencia muito considerável nos projetos de circuitos.

Os experimentos mostraram que é possível alcançar reduções em número de transistores em diferentes tecnologiase assim alcançar reduções em consumo de energia e em alguns casos em área ao combinar o uso da ferramenta de síntese automática de leuiates ASTRAN com a aplicação de uma técnica de otimização diretamente na *netlist* dos circuitos. O uso de SCCG para otimizar a *netlist* de circuitos proporcionou essas reduções, entretanto o impacto em *timing* foi muito significativo em todos os circuitos do experimento para as duas tecnologias.

5.1 Trabalhos Futuros

Uma abordagem a ser explorada é o uso do ASTRAN para gerar células com lógica de transistores de passagem. As células complexas desse trabalho foram dimensionadas utlizando o método de esforço lógico, explorar outros métodos de dimensionamento de transistores é outra proposta futura.Este trabalho adaptou o ASTRAN para as tecnologias de 600*nm* e 180*nm* aumentando assim o suporte da ferramenta além de 350*nm*, 65*nm* e 45*nm*. Outra proposta é adaptar o ASTRAN para tecnologias abaixo de 45*nm* e aplicar as otimizações por uso de SCCG.

Os circuitos de *bechmark* otimizados utilizavam os leiautes das células complexas geradas pelo ASTRAN e as células da biblioteca standard cell da XFAB para as outras funções lógicas. O trabalho segue até a etapa de gerar os resultados através da ferramenta RTL Compiler. A partir desse ponto uma proposta é seguir até o final da síntese física através de uma ferramenta comercial ou acadêmica e assim realizar a avaliação em termos de potência dinâmica e de *leakage*, área e atraso. Outra proposta é que todas as células que os circuitos otimizados utilizem sejam geradas pelas ferramenta ASTRAN, realizando novas comparações com circuitos gerados pela metodologia *standard cell*.

REFERÊNCIAS

ANNE, Naresh. **Design and characterization of a standard cell library for the FREEPDK45 process**, M.S Thesis, Oklahoma State University, 2010.

ASTRAN. Disponível em: <http://www.inf.ufrgs.br/~amziesemerj/ icpd/>. Acesso em: abr, 2014.

CADENCE. **Virtuoso Layout Suite**. Disponível em : <http://www.cadence.com>. Acesso em : ago, 2014.

CADENCE. **VirtuosoLiberate.** Disponível em : <http://www.cadence.com>. Acesso em : ago, 2016.

CADENCE. **EncounterRTLCompiler.** Disponível em : <http://www.cadence.com>. Acesso em : ago, 2014.

CARRO, L. **Gerador Parametrizável de Partes Operativas CMOS**. 1989. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

CHEN, W.-K. **Computer Aided Design and Design Automation**. 3rd.ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.

CONCEIÇÃO, C.; POSSER, G.; REIS, R. Reducing the Number of Transistors with Gate Clustering. In: Latin American Symposium on Circuits and Systems, LASCAS, 2016. **Proceedings** . . . USA: IEEE, 2016. P.1–4.

DETJENS, E. et al. Technology mapping in MIS. In: ACM/IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1987, Los Alamitos, CA, USA. **Proceedings**. . . New York: ACM Press, 1987. p.116–119.

DUECK, G.; SCHEUER, T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. **J. Comput. Phys**., San Diego, CA, USA, v.90, n.1, p.161–175, 1990.

FERREIRA, F.; MORAES, F.; REIS, R. LASCA - Interconnect Parasitic Extraction Tool for Deep-Submicron IC Design. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN. **Anais**. . . [S.l.: s.n.], 2000.p.327–332.

FLACH, G.; HENTSCHKE, R.; REIS, R. Algorithms for improvement of RotDL router. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2004, Ijuí: Unijuí. **Anais**. . . [S.l.: s.n.], 2004.

FRAGOSO, J. **WTROPIC - um Gerador Automático de Macro Células CMOS Acessível via WWW**. 2001. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

GÜNTZEL, J. L. et al. A Novel Approach for ASIC Layout Generation. In: MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, Rio de Janeiro. **Anais**... [S.l.: s.n.], 1995.

GÜNTZEL, J. L. **Geração de Circuitos Utilizando Matrizes de Células Pré-difundidas**. 1993. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

GÜNTZEL, J. L.; RIBAS, R.; REIS, R. MARCELA - Uma Nova Abordagem para Pré-difundidos. In: SBMicro, Belo Horizonte. **Anais**. . . [S.l.: s.n.], 1991.p.534–543.

HENTSCHKE, R. **Algorithms for Wire Length Improvement of VLSI Circuits with Concern to Critical Paths**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS - Brasil.
HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

ITC'99. **Circuitos de Benchmark ITC'99**.Disponívelem: <http://www.cerc.utexas.edu/itc99-benchmarks/bench.html>. Acessoem :jan, 2015.

JOHANN, M.; REIS, R. MARTE –MazeRouTerEnvironment. In: SBMicro, Campinas, São Paulo. **Anais**. . . [S.l.: s.n.], 1993.p.XII.37 – XII.39.

LAZZARI, C. **Automatic Layout GenerationofStatic CMOS CircuitsTargetingDelayand Power Reduction**.2003.Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

LEFEBVRE, M.; MARPLE, D.; SECHEN, C. The future of custom cell generation inphysical synthesis. In: DESIGN AUTOMATION, DAC, 34., 1997, Anaheim, California,United States. **Proceedings**. . . New York: ACM Press, 1997. p.446–451.

LLOPART, X. Méthodologie de caractérisation dês cellules numériques. In: Ecole IN2P3 de microélectronique 2015, Paris, França. **Anais**. . . [S.l.: s.n.], 2015. Disponível em: < http://www.in2p3.fr/actions/formation/microelectronique15/Support_microelc-15.html >. Acesso em : Ago. 2016.

LUBASZEWSKI, M. **Geração Automática de Lógica Aleatória Utilizando a Metodologia TRANCA**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MCMURCHIE, L.; EBELING, C. PathFinder: a negotiation-based performance- driven router for fpgas. In: ACM INTERNATIONAL SYMPOSIUM ON FIELD-PROGRAMMABLE GATE ARRAYS, FPGA, 3., 1995, Monterey, California, United States.**Proceedings**. . . New York: ACM Press, 1995. p.111–117.

MEINHARDT, C. **Geração de Leiautes Regulares Baseados em Matrizes de Células. 2006. Dissertação** (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

MENTOR GRAPHICS. **Calibre** . Disponível em: <http://www.mentor.com>. Acesso em : set, 2014.

MORAES, F. G. TRAGO - **Síntese Automática de Leiaute para Circuitos em Lógica Aleatória**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MORAES, F. **SynthèseTopologique de Macro-CellulesenTechnologie.** CMOS. 1994. Doutor em Ciência da Computação — Université Montpellier II, França.

MORAES, F.; REIS, R.; LIMA, F.An Efficient Layout Style for Three-Metal CMOS Macro-Cells. In: VLSI'97, Gramado. **Anais**. . . [S.l.: s.n.], 1997.p.415–426.

NANGATE Library Creator. Disponível em: <http://www.nangate.com/>. Acessoem: jul, 2017.

NANGATE. Nangate Open Cell Library v1.0, FreePDK v1.3 Package.Disponívelem: <http://www.nangate.com>. Acessoem : ago, 2014.

NISHIZAWA, S.; ISHIHARA, T.; ONODERA, H.; Layout Generator with Flexible Grid Assignment for Area Efficient Standard Cell. **IPSJ Transactions on System LSI Design Methodology (T-SLDM),** Vol.8 , pp. 131-135, 2015. Disponível em : <https://www.jstage.jst.go.jp/article/ipsjtsldm/8/0/8_131/_article/references>. Acesso em : jul, 2017.

NCSU-FreePDK45-1.4 design kit. Disponível em: < http://www.eda.ncsu.edu/wiki/ FreePDK45:Contents >. Acesso em : ago, 2014.

POSSANI, V.; MARQUES, F.; ROSA JUNIOR, L. DA; CALLEGARO, V.; REIS, A.; RIBAS, R. Transistor-level optimization of CMOS complex gates. In: Latin American Symposium on Circuits and Systems, LASCAS, 2013. **Proceedings** . . . USA: IEEE, 2013. P.1–4.

POSSER, G. **Dimensionamento de Portas Logicas usando programação geometrica**. 2011. Dissertacao (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Computação, UFRGS, Porto Alegre, RS.

POSSER, G. et al., ZIESEMER, A., GUIMARESJr,D., WILKE G., REIS, R., A Study on Layout Quality of Automatic Generated Cells. In: ICECS 2010, Atenas, Grécia. **Anais**. . . [S.l.: s.n.], 2010.

REIS, A. I. **Geração de Células Transparentes**. 1993. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

PROGENESYS. Disponível em: <http://www.arm.com>. Acesso em: jul, 2017.

REIS, A. I.; REIS, R. TRAMOII - Proposta para um Gerador de Leiaute Baseado em Células para Circuitos CMOS Digitais com Dois Níveis de Metal. In: SIMPÓSIO BRASILEIRO DE CONCEPÇAO DE CIRCUITOS INTEGRADOS, Jaguariúna. **Anais**. . . [S.l.: s.n.], 1991.p.90–99.

REIS, R.A New Standard Cell CAD Methodology. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, Portland, Oregon. **Anais**... [S.1.:s.n.], 1987. p.385-388.

REIS, R.; GOMES, R.; LUBASZEWSKI, M. An Efficient Design Methodology for Standard Cell Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, Espoo. **Anais**. . . [S.l.: s.n.], 1988.p.1213–1216.

SANTOS, G.; JOHANN, M.; REIS, R. Channel Based Routing in Channel-less Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2006. **Proceedings**. . . USA: IEEE, 2006. p.4 pp.–340.

SAWICKI, S. et al. Studying the influence of I/O pads placement on wirelength and 3D- Vias of VLSI 3D integrated circuits. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2007, Porto Alegre. **Anais**. . . [S.l.: s.n.], 2007.p.89–92.

SENTOVICH, E.;SINGH, K.;LAVAGNO, L.;MOON, C.;MURGAI,R.;SALDANHA, A.;SAVOJ,H.;STEPHAN, P.;BRAYTON, R.;SANGIOVANNI-VINCENTELLI, A. L. “Sis: A system for sequential circuit synthesis,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M92/41, 1992. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/2010.html>. Acessoem: abr, 2017.

SHARMA, R.; **Characterization and Modeling of Digital Circuits.** [S.l.]: CreateSpace Independent Publishing Platform, 2015.

SMANIOTTO, G.; MACHADO, J. ; MOREIRA, M.; ZIESEMER, A.; MARQUES, F.; JUNIOR, L.Optimizing cell area by applying an alternative transistor folding technique in an open source physical synthesis CAD tool. In: Latin American Symposium on Circuits and Systems, LASCAS, 2016. Proceedings . . . USA: IEEE, 2016. P.1–4.

SUSIN, A. A. **Etudedes Parties Operatives a Elements Modulaires pour Processeurs Monolithiques**. 1981. Tese (Doutorado em Engenharia) — Université Grenoble, Greno- ble, France.

SUTHERLAND, I.; SPROULL, B.; HARRIS, D. LogicalEffort: DesigningFast CMOS Circuits. San Francisco, CA, USA: Morgan KaufmannPublishers Inc., 1999.

WILKE, G. et al. Finding the Critical Delay of Combinational Blocks by Floating Vector Simulation and Path Tracing. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, SBCCI, 15, Porto Alegre, RS. **Anais**. . . [S.l.: s.n.], 2002. p.277–282.

ZIESEMER, A. M. **Geração Automática de Partes Operativas de Circuitos VLSI**. 2007. Dissertação (Mestrado em Ciência da Computação) — Programa de Pós-Graduação em Computação, UFRGS, Porto Alegre, RS

ZIESEMER, A. M. **Síntese automática do leiaute de redes de transistores**. 2014. Tese (Doutorado em Ciência da Computação) — Programa de Pós Graduação em Microeletrônica, UFRGS, Porto Alegre, RS

ZIESEMER, A.; REIS, R.; MOREIRA, M.T.; ARENDT, M.E.; CALAZANS, N.L.V. Automatic layout synthesis with ASTRAN applied to asynchronous cells. In: IEEE 5th Latin American Symposium on Circuits and Systems, LASCAS, 2014.

X-FAB, XC06 design kit. Disponível em: <http://www.xfab.com/>. Acesso em : dez, 2015.

X-FAB, XC018 design kit. Disponível em: <http://www.xfab.com/>. Acessoem :jan , 2017.

APÊNDICE A ARQUIVOS DE CONFIGURAÇÃO DO ASTRAN

A.1 Regras de projeto da ferramenta ASTRAN para a tecnologia de 180*nm*

1.	TECHNAME	tech180	
2.	MINSTEP	0.005	
3.	VDD	1.8	
4.	MLAYERS	4	
5.	S1P1P1	0.25	* Minimum POLY1 spacing
6.	S2P1P1	0.25	* Minimum GATE to GATE spacing(*)
7.	S2P1P1_1	0.25	* Gate length for rule S2P1P1.2
8.	S2P1P1_2	0.25	* GATE to GATE spacing if L>S2P1P1.1
9.	S3P1P1	0.25	* Minimum POLY1 spacing in a dense line end(*)
10.	S1DFP1	0.10	* Minimum POLY1 spacing to DIFF
11.	S2DFP1	0.10	* Minimum POLY1 spacing to DIFF if W<S3DFP1(*)
12.	S3DFP1	0.10	* Transistor Width for rule S2DFP1(*)
13.	E1P1DF	0.22	* Minimum POLY1 extension of GATE(*)
14.	E2P1DF	0.22	* Minimum POLY1 extension of GATE in L shape diff(*)
15.	E1DFP1	0.32	* Minimum DIFF extension of GATE
16.	E1DNP1	0.32	*Minimum NDIFF extension of GATE when butted to PDIFF
17.	E1DDP1	0.32	*Minimum PDIFF extension of GATE when butted to NDIFF
18.	R1P1	0	* Maximum ratio of POLY area to touched GATE area
19.	S1DFP2	0	* Minimum POLY2 spacing to DIFF
20.	S1P1P2	0	* Minimum POLY1 spacing to POLY2
21.	W2CT	0.22	* Fixed CONT size
22.	E1M1CT	0.06	* Minimum MET1 enclosure of CONT
23.	E2M1CT	0.06	* Minimum MET1 extension on CONT on at least 2 opposite sides
24.	E3M1CT	0.06	* Minimum MET1 enclosure of CONT on all sides if E2M1CT is not fulfilled(*)
25.	E1DFCT	0.10	* Minimum DIFF enclosure of CONT
26.	E2DFCT	0.10	* Minimum DIFF enclosure of CONT (min. 2 oposite sides)(*)
27.	E1P1CT	0.10	* Minimum POLY1 enclosure of CONT
28.	E2P1CT	0.10	* Minimum POLY1 extension on CONT on all sides at least 2 opposite sides
29.	E3P1CT	0.10	* Minimum POLY1 enclosure of CONT if E2P1CT is not

	fulfilled		
30.	E1P2CT	0	* Minimum POLY2 enclosure of CONT
31.	S1CTP1	0.16	* Minimum DIFFCON spacing of GATE
32.	S1CTDP	0.20	* Minimum NDIFFCON spacing of PDIFF
33.	S1CTDN	0.20	* Minimum PDIFFCON spacing of NDIFF
34.	S1CTDF	0.20	* Minimum POLY1CON spacing of DIFF
35.	S1CTP2	0	* Minimum POLY1CON spacing of POLY2
36.	R1M1	0	* Minimum ratio of MET1 area to die area
37.	R2M1	0	* Maximum ratio of MET1 area to connected GATE and
	CPOLY area		
38.	W2VI	0.26	* Fixed VIA size
39.	E1M1VI	0.01	* Minimum MET1 enclosure of VIA
40.	E2M1VI	0.06	* Minimum MET1 enclosure of VIA on at least 2 opposite
	sides(*)		
41.	E1M2VI	0.01	* Minimum MET2 enclosure of VIA
42.	S1M2M1	0	* Minimum MET2 spacing to MET1 over CPOLY
43.	S1M1M2	0	* Minimum MET1 spacing to MET2 over CPOLY
44.	R2M2	0	* Maximum ratio of MET2 area to connected GATE and
	CPOLY area		
45.	W2P1	0.18	* Minimum GATE length
46.	W2DF	0.22	* Minimum DIFF width
47.	W2V2	0.26	* Fixed VIA2 size
48.	S1M1M1	0.23	* Minimum MET1 spacing to MET1
49.	S2M1M1	0.60	* Minimum Wide MET1 spacing to MET1
50.	S3M1M1	0.23	* Minimum MET1 spacing in a dense line end(*)
51.	E1M2V2	0.01	* Minimum MET2 enclose of VIA2
52.	S1CTCT	0.25	* Minumum aligned CONT spacing
53.	S2CTCT	0.25	* Minumum shorted CONT spacing
54.	S3CTCT	0.25	* Minumum misaligned CONT spacing
55.	E1INDF	0.18	* Minimum NPLUS extension of DIFF
56.	E1IPDF	0.18	* Minimum PPLUS extension of DIFF(*)
57.	E1WNDP	0.43	* Minimum NTUB enclosure of PDIFF
58.	S1M2M2	0.28	* Minimum MET2 spacing to MET2
59.	S2M2M2	0.60	* Minimum MET2 spacing to WIDE_MET2
60.	A1M1	0.202	* Minimum MET1 area
61.	W1M1	0.23	* Minimum MET1 width

62.	W1M2	0.28	* Minimum MET2 width
63.	W1M3	0.28	* Minimum MET3 width
64.	W1M4	0.28	* Minimum MET4 width
65.	W1M5	0.28	* Minimum MET5 width
66.	W1M6	0	* Minimum MET6 width
67.	W1M7	0	* Minimum MET7 width
68.	W1M8	0	* Minimum MET8 width
69.	W1M9	0	* Minimum MET9 width
70.	W1M10	0	* Minimum MET10 width
71.	A1DF	0.202	* Minimum DIFF area(*)
72.	S1DFDF	0.28	* Minimum DIFF spacing to DIFF
73.	S2DFDF	0.28	* Minimum U shape DIFF spacing
74.	S1DNWN	0.43	* Minimum NDIFF spacing to NTUB
75.	S1VIVI	0.26	* Minimum VIA spacingto VIA
76.	S1V2V2	0.26	* Minimum VIA2 spacing to VIA2
77.	S1M3M3	0.28	* Minimum MET3 spacing to MET3
78.	E1M3V2	0.01	* Minimum MET3 enclosure of VIA2
79.	E1M3V3	0.01	* Minimum MET3 enclosure of VIA3
80.	W2V3	0.26	* Fixed VIA3 size
81.	E1M4V3	0.01	* Minimum MET4 enclosure of VIA3
82.	S1M4M4	0.28	* Minimum MET4 spacing to MET4
83.	W2V4	0.26	* Fixed VIA4 size
84.	E1M4V4	0.01	* Minimum MET4 enclosure of VIA4
85.	E1M5V4	0.01	* Minimum MET5 enclosure of VIA4
86.	S1M5M5	0.28	* Minimum MET5 spacing to MET5
87.	W2V5	0	* Fixed VIA5 size
88.	S1IPIP	0.44	* PPLUS spacing to PPLUS(pimp)
89.	S1ININ	0.44	* NPLUS spacing to NPLUS(nimp)
90.	W2V6	0	* Fixed VIA6 size
91.	W2V7	0	* Fixed VIA7 size
92.	W2V8	0	* Fixed VIA8 size
93.	W2V9	0	* Fixed VIA9 size
94.	W2V10	0	* Fixed VIA10 size

APÊNDICE B ARQUIVOS SPICE

B.1 Exemplo de uma netlist extraída da célula INX1 usada na etapa da caracterização

```
.SUBCKT INX1 vdd gnd A Q
MavD2_1   Q#3   A#5   gnd#1 gnd#1      ne     L=0.18U     W=0.66U
+ AD=0.3168P     AS=0.4664 P       PD=2.28U   PS=3.88U
+ wtot=6.6e-07 nrs=0.757576 nrd=0.757576 ng=1 mtot=1 m=1
MavD3_1   Q#1   A#6   vdd#3 vdd#1      pe     L=0.1845U  W=1.6865U
+ AD=0.6288P     AS=0.648P  PD=4.10627U      PS=4.10627U
+ wtot=1.68627e-06 nrs=0.296512 nrd=0.296512 ng=1 mtot=1 m=1
*
RESISTOR AND CAP/DIODE CARDS
*
Re2   vdd   vdd#1   5.5000
Re1   A#6   A       43.7223
Rd8   vdd   vdd#3   11.3791
Rd9   A#5   A       28.7111
Rk2   gnd#2 gnd     5.111E-02
Rk3   Q     Q#3     11.0000
Rj1   gnd#2 gnd#1   11.0000
Rl2   gnd   gnd#1   5.5420
Rl4   Q     Q#1     11.6318
*
CAPACITOR CARDS
*
C1    vdd#3 A#6   2.23679E-16
C2    A#6   Q#1   1.84306E-16
C3    vdd#1 Q#1   7.63451E-17
C4    vdd#1 Q#3   1.32582E-18
C5    vdd#1 gnd   2.06107E-18
C6    gnd   vdd   1.03423E-18
C7    A#6   gnd#1 5.04798E-19
C8    Q#3   gnd#1 1.17671E-16
C9    Q#3   A#5   2.06E-17
```

```
C10   gnd#1 A#5   1.62276E-16
C11   vdd#3 A     4.36253E-17
C12   vdd#1 A     7.49632E-17
C13   Q#1   A     7.56031E-17
C14   Q#3   A     4.69695E-17
C15   gnd   A     3.12789E-18
C16   gnd#1 A     8.43383E-17
*
.ENDS INX1
```

APÊNDICE C ARQUIVO USADO NA IMPORTAÇÃO DO GDS PARA A FERRAMENTA VIRTUOSO

C.1 Arquivo de *layer map* para 180*nm* e 600*nm*

```
# layer      purpose  layer  | datatyp |
#---------+---------+--------+---------+
NGD         drawing     2        0
CCIMP       drawing     3        0
DNWELL      drawing     4        0
NWELL       drawing     5        0
SNWELL      drawing     6        0
PWELL       drawing     7        0
#FIMP       drawing     8        0
POLY0       drawing     9        0
DIFF        drawing     10       0
PGD         drawing     13       0
PMVVT       drawing     14       0
NDIMP       drawing     15       0
TUOX        drawing     16       0
TUIMP       drawing     17       0
OPTO        drawing     19       0
POLY1       drawing     20       0
FD          drawing     22       0
NIMP        drawing     23       0
PIMP        drawing     24       0
```

22. STACK drawing 26 0
23. SAS drawing 27 0
24. LINC drawing 29 0
25. NOPLDD drawing 30 0
26. ESD drawing 31 0
27. #P5VVT drawing 33 0
28. CONT drawing 34 0
29. MET1 drawing 35 0
30. VIA drawing 36 0
31. MET2 drawing 37 0
32. #VIA2 drawing 38 0
33. #MET3 drawing 39 0
34. PAD drawing 40 0
35. HV drawing 41 0
36. #SFCDEF drawing 42 0
37. ZAPDEF drawing 43 0
38. #ONO drawing 44 0
39. FLARR drawing 45 0
40. EEARR drawing 46 0
41. DIODEF drawing 47 0
42. MV drawing 48 0
43. RESDEF drawing 49 0
44. RESTRM drawing 50 0
45. BIPO drawing 51 0
46. HALL drawing 52 0
47. NOFIMP drawing 53 0
48. CETXT drawing 54 0
49. CAPDEF drawing 55 0
50. BLTXT drawing 56 0
51. M1HOLE drawing 57 0
52. M2HOLE drawing 58 0
53. PADTXT drawing 59 0
54. #M3HOLE drawing 61 0
55. LOCKED drawing 65 0
56. M1RDEF drawing 66 0
57. M2RDEF drawing 67 0

```
58.  #M3RDEF  drawing  68  0
59.  PHODEF  drawing  70  0
60.  NOPIM  drawing  74  0
61.  #NOSAPW  drawing  75  0
62.  NOSLD  drawing  76  0
63.  #VIAL  drawing  80  0
64.  #METL  drawing  81  0
65.  #MLHOLE  drawing  82  0
66.  NOFILLM  drawing  89  0
67.  M1TXT  drawing  91  0
68.  M2TXT  drawing  92  0
69.  prBoundarydrawing  190  0
70.  SUBCUT  drawing  191  0
71.  textdrawing  230  0
```

APÊNDICE D ARQUIVOS TCL

D.1 Exemplo de *Template* no formato TCL usado na ferramenta de caracterização Virtuoso LIBERATE

```
1.  set_varslew_lower_rise 0.1
2.  set_varslew_lower_fall 0.1
3.  set_varslew_upper_rise 0.9
4.  set_varslew_upper_fall 0.9
5.  set_varmeasure_slew_lower_rise 0.1
6.  set_varmeasure_slew_lower_fall 0.1
7.  set_varmeasure_slew_upper_rise 0.9
8.  set_varmeasure_slew_upper_fall 0.9
9.  set_vardelay_inp_rise 0.5
10. set_vardelay_inp_fall 0.5
11. set_vardelay_out_rise 0.5
12. set_vardelay_out_fall 0.5
13. set_vardef_arc_msg_level 0
14. set_varmax_transition 1.5e-09
15. set_varmin_transition 0.02e-09
16. set_varmin_output_cap 0.0010e-12
```

```
17.     set cells { \
18.     GATE_0X1 \
19.     GATE_1X1 \
20.     GATE_2X1 \
21.     }
22.     define_template -typedelay \
23.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
24.     -index_2 {0.0010 0.0309 0.0619 0.1237 0.2475 0.4950 0.9900}\
25.     delay_template_7x7_X1
26.     define_template -typedelay \
27.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
28.     -index_2 {0.0010 0.0309 0.0619 0.1237 0.2475 0.4950 0.9900}\
29.     delay_template_7x7_X2
30.     define_template -typedelay \
31.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
32.     -index_2 {0.0010 0.0464 0.0928 0.1856 0.3712 0.7425 1.4850}\
33.     delay_template_7x7_X3
34.     define_template -typepower \
35.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
36.     -index_2 {0.0010 0.0309 0.0619 0.1237 0.2475 0.4950 0.9900}\
37.     power_template_7x7_X1
38.     define_template -typepower \
39.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
40.     -index_2 {0.0010 0.0309 0.0619 0.1237 0.2475 0.4950 0.9900}\
41.     power_template_7x7_X2
42.     define_template -typepower \
43.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
44.     -index_2 {0.0010 0.0464 0.0928 0.1856 0.3712 0.7425 1.4850}\
45.     power_template_7x7_X3
46.     define_template -typeconstraint \
47.     -index_1 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
48.     -index_2 {0.02 0.1 0.4 0.9 1.5 2.2 3}\
49.     constraint_template_7x7_X1
50.     define_cell \
51.     -input { A0 A1 A2 A5 A6 A7 } \
```

```
52.     -output { Q } \
53.     -pinlist { A0 A1 A2 A5 A6 A7 Q } \
54.     -delay delay_template_7x7_X2 \
55.     -power power_template_7x7_X2 \
56.     GATE_0X1
57.     define_cell \
58.     -input { A0 A1 A2 A3 A4 A5 } \
59.     -output { Q } \
60.     -pinlist { A0 A1 A2 A3 A4 A5 Q } \
61.     -delay delay_template_7x7_X2 \
62.     -power power_template_7x7_X2 \
63.     GATE_1X1
64.     define_cell \
65.     -input { A2 A5 A6 A7 A8 A9 } \
66.     -output { Q } \
67.     -pinlist { A2 A5 A6 A7 A8 A9 Q } \
68.     -delay delay_template_7x7_X2 \
69.     -power power_template_7x7_X2 \
70.     GATE_2X1
```

D.2 Exemplo de *Script* de execução no formato TCL usado na ferramenta de caracterização Virtuoso LIBERATE

```
1.  # EXEMPLO para o bechmark B02_PL01 do ITC99
2.
3.  # Set the run directory.
4.  setrundir [pwd]
5.
6.  # Create the directories Liberate will write to.
7.  execmkdir -p ${rundir}/LDB
8.  execmkdir -p ${rundir}/LIBRARY
9.  execmkdir -p ${rundir}/DATASHEET
10. execmkdir -p ${rundir}/VERILOG
11.
12. ## Load Spice models and subckts ## (Modificado by Gisell)
13. setspicefiles ${rundir}/MODELS/xc180_tech.scs
```

```
14.
15.     ## Load template information for each cell ## (Verificado by Gisell)
16.     source ${rundir}/TEMPLATE/template_st.tcl
17.
18.     set_varextsim_model_include "${rundir}/MODELS/xc180_tech.scs"
19.
20.     ## Define os tipos, elementos e pinos dos transistores e diodos (Verificado by Gisell)
21.     define_leafcell -type nmos -pin_position { 0 1 2 3 } -element ne
22.     define_leafcell -type pmos -pin_position { 0 1 2 3 } -element pe
23.     define_leafcell -type diode -pjperi -pin_position { 0 1 } -element dp
24.     define_leafcell -type diode -pjperi -pin_position { 0 1 } -element dn
25.     define_leafcell -type diode -pjperi -pin_position { 0 1 } -element dnw
26.
27.     set_varextsim_flatten_netlist 0
28.     set_varspectre_use_char_opt_license 0
29.     set_units -capacitance 1pf
30.     set_varsim_estimate_duration 0
31.     set_varsim_duration 1e-7
32.     set_varmax_leakage_vector 5000
33.
34.     #set_operating_condition : temperature and voltages (Modificado by Gisell)
35.     set_operating_condition -name _typ_1_8V_25C -voltage 1.80 -temp 25
36.
37.     foreach cell $cells {
38.     lappendspicefiles ${rundir}/netlists/${cell}.sp
39.     }
40.     read_spice -format spice $spicefiles
41.
42.     ## To specify the cell netlists ## (Modificado by Gisell)
43.     #read_spice -format spice $spicefiles
44.
45.     set_varextsim_deck_dir "${rundir}/LOG/"
46.     ##set_varextsim_save_failed "${rundir}/Falha/"
47.
48.     ## Characterize the library for NLDM (default), CCS and ECSM timing (Revisado by Gisell)
49.     char_library -cells $cells -extsimspectre
```

```
50.
51.     #Create Liberty (Modificado by Gisell)
52.     write_library -overwrite ${rundir}/LIBRARY/B02_complex.lib
53.
54.     ## Save characterization database for post-processing ## (Modificado by Gisell)
55.     write_ldb ${rundir}/LDB/B02_complex.ldb
```

APÊNDICE E ARQUIVOS DE CONFIGURAÇÃO DO RTL COMPILER

E.1 Arquivo de *constraints*

```
1.  # define o clock de 400MHz
2.  create_clock -domain clock_domain -period 2.5 -name clock -waveform {0.0 1.25}
    [get_port "clock"]
3.
4.  # define o tempo de subida e descida do clock em 100ps
5.  set_clock_latency 1.0 [all_clocks]
6.  setclock_ports [get_port {clock}]
7.
8.  set_clock_transition -rise 0.1 [get_clocks "clock"]
9.  set_clock_transition -fall 0.1 [get_clocks "clock"]
10. # modela o atraso das entradas dos modulos
11. set_input_delay -clock clock 0.2 [all_inputs]
12. set_output_delay -clock clock 0.2 [all_outputs]
13.
14. # modela a incerteza do clock em 200ps para setup e 100ps para hold
15. set_clock_uncertainty 0.2 -setup [ all_clocks ]
16. set_clock_uncertainty 0.2 -hold [ all_clocks ]
```

E.2 *Script* em formato TCL

```
1.  ################################################################
2.  ## Library setup
3.  ################################################################
```

```
4.  set DESIGN b14
5.  set_attributelib_search_path                                    {
    /home/gme/gbmoura/xfab/virtuoso/RTL_Compiler/Xfab_caracterizado }
6.  set_attribute library { D_Cells_xfab.lib }
7.  set_attributeinformation_level 9
8.  set_attributehdl_error_on_latch true
9.  ################################################################
10. ## Load Design
11. ################################################################
12. # Read HDL and elaborate
13. read_hdl -vhdl  b14.vhd
14.
15. elaborate $DESIGN
16. puts "Runtime & Memory after 'read_hdl'"
17. ################################################################
18. ## Constraints Setup
19. ################################################################
20. # Read SDC file
21. #clockchamadocomo clock
22. read_sdcconstraints.sdc
23.
24. puts "Runtime & Memory after 'Read SDC file'"
25. ################################################################
    ####### Synthesizing to generic
26. ################################################################
27. #Synthesize
28. synthesize -to_generic
29. #synthesize -to_generic -eff $SYN_EFF
30.
31. #timestat GENERIC
32. #reportdatapath> $_REPORTS_PATH/${DESIGN}_datapath_generic.rpt
33. puts "Runtime & Memory after 'synthesize -to_generic'"
34.
```

```
35. ##########################################################################
    ####### write Encounter file set (verilog, SDC, config, etc.)
36. ##########################################################################
37. #write_encounter design -basename<path & base filename> -lef<lef_file(s)>
38.
39. report power
40. report area
41. report timing
42.
43. write_hdl> ./${DESIGN}_m.hvsyn
44. write_sdc> ./${DESIGN}_m.sdc
45. write_script> ./${DESIGN}_m.script
46. ##########################################################################
    ### write_do_lec
47. ### generate script for LEC formal verification tool
48. ##########################################################################

49. write_do_lec -no_exit -golden_design ./${DESIGN}_global_mapped.v -revised_design
    ./${DESIGN}_m.hvsyn -logfile  ./globalmap2final.lec.log > ./globalmap2final.lec.do
50.
51. ##Uncomment if the RTL is to be compared with the final netlist..
52. write_do_lec -no_exit -revised_design ./${DESIGN}_m.hvsyn -logfile ./rtl2final.lec.log
    > ./rtl2final.lec.do
53.
54. write_encounter design $DESIGN -basenameRTL_Compiler/b01
55.
56. puts "Final Runtime & Memory."
57. timestat FINAL
58. puts "============================"
59. puts "Synthesis Finished ....."
60. puts "============================"
61. ##quit
```

E.3 Exemplo de *netlist* estrutural sem otimização do *bechmark* B01 do ITC"99

```
1.  // Generated by Cadence Encounter(R) RTL Compiler RC12.24 - v12.20-s034_1
2.
3.  module b01(line1, line2, reset, outp, overflw, clock);
4.  input line1, line2, reset, clock;
5.  output outp, overflw;
6.  wire line1, line2, reset, clock;
7.  wireoutp, overflw;
8.  wire [2:0] stato;
9.  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
10. wire n_8, n_9, n_10, n_11, n_12, n_13, n_14, n_15;
11. wire n_16, n_17, n_18, n_19, n_20, n_21, n_22, n_23;
12. wire n_24, n_25, n_26, n_27, n_28, n_29, n_30, n_31;
13. wire n_32, n_33, n_34, n_35, n_36, n_37, n_38, n_39;
14. DFRRQX1 \stato_reg[0] (.RN (n_39), .C (clock), .D (n_38), .Q
15. (stato[0]));
16. DFRRQX1 \stato_reg[1] (.RN (n_39), .C (clock), .D (n_37), .Q
17. (stato[1]));
18. OR5X1 p1158A(.A (n_34), .B (n_35), .C (n_9), .D (n_22), .E (n_29), .Q
19. (n_38));
20. NA3X1 p1304A(.A (n_31), .B (n_20), .C (n_32), .Q (n_37));
21. DFRRQX1 \stato_reg[2] (.RN (n_39), .C (clock), .D (n_36), .Q
22. (stato[2]));
23. SDFRRQX0 outp_reg(.RN (n_39), .C (clock), .D (n_25), .SD (n_24), .SE
24. (n_28), .Q (outp));
25. NA2X1 p1424A(.A (n_30), .B (n_23), .Q (n_36));
26. NO2I1X1 p1158A770(.B (n_16), .AN (n_33), .Q (n_35));
27. NO2X1 p1222A(.A (n_33), .B (line2), .Q (n_34));
28. NA2X1 p1304A771(.A (n_26), .B (stato[2]), .Q (n_32));
29. NO2X1 p1410A(.A (n_15), .B (n_14), .Q (n_31));
30. NA2X1 p1446A(.A (n_18), .B (n_27), .Q (n_30));
31. NO2I1X1 p1368A(.B (n_12), .AN (n_28), .Q (n_29));
```

```
32.  NA2X1 p1158A772(.A (n_13), .B (n_27), .Q (n_33));
33.  NA2X1 p1304A773(.A (n_5), .B (n_11), .Q (n_26));
34.  INX1 p1335A(.A (n_24), .Q (n_25));
35.  NA2X1 p1424A774(.A (n_21), .B (n_6), .Q (n_23));
36.  DFRRQX0 overflw_reg(.RN (n_39), .C (clock), .D (n_7), .Q (overflw));
37.  NO2I1X1 p1270A(.B (n_19), .AN (n_21), .Q (n_22));
38.  NA2X1 p1404A(.A (n_21), .B (n_19), .Q (n_20));
39.  NA2X1 p1272A(.A (n_17), .B (stato[2]), .Q (n_24));
40.  NA2X1 p1446A776(.A (n_17), .B (n_16), .Q (n_18));
41.  NO2X1 p1422A(.A (n_17), .B (stato[2]), .Q (n_15));
42.  AND3X1 p1409A(.A (n_10), .B (stato[0]), .C (n_16), .Q (n_14));
43.  NA2X1 p1912A(.A (n_1), .B (n_3), .Q (n_28));
44.  NA2I1X0 p1158A777(.B (n_12), .AN (n_8), .Q (n_13));
45.  OR2X1 p1304A778(.A (n_10), .B (n_16), .Q (n_11));
46.  AND3X1 p1344A(.A (n_8), .B (n_2), .C (line2), .Q (n_9));
47.  NO2X1 p1548A(.A (n_12), .B (stato[2]), .Q (n_7));
48.  NA2X1 p1481A(.A (n_4), .B (n_19), .Q (n_6));
49.  NA2I1X0 p1424A779(.B (stato[0]), .AN (n_19), .Q (n_5));
50.  NO2X1 p1270A780(.A (n_27), .B (stato[1]), .Q (n_21));
51.  NA2X1 p1272A781(.A (n_4), .B (stato[1]), .Q (n_17));
52.  NA2X1 p1912A782(.A (n_2), .B (line2), .Q (n_3));
53.  NA2X1 p1926A(.A (n_0), .B (line1), .Q (n_1));
54.  NO2X1 p1160A(.A (stato[1]), .B (stato[0]), .Q (n_8));
55.  NA2X1 p1157A(.A (stato[1]), .B (stato[0]), .Q (n_12));
56.  NO2X1 p1948A(.A (line1), .B (line2), .Q (n_19));
57.  NA2X1 p1965A(.A (line1), .B (line2), .Q (n_16));
58.  INX1 p1304A783(.A (stato[1]), .Q (n_10));
59.  INX1 p1269A(.A (stato[2]), .Q (n_27));
60.  INX1 p1272A784(.A (stato[0]), .Q (n_4));
61.  INX1 p214748365A(.A (reset), .Q (n_39));
62.  INX1 p1911A(.A (line1), .Q (n_2));
63.  INX1 p1926A785(.A (line2), .Q (n_0));
64.  endmodule
```

E.4 Exemplo de *netlist* estrutural otimizada com restrição de entradas e transistores em série do *bechmark* B01 do ITC'99

```
1.  module b01 ( outp,overflw,line1,line2,reset,clock );
2.  input line1,line2,reset,clock;
3.  output outp,overflw;
4.  wire
5.  line1,line2,reset,clock,outp,overflw,n_1,n_2,n_3,n_4,n_7,n_8,n_9,n_10,n_12,n_13,n
6.  4,n_15,n_16,n_17,n_19,n_21,n_22,n_24,n_25,n_26,n_27,n_28,n_29,n_33,n_34,n_35
7.  n_36,n_37,n_38,n_39,n_27_inv,stato_0__inv,line2_inv,n_15_inv,stato_2__inv,n_14
8.  nv,n_19_inv,n_21_inv,n_26_inv;
9.  wire [2:0]stato;
10. DFRRQX1 \stato_reg[0](
11. .RN(n_39),
12. .C(clock),
13. .D(n_38),
14. .Q(stato[0]));
15. DFRRQX1 \stato_reg[1](
16. .RN(n_39),
17. .C(clock),
18. .D(n_37),
19. .Q(stato[1]));
20. OR5X1 p1158A(
21. .C(n_9),
22. .D(n_22),
23. .A(n_34),
24. .B(n_35),
25. .E(n_29),
26. .Q(n_38));
27. DFRRQX1 \stato_reg[2](
28. .RN(n_39),
29. .C(clock),
30. .D(n_36),
```

```
31.     .Q(stato[2]));
32.     SDFRRQX0 outp_reg(
33.     .RN(n_39),
34.     .C(clock),
35.     .D(n_25),
36.     .SD(n_24),
37.     .SE(n_28),
38.     .Q(outp));
39.     NO2I1X1 p1158A770(
40.     .B(n_16),
41.     .AN(n_33),
42.     .Q(n_35));
43.     NO2X1 p1222A(
44.     .A(n_33),
45.     .B(line2),
46.     .Q(n_34));
47.     NO2I1X1 p1368A(
48.     .B(n_12),
49.     .AN(n_28),
50.     .Q(n_29));
51.     NA2X1 p1158A772(
52.     .A(n_13),
53.     .B(n_27),
54.     .Q(n_33));
55.     INX1 p1335A(
56.     .A(n_24),
57.     .Q(n_25));
58.     DFRRQX0 overflw_reg(
59.     .RN(n_39),
60.     .C(clock),
61.     .D(n_7),
62.     .Q(overflw));
63.     NO2I1X1 p1270A(
```

```
64.     .B(n_19),
65.     .AN(n_21),
66.     .Q(n_22));
67.     NA2X1 p1272A(
68.     .A(n_17),
69.     .B(stato[2]),
70.     .Q(n_24));
71.     NO2X1 p1422A(
72.     .A(n_17),
73.     .B(stato[2]),
74.     .Q(n_15));
75.     AND3X1 p1409A(
76.     .C(n_16),
77.     .A(n_10),
78.     .B(stato[0]),
79.     .Q(n_14));
80.     NA2X1 p1912A(
81.     .A(n_1),
82.     .B(n_3),
83.     .Q(n_28));
84.     NA2I1X1 p1158A777(
85.     .B(n_12),
86.     .AN(n_8),
87.     .Q(n_13));
88.     AND3X1 p1344A(
89.     .C(line2),
90.     .A(n_8),
91.     .B(n_2),
92.     .Q(n_9));
93.     NO2X1 p1548A(
94.     .A(n_12),
95.     .B(stato[2]),
96.     .Q(n_7));
```

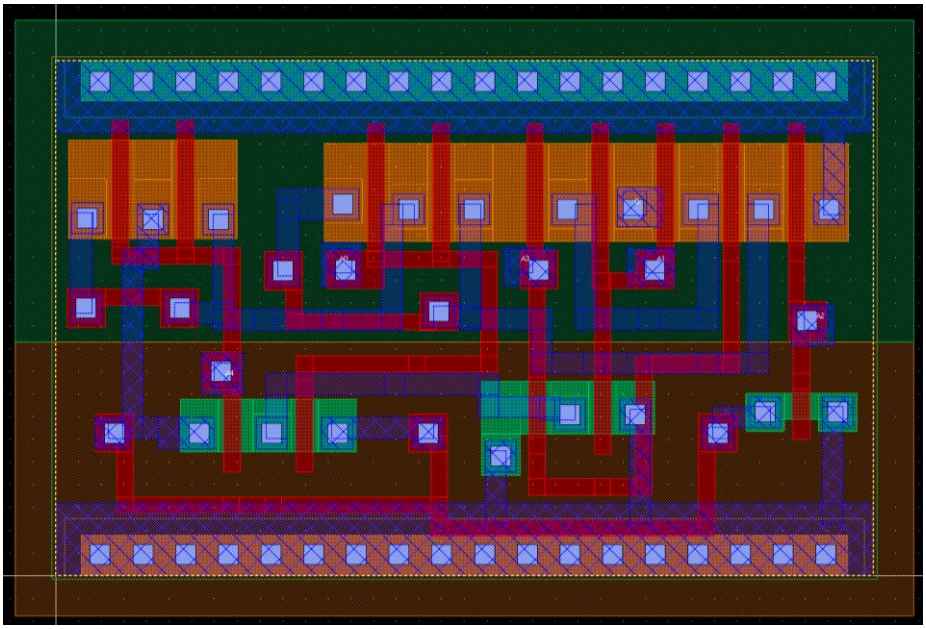
```
97.   NO2X1 p1270A780(
98.   .A(n_27),
99.   .B(stato[1]),
100.  .Q(n_21));
101.  NA2X1 p1272A781(
102.  .A(n_4),
103.  .B(stato[1]),
104.  .Q(n_17));
105.  NA2X1 p1912A782(
106.  .A(n_2),
107.  .B(line2),
108.  .Q(n_3));
109.  NO2X1 p1160A(
110.  .A(stato[1]),
111.  .B(stato[0]),
112.  .Q(n_8));
113.  NA2X1 p1157A(
114.  .A(stato[1]),
115.  .B(stato[0]),
116.  .Q(n_12));
117.  NO2X1 p1948A(
118.  .A(line1),
119.  .B(line2),
120.  .Q(n_19));
121.  NA2X1 p1965A(
122.  .A(line1),
123.  .B(line2),
124.  .Q(n_16));
125.  INX1 p1304A783(
126.  .A(stato[1]),
127.  .Q(n_10));
128.  INX1 p1269A(
129.  .A(stato[2]),
```

```
130.  .Q(n_27));
131.  INX1 p1272A784(
132.  .A(stato[0]),
133.  .Q(n_4));
134.  INX1 p214748365A(
135.  .A(reset),
136.  .Q(n_39));
137.  INX1 p1911A(
138.  .A(line1),
139.  .Q(n_2));
140.  GATE_0X1 gate_0(
141.  .A5(n_16),
142.  .A4(n_10),
143.  .A6(n_19),
144.  .A7(stato_0__inv),
145.  .out(n_26));
146.  GATE_1X1 gate_1(
147.  .A6(stato_2__inv),
148.  .A9(n_19_inv),
149.  .A10(n_21_inv),
150.  .A11(n_26_inv),
151.  .A8(n_15_inv),
152.  .A7(n_14_inv),
153.  .Q(n_37));
154.  INX0 inv_n_27_inv(
155.  .A(n_27),
156.  .Q(n_27_inv));
157.  INX0 inv_line2_inv(
158.  .A(line2),
159.  .Q(line2_inv));
160.  NA2X1 gate_2(
161.  .A(line1),
162.  .B(line2_inv),
```

```
163.    .Q(n_1));
164.    INX0 inv_n_15_inv(
165.    .A(n_15),
166.    .Q(n_15_inv));
167.    INX0 inv_n_19_inv(
168.    .A(n_19),
169.    .Q(n_19_inv));
170.    INX0 inv_n_21_inv(
171.    .A(n_21),
172.    .Q(n_21_inv));
173.    INX0 inv_n_26_inv(
174.    .A(n_26),
175.    .Q(n_26_inv));
176.    INX0 inv_stato_2__inv(
177.    .A(stato[2]),
178.    .Q(stato_2__inv));
179.    INX0 inv_n_14_inv(
180.    .A(n_14),
181.    .Q(n_14_inv));
182.    INX0 inv_stato_0__inv(
183.    .A(stato[0]),
184.    .Q(stato_0__inv));
185.    GATE_3X1 gate_3(
186.    .A1(n_4),
187.    .A6(n_17),
188.    .A5(n_16),
189.    .A7(n_19),
190.    .A8(n_21_inv),
191.    .A9(n_27_inv),
192.    .Q(n_36));
193.  Endmodule
```

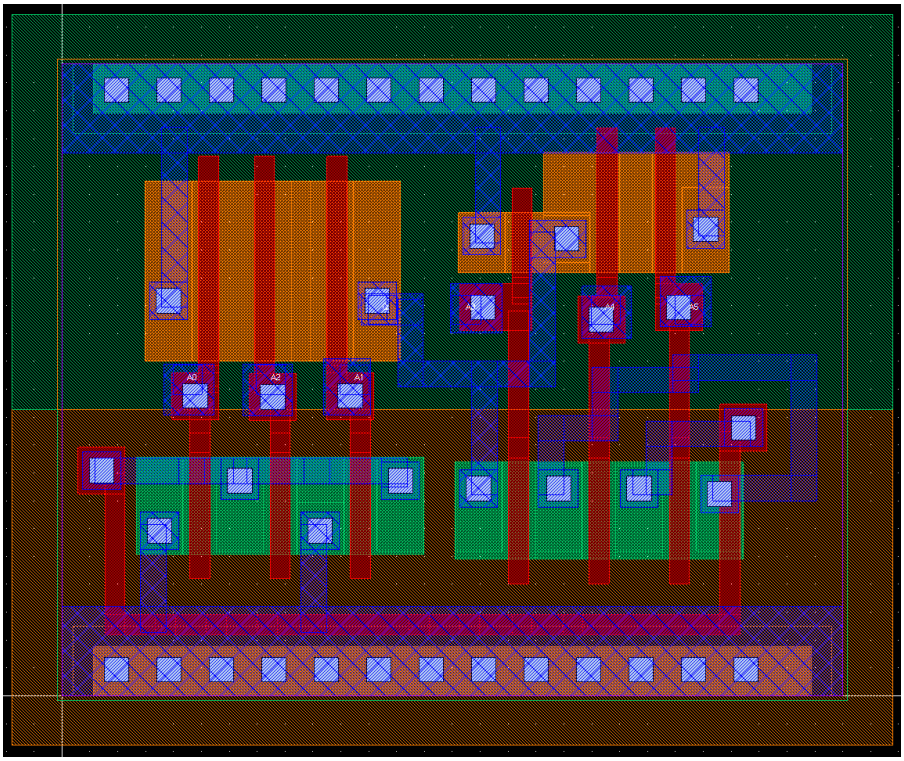
APÊNDICE F LEIAUTES DE PORTAS COMPLEXAS GERADOS PELA FERRAMENTA ASTRAN EM 180NM

Figura G.1 – Porta complexa da expressão lógica !(A2+((A0+A4)*(A3+A1))).



Fonte: Figura elaborada pelo autor.

Figura G.2 – Porta complexa da expressão lógica !((A3*(A5+A4))*((A0+A2)+A1)).



Fonte: Figura elaborada pelo autor.