PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
FACULDADE DE INFORMÁTICA

ASCEND-FREEPDK45: PROJETO E IMPLEMENTAÇÃO DE UMA BIBLIOTECA DE CÉLULAS ABERTA PARA CIRCUITOS ASSÍNCRONOS

CARLOS HENRIQUE MENEZES OLIVEIRA

Monografia apresentada como requisito parcial à obtenção do grau de Engenheiro de Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Msc. Matheus Trevisan Moreira
Co-Orientador: Prof. Dr. Ney Laert Vilar Calazans

Porto Alegre
2015

DEDICATÓRIA

Dedico este trabalho a minha família, a minha namorada e todos meus amigos que me acompanharam durante a minha graduação.

AGRADECIMENTOS

Agradeço primeiramente a minha família, meus pais, Senir e Rita, ao meu irmão, João e sobretudo ao meu vô, vô Jaime que sempre foi um grande incentivador do meu estudo desde quando eu era criança e o mesmo me levava para escola e me ensinava as lições de matemática. Pensando que o ser pensante que sou hoje agradeço a todos que me ajudaram com ensinamentos até aqui. Agradeço aos meus professores do CEFET e também ao pessoal do estágio realizado na TV Sudoeste. Além destes, agradeço ao professores da PUCRS que tanto contribuíram para o meu aprendizado na minha graduação, e também a Ney Calazans pelo convite de ingressar como bolsista de iniciação científica no GAPH e pela co-orientação neste trabalho. Dedico aos meus amigos que me ajudaram em diversas etapas da minha graduação, aos amigos do RONDON que realizaram comigo a melhor experiência da graduação. E como não lembrar dos amigos que fiz na França e também todo o aprendizado que adquiri em terras francesas. E no tempo de escrita desse trabalho de final de curso, queria muito agradecer pelos vários dias de companhia ao meu lado dos meus amigos e companheiros do GAPH, sobretudo Jurinha e Waltinho que estão comigo desde os primeiros dias dessa graduação, junto ao Edcity, e os que sempre me acompanham na *meiuca*. Agradeço a minha namorada Bruna por ter sido paciente e ter me dado diversos dias de alegria e alívio durante todo esse processo. Quero agradecer pela ajuda de todos meus companheiros do GAPH e dos que de alguma forma me ajudaram a realizar este trabalho, LHeck e Bortolon. E ao meu orientador que me acompanha desde os primeiros passos na iniciação científica, e que me mostrou boa parte do que sei em assíncronos e microeletrônica.

ASCEND-FREEPDK45: PROJETO E IMPLEMENTAÇÃO DE UMA BIBLIOTECA DE CÉLULAS ABERTA PARA CIRCUITOS ASSÍNCRONOS

RESUMO

A análise do estado da arte em circuitos assíncronos revela a carência de recursos de apoio ao projeto. Dentre os recursos que fazem falta ressalta-se a escassez de bibliotecas de células assíncronas para apoiar o projeto semi-dedicado. Bibliotecas de componentes assíncronos, quando citadas na literatura, frequentemente surgem como desenvolvimentos por demanda de projetos específicos, não servindo para uso geral, além de não se caracterizarem como bibliotecas de livre acesso. A biblioteca aberta de células assíncronas ASCEnD-FreePDK45 visa suprir algumas das carências citadas. ASCEnD-FreePDK45 parte de um *design kit* de acesso livre, o FreePDK45, desenvolvido na North Carolina State University (NCSU). Esta biblioteca é totalmente compatível com a biblioteca aberta NanGate-FreePDK45 (mais de 170 células com 62 funções distintas, baseada no mesmo *design kit*). ASCEnD-FreePDK45 foi desenvolvida com o fluxo ASCEnD-A, específico para criar biblioteca de células assíncronas. Ela contém hoje 28 células que implementam 10 funções distintas das famílias lógicas NCL e NCL+, e cujo objetivo primário é dar suporte ao *template* de projeto assíncrono SDDS-NCL. Neste trabalho, demonstra-se o uso da biblioteca ASCEnD-FreePDK45 e sua integração com a biblioteca NanGate-FreePDK45 através de um conjunto de experimentos sobre um estudo de caso de implementação de circuito. Trata-se de um somador Kogge-Stone de 32 bits, sintetizado com sucesso em várias versões para atender a diferentes restrições de projeto.

Palavras-Chave: Circuitos assíncronos, Biblioteca de células aberta.

ASCEND-FREEPDK45: DESIGN AND IMPLEMENTATION OF AN OPEN CELL LIBRARY FOR ASYNCHRONOUS CIRCUITS

ABSTRACT

An analysis of the state of art in asynchronous circuits reveals a lack of resources to support their design. Among the missing resources, stands out the scarcity of cell libraries of asynchronous components to support semi-custom design. When asynchronous components cell libraries do appear in the literature, they often come as a demand from specific circuits design and not as general purpose design resources. Also, these resources are never open access. The open asynchronous cell library ASCEnD-FreePDK45 targets to fulfill these missing spots in the design of asynchronous circuits. ASCEnD-FreePDK45 is based on the open access FreePDK45 design kit, developed in the North Carolina State University (NCSU). This library is fully compatible with the open access NanGate-FreePDK45 library (more than 170 cells implementing 62 distinct functions). ASCEnD-FreePDK45 was developed with the ASCEnD-A design flow, which has been specifically proposed to support the construction of asynchronous cell libraries. The library contains today 28 cells that implement 10 distinct functions from the NCL and NCL+ logic families. The primary target is to support the SDDS-NCL asynchronous design template. This work demonstrates the use of the ASCEnD-FreePDK45 and its integration with the NanGate-FreePDK45 library through a set of experiments over a case study circuit implementation. The case study is a 32-bit Kogge-Stone adder, successfully synthesized in several versions to fulfill different design constraints.

Keywords: Asynchronous circuits, Open cell library.

LISTA DE FIGURAS

Figura 1.1 – Tipos de células que compõe a biblioteca ASCEnD-FreePDK45. . . . 14
Figura 2.1 – Um exemplo de layout de um componente eletrônico, no caso um inversor. 16
Figura 3.1 – Os diferentes métodos de projetos. 20
Figura 4.1 – Visão geral do fluxo ASCEnD-A de projeto de células. [1] 27
Figura 4.2 – Etapa de configuração do ambiente e a especificação do circuíto. [1] 28
Figura 4.3 – Etapa de dimensionamento do circuito. [1] 29
Figura 4.4 – Detalhamento da etapa onde se realiza a criação e importação do layout e do esquemático. [1] 30
Figura 4.5 – Fluxo onde se caracteriza uma biblioteca. [1] 30
Figura 5.1 – *Layout* da célula INCLP1W11OF2X1 33
Figura 5.2 – *Layout* da célula INCLP1W11OF2X2 33
Figura 5.3 – *Layout* da célula INCLP1W11OF2X4 33
Figura 5.4 – Modelo 3D simplificado de um transistor MOS. 33
Figura 5.5 – *Layout* com as as medidas das regras de projeto do *gate*. 34
Figura 5.6 – *Layout* com as as medidas das regras de projeto para a camada de difusão. 35
Figura 5.7 – *Layout* com as as medidas das regras de projeto para a camada de contato. 36
Figura 5.8 – *Layout* com as as medidas das regras de projeto para a camada de metal1. 37
Figura 5.9 – Arquitetura geral das células da biblioteca ASCEnD-FreePDK45 (idêntica à da biblioteca NanGate-FreePDK45). 38
Figura 5.10 – Ilustração do processo de *folding* de um transistor. 39
Figura 5.11 – Exemplo de descrição Spice para a porta INCL1111OF2X1, entrada do programa ROGen. 41
Figura 5.12 – Circuito de simulação gerado pelo ROGen utilizando o INCL1W11OF2X1, apresentando a configurações com *fan out* 4. 42
Figura 5.13 – Gráfico exportado do CeS, que mostra o melhor dimensionamento para a célula INCL1W11OF2X1. 43
Figura 5.14 – Processo de importação do *layout* de uma célula. 44

Figura 5.15 – Após a verificação de erros de DRC, apresenta-se um erro de regra de projeto na célula RINCL2W11OF2X1, relativo a um polígono de polissilício, como se vê no relatório a esquerda da figura. 45
Figura 5.16 – O erro de DRC corrigido e o relatório indicando que todas as regras foram verificadas sem erro. 45
Figura 5.17 – *V*ista abstrata da célula INCL1W11OF2X1. 46
Figura 5.18 – O fluxograma que explicita o método de denominação das células da biblioteca ASCEnD-FreePDK45. As escolhas em azul são opcionais e aquelas em vermelho são obrigatórias. 49
Figura 5.19 – Exemplo de *datasheet* da biblioteca ASCEnD-FreePDK45, para a célula INCL1W11OF2X1 (Parte 1). 50
Figura 5.20 – Exemplo de *datasheet* da biblioteca ASCEnD-FreePDK45, para a célula INCL1W11OF2X1 (Parte 2). 51

LISTA DE TABELAS

Tabela 3.1 – Tabela verdade de um C-element convencional de 2 entradas *A* e *B* e saída *Y*. O índice nos valores de algumas saídas designa valores atuais (n) e anteriores ($n-1$). 24
Tabela 3.2 – Tabela verdade de uma porta NCL2-de-3 (porta de 3 entradas com *threshold* 2). 25
Tabela 3.3 – Tabela verdade de uma porta NCL+2-de-3 (porta de 3 entradas com *threshold* 2). 26
Tabela 4.1 – Parâmetros de configuração da ferramenta ROGen. 31
Tabela 5.1 – Regras de *design* do *gate*. 34
Tabela 5.2 – Regras de projeto para a camada de difusão. 35
Tabela 5.3 – Regras de projeto para a camada de contato 36
Tabela 5.4 – Regras de *design* do Metal1 37
Tabela 5.5 – Tabela com as correspondências entre o Virtuoso e o arquivo GDS . 44
Tabela 5.6 – Lista das células da ASCEnD-FreePDK45. 47
Tabela 5.7 – Alguns resultados da síntese de diferentes versões de um somador Kogge-Stone de 32 bits utilizando a biblioteca ASCEnD-FreePDK45. 48

LISTA DE SIGLAS

ASCEND-A – ASCEnD-Astran

CI – Circuito Integrado

DRC – *Design Rules Check*

EDA – *Eletronic Design Automation*

FO4 – *Fan-out of 4*

FPGA – *Field Programmable Gate Arrays*

INCL – NCL Invertido

INCL+ – NCL+ Invertido

ITRS – *International Technology Roadmap for Semiconductors*

LEF – *Library Exchange Format*

LVS – *Layout Versus Schematic*

NCL – *Null Convention Logic*

NCL – Null Convention Logic

NCL+ – *Return-to-one Convention Logic*

NCLP – *Return-to-one Convention Logic*

NCSU – *North Carolina State University*

PDK – *Process Design Kit*

QDI – Quasi Delay Insensitive

RINCL – NCL Invertido com função de *Reset*

RNCL – NCL com função de *Reset*

SDDS – *Spatially Distributed Dual-Spacer*

SINCL – NCL Invertido com função de *Set*

SNCL – NCL com função de *Set*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 MOTIVAÇÃO 12
1.2 OBJETIVOS 14

2 TRABALHOS RELACIONADOS 16
2.1 FREEPDK45 E A BIBLIOTECA DA NANGATE 16
2.2 ASCEND-ST65 17
2.3 TAL_130NM (TIMA/LETI) 17
2.4 ASPRO-216 18
2.5 DISCUSSÃO 18

3 PROJETO DE CIS ASSÍNCRONOS BASEADO EM CÉLULAS 20
3.1 MÉTODOS DE PROJETO DE CIS 20
3.2 PROJETO DE BIBLIOTECAS DE CÉLULAS 22
3.3 CÉLULAS ASSÍNCRONAS TÍPICAS 24

4 FLUXO ASCEND 27
4.1 ESPECIFICAÇÃO 28
4.2 DIMENSIONAMENTO 28
4.3 DESIGN 29
4.4 CARACTERIZAÇÃO 29

5 IMPLEMENTAÇÃO DA BIBLIOTECA ASCEND-FREEPDK45 32
5.1 ESPECIFICAÇÃO DAS CÉLULAS 32
5.2 DEFINIÇÃO DE REGRAS DE PROJETO 32
5.3 DEFINIÇÃO DA ARQUITETURA DE CÉLULAS 38
5.4 ADAPTAÇÃO DO FLUXO ASCEND-A 39
5.4.1 DIMENSIONAMENTO COM ROGEN E CES 40
5.4.2 CRIAÇÃO DE *LAYOUTS* UTILIZANDO ASTRAN 42
5.4.3 CARACTERIZAÇÃO DA BIBLIOTECA 45
5.5 COMPOSIÇÃO DA BIBLIOTECA ASCEND-FREE-PDK45 (V.1.0) 46

6 CONCLUSÃO E TRABALHOS FUTUROS 52

REFERÊNCIAS **54**

ANEXO A – Regras de *design* da FreePDK45. . . . . **58**

1. INTRODUÇÃO

1.1 Motivação

Durante as ultimas décadas o foco da industria de semicondutores esteve majoritariamente relacionado ao projeto de circuitos síncronos [2]. Circuitos síncronos são definidos como circuitos que utilizam um único sinal global para controlar todos os seus registradores [3]. Ou seja, toda troca de informação e sincronização é governada por esse sinal, comumente chamado de sinal de relógio, ou em inglês, *clock*.

Diversos problemas dificultam o projeto de circuitos síncronos em tecnologias atuais [4]. Muitos desses problemas estão ligados à distribuição do sinal de relógio, como o escorregamento de relógio, definido como a variação do instante em que o relógio chega à diferentes registradores. Esse problema é causado pelas diferenças nos caminhos de interconexão percorridos por este sinal global até atingir cada registrador individual. Outro problema que afeta o sinal de relógio é conhecido como *jitter*. Nesse caso, o sinal sofre uma variação na duração de seus pulsos, podendo aumentar ou diminuir o período. Para evitar que estes e outros problemas comprometam a funcionalidade do circuito, projetistas de CIs recorrem a técnicas específicas de projeto e de distribuição do sinal de relógio [3]. Essas técnicas exigem a adição de hardware, o que pode causar impactos nos parâmetros de eficiência energética e desempenho do projeto. Segundo Tiwari et al. [5], o hardware necessário para a distribuição do sinal de relógio em um microprocessador é responsável por em média 45% da potência total finalmente dissipada pelo componente. Além disso, segundo Wimer [6], esse valor pode se elevar a até 75% em alguns CIs.

Circuitos assíncronos, por outro lado, representam hoje uma fatia muito pequena dos CIs comerciais. Uma tendência esperada é que estes ganhem cada vez mais espaço no mercado, devido ao fato de permitirem ganhos em comparação com circuitos síncronos em diversos contextos de uso e tecnologias atuais [2]. De fato, circuitos assíncronos potencialmente permitem obter vantagens em aspectos como [4]: (i) redução de potência; (ii) velocidade de operação; e (iii) problemas relacionados ao sinal de relógio, pela sua eliminação. Porém, existem dificuldades para trabalhar com circuitos assíncronos, como a questão de como garantir a temporização adequada das ações do circuito, a área ocupada por estes quando se empregam técnicas de projeto robustas a atrasos quaisquer e sua variação, entre outras dificuldades. Estas dependem da forma de projeto adotada para circuitos assíncronos, cujas características podem variar amplamente, ao contrário do que ocorre quando se emprega projeto síncrono. Circuitos assíncronos trazem a necessidade de dispor de células e métodos de síntese distintos. Quando se trata de métodos de síntese de circuitos assíncronos, normalmente se aceita que estes exigem técnicas de comunica-

ção e sincronização locais (em inglês, *local handshaking*), ao invés de se basearem na existência de um clock global. Para cada *template* assíncrono usado é necessário atender alguns requisitos diferentes dos atendidos em projetos síncronos. Por exemplo, circuítos assíncronos podem operar utilizando codificações que não são Booleanas (também denominadas *single rail*), optando por codificações insensíveis a atrasos. Exemplos das últimas são codificações *multi-rail* como dual rail e 1-de-4. Outra possibilidade é esses circuitos usarem codificações Booleanas tradicionais, mas necessitarem de cuidado extra na definição de restrições temporais, onde linhas de atraso devem ser adicionadas a sinais de controle selecionados [7].

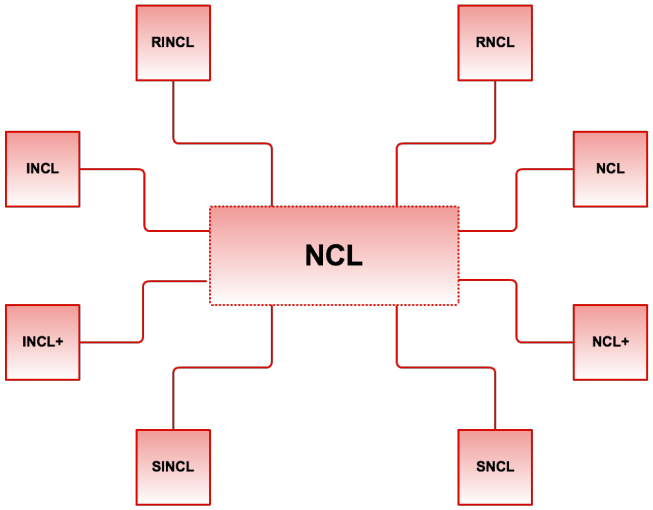
Com a análise do momento atual nota-se que existe uma carência de ferramentas de apoio ao projeto de circuitos de assíncronos [4] [8] [9]. Tal carência fica clara, quando se analisa o suporte ao projeto de circuitos síncronos. Para os últimos existem diferentes ferramentas para simulação, verificação e síntese que facilitam o projeto síncrono. Além disso, diferentes bibliotecas de componentes básicos estão disponíveis para projetistas de CIs síncronos, as chamadas de bibliotecas de células padronizadas (em inglês, *standard cells*). Uma biblioteca de células padronizada contém elementos básicos pré-caracterizados em número e diversidade suficientes para implementar qualquer circuito síncrono. Contudo, não existem bibliotecas comerciais contendo os componentes necessários aos diferente *templates* utilizados em projeto assíncrono, tais como células NCL [10],NCL+ [11], C-elements [12], módulos PCHB, LSSD [13], elementos toggle e Q-flops, entre outros. Alguns trabalhos tentam resolver os problemas de apoiar métodos de projeto assíncronos, como a ferramenta de síntese Proteus [14], a ferramenta de atendimento de restrições de temporização ACDC [9], e os *templates* Blade [15] e SDDS-NCL [16]. No que diz respeito a propostas de bibliotecas de células assíncronas existem exemplos não comerciais, tais como a biblioteca TAL(TIMA/LETI) [17] e a biblioteca ASCEnD-STM65 [8]. Porém, tais bibliotecas não são de fácil acesso, pois estão projetadas em tecnologias não disponíveis livremente ou não são elas mesmas de domínio público.

Neste cenário, projetistas de circuitos assíncronos adaptam-se, utilizando ferramentas originalmente projetadas para projetos síncronos e implementando células na medida da necessidade de seus projetos específicos. Para tanto, ferramentas tradicionais podem ter de ser adaptadas, conforme discutido em [18] e [19]. Apesar de alguns trabalhos contornarem esta limitação, a falta de bibliotecas de células para circuitos assíncronos é um problema mais difícil de ser contornado. De acordo com Gulati e Brunvand [20], o uso de bibliotecas convencionais para o projeto de circuitos assíncronos não somente compromete a qualidade do circuito projetado, mas também pode comprometer sua funcionalidade. Dessa forma, existe uma grande necessidade de novos métodos, ferramentas e bibliotecas que permitam o projeto de circuitos assíncronos. A necessidade de bibliotecas assíncronas de fácil acesso motivou a criação da biblioteca ASCEnD-FreePDK45 proposta neste trabalho. Esta baseia-se no Free PDK 45nm um Process Design Kit (PDK) de livre acesso para o

nodo tecnológico de fabricação de CIs de 45nm. Este PDK é disponibilizado pela North Carolina State University (NCSU) e possui uma biblioteca de células padronizada desenvolvida especialmente para ele pela empresa NanGate, também de acesso livre.

1.2 Objetivos

Dada a motivação acima, esse trabalho busca reduzir a carência por bibliotecas de células para o projeto de circuitos assíncronos. Para tanto, propõe-se a ASCEnD-FreePDK45, uma biblioteca aberta que habilita o projeto de CIs assíncronos usando um conjunto de *templates* baseados em *Null Convention Logic* (NCL) e em *Return-to-one Convention Logic*(NCL+), que estão especificados na seção 3.3. A ASCEnD-FreePDK45 de fato disponibiliza um conjunto básico de células NCL e NCL+ dos tipos especificados na Figura 1.1.



Nome da célula	Especificação
INCL	NCL Invertido
INCL+	NCL+ Invertido
NCL	Null Convention Logic
NCL+	Return-to-one Convention Logic
RINCL	NCL Invertido com função de Reset
RNCL	NCL com função de Reset
SINCL	NCL Invertido com função de Set
SNCL	NCL com função de Set

Figura 1.1 – Tipos de células que compõe a biblioteca ASCEnD-FreePDK45.

O grupo de pesquisa GAPH desenvolveu anteriormente a biblioteca a ASCEnD-ST65 [8] utilizando o fluxo ASCEnD [8], proposto pelo mesmo grupo. A biblioteca ASCEnD-ST65 usa como base o PDK STM-65nm da fundição ST Microelectronics e emprega o editor Virtuoso 5.1 da Cadence. O acesso a esta tecnologia depende de Non Disclosure Agreement (NDA) assinado entre o GAPH e a ST Microelectronics. A criação da biblioteca ASCEnD-FreePDK45 foi realizada utilizando o mesmo fluxo, que porém necessitou de adaptações para tornar-se compatível com o FreePDK45 [21] e também com a versão mais recente (V. 6.1.5) do software Virtuoso. Pode-se dividir os objetivos desse trabalho em duas classes, estratégicos e específicos, onde os objetivos estratégicos são os seguintes:

- Adaptação do fluxo ASCEnD-A para dar suporte ao FreePDK45;

- Geração de uma biblioteca de células para permitir o projeto de circuitos assíncronos usando o FreePDK45.

Para alcançar esses objetivos, definiram-se os seguintes objetivos específicos:

- Dominar conceitos de fluxos de projeto de CIs (baseados em células);
- Ter comando sobre conceitos básicos de projeto de circuitos assíncronos;
- Identificar um subconjunto mínimo de células necessárias para o projeto de circuitos assíncronos, conforme especificado na ASCEnD-ST65;
- Compreender o fluxo de projeto ASCEnD-A [22], desenvolvido para automatizar o processo de geração de células para circuitos assíncronos;
- Explorar o FreePDK45 e a biblioteca projetada pela NanGate [23] para essa tecnologia;
- Projetar a biblioteca ASCEnD-FreePDK45;
- Validar a biblioteca ASCEnD-FreePDK45;
- Disponibilizar a biblioteca desenvolvida via acesso livre para a comunidade de projeto de CIs.

2. TRABALHOS RELACIONADOS

2.1 FreePDK45 e a Biblioteca da NanGate

O acesso a um PDK é fundamental para que se consiga fabricar um CI. Isso se deve ao fato de um PDK conter as definições de regras e passos que devem ser executados para projetar um CI que possa ser enviado para fabricação. Um PDK contém os arquivos referentes à tecnologia, scripts que permitem o desenho do *layout* e as regras de projeto. Layout é a representação das máscaras que serão utilizadas para fabricação de um CI. Normalmente um layout apresenta diversos polígonos de cores diferentes, onde cada cor representa uma máscara que corresponde a uma camada de um tipo de material específico utilizado na confecção de componentes eletrônicos, como se vê na Figura 2.1.

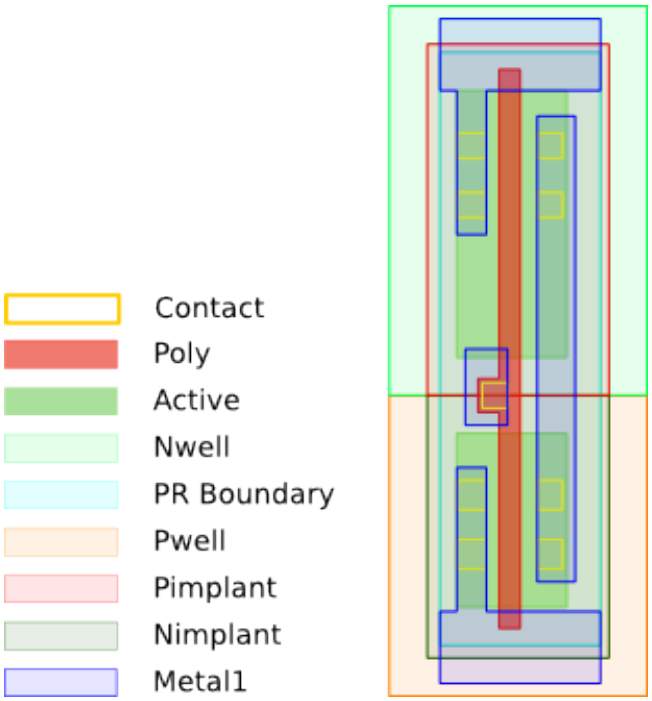


Figura 2.1 – Um exemplo de layout de um componente eletrônico, no caso um inversor.

Definindo sucintamente, regras de projeto representam os limites físicos do processo de fabricação de chips e as restrições para cada camada do *layout*.

Entretanto, o acesso à PDKs é, tipicamente, dificultado por restrições impostas pelas fábricas, detentoras da propriedade intelectual destes. Pensando nisso, o grupo de automatização de design eletrônico da Universidade do Estado da Carolina do Norte (North Carolina State University - Eletronic Design Automation Group) propôs um PDK aberto e de desenvolvimento colaborativo, baseado em um nodo tecnológico de 45nm real. Esse PDK

foi chamado FreePDK45 [21] e é voltado, sobretudo, para atividades de pesquisa e ensino de microeletrônica.

As especificações tecnológicas do FreePDK45 foram criadas com base em artigos publicados, em um modelo preditivo tecnológico e em regras de escalamento tecnológico. Isso é fundamentalmente diferentemente de PDKs industriais, que são desenvolvidos baseados no processo de fabricação de CIs reais. Devido a isso, projetos realizados com base na FreePDK45 não é fabricável, porém é utilizado em muitos ambientes acadêmicos por pesquisadores e universidades como material didático fundamental de ensino de microeletrônica.

A partir da necessidade de uma biblioteca de células de acesso livre compatível com o FreePDK45, a NanGate criou a biblioteca "NanGate FreePDK45" [23]. Essa biblioteca contém um conjunto básico de 170 células com 62 funções diferentes, incluindo operações lógicas básicas, células sequenciais e células mais complexas como *scan flops*, mas ela não dá nenhum suporte específico à metodologias de projeto de circuítos assíncronos.

2.2 ASCEnD-ST65

A ASCEnD-ST65 é uma biblioteca de células assíncronas desenvolvida para o *design kit* de 65nm da fundição ST Microeletronics. Essa biblioteca foi criada visando a utilização de células assíncronas para ambientes de síntese como os descritos em [16] e [9]. Ela foi projetada e é mantida pelo grupo GAPH da PUCRS, mesmo grupo do Autor desse trabalho. De fato, o Autor participou do processo de criação da biblioteca, desenvolvendo atividades de geração de *layout* de células.

A ASCEnD-ST65 contém 613 células, incluindo diferentes funcionalidades para circuitos assíncronos e diferentes dimensionamentos de transistores. Dessa forma, a biblioteca permite explorar otimizações em termos de eficiência energética e velocidade de operação [24]. Infelizmente, essa biblioteca não pode ser distribuída para todos interessados, pois foi desenvolvida utilizando um *design kit* sujeito a NDAs assinados entre o GAPH e a ST Microelectronics.

2.3 TAL_130nm (TIMA/LETI)

TAL é uma biblioteca de células dedicada ao projeto de circuitos assíncronos QDI, desenvolvida pelos laboratórios frances TIMA do INPG de Grenoble e o CEA-LETI da mesma cidade. Essa biblioteca foi projetada para o nodo tecnológico de 130nm, conforme detalhado por Maurine e outros [17]. Nesse mesmo trabalho, os Autores exploram o fluxo utilizado para a criação da biblioteca. Infelizmente, novamente, devido à utilização de

um PDK que não é de livre acesso e a restrições de propriedade intelectual dos próprios laboratórios, a biblioteca TAL não é distribuída abertamente.

2.4 ASPRO-216

O ASPRO-216 [25] é um microprocessador assíncrono de 16 bits, baseado em um *template* QDI. O projeto desse processador que está baseado em uma biblioteca de *standard cells* com cerca de 40 células especificamente criadas para o projeto deste microprocessador assíncrono. A escolha das células assíncronas dessa biblioteca foi feita de acordo com a necessidade do projeto, logo estas não foram criadas de maneira a dar suporte a projetos assíncronos em geral. Tais células incluem *C-elements* com 2 e 3 portas, com *set* e/ou *reset*) em conjunto com uma biblioteca convencional, fornecida juntamente com o PDK da tecnologia alvo, 0.25 micrômetros da CNET/SGS-Thomson.

2.5 Discussão

Circuitos assíncronos estão ganhando cada vez mais espaço em pesquisa e aplicações industriais selecionadas. Prova disso é a previsão da ITRS que diz que os protocolos de comunicação assíncronos ganharão uma parte do mercado de CIs brevemente [8]. Entretanto, o suporte ao projeto automatizado de circuitos assíncronos é pequeno e existe uma clara necessidade de um suporte maior, em especial para bibliotecas de células. De fato, grande parte do projeto automatizado de circuitos integrados requere bibliotecas de células, no caso de circuitos assíncronos bibliotecas como as propostas em [8] e [17]. Porém essas bibliotecas foram desenvolvidas em *design kits* privados e não são de acesso universal, pois estes *design kits* exigem o estabelecimento de acordos entre instituições que desejem acessá-los. Isso dificulta a distribuição e utilização dessas bibliotecas, fazendo com que o processo de validação de projetos assíncronos exija a criação de uma biblioteca específica, como foi o caso no projeto do ASPRO-216 [25] e do projeto mencionado em [26], onde foram utilizadas bibliotecas tanto da fábrica quanto específicas. Contudo, o fato destas bibliotecas de células assíncronas serem criadas a partir da necessidade dos projetos dificulta o seu uso projetos diferentes das aplicações para os quais foram elaboradas. Como resultado, projetistas que buscam usar circuitos assíncronos são obrigados a produzir suas próprias bibliotecas de células, e quando o fazem utilizam de métodos manuais de criação. A utilização de um fluxo automatizado de criação de células permite a finalização desta em um tempo menor e com uma padronização entre as células, algo mais difícil de se obter ao produzir *layouts* de célula de forma totalmente manual. A transformação do fluxo ASCEnD [8] para o fluxo ASCEnD-A [22], onde se utiliza a ferramenta de geração automática de *layouts*

Astran [27], permitiu criar bibliotecas de células mais rapidamente e com maior facilidade de configuração. Para distribuir bibliotecas livremente, é necessário utilizar um *design kit* também livre como o FreePDK45. Isto, aliado ao porte do ASCEnD-A para essa tecnologia, permite criar células assíncronas quaisquer para distribuição livre. A biblioteca ASCEnD-FreePDK45 possui este objetivo maior, a distribuição de uma biblioteca assíncrona de forma livre, que vem assim complementar a biblioteca síncrona, NanGate-FreePDK45 [23]. Essa distribuição permite a projetistas acesso facilitado a uma biblioteca assíncrona, o que auxilia no desenvolvimento de técnicas de projetos baseadas em circuitos assíncronos. Em última anáslise este trabalho é uma contribuição ao avanço no desenvolvimento de uso de projeto assíncrono e ao acesso a suas vantagens potenciais.

3. PROJETO DE CIS ASSÍNCRONOS BASEADO EM CÉLULAS

Este capitulo visa apresentar os conceitos básicos necessários para se criar uma biblioteca de células assíncronas, por isso ela abordará os métodos de projeto de circuitos integrados, o projeto de bibliotecas de células e as células assíncronas.

3.1 Métodos de projeto de CIs

O primeiro passo no projeto de um sistema digital é a definição do tipo de método de implementação que será utilizado para tal projeto, que pode ser *FPGA*, *gate array*, *cell-based*, *full-custom* entre outros, como vemos na Figura 3.1 [28]. Para que tal definição tenha lugar deve ser feita uma análise do custo-benefício de cada método, pois existem alguns que utilizam recursos pré-projetados, pré-caracterizados ou mesmo pré-fabricados e outras que não permitem o projetista ignorar nenhuma das etapas de fabricação.

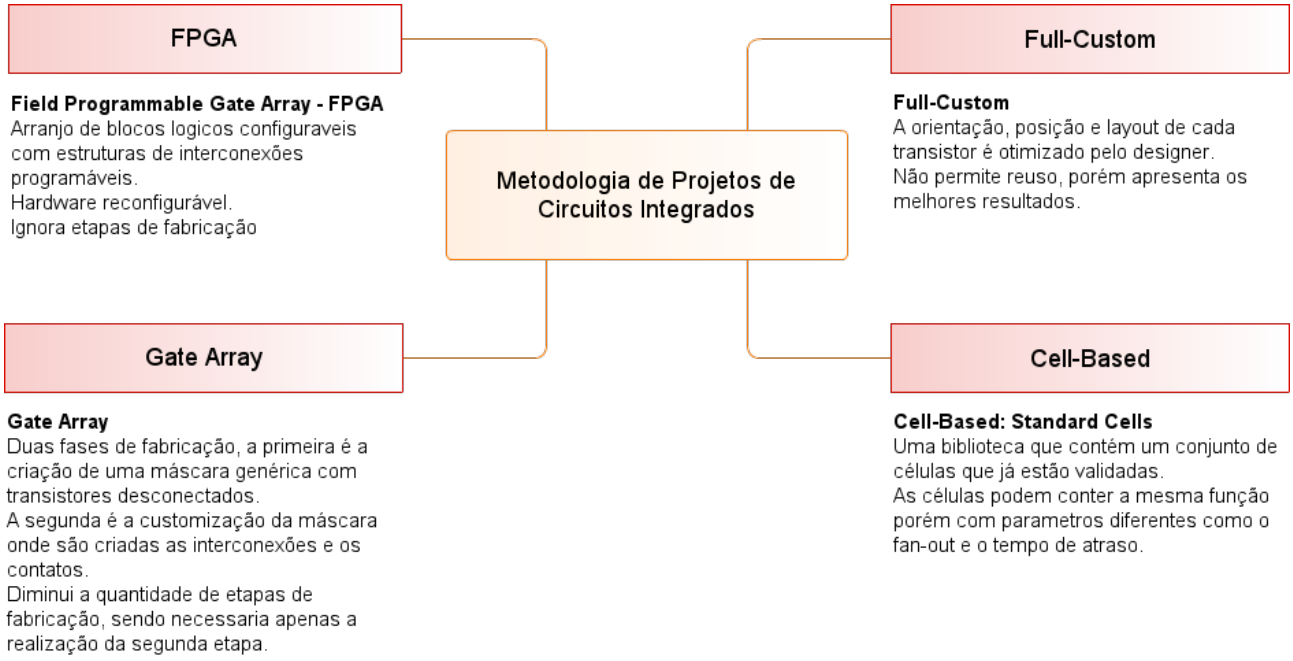


Figura 3.1 – Os diferentes métodos de projetos.

Um FPGA é um circuito integrado que pode ter sua funcionalidade configurada após a sua fabricação. Estes costumam ser formadas por três tipos de elementos: (i) uma grande quantidade de unidades lógicas configuráveis, que podem ser interligadas através de (ii) blocos de interconexão configuráveis internos ao CI. FPGAs possuem ainda interfaces de comunicação externas (pinos) que também podem ser configuradas. Todo o processo de configuração destes CIs inicia-se pelo uso de descrições de hardware em alto

nível. Este modelo apresenta uma bom desempenho com um nível de dificuldade de implementação baixo. Entretanto, o uso de FPGAs causa penalidades em termos de área e consumo de energia, a ponto de inviabilizar seu uso em aplicações com restrições elevadas de desempenho e/ou baixo consumo de energia, por exemplo.

O método mais flexível de projeto de CIs é o *full-custom*, que hoje é utilizado para implementar chips de altíssimo desempenho, tais como processadores comerciais de ponta e memórias semicondutoras. Neste estilo de projeto, o projetista controla o desenvolvimento desde o nível de transistores e não existe uma grande modularização do layout, que pode ser realizado manualmente, transistor a transistor. Por exemplo, este foi o método usado no projeto do primeiro microprocessador comercial, o Intel 4004. O projeto de um CI *full-custom* normalmente apresenta os melhores resultados em área, desempenho e consumo de energia em comparação com os demais métodos. Porém, este estilo de projeto requer uma mão de obra extremamente qualificada e deixa a desejar no reuso e modularidade. O tempo de projeto também pode ser proibitivo para muitas classes de aplicação.

Outro método existente são os *Cell-Based* que consistem na criação de um conjunto de células com funções lógicas diferentes, chamadas *standard-cells*. Este método foi criado para simplificar o processo de criação de CIs. Com o aumento de quantidade de transistores dentro de um chip, se tornou mais difícil a utilização do método*full-custom*. Para viabilizar a produção de circuitos com grande quantidades de transistores, divide-se o projeto em uma grande quantidade interconectada de blocos, que são as células. Células são circuitos de pequeno porte previamente projetadas, caracterizadas e testadas. Essas células são desenvolvidas utilizando o método *full-custom*, ou seja elas são dimensionadas, projetas e validadas individualmente, transistor a transistor. Este método é obviamente mais produtivo que o método *full-custom*, por ser baseado na reutilização maciça de componentes.

Uma biblioteca de *Standard-Cells* conta normalmente com uma grande variedade de portas lógicas, *flip-flops* e *PADs* de entrada e saída. Em alguns casos, podem ainda conter componentes mais complexos como *latches*, somadores, subtratores, etc. Estas bibliotecas apresentam células com bom rendimento em área, desempenho e dissipação de energia. O método Cell-Based tende a ser o mais utilizado na fabricação de chips pois permite ao projetista re-utilizar os layouts das células que já estão validadas para criar CIs, o que conduz a resultados satisfatórios comparados ao *full-custom*, e com resultados bastante superiores a utilizar FPGAs [29].

Existem outras metodologia de circuitos integrados como a utilização de microprocessadores que é a maneira mais rápida na criação de aplicações usando sistemas digitais. Existem diversas marcas de microprocessadores no mercado com configurações de clock e memória diversas, que permitem explorar otimizações para diferentes necessidades. Além disso, não é necessário um profissional extremamente qualificado em projeto de circuitos integrados para implementar aplicações usando este tipo de solução. Entretanto, o de-

sempenho e o consumo energético sofrem penalidades consideráveis nestes métodos de projeto [29].

3.2 PROJETO DE BIBLIOTECAS DE CÉLULAS

Para projetar uma biblioteca de células após a definição de quais células gerar, deve-se realizar o dimensionamento dos transistores. Para dimensionar uma célula existem vários métodos, tais como utilizar informações de *fan-out* para calcular o atraso da porta e assim dimensionar os transistores da cada célula. Outro método consiste em utilizar *software* tais como o ROGen [22], onde através de simulações com as células utilizando diferentes tamanhos de transistores se busca o melhor dimensionamento através de uma função de custo ou uma característica específica. Um exemplo de tal função é uma que avalie qual o melhor dimensionamento para obter a menor potência dinâmica.

Após realizar o dimensionamento, produz-se o *layout* das células. Para isto é necessária a criação de uma arquitetura padrão, com a definição de:

- Passo de roteamento vertical e horizontal;
- Altura da célula;
- Altura das linhas de alimentação;
- Dimensionamento das camadas de substrato.

Essas definições são escolhas feitas pelo projetista levando em consideração a tecnologia escolhida, já que cada uma destas apresenta regras distintas, o que é abordado em mais detalhe no Capítulo 5. Para finalizar esta etapa, realizam-se mais alguns passos de verificação, onde se inclui a conferência de que o *layout* respeita as regras da tecnologia, no que se denomina *Design Rules Check*; também se verifica que o *layout* corresponde a um circuito cuja funcionalidade é a mesma do esquemático original, no que se chama *Layout versus Schematic*.

Com o *layout* finalizado, pode-se dele extrair diversas informações, formatadas para diferentes funções. incluindo:

- Vista Abstrata
  É um representação abstrata do *layout*, onde se representa apenas um subconjunto de camadas. Normalmente representam-se aqui os terminais, os pinos, informação de bloqueios (*blockage*) de camadas de metal e os limites do *layout*. Esta vista é útil para o roteamento de um circuito que utiliza as células, pois ela mostra o posicionamento dos pinos de entrada e saída de cada célula e a abstração reduz o uso de recursos

computacionais no roteamento do circuito. Normalmente, essa vista é gerada em formato LEF da Cadence [30].

- Descrição Pós-Síntese
  Para simulações e análises mais precisas do comportamento de um CI são necessários, além da descrição do circuito a nível de transistores, os valores das cargas parasitas presentes dentro do *layout*. Cargas parasitas são informações sobre efeitos indutivos, resistivos ou capacitivos gerados não somente pelos transistores da célula mas também por outras partes da mesma como nos terminais e fiação. Para se extrair essa descrição o programa mais adotado é o Calibre PEX da Mentor, que gera um arquivo Spice com os parasitas.

O último passo para a criação de uma biblioteca é a sua caracterização, onde são feitas as medidas elétricas e temporais de cada célula da biblioteca. As células caracterizadas servem de guia para as ferramentas utilizadas no fluxo de criação de CIs, da síntese lógica à síntese física. A caracterização deve ser feita levando-se em conta variações de processo, temperatura e tensão de alimentação, e a qualidade da biblioteca depende diretamente da qualidade da caracterização. O modelo mais utilizado para criar a biblioteca caracterizada é o Liberty [31].

A qualidade de uma célula pode ser avaliada de acordo com diversas características, tais como:

- Cargas parasitas:
  São as cargas que podem ser indutivas, resistivas ou capacitivas presentes não somente no transistor, mas também em outras partes da célula, como os terminais.

- Capacitância de entrada:
  A capacitância mínima que necessita ser carregada para que uma célula propague o efeito de um sinal em sua(s) entrada(s).

- Atraso de propagação:
  O tempo que uma porta lógica leva para trocar a saída devido a uma troca de valores em alguma de suas entradas.

- Atraso de transição
  É o tempo que leva para uma saída chavear de um estado para outro, 1 para 0 e 0 para 1.

- Consumo estático:
  O consumo que ocorre mesmo quando não há chaveamento; é calculado multiplicando-se a tensão de alimentação pela corrente de fuga.

- Consumo dinâmico:
  É a dissipação de potência em função dos chaveamentos das portas.

3.3 Células Assíncronas Típicas

Circuitos assíncronos necessitam de elementos de base que permitam a construção de blocos que se comunicam através de protocolos de *handshake* local. Tais elementos são chamados de células assíncronas e sua natureza e composição variam conforme o estilo de projeto utilizado e os níveis de otimização explorados. Este trabalho aborda as seguintes classes de componentes:

- Muller C-elements

  São componentes utilizados em circuítos assíncronos para construir sincronizadores de eventos. Este elemento foi proposto por Muller e Bartky em [12] e opera conforme exemplificado na Tabela 3.1 para o C-element mais simples. Neste, para que a saída esteja em nível lógico 1 todas entradas devem estar em 1, e para que a saída esteja em nível lógico 0 todas entradas devem estar em 0. Nos demais casos a saída permanece com seu valor anterior. *C-elements* são elementos fundamentais em diversos *templates* assíncronos. Somente no laboratório GAPH da PUCRS foram utilizados , por exemplo, na implementação de redes intrachip [9][32][33][34] e em projeto de processadores assíncronos utilizando o template Blade [15]. Fora isso, foram criadas outras aplicações como o ARM assíncrono [35], também circuitos AES utilizando NCL [36].

A	B	Y_n
0	0	0
0	1	Y_{n-1}
1	0	Y_{n-1}
1	1	1

Tabela 3.1 – Tabela verdade de um C-element convencional de 2 entradas A e B e saída Y. O índice nos valores de algumas saídas designa valores atuais (n) e anteriores ($n-1$).

- NCL - Null Convention Logic

  Células NCL são utilizadas em diversos templates baseados na lógica NCL [10], que empregam os chamados *threshold gates*. Estas células possuem n entradas e sua funcionalidade depende de um limiar (*threshold m*, que determina a quantidade mínima de entradas em nível lógico 1 para que a saída vá para o nível lógico 1. Além disso, caso todas entradas estejam em nível logico 0 a saída vai para o nível logico 0. Para os demais casos, a saída mantém seu valor anterior. A título de exemplo, a Tabela 3.2 mostra a funcionalidade de um NCL de 3 entradas e *threshold* 2, ou seja: A saída vai para o nível lógico 1 quando 2 ou mais entradas estão em 1.

A saída vai para o nível lógico 0, quando todas as 3 entradas estão em 0.
Em outros casos, a saída mantém o valor anterior (Y_{n-1}).
Células NCL despertaram recentemente o interesse comercial da empresa *Wave Semiconductor*, que pretende comercializar *Azure* um CI a ser projetado com lógica NCL [37]. Além disso diversos CIs em lógica NCL foram projetados pela empresa Theseus no passado [10] [38].

A	B	C	Y_n
0	0	0	0
0	0	1	Y_{n-1}
0	1	0	Y_{n-1}
0	1	1	1
1	0	0	Y_{n-1}
1	0	1	1
1	1	0	1
1	1	1	1

Tabela 3.2 – Tabela verdade de uma porta NCL2-de-3 (porta de 3 entradas com *threshold* 2).

- NCL+ - Return-to-One Null Convention Logic

  Portas NCL+ são semelhantes a portas NCLs, porém com suporte ao protocolo *return-to-one* ao contrário das NCL que assumem um protocolo *return-to-zero* [39]. Essas células possuem um comportamento dual em relação a portas NCL, conforme representa o exemplo da Tabela 3.3. Uma célula NCL+ [11] tem *n* entradas e *threshold m*, onde *m* determina o quantidade mínima de entradas em nível lógico 0 para que a saída vá para nível lógico 0. Caso todas entradas estejam em nível logico 1 a saída vai para o nível logico 1. Para outros caso a valor anterior da saída é mantido. A Tabela 3.3 mostra que:
  A saída recebe o nível lógico 0 quando 2 ou mais entradas estão em 0.
  A saída recebe o nível lógico 1 apenas quando todas as 3 entradas estão em 1.
  Nos demais casos, a saída mantém o seu valor anterior (Y_{n-1}).
  Células NCL+ foram propostas para ser uma alternativa à utilização do NCL em [18]. Verificou-se que a utilização do protocolo RTO apresenta maior eficiência no consumo de potência estática, sendo asssim NCL+ uma opção para utilização em circuitos assíncronos. A análise conduzida em [16] ressalta a utilização do NCL+ em projetos de circuitos assíncronos que utilizam o *template* SDDS-NCL, que combina lógicas NCL e NCL+. O estudo de caso daquela referência foi um somador *Kogge-Stone* de 8 bits utilizando este *template* [40].

- MUTEX

A	B	C	Y_n
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	Y_{n-1}
1	0	0	0
1	0	1	Y_{n-1}
1	1	0	Y_{n-1}
1	1	1	1

Tabela 3.3 – Tabela verdade de uma porta NCL+2-de-3 (porta de 3 entradas com *threshold* 2).

O MUTEX é um elemento de exclusão mútua utilizado na construção de árbitros. Este elemento permite lidar com múltiplas requisições em sistemas que não possuem uma referência temporal comum, controlando a introdução de metaestabilidade no circuito.

- Outros elementos

  Dependendo do projeto e do *template* utilizado na construção de um CI assíncrono, outros tipos de células podem ser necessárias tais como: células dinâmicas PCHB, LSSD, toggle, Q-flops, Sleep Convention Logic, etc.

4. FLUXO ASCEND

O fluxo ASCEnD-A [22], cuja visão global aparece na Figura 4.1, é um fluxo que visa a criação de componentes necessários para o projeto de circuitos assíncronos no nível de célula. Em outras palavras, ele apoia a construção de bibliotecas de células para suporte ao projeto de circuitos assíncronos no estilo semi-dedicado de projeto (*semi-custom*. O fluxo ASCEnD-A possui quatro macro etapas (mostradas na Figura 4.1 como retângulos de cantos arredondados coloridos): (i) Especificação, (ii) Dimensionamento, (iii) Design e (iv) Caracterização. Estas etapas foram criadas de acordo com a sua funcionalidade no fluxo ASCEnD-A. Na figura 4.2 quando se trata em *Cell Library Templates* se refere a especificação das células da biblioteca, na ilustração 4.3 *Cell Sizing* é o processo de dimensionamento das células, já na imagem 4.4 *Cell Layout* se refere a criação do *layout*, também chamado de *design* e na figura 4.5 *Cell Characterization* é a etapa de caracterização da células.

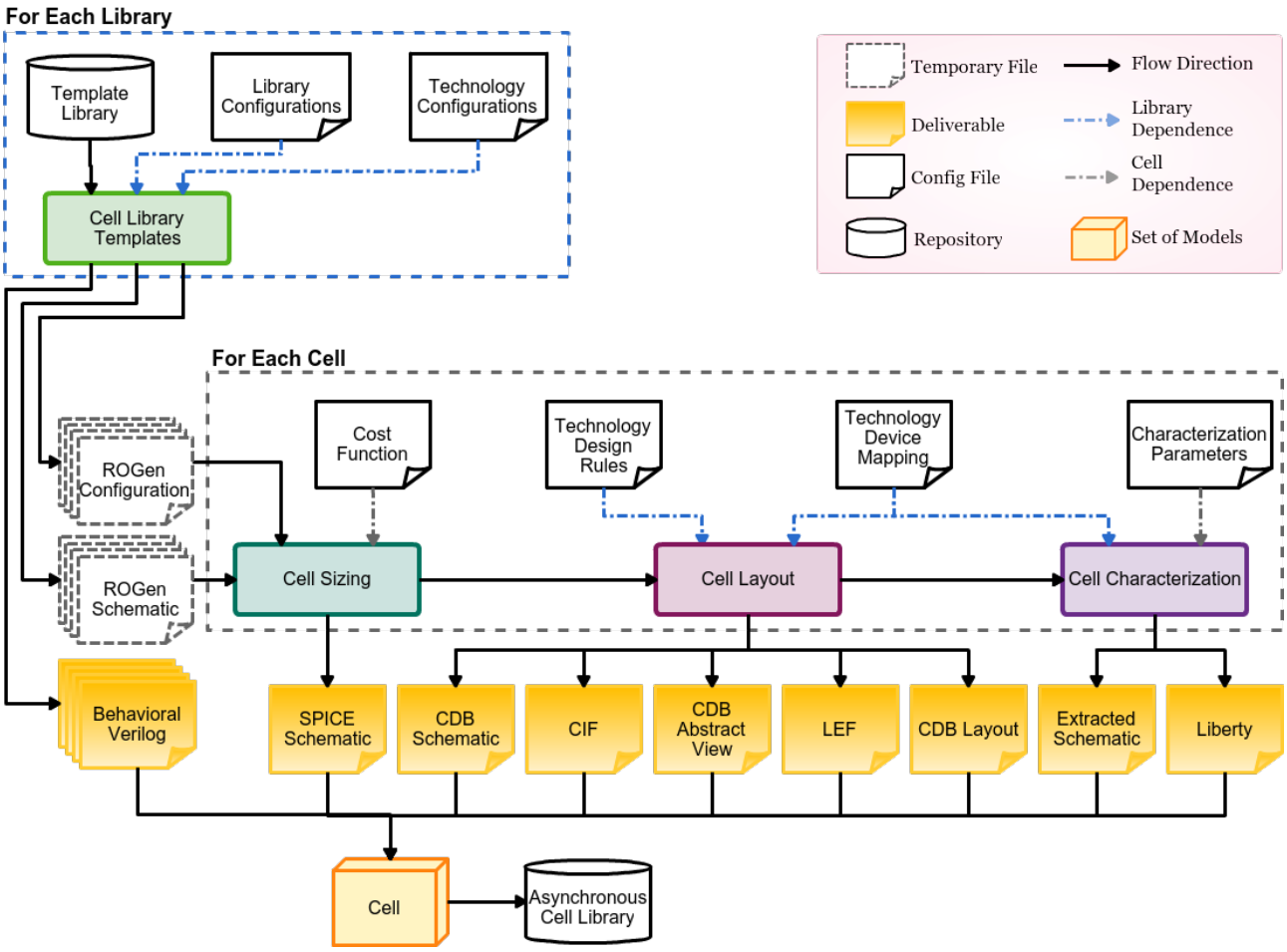


Figura 4.1 – Visão geral do fluxo ASCEnD-A de projeto de células. [1]

4.1 Especificação

O primeiro passo do fluxo é a configuração do ambiente, que consiste na escolha e especificação da tecnologia que o fluxo utilizará e também a escolha de quais células serão geradas. Para cada função lógica, cria-se um *template* descrito em SPICE com os parâmetros necessários para a execução das ferramentas e scripts do fluxo criarem as células de acordo com sua configuração, estes passos estão exibidos na figura 4.2

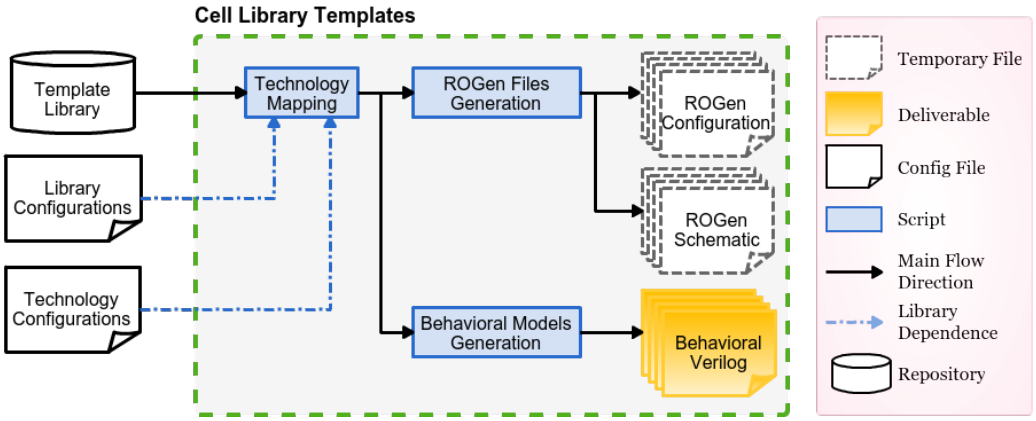


Figura 4.2 – Etapa de configuração do ambiente e a especificação do circuíto. [1]

4.2 Dimensionamento

Para dimensionar célula criadas foram propostas duas ferramentas: ROGen e CeS [41]. ROGen cria um oscilador em anel com várias instâncias da célula a ser dimensionada, para que possa fazer a simulação deste circuito. A cada simulação as dimensões dos transistores da célula projetada são variadas de acordo com configurações definidas pelo projetista. As simulações utilizam a ferramenta Spectre/MMsim da Cadence. ROGen é configurável através de um arquivo que especifica as diversas características do ambiente que será gerado. Este arquivo se baseia nos parâmetros mostrados na Tabela 4.1. O arquivo de saída do ROGen é o arquivo com os resultados da simulação. Este arquivo é a entrada de uma segunda ferramenta, chamada CeS (de *Cell Specifier*), que seleciona os melhores resultados de acordo com uma função de custo, definida pelo projetista. Analisando os resultados, a ferramenta indica os tamanhos de transistores a serem utilizados, concluindo o dimensionamento do circuito.

Para realização dessa etapa são necessárias três entradas:

- Descrição SPICE do circuito com os parâmetros do ROGen.
- Arquivo de configuração do ROGen.

- Função custo, que pode ser criada a partir de 16 parâmetros do CeS, tais como frequência e potência dinâmica.

Este fluxo pode ser visualizado em 4.3.

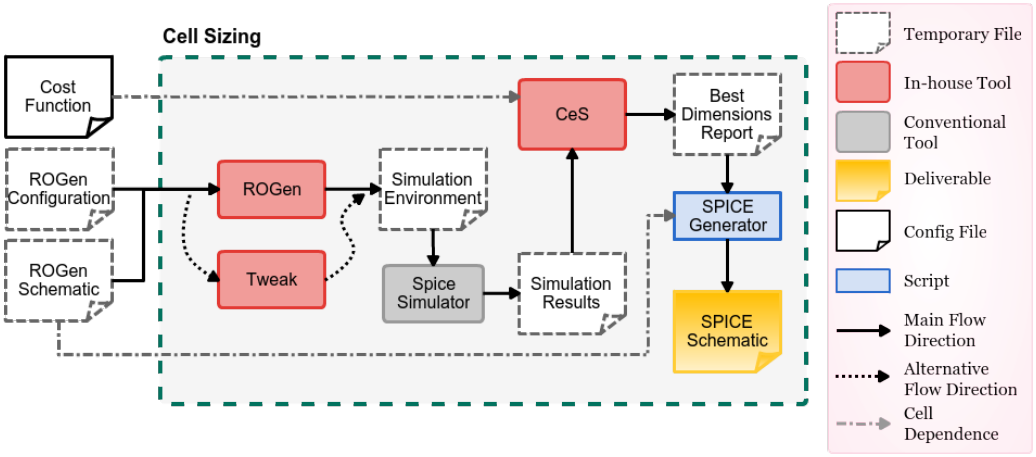


Figura 4.3 – Etapa de dimensionamento do circuito. [1]

4.3 Design

Com a descrição SPICE da célula dimensionada, o fluxo ASCEnD-A [22] é capaz de gerar o *layout* desta utilizando a ferramenta ASTRAN [27], uma ferramenta desenvolvida pelo Grupo de Microeletrônica da UFRGS, que gera automaticamente células e que dá suporte a nodos tecnológicos STM-65 [22] e FreePDK45 [42], entre outros. Após isso, o layout gerado é importado, juntamente com esquemático para o ambiente CADENCE Virtuoso. A Figura 4.4 mostra todos passos necessários para realizar esta etapa. Note que, para finalização dos *layouts*, são necessárias as verificações que garantem que o layout respeita as regras de projeto, do inglês *Design Rules Check*, e se o esquemático e o layout correspondem à uma mesma funcionalidade usando a ferramenta *Layout Versus Schematic*. Após a verificação completa da célula é feita a extração desta para o formato LEF.

4.4 Caracterização

Para a caracterização de células assíncronas foi necessário realizar adaptações, pois estas apresentam características que impedem que ferramentas de caracterização comerciais de reconhecê-las adequadamente. Devido a isso, foi criado o programa Li-ChEn [43], uma ferramenta que permite a partir do *netlist* extraído do *layout* realizar a caracterização da célula. Neste processo exibido na figura 4.5, os dados de tempo e potência

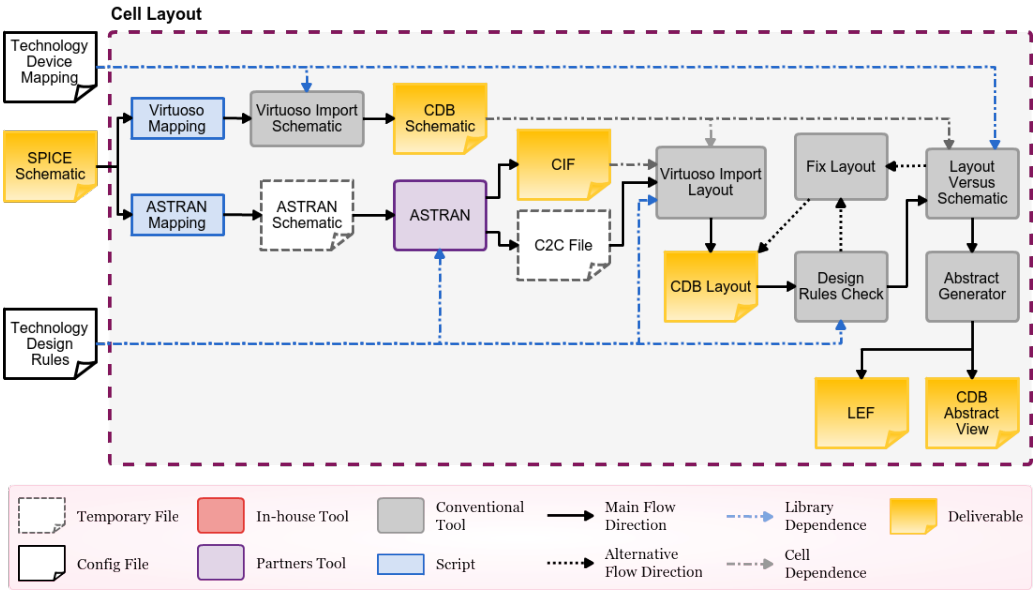


Figura 4.4 – Detalhamento da etapa onde se realiza a criação e importação do layout e do esquemático. [1]

de cada célula são exportados para ao formato *Liberty* [31]. Para finalizar a biblioteca é necessária a criação de alguns modelos como o *Verilog comportamental*, *abstract* e *symbol*, que são gerados utilizando ferramentas convencionais ou reaproveitando *templates* disponíveis no fluxo ASCEnD-A.

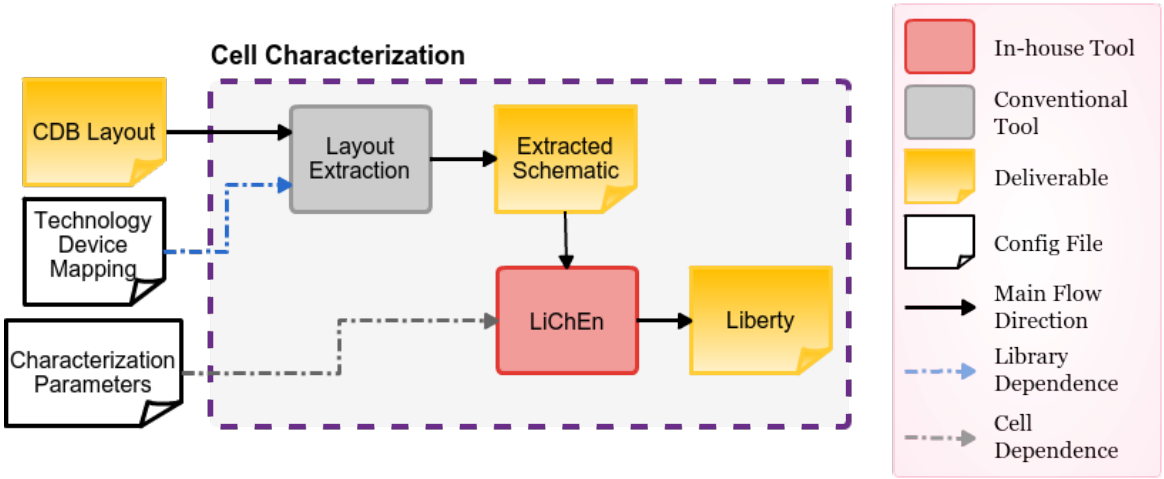


Figura 4.5 – Fluxo onde se caracteriza uma biblioteca. [1]

Parâmetros	Descrição
$tech	Tecnologia
$models	Biblioteca da tecnologia para o simulador
$models_lang	Linguagem do simulador
$nand	Nome da NAND, posição das entradas, saídas, gnd,vdd
$nand_file	Arquivo Spice da NAND
$inv	Nome do INVERSOR posição das entradas, saídas, gnd,vdd
$inv_file	Arquivo Spice do Inversor
$c_el_num_in	Quantidade de entradas do C-Element
$c_el_config	Sinais de controle: Set(s) ou Reset(r) ou Set and Reset(rs) ou Nothing(n)
$c_el_rst	Reset ativo alto/baixo
$c_el_set	Set ativo alto/baixo
$c_el_inv	Saída invertida
$c_el	Nome do C-Element, posição das entradas, saídas,gnd,vdd
$c_el_file	Arquivo Spice do C-Element
$c_el_num	Quantidade de C-Elements no anel
$volt	Tensão de operação
$leak_start_time	Início da medição de *leakage*
$leak_end_time	Final da medição de *leakage*
$dyn_start_time	Início da medição de potência dinâmica
dyn_end_time	Final da medição de potência dinâmica
$dyn_switch_event	Ciclos até iniciar medições
$time_unit	Unidade de tempo
$sim_step	Incremento da simulação
$sim_time	Tempo de simulação
$in_slew	Rampa de entrada
$rise_trans	Tensão transição de subida
$fall_trans	Tensão transição de descida
$log_1	Tensão do nível lógico 1
$log_0	Tensão do nível lógico 0
$max_1fing_n	Tamanho máximo de um *finger* NMOS
$max_1fing_p	Tamanho máximo de um *finger* PMOS
$min_n	Tamanho mínimo NMOS
$max_n	Tamanho máximo NMOS
$n_step	Incremento NMOS
$min_p	Tamanho Minimo PMOS
$max_p	Tamanho Máximo PMOS
$p_step	Incremento PMOS
$temp	Temperatura de simulação
$fan_out	Fan out
$threshold	*Threshold*(NCL)
$protocol	0 RTZ, 1 RTO

Tabela 4.1 – Parâmetros de configuração da ferramenta ROGen.

5. IMPLEMENTAÇÃO DA BIBLIOTECA ASCEND-FREEPDK45

5.1 Especificação das células

A escolha das células da ASCEnD-FreePDK45 foi feita com base no fluxo SDDS-NCL [16] [44] que utiliza células NCL e NCL+ para a síntese de circuitos assíncronos. A composição da biblioteca inclui células NCL e NCL+ de 2 e 3 portas de entradas, quando sua função contém *set* ou *reset*, conforme se vê na Figura 1.1. Com essa configuração tem-se suporte ao projeto de circuitos QDI usando o fluxo SDDS-NCL proposto pelo GAPH e que será utilizado para validação da biblioteca.

A biblioteca conta com células com 3 capacidade de corrente distintas (em inglês, *driving strengths* ou simplesmente *drives*): X1,X2 e X4. A capacidade de corrente tem relação com a capacidade de carga da célula e com unidades de tempo. Por exemplo, com *drive* X4 uma célula pode carregar 4 vezes mais portas que uma célula com *drive* X1 em tempo similar. Note que isso influencia no dimensionamento da células e sua respectiva carga. Todas as simulações para dimensionamento foram baseadas nos valores de carga apresentados na biblioteca NanGate-FreePDK45, onde a capacidade de carga de uma célula X1 da ASCEnD-FreePDK45 será a mesma de uma célula X1 da NanGate-FreePDK45. Conforme vemos nas Figuras 5.1, 5.2 e 5.3, a mesma funcionalidade INCLP1W11OF2 corresponde a células cujo *layout* varia de acordo com a variação do *drive*. No *layout*, nota-se isso através da variação da altura da difusão, em verde nas Figuras.

5.2 Definição de regras de projeto

A FreePDK45 [21] tem seu conjunto de regras de projeto apresentado no Anexo A. Nesta Seção apresenta-se um conjunto básico de regras para as seguintes camadas: Polissilício, Difusão, Contato e Metal1. Essas regras foram utilizadas como base para o projeto da biblioteca ASCEnD-FreePDK45.

Para a geração de *layouts* com o ASTRAN [27] é necessário configurar o arquivo de regras da tecnologia de referência. As próximas figuras mostram as regras da FreePDK45 e seu respectivo parâmetro no ASTRAN. Um modelo 3D simplificado de um transistor CMOS é mostrado na Figura 5.4, adaptada de [45], e tem três elementos básicos, a porta, o dreno e a fonte (em inglês: *gate*, *drain* e *source*). A camada em vermelho é chamada de polissilício e corresponde à porta do transistor. Os pinos de fonte e dreno estão conectados à camada de difusão, em verde. Para conectar os pinos e as camadas utiliza-

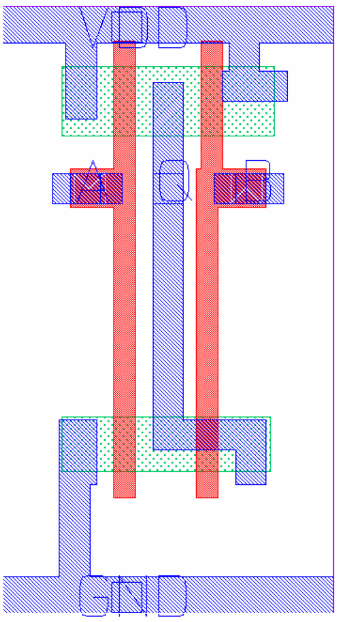


Figura 5.1 – *Layout* da célula INCLP1W11OF2X1
.

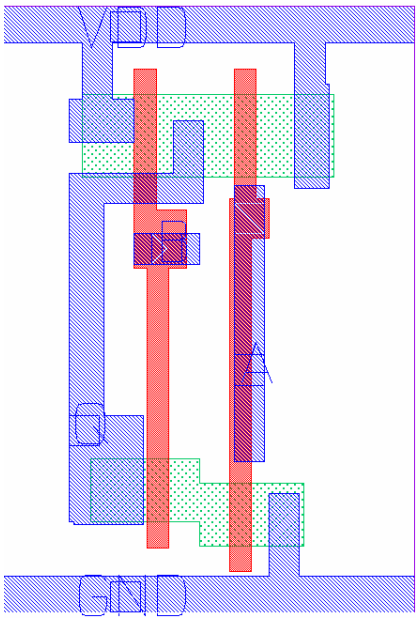


Figura 5.2 – *Layout* da célula INCLP1W11OF2X2
.

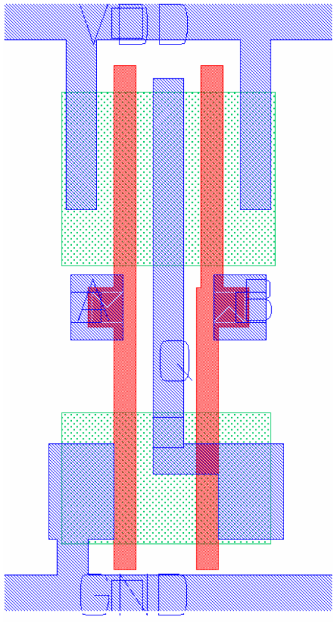


Figura 5.3 – *Layout* da célula INCLP1W11OF2X4
.

se contatos, e para fazer as ligações entre os diversos pontos do circuito pode-se utilizar diversas camadas como o polissilício, metais ou vias.

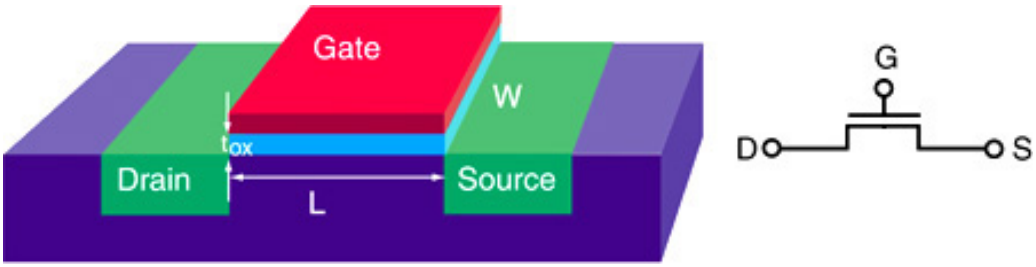


Figura 5.4 – Modelo 3D simplificado de um transistor MOS.

As próximas paginas detalham as regras para cada uma das camadas citadas acima. As Figuras 5.5, 5.6, 5.7 e 5.8 foram adaptadas de [46].

- Polissilício

  O polissilício é uma camada de projeto utilizada para gerar a porta do transistor. NA confecção do *layout* de células, quando essa camada é colocada sobre uma camada de difusão, forma-se um transistor. De acordo com sua ativação, este dispositivo permite ou não a condução de corrente entre o dreno e a fonte do transistor, e a largura da camada de difusão largura determina a largura do transistor (W). Na Figura 5.5 existem duas camadas de polissilício (em vermelho) sobre a difusão(em laranja), formando assim dois transistores. Nessa tecnologia o menor comprimento (L) de um transistor é 50 nm, vide a regra POLY.1 da Tabela 5.1.

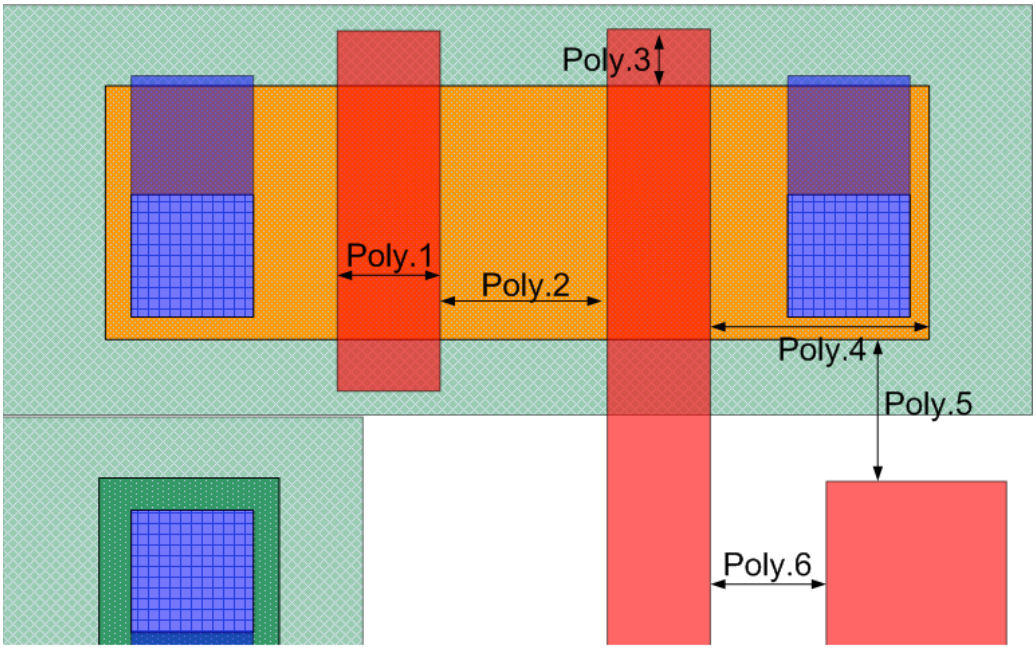


Figura 5.5 – *Layout* com as as medidas das regras de projeto do *gate*.

FreePDK45 Regra	ASTRAN Parâmetro	Valor	Descrição
POLY.1	E1P1DF	50 nm	Largura mínima do polissilício
POLY.2	S2P1P1	140 nm	Distância mínima entre dois gates sobre uma difusão.
POLY.3	E1DFP1	55 nm	Altura mínima do gate após a difusão.
POLY.4		70 nm	Distância mínima entre um gate e o fim da difusão.
POLY.5	S1DFP1	50 nm	Distância mínima entre a difusão e o polissilício.
POLY.6	S1P1P1	75 nm	Distância mínima entre dois polissilícios.

Tabela 5.1 – Regras de *design* do *gate*.

- Difusão
  A camada de difusão é onde estão ligados o dreno e a fonte do transistor. Sua altura determina, entre outras características, a capacitância máxima de saída do transistor, e é essa altura que é dimensionada durante a etapa de dimensionamento das células. A difusão pode ser polarizada tanto para o substrato NWELL ou PWELL. Dependendo da polarização, a ativação da porta ocorre ou no nível lógico 0 ou no nível lógico 1. A regra ACTIVE.4, da Tabela 5.2, ressalta a importância da difusão estar dentro dos substratos, como ocorre na Figura 5.6.

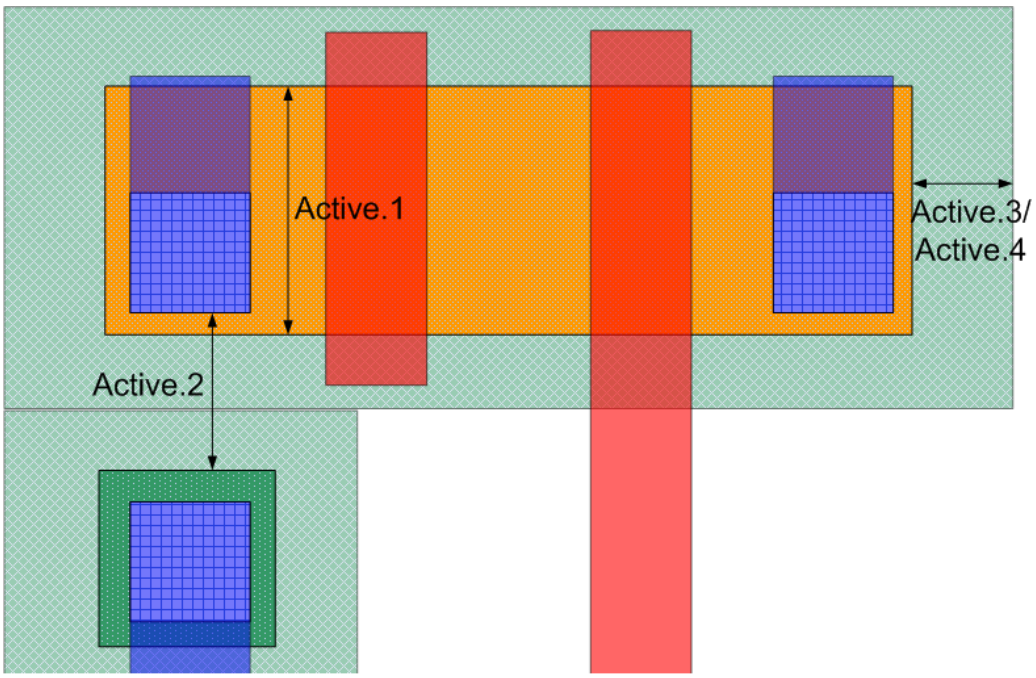


Figura 5.6 – *Layout* com as as medidas das regras de projeto para a camada de difusão.

FreePDK45 Regra	ASTRAN Parâmetro	Valor	Descrição
ACTIVE.1	W2DF	90 nm	Altura mínima da difusão.
ACTIVE.2	S1DFDF	80 nm	Distância mínima entre difusões.
ACTIVE.3	E1INDF E1IPDF	55 nm	Distância mínima emtre a difusão e o fim do NWELL ou PWELL.
ACTIVE.4	Nenhum		Nota: A difusão deve estar dentro do NWELL ou PWELL.

Tabela 5.2 – Regras de projeto para a camada de difusão.

- Contato
  Esta camada é responsável por fazer a ligação entre camadas distintas, como uma ligação entre a difusão e metal1, mostrado sobre o marcador CONTACT.1, na Figura 5.7. A distância mínima de uma camada em torno do contato pode variar, de acordo com o tipo da camada. No caso da FreePDK45 não se altera como se pode notar nas regras CONTACT.4 e CONTACT.5 da Tabela 5.3, onde seja para difusão ou para polissilício, a distância mínima é a mesma.

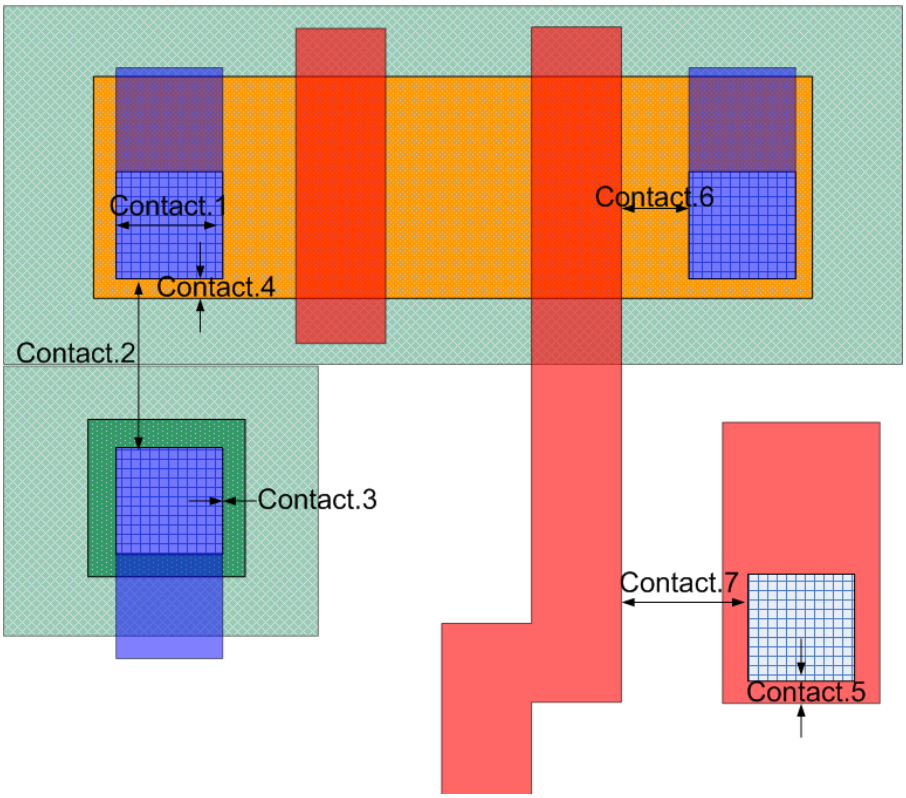


Figura 5.7 – *Layout* com as as medidas das regras de projeto para a camada de contato.

FreePDK45 Regra	ASTRAN Parâmetro	Valor	Descrição
CONTACT.1	W2CT	65 nm	Altura mínima do contato.
CONTACT.2	S1CTCT	75 nm	Distância mínima entre contatos.
CONTACT.3	Nenhum		Nota: o contato deve estar dentro de metal1, polissilício ou difusão.
CONTACT.4	E1DFCT	5 nm	Extensão mínima da difusão ao redor do contato.
CONTACT.5	E1P1CT	5 nm	Extensão mínima do polissilício ao redor do contato.
CONTACT.6	S1CTP1	35 nm	Distância mínima entre contato e gate.
CONTACT.7		90 nm	Distância mínima entre um contato sobre o polissilício e outro polissilício.

Tabela 5.3 – Regras de projeto para a camada de contato

- Metal1
  Camada que interliga diversos pontos de um *layout*, fazendo ligações com polissilício e difusão. Essa camada é também onde são conectados os pinos de entrada e saída e aquela na qual são confeccionadas as bandas de alimentação. A Tabela 5.4 mostra as regras de projeto para utilização do polissilício, referenciando as marcações feitas na Figura 5.8.

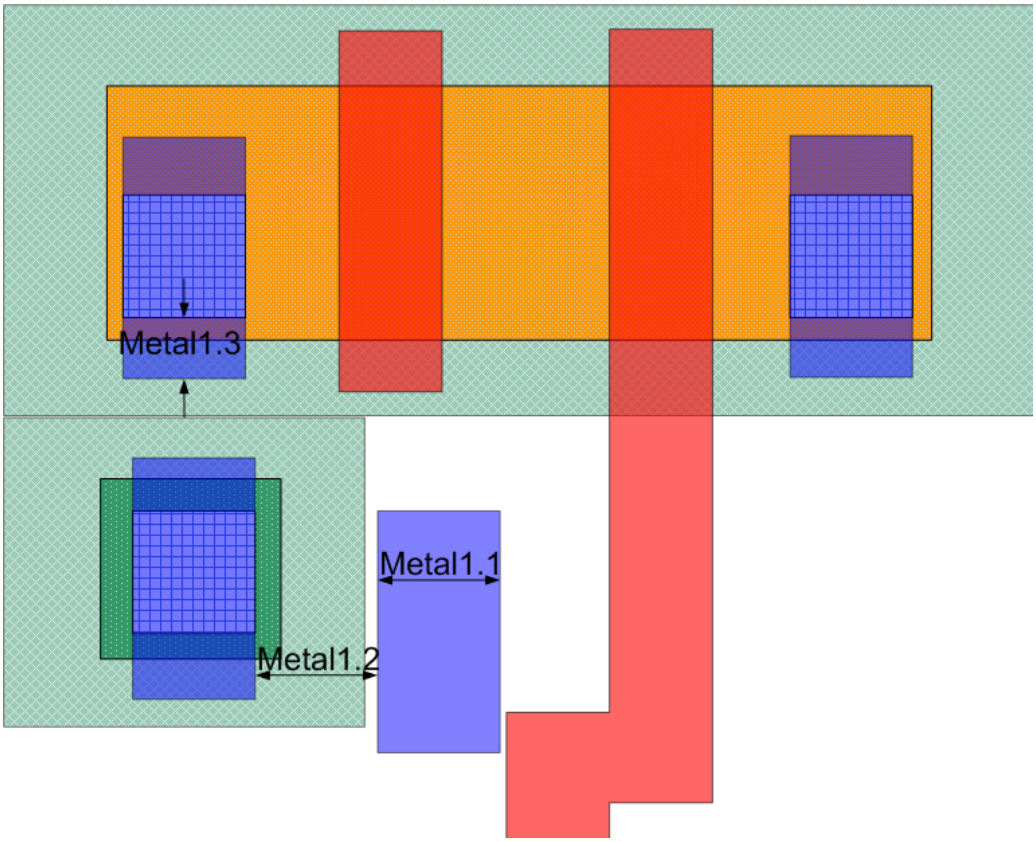


Figura 5.8 – *Layout* com as as medidas das regras de projeto para a camada de metal1.

FreePDK45 Regra	ASTRAN Parâmetro	Valor	Descrição
METAL1.1	W1M1	65 nm	Largura mínima de metal1
METAL1.2	S1M1M1	65 nm	Distância mínima entre metal1
METAL1.3	E2M1CT	35 nm	Extensão mínima de metal1 ao redor de dois lados opostos de um contato.
METAL1.4	E2M1VI	35 nm	Extensão mínima de metal1 ao redor de dois lados opostos de uma VIA1.
METAL1.5	S2M1M1	90 nm	Distância mínima entre metal1, com um deles apresentando largura maior que 90 mn.

Tabela 5.4 – Regras de *design* do Metal1

5.3 Definição da arquitetura de células

A arquitetura da biblioteca ASCEnD-FreePDK45 foi proposta visando a compatibilidade com as células da biblioteca NanGate-FreePDK45 [23], de forma a permitir a criação de projetos de CIs utilizando as duas bibliotecas. Dessa forma, células já implementadas na NanGate-FreePDK45, como *buffers*, inversores. portas lógicas, etc. podem ser reutilizadas em projetos de circuitos assíncronos. A arquitetura proposta para as células da biblioteca ASCEnD-FreePDK45 é apresentada na Figura 5.9. Outros fatores, tais como o tamanho máximo de um *finger* de um transistor também são idênticos aos equivalentes da biblioteca de células da NanGate-FreePDK45.

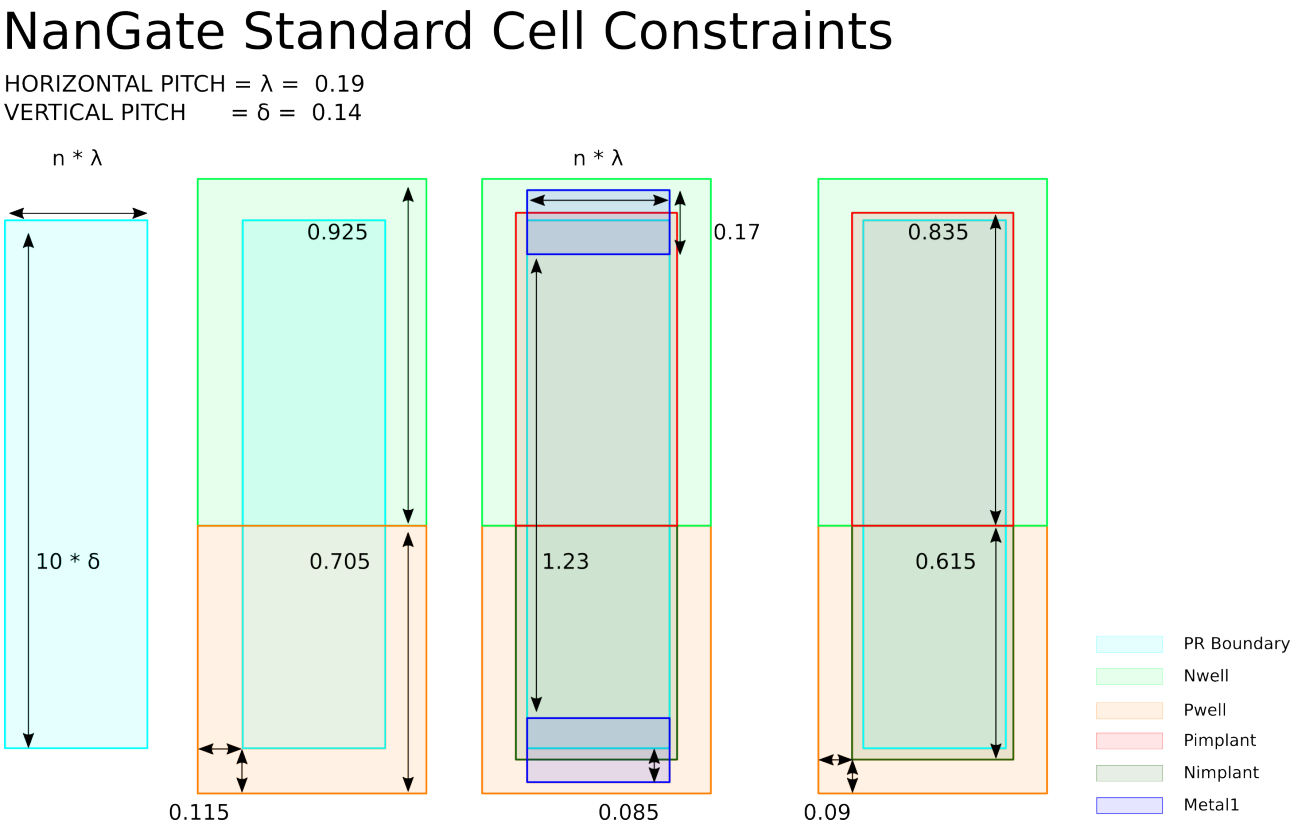


Figura 5.9 – Arquitetura geral das células da biblioteca ASCEnD-FreePDK45 (idêntica à da biblioteca NanGate-FreePDK45).

Quando um transistor tem dimensionamento maior que o tamanho máximo de um *finger*, é feito um processo chamado *folding*, onde seu dimensionamento é dividido por 2 e ao invés de ter apenas um transistor *layout*, criam-se dois transistores em paralelo, cada um dos quais é chamado de *finger* como mostra a Figura 5.10. O *finger* determina a quantidade de vezes que uma camada paralela de polissilício atravessa a camada de difusão. Este processo pode ser repetido mais vezes até que o transistor fique com seu dimensionamento menor que o tamanho máximo de um *finger*.

O ASTRAN necessita de alguns valores presentes na arquitetura, incluindo:

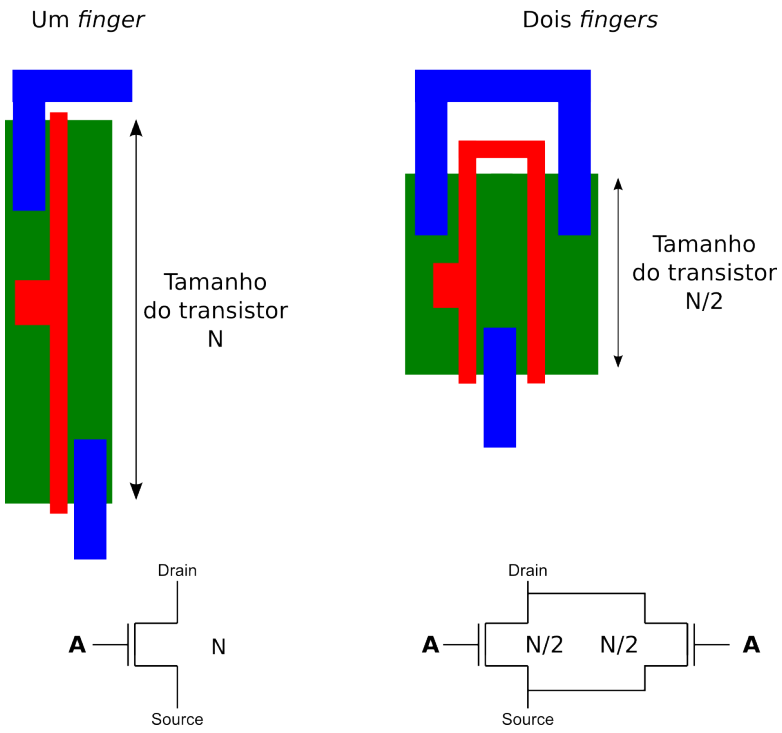


Figura 5.10 – Ilustração do processo de *folding* de um transistor.

- Tamanho das bandas de alimentação - na Figura 5.9 as camadas de metal1, são as bandas de alimentação onde serão conectados os pinos de alimentação, como se vê na Figura 5.1 onde estão os pinos VDD e GND;
- Altura da célula - esta altura é limitada pela camada de *prBoundary* na Figura 5.9;
- Grade de manufatura - é a menor resolução do processo de fabricação da tecnologia, na FreePDK45 é 2.5 nm;
- Distâncias das camadas de subtrato - as camadas de substrato são as camadas NWELL, PWELL,Nimplant e Pimplant que são utilizadas para polarizar os transistores que podem ser PMOS ou NMOS.

5.4 Adaptação do fluxo ASCEnD-a

O fluxo ASCEnD-A foi inicialmente desenvolvido para operar de forma genérica, mas teve como alvo incial o PDK da tecnologia STMicroelectronics 65nm. Neste trabalho, fez-se entretanto necessária a adaptação deste fluxo para o FreePDK45 [21]. Nem todas etapas do fluxo necessitaram de alterações. A maior parte das etapas apenas exigiu configurações distintas. Nas próximas Seções aborda-se os passos da criação da biblioteca, relatando eventuais dificuldades encontradas que exigiram adaptação do fluxo ASCEnD-

A. Em última análise, diversos *scripts* tiveram que ser reformulados e as configurações de tecnologia foram adaptadas.

5.4.1 Dimensionamento com ROGen e CeS

Antes de dimensionar a biblioteca, foi feita a especificação das células a serem geradas conforme a Seção 5.1 descreveu , junto com a definição da tecnologia alvo, a FreePDK45. Para dimensionar as células, outras escolhas tiveram de ser feitas. Para configurar o ambiente de simulação e de dimensionamento utilizado no ROGen, escolheu-se, por exemplo:

- Temperatura: 25 graus Celsius
- Tensão de alimentação: 1 volt
- Tamanho do *gate*: 50nm
- Largura máxima de um *finger* do transistor N: 300nm
- Largura máxima de um *finger* do transistor P: 400nm
- Incremento do tamanho dos transistores: 10 nm
- Tempo de simulação: 10 ns

Na especificação da biblioteca definiu-se também também quais seriam os *drives* a endereçar nesta versão da biblioteca. Com essa informação, ROGen foi utilizado para selecionar o tamanho mínimo e máximo dos transistores da *NanGate* [23], levando em consideração também o *drive* da célula.

Após sua execução, ROGen gera dois arquivos:

- Descrição Spice da célula com variáveis no tamanho do transistor - a Figura 5.11 apresenta um descrição Spice utilizada pelo ROGen, onde os transistores da *FreePDK* apresentam as variáveis de tamanho do transistor(pparam ou nparam) e a quantidade de *fingers* do transistor(pnum ou nnum).
- Arquivo de simulação - com as descrições Spice do inversor, da *nand2* e da célula em projeto gera-se o oscilador em anel para simulação. Este oscilador em anel é configurado seguindo a especificação do *fan out* e a quantidade de vezes que a célula deve ser repetida. Na Figura 5.12 essas configurações estão representadas nas áreas delimitadas por retângulos tracejados em azul e vermelho, respectivamente.

```
*****************************************************************
* Inverted NCL Gate 1w11, threshold: 1 weights [1, 1] inputs 2
* Generated by Matheus Trevisan Moreira
* GAPH/PUCRS - Porto Alegre, Brazil
* 2014-04-24 18:08:50
*****************************************************************
.subckt ST_INCL1W11OF2X1 A B Q GND GNDS VDD VDDS
*.pininfo A:I B:I Q:O GND:G GNDS:G VDD:P VDDS:P
MPS00 VDD  A    pl00 VDDS PMOS_VTL W=pparam m=pnum L=0.05u
MPS01 pl00 B    Q    VDDS PMOS_VTL W=pparam m=pnum L=0.05u
MNS00 Q    A    GND  GNDS NMOS_VTL W=nparam m=nnum L=0.05u
MNS01 Q    B    GND  GNDS NMOS_VTL W=nparam m=nnum L=0.05u
*Transistors count: NMOS 4 PMOS 4 TOTAL 8
.ends
*ROGen:T 1
*ROGen:I 2
*ROGen:P in in out gnd gnds vdd vdds
*ROGen:RP0 A B
*ROGen:RP1 A B
```

Figura 5.11 – Exemplo de descrição Spice para a porta INCL1111OF2X1, entrada do programa ROGen.

Para simular os arquivos gerados pelo ROGen, foi utilizada a ferramenta SPECTRE. A simulação foi executada utilizando computação em grid. Simulações são assim conduzidas em paralelo em diversos computadores e depois copiadas para o computador que contém o ambiente de simulação. Com as simulações prontas, utiliza-se o programa CeS para definir qual é o melhor dimensionamento para cada célula. Utilizou-se a Equação 5.1, que avalia a eficiência energética para o circuito. Nesta Equação, *propr* é o tempo de propagação de subida, *propf* é o tempo de propagação de descida e *dynpwr* é a potência dinâmica consumida.

$$\left(\frac{1}{\frac{propr + propf}{2} * dynpwr} \right) \tag{5.1}$$

Após selecionar o melhor resultado de cada simulação, o CeS gera um gráfico que mostra a evolução de valores da Equação, para cada dimensionamento dos transistores em cada simulação, e informa o melhor dimensionamento para aquela célula. Vale ressaltar que o CeS apresenta um comportamento não necessariamente monotônico, como se pode perceber na análise da Figura 5.13. Nota-se no gráfico alguns valores que não representam verdadeiramente o melhor dimensionamento, devido a inconsistências inevitáveis nos passos de variação do tamanho dos transistores *NMOS* e *PMOS*. Devido a isto, é necessário lançar mão da visualização do gráfico antes de confirmar o valor de dimensionamento para cada célula. O gráfico da Figura 5.13 mostra que o dimensionamento ideal

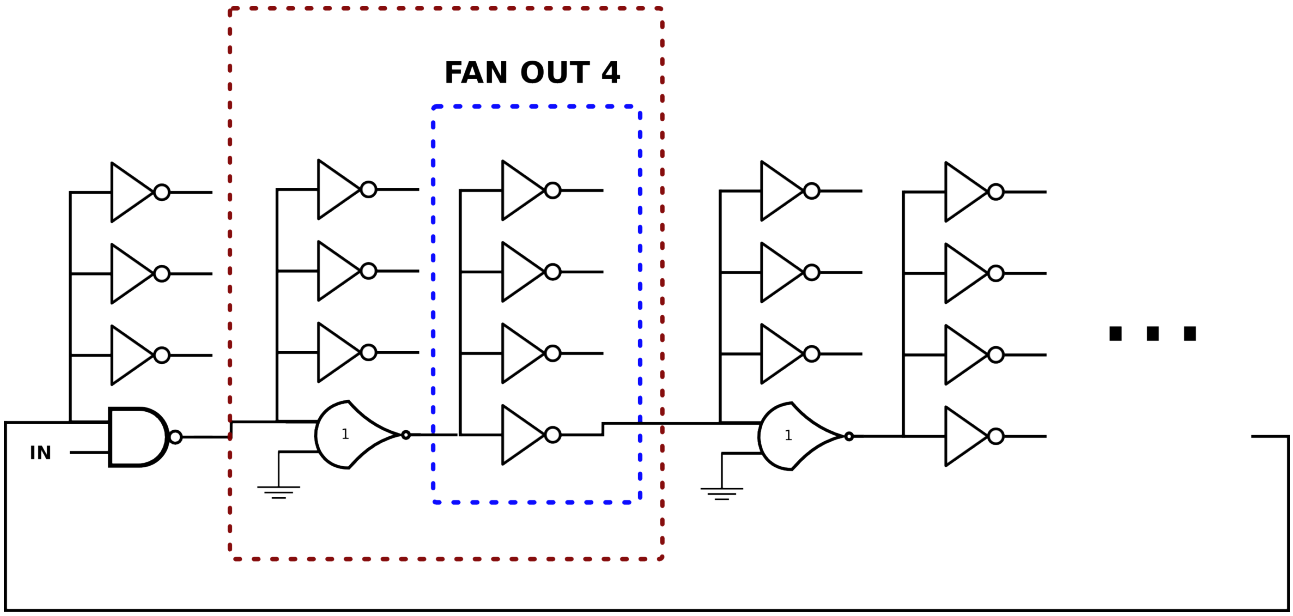


Figura 5.12 – Circuito de simulação gerado pelo ROGen utilizando o INCL1W11OF2X1, apresentando a configurações com *fan out* 4.

para a célula INCL1W11OF2X1 são transistores *PMOS* de 100nm e os *NMOS* de 85nm, e nota-se que por essa célula possuir *drive* X1, não requer transistores de grande porte para suprir o *fan out* 4 para si.

5.4.2 Criação de *layouts* utilizando ASTRAN

O programa ASTRAN [27] precisou ser adaptado em dois aspectos para ser compatível com a biblioteca ASCEnD-FreePDK45. A primeira adaptação foi torná-lo compatível com o FreePDK45 [21], o que foi bastante simples pois já haviam sido feitos testes usando o PDK. Porém, foram necessárias algumas alterações nos arquivos de regras do ASTRAN. Além disso foi necessária a migração da arquitetura do *layout* apresentada em 5.9 para que o ASTRAN gerasse as células de acordo com a estrutura da *standard cell* escolhida. A outra alteração, que demandou mais tempo de estudo e desenvolvimento de *scripts*, foi a migração para o Virtuoso 6.1. Essa ferramenta tem como principal mudança em relação as versões anteriores o fato de seu banco de dados ser o *Open Acess(OA) database*. Logo, todas as etapas do fluxo ASCEnD-A que se relacionavam com o ASTRAN precisaram ser refeitas.

O grande problema dessa etapa foi a importação do *layout* vindo do ASTRAN, pois o mesmo é gerado no formato CIF e o Virtuoso descontinuou o suporte a este formato para importação, fazendo-se necessária a importação utilizando arquivos no formato GDS. Porém, ASTRAN não gera *layouts* corretamente neste formato. A ferramenta não cria os pinos corretamente e a tabela de correspondência entre as camadas do GDS e as cama-

das do *layout* do FreePDK, mostrada na Tabela 5.5 era inconsistente. Após essas etapas de depuração, uma nova versão do ASTRAN foi disponibilizada pelos desenvolvedores da ferramenta e alguns ajustes foram realizados, incluindo a criação dos pinos em GDS e a adição da camada de *nimplant*. Com isso foi possível importar o *layout*, porém o *script* de importação teve que ser alterado para que passasse a fazer o escalamento do *layout*. Isso se deve ao fato de o arquivo GDS é gerado com uma escala 2.5 vezes menor que o tamanho real.

As figuras 5.14 mostram o processo de importação do *layout* onde a imagem a) mostra um arquivo GDS, b) a figura importada, e a c) após o escalonamento do *layout*.

Com o *layout* importado, a próxima etapa foi a correção de erros de *DRC*. Devido a este *PDK* não ser fabricável ele tem menos menos regras de projeto que PDK reais. Logo, a quantidade de erros de *DRC* é menor e a etapa de correção de erros foi conduzida de forma rápida. Entretanto, um erro do ASTRAN alterava a altura das camadas de *implant*, o que teve de ser verificado em cada layout e corrigido manualmente, quando necessário. As Figuras 5.15 e 5.16 mostram a correção de um erro de *DRC*.

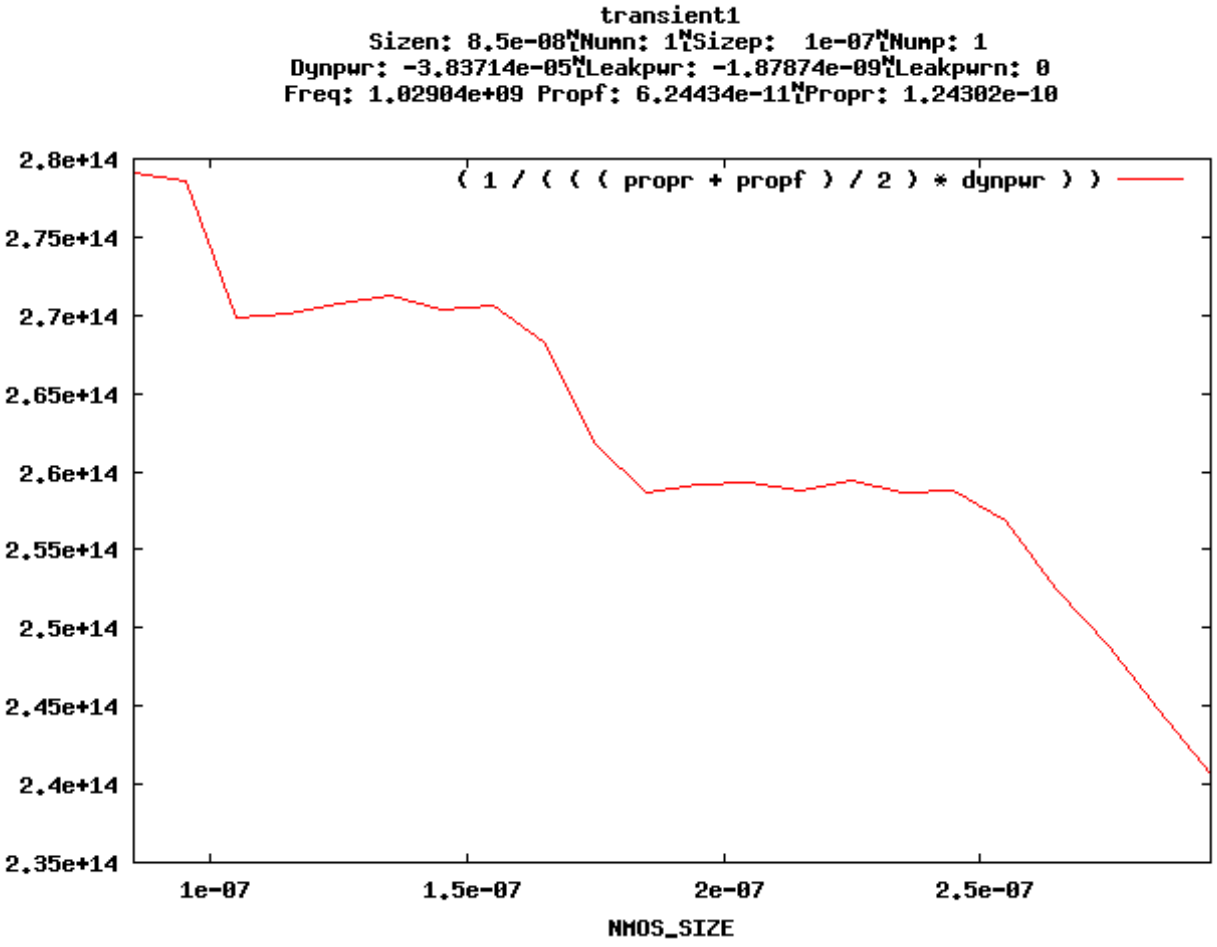
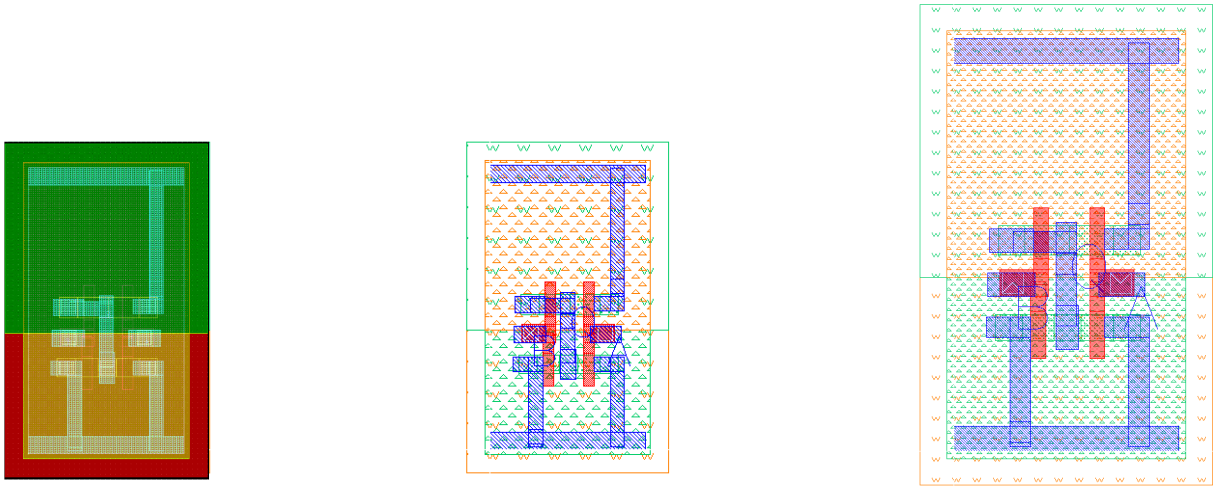


Figura 5.13 – Gráfico exportado do CeS, que mostra o melhor dimensionamento para a célula INCL1W11OF2X1.



(a) *Layout* em formato GDS, (b) *Layout* importado para o Virtuoso. (c) *Layout* com a escala correta.

Figura 5.14 – Processo de importação do *layout* de uma célula.

CAMADA VIRTUOSO	CAMADA GDS	TIPO DE CAMADA
active	1	drawing
nwell	3	drawing
nimplant	4	drawing
pimplant	5	drawing
poly	9	drawing
contact	10	drawing
metal1	11	drawing
prBoundary	235	drawing
metal1	18	pin
pwell	2	drawing

Tabela 5.5 – Tabela com as correspondências entre o Virtuoso e o arquivo GDS

O último passo na criação do *layout* é a criação da vista abstrata, que fornece apenas as posições dos pinos dentro de cada célula e metais de roteamento interno, como se constata na Figura 5.17. O *layout* da vista abstrata mostra os pinos. No caso das células da ASCEnD-FreePDK, os pinos são criados na camada de metal1 de tipo *drawing*. Por isso, apenas a camada de metal1 é mostrada nessa vista.

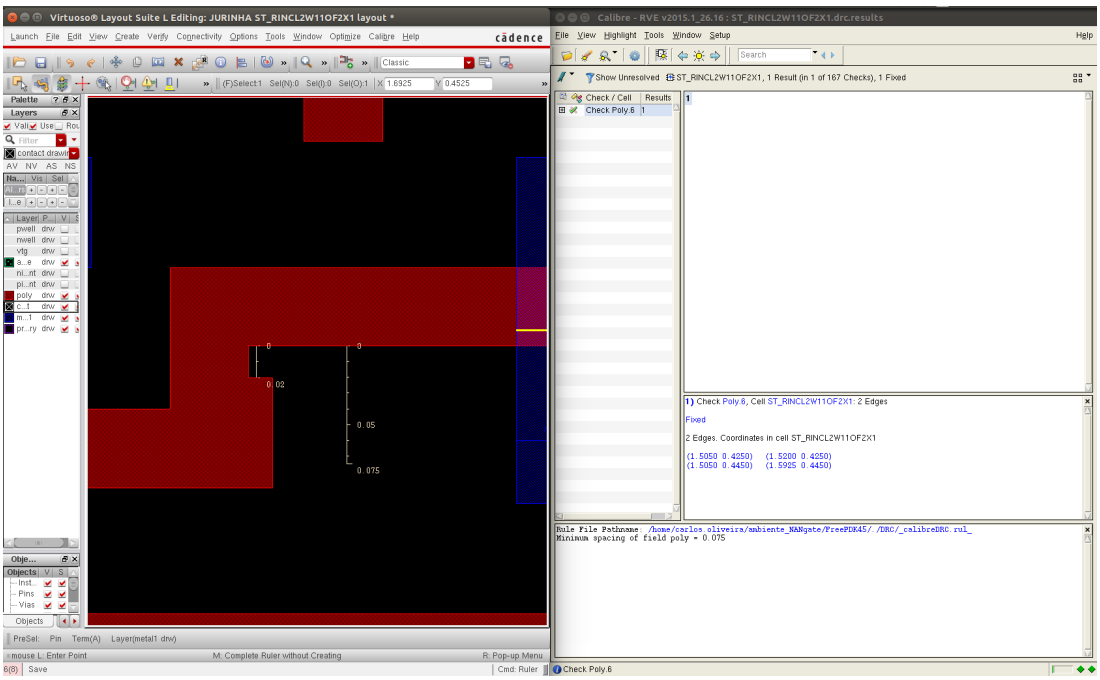


Figura 5.15 – Após a verificação de erros de DRC, apresenta-se um erro de regra de projeto na célula RINCL2W11OF2X1, relativo a um polígono de polissilício, como se vê no relatório a esquerda da figura.

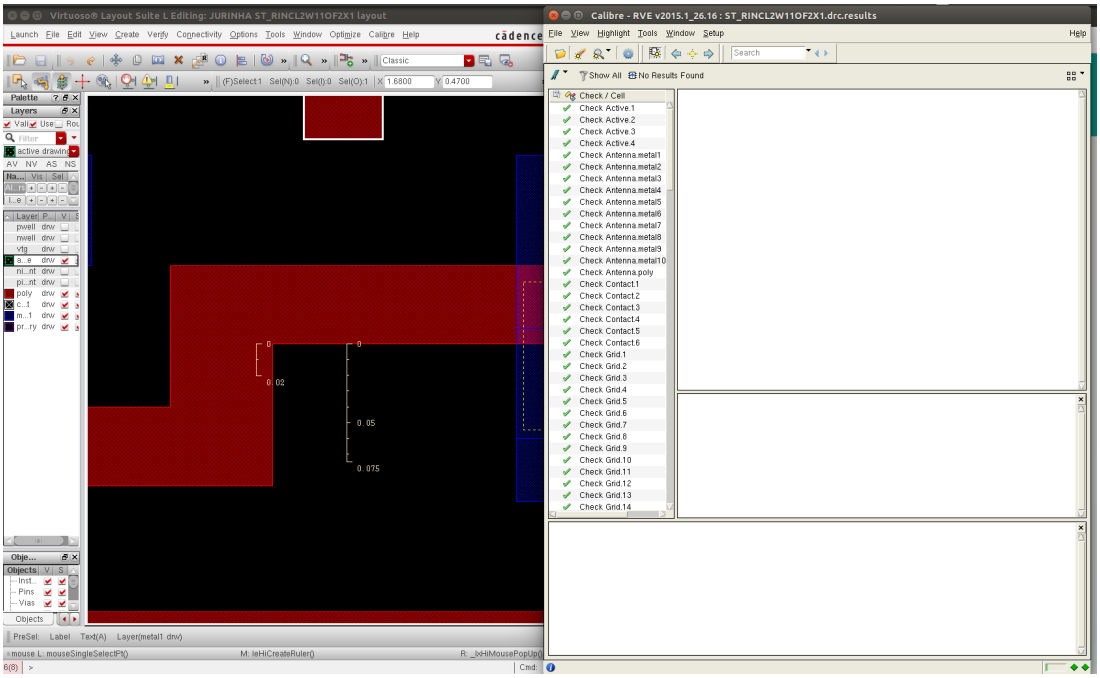


Figura 5.16 – O erro de DRC corrigido e o relatório indicando que todas as regras foram verificadas sem erro.

5.4.3 Caracterização da biblioteca

Para caracterizar a biblioteca foi utilizado um *script* do fluxo ASCEnD, que recebe o nome da célula, a biblioteca e qual tipo de extração a realizar. Para extração da ASCEnD-

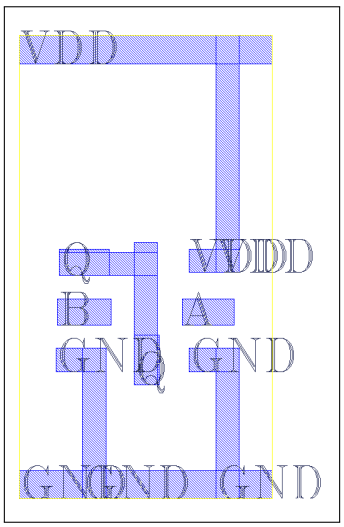


Figura 5.17 – *V*ista abstrata da célula INCL1W11OF2X1.

FreePDK, utilizou-se a extração capacitiva e resistiva. Para realizar de fato a caracterização, foi utilizado o LiChEn e, para tanto criou-se um *script* que configura todos os parâmetros do LiChEn, tais como as capacitâncias de carga e os tempos de simulação, e inclui o arquivo Spice utilizado na caracterização. Note-se que este arquivo pode ser a descrição gerada pela extração das capacitâncias e resistências ou a descrição sem os efeitos parasitas. Após a configuração e execução do LiChEn, a ferramenta criou um arquivo Liberty que contém informações sobre as características elétricas da célula. A partir destes dados foi gerado o modelo de página do *datasheet* da biblioteca, ilustrado nas Figuras 5.19 e 5.20.

5.5 Composição da Biblioteca ASCEnD-Free-PDK45 (v.1.0)

A biblioteca ASCEnD-Free-PDK45 apresenta um número reduzido de células, devido a dificuldades encontradas na adaptação do fluxo para outra versão do Virtuoso, bem como devido ao tempo disponível para realizar este trabalho. Outro fator que determinou a extensão do trabalho relatado aqui foi a necessidade de adaptação do ASTRAN para o modelo de *standard cell* proposto pela ASCEnD-FreePDK45, bem como a necessidade de configuração do ambiente de criação de *layouts* para a FreePDK45. Contudo, o processo de criação dessa biblioteca permitiu otimizar e automatizar várias etapas do fluxo ASCEnD-A, o que permitirá a criação de outras células que deem suporte a outros *templates* assíncronos de maneira muito mais eficiente no FreePDK45.

A Tabela 5.6 mostra todas as células que estão prontas para uso no momento da escrita deste documento. A determinação do nome de cada célula da biblioteca é baseada no método descrito pelo fluxograma da Figura 5.18. Este analisa 8 características de uma célula:

- Se o circuito é estático ou simétrico;
- Se tem função de *set* ou *reset*;
- Se é NCL ou NCL+(NCLP).
- O *threshold* da célula;
- Os pesos de cada entrada;
- A quantidade de entradas;
- O tamanho do *drive* para a célula;

Ao analisar a célula ST_SINCL2W11OF2X2, verifica-se que se trata de um circuito estático, com função de *set*, correspondendo a uma porta NCL com saída invertida, que possui *threshold* 2, o com peso 1 para cada uma de suas entradas. O *drive* da célula é 2.

NCL	INCL	NCLP	INCLP
ST_NCL1W11OF2X1	ST_INCL1W11OF2X1	ST_NCLP1W11OF2X1	ST_INCLP1W11OF2X1
ST_NCL1W11OF2X2	ST_INCL1W11OF2X2	ST_NCLP1W11OF2X2	ST_INCLP1W11OF2X2
ST_NCL1W11OF2X4	ST_INCL1W11OF2X4	ST_NCLP1W11OF2X4	ST_INCLP1W11OF2X4
ST_NCL2W11OF2X1	ST_INCL2W11OF2X1		
ST_NCL2W11OF2X2	ST_INCL2W11OF2X2		
ST_NCL2W11OF2X4	ST_INCL2W11OF2X4		
ST_RNCL2W11OF2X1	ST_RINCL2W11OF2X2		
ST_RNCL2W11OF2X2	ST_SINCL2W11OF2X1		
ST_RNCL2W11OF2X4	ST_SINCL2W11OF2X2		
ST_SNCL2W11OF2X1	ST_SINCL2W11OF2X4		
ST_SNCL2W11OF2X2			

Tabela 5.6 – Lista das células da ASCEnD-FreePDK45.

Utilizando as células projetadas neste trabalho e da biblioteca NanGate-FreePDK45, um estudo de caso de circuito foi sintetizado utilizando o fluxo de síntese proposto em [18]. Este estudo de caso foi um somador Kogge-Stone de 32 bits. Ele foi sintetizado utilizando diferentes restrições de projetos. Tais restrições foram baseadas na latência alvo do somador. Essas diferentes versões do somador permitem a demonstração da compatibilidade da biblioteca gerada com o fluxo de síntese e sua capacidade de permitir a exploração de diferentes métricas no projeto de circuitos QDI e com a biblioteca da NanGate-FreePDK45. A Tabela 5.7 apresenta os valores obtidos para: o número de portas lógicas após a síntese; número de portas lógicas no caminho crítico; e valores de potência estática, dinâmica e total do circuito. Conforme a Tabela demonstra, quanto menor a latência (consequentemente, maior a velocidade do circuito), maior a potência e o número de portas lógicas no circuito. Dessa forma, fica clara a possibilidade de exploração de diferentes necessidades de projeto utilizando a biblioteca ASCEnD-FreePDK45.

Latência (ns)	Portas	Cam. Crít.	Pot. Est. (uW)	Pot. Din. (uW)	Pot. Tot. (uW)
1.5	2822	22	23.814	500.436	524.251
2	2633	32	22.324	478.061	500.386
2.5	2604	34	22.126	482.808	504.934

Tabela 5.7 – Alguns resultados da síntese de diferentes versões de um somador Kogge-Stone de 32 bits utilizando a biblioteca ASCEnD-FreePDK45.

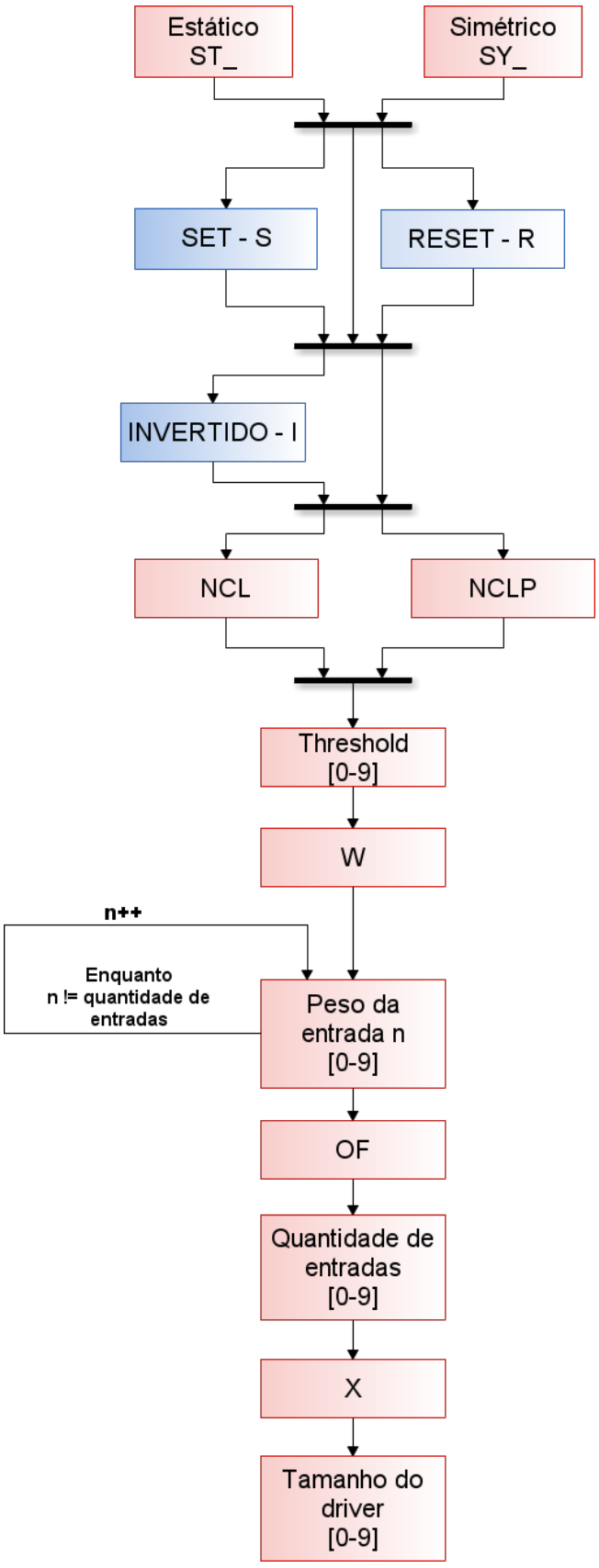
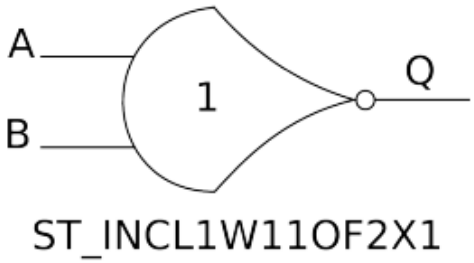


Figura 5.18 – O fluxograma que explicita o método de denominação das células da biblioteca ASCEnD-FreePDK45. As escolhas em azul são opcionais e aquelas em vermelho são obrigatórias.

ASCEND-FREEPDK45

Name	ST_INCL1W11OF2X1
Strength	1
Cell Area	1.6137 nm²
Equation	Q=INCL1(A,B)
Virtual Equation	Q=~(A+B)
Input[weight]	A[1] B[1]
Output	Q
Power and Ground Pins	VDD VDDS GND GNDS
Type	INCL



ST_INCL1W11OF2X1

Leakage[pW]
856.285

Capacitance[fF]	
A	0.000276413
B	0.000281022

Propagation Delay[ns]						
ST_INCL1W11OF2X1		Load Capacitance[pf]				
		Slope [ns]	0.0005	0.001	0.03	0.08
A to Q	FALL	0.005	0.01667	0.0244	0.468813	1.23486
		0.01	0.01742	0.0252	0.469794	1.2355
		0.02	0.01883	0.0266	0.471298	1.23695
		0.05	0.02281	0.0307	0.475438	1.24132
		0.08	0.02593	0.0347	0.479537	1.24546
	RISE	0.005	0.00783	0.0112	0.197886	0.519401
		0.01	0.00868	0.012	0.198714	0.520258
		0.02	0.01051	0.0138	0.200352	0.521795
		0.05	0.01433	0.0188	0.205324	0.526976
		0.08	0.01651	0.022	0.211034	0.53267
B to Q	FALL	0.005	0.01417	0.022	0.466449	1.23214
		0.01	0.01475	0.0226	0.467095	1.23292
		0.02	0.01631	0.024	0.468595	1.23443
		0.05	0.02162	0.0293	0.473226	1.23876
		0.08	0.02563	0.0347	0.477869	1.24349
	RISE	0.005	0.00672	0.01	0.196578	0.518189
		0.01	0.00759	0.0109	0.197451	0.519063
		0.02	0.00937	0.0126	0.199036	0.520495
		0.05	0.01233	0.0172	0.204053	0.525546
		0.08	0.01392	0.0198	0.209817	0.531402

Figura 5.19 – Exemplo de *datasheet* da biblioteca ASCEnD-FreePDK45, para a célula INCL1W11OF2X1 (Parte 1).

Output Transition [ns]						
ST_INCL1W11OF2X1		Slope [ns]	0.0005	0.001	0.03	0.08
A to Q	FALL	0.005	0.02429	0.0398	0.940277	2.49238
		0.01	0.0243	0.0398	0.940219	2.49251
		0.02	0.02453	0.0398	0.940135	2.49223
		0.05	0.02751	0.042	0.94008	2.49219
		0.08	0.03178	0.0454	0.939958	2.49213
	RISE	0.005	0.01123	0.0175	0.381037	1.00724
		0.01	0.01155	0.0176	0.380781	1.00726
		0.02	0.01309	0.0187	0.380817	1.00705
		0.05	0.01934	0.0244	0.38062	1.00725
		0.08	0.02475	0.0307	0.380482	1.00729
B to Q	FALL	0.005	0.02427	0.0398	0.940359	2.49245
		0.01	0.02425	0.0398	0.94009	2.49219
		0.02	0.02518	0.04	0.940047	2.49214
		0.05	0.02996	0.0438	0.940097	2.49191
		0.08	0.03536	0.0487	0.939959	2.49221
	RISE	0.005	0.0093	0.0156	0.379533	1.00713
		0.01	0.00968	0.0157	0.37955	1.00708
		0.02	0.01134	0.0169	0.379704	1.00737
		0.05	0.01756	0.0229	0.379761	1.00693
		0.08	0.02279	0.029	0.379376	1.00723

Dynamic Power Consuption [pW/GHz]						
ST_INCL1W11OF2X1		Slope [ns]	0.0005	0.001	0.03	0.08
A to Q	FALL	0.005	0.00083	0.0011	0.0155681	0.040518
		0.01	0.00082	0.0011	0.0155727	0.0405427
		0.02	0.00081	0.0011	0.0155701	0.0405397
		0.05	0.00081	0.0011	0.0155712	0.04054
		0.08	0.00083	0.0011	0.0155694	0.0405416
	RISE	0.005	0.00032	0.0006	0.0150721	0.0400722
		0.01	0.00033	0.0006	0.0150741	0.0400741
		0.02	0.00033	0.0006	0.0150753	0.0400751
		0.05	0.00031	0.0006	0.0150745	0.0400746
		0.08	0.00028	0.0005	0.0150727	0.0400745
B to Q	FALL	0.005	0.00063	0.0009	0.0153801	0.0403463
		0.01	0.00062	0.0009	0.0153767	0.0403471
		0.02	0.00062	0.0009	0.0153746	0.040347
		0.05	0.00064	0.0009	0.0153752	0.0403391
		0.08	0.00068	0.0009	0.0153745	0.0403447
	RISE	0.005	0.00127	0.001	0.013474	0.0384742
		0.01	0.00126	0.001	0.0134724	0.0384725
		0.02	0.00126	0.001	0.0134718	0.0384718
		0.05	0.00128	0.001	0.0134706	0.0384708
		0.08	0.00131	0.001	0.0134689	0.0384706

Figura 5.20 – Exemplo de *datasheet* da biblioteca ASCEnD-FreePDK45, para a célula INCL1W11OF2X1 (Parte 2).

6. CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho de conclusão de curso foi realizado durante o segundo semestre de 2015 e teve como objetivo a criação de uma biblioteca de células abertas de circuitos assíncronos, a ASCEnD-FreePDK45. A biblioteca foi criada no FreePDK45, e é compatível com a biblioteca NanGate-FreePDK45. Foram criadas 28 células assíncronas, que são NCL ou NCL+ e dão suporte ao *template* assíncrono SDDS-NCL. A biblioteca foi criada utilizando o fluxo ASCEnD-A. A fluxo ASCEnD-A foi adaptado para o FreePDK45, para a versão 6.1 do Cadence Virtuoso e também para arquitetura da NanGate-FreePDK45. Esta adaptação possibilita uma maior flexibilidade ao fluxo ASCEnD-A, visto que o fluxo automatizado é composto por diversos *scripts* que foram desenvolvidos buscando a máxima generalidade. A migração para novas tecnologias ou a criação de novas células podem ser obtidas agora mais facilmente. A validação do fluxo adaptado foi feita através da criação desta biblioteca, e foi o momento critico deste trabalho, onde todas as ferramentas alteradas precisaram ser testadas em conjunto e durante este processo precisaram ferramentas serem refeitas. E tiveram varias etapas do fluxo que foram reexecutadas devido a propagação dos erros de etapas anteriores, como ao verificar os erros no calculo do dimensionamento vindo do CeS. Fora isso o processo de criação dos *layouts* utilizando o ASTRAN apresentou diversos *bugs*, como a ausência de pinos e a alteração da altura das células, que foram solucionadas pelo autor do *software*, e precisou ser refeita três vezes para que chegasse a uma versão mais estável e compatível com o *layout* esperado. Como as ferramentas de caracterização disponíveis não são compatíveis com as células assíncronas foi necessária a utilização do LiChen que apresenta uma limitação que é a conversão das funções das células assíncronas para funções lógicas.

A criação desta biblioteca foi possível graças aos conhecimentos desenvolvidos em diversas disciplinas do curso de Engenharia de Computação da PUCRS, sobretudo as disciplinas de Microeletrônica e Projeto de Sistemas Integrados I e II. A disciplina de Microeletrônica formulou toda a base teórica necessária para o trabalho com as diferentes formas como se apresentavam as células durante o fluxo de criação destas. A mesma disciplina abordou o processo de análise de qual tipo de transistor usar, a influência dos dimensionamentos sobre a carga de cada célula e também os conceitos básicos por trás de transistores MOS. Além disso, no processo de criação do *layout* das células, foi onde o que se aprendeu nos laboratórios da disciplina foi aplicado: as noções de bibliotecas de células, as regras de projeto e as verificações de DRC e LVS. As disciplinas de Projeto de Sistemas Integrados (I e II), permitiram analisar e dominar o processo de projeto de CIs. Desta forma, pude dominar o processo de desenvolvimento de uma biblioteca de células. Também pude avaliar a biblioteca em uso, através do estudo de caso de circuito realizado com a biblioteca, o somador Kogge-Stone de 32 bits. Porém nem todo conhecimento necessário estava incluído dentro do que foi abordado na faculdade. Foram desenvolvidos programas em

shell script, *awk* e *Cadence Skill*, que são linguagens que não foram apresentadas durante a graduação. Fora isto, há também o grande corpo de conhecimentos adquiridos durante o trabalho de conclusão do curso, relacionado sobretudo ao projeto de circuitos assíncronos.

A biblioteca ASCEnD-FreePDK45 está disponibilizada publicamente através do repositório
(https://corfu.pucrs.br/svn/ascend/ncsu-freepdk45/), e contém todos modelos necessários para síntese de circuitos assíncronos, tais como arquivos Liberty (.lib), vistas abstratas (.lef), descrições Spice pré- e pós-síntese, *layouts* em GDSII, esquemáticos em PNG além dos *datasheets* para cada célula da biblioteca ASCEnD-FreePDK45.

Além disso o grupo GAPH está em contato com a empresa Wave
(http://www.wavesemi.com/), que comercializa produtos baseados em lógica NCL, e eles estão interessados em nossa biblioteca para o FreePDK45. Como a ASCEnD-A e o FreePDK45 são material de livre acesso livre, a distribuição dessa biblioteca para fins educacionais e de pesquisa são o foco principal deste trabalho.

O próximo passo para esta biblioteca é a continuação do processo de criação de novas células para dar melhor suporte ao *template* SDDS-NCL, bem como para dar suporte a outros *templates* assíncronos. Já em relação ao trabalho realizado como evolução do fluxo ASCEnD-A, divisa-se um conjunto de evoluções, tal como expandir o leque biblioteca assíncronas para tecnologias fabricáveis como a TSMC 180nm na qual o GAPH já prototipou CIs assíncronos com sucesso.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Moreira, matheus t and oliveira, carlos hm and guazelli, ricardo. figuras referentes ao fluxo ascend-a. https://cacoo.com/diagrams/uZm73JFqlUMkkLpb. Nota: Acesso restrito aos autores. Acessado: 14/11/2015.

[2] ITRS. International Technology Roadmap for Semiconductors. Technical report, http://www.itrs.net/home.html, 2011.

[3] Jan M. Rabaey and Anantha P. Chandrakasan and Borivoje Nikolic. *Digital integrated circuits*, volume 2. 2002.

[4] Peter A. Beerel, Recep O. Ozdag, and Marcos Ferretti. *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

[5] Vivek Tiwari, Deo Singh, Suresh Rajgopal, Gaurav Mehta, Rakesh Patel, and Franklin Baez. Reducing power in high-performance microprocessors. In *Proceedings of the 35th annual Design Automation Conference*, pages 732–737, 1998.

[6] Shmuel Wimer. Easy and difficult exact covering problems arising in vlsi power reduction by clock gating. *Discrete Optimization*, 14:104–110, 2014.

[7] Jens Sparso and Steve Furber. *Principles of Asynchronous Circuit Design - A Systems Perspective*. Springer, 2002.

[8] Matheus T. Moreira. Design and Implementation of a Standard Cell Library for Building Asynchronous ASICs. Trabalho de Conclusão de Curso, Engenharia de Computação. Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul, 2010.

[9] Matheus Gibiluka. Design and implementation of an asynchronous noc router using a transition-signaling bundled-data protocol. Trabalho de Conclusão de Curso, Engenharia de Computação. Faculdade de Informática - Pontifícia Universidade Católica do Rio Grande do Sul, 2013.

[10] Karl M Fant, Scott Brandt, et al. NULL Convention Logic^{TM}: a complete and consistent logic for asynchronous digital circuit synthesis. In *International Conference on Application Specific Systems, Architectures and Processors (ASAP96)*, pages 261–273, 1996.

[11] Matheus T. Moreira, Carlos H. M. Oliveira, Ricardo C. Porto, and Ney L. V. Calazans. NCL+: Return-to-one null convention logic. In *IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 836–839. IEEE, 2013.

[12] David E. Muller and W. Scott Bartky. A Theory of Asynchronous Circuits I. In *International Symposium on the Theory of Switching - Part I*, 1957.

[13] Edward B. Eichelberger and Thomas W. Williams. A logic design structure for lsi testability. In *14th Design Automation Conference (DAC)*, pages 462–468, 1977.

[14] Peter A. Beerel, Georgios D. Dimou, and Andrew M. Lines. Proteus: An ASIC flow for GHz asynchronous designs. *IEEE Design & Test of Computers*, (5):36–51, 2011.

[15] Dylan Hand, Matheus T. Moreira, Hsin-Ho Huang, Danlei Chen, Frederico Butzke, Zhichao Li, Matheus Gibiluka, Melvin Breuer, Ney L. V. Calazans, and Peter A. Beerel. Blade–A Timing Violation Resilient Asynchronous Template. In *21st IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 21–28, 2015.

[16] Matheus T. Moreira, Guilherme Trojan, Fernando G. Moraes, and Ney L. V. Calazans. Spatially Distributed Dual-Spacer Null Convention Logic Design. *Journal of Low Power Electronics*, 10(3):313–320, 2014.

[17] Philippe Maurine, J. B. Rigaud, F. Bouesse, G. Sicard, and M. Renaudin. Tal: une bibliothèque de cellules pour le design de circuits asynchrones qdi. *4iemes Journées Francophones d'Etudes Faible Tension, Faible Consommation*, pages 41–49, 2003.

[18] Matheus Moreira, Augusto Neutzling, Miguel Martins, André Reis, Renato Ribas, and Ney Calazans. Semi-custom NCL Design with Commercial EDA Frameworks: Is it possible? In *20th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 53–60, 2014.

[19] Matheus Gibiluka, Matheus Moreira, and Ney Calazans. A Bundled-Data Asynchronous Circuit Synthesis Flow Using a Commercial EDA Framework. In *18th Euromicro Conference on Digital System Design (DSD)*, 2015.

[20] Gaurav Gulati and Erik Brunvand. Design of a cell library for asynchronous microengines. In *15th ACM Great Lakes symposium on VLSI*, pages 385–389, 2005.

[21] James E. Stine, Ivan Castellanos, Michael Wood, Jeff Henson, Fred Love, W. Rhett Davis, Paul D. Franzon, Michael Bucher, Sunil Basavarajaiah, Julie Oh, et al. Freepdk: An open-source variation-aware design kit. In *IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pages 173–174, 2007.

[22] Matheus T. Moreira, Michel Arendt, Adriel Ziesemer, Ricardo A. L. Reis, and Ney L. V. Calazans. Automated Synthesis of Cell Libraries for Asynchronous Circuits. In *27th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–7, 2014.

[23] NanGate FreePDK45 Open Cell Library. http://www.nangate.com. Acessado: 08/08/2015.

[24] Matheus Moreira, Bruno Oliveira, Julian Pontes, Fernando Moraes, and Ney Calazans. Adapting a C-element Design Flow for Low Power. In *18th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 45–48. IEEE, 2011.

[25] Marc Renaudin, Pascal Vivet, and Frédéric Robin. ASPRO-216: a Standard-Cell QDI 16-bit RISC Asynchronous Microprocessor. In *1998 Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 22–31, 1998.

[26] Marc Renaudin, Bachar E. Hassan, and Alain Guyot. A New Asynchronous Pipeline Scheme: Application to the Design of a Self-timed Ring Divider. *IEEE Journal of Solid-State Circuits*, 31(7):1001–1013, 1996.

[27] Adriel Ziesemer, Ricardo A. L. Reis, Matheus T. Moreira, Michel E. Arendt, and Ney L. V. Calazans. Automatic Layout Synthesis with ASTRAN applied to Asynchronous Cells. In *IEEE 5th Latin American Symposium on Circuits and Systems (LASCAS)*, pages 1–4, 2014.

[28] Sung-Mo Kang and Yusuf Leblebici. *CMOS Digital Integrated Circuits*. Tata McGraw-Hill Education, 2003.

[29] Neil H. E. Weste and David M. Harris. *CMOS VLSI Design: A Circuits And Systems Perspective*. Pearson Education, India, 2006.

[30] Marko Dimitrijević, Borisav Jovanović, Bojan Anđelković, Milan Savić, and Miljana Sokolović. Experiences in Using CADENCE - the Industry Standard for Integrated Circuits Design. In *XLVII ETRAN Conference*, 2003.

[31] Open Source Liberty. http://www.opensourceliberty.org. Acessado: 02/09/2015.

[32] Matheus Gibiluka, Matehus Moreira, and Ney Calazans. BAT-Hermes: A Transition-Signaling Bundled-Data NoC Router. In *VI Latin American Symposium on Circuits & Systems (LASCAS)*, pages 1–4, 2015.

[33] Julian Pontes, Matheus Moreira, Fernando Moraes, and Ney Calazans. Hermes-A – An Asynchronous NoC Router with Distributed Routing. In *Power and Timing Modeling, Optimization, and Simulation (PATMOS)*, pages 150–159. 2011.

[34] Matheus T. Moreira, Felipe Magalhaes, Matheus Gibiluka, Fabiano P. Hessel, and Ney L. V. Calazans. BaBaNoC: An Asynchronous Network-on-Chip described in Balsa. In *International Symposium on Rapid System Prototyping (RSP)*, pages 37–43, 2013.

[35] Jordi Cortadella, Alex Yakovlev, and Jim Garside. Logic design of asynchronous circuits. In *7th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 26–27, 2002.

[36] Danil Sokolov, Julian Murphy, Alex Bystrov, and Alex Yakovlev. Improving the Security of Dual-rail Circuits. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 282–297. 2004.

[37] Wave Semiconductor. http://www.wavesemi.com/. Acessado: 05/11/2015.

[38] Michiel Ligthart, Karl Fant, Ross Smith, Alexander Taubin, and Alex Kondratyev. Asynchronous Design using Commercial HDL Synthesis Tools. In *Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, pages 114–125, 2000.

[39] Matheus T. Moreira, Ricardo A. Guazzelli, Ney L. V. Calazans, et al. Return-to-one Protocol for reducing Static Power in C-elements of QDI Circuits employing m-of-n Codes. In *25th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6, 2012.

[40] Matheus T. Moreira, Julian J. H. Pontes, and Ney L. V. Calazans. Tradeoffs between RTO and RTZ in WCHB QDI Asynchronous Design. In *15th International Symposium on Quality Electronic Design (ISQED)*, pages 692–699, 2014.

[41] Matheus Moreira, Michel Arendt, Adriel Ziesemer, Ricardo Reis, and Ney Calazans. Automated synthesis of cell libraries for semi-custom asynchronous design.

[42] Gisell B. Moura, Adriel Ziesemer Jr, and Ricardo A. L. Reis. A Study on the Automatic Synthesis of Layout with ASTRAN using FreePDK 45nm. In *VI Latin American Symposium on Circuits & Systems (LASCAS)*, pages 1–4, 2015.

[43] Matheus Trevisan Moreira, Carlos Henrique Menezes Oliveira, Ney Laert Vilar Calazans, and Luciano Copello Ost. Lichen: Automated electrical characterization of asynchronous standard cell libraries. In *Digital System Design (DSD), 2013 Euromicro Conference on*, pages 933–940. IEEE, 2013.

[44] Ricardo A Guazzelli, Fernando G Moraes, Ney LV Calazans, and Matheus T Moreira. Sdds-ncl design: Analysis of supply voltage scaling. In *Proceedings of the 28th Symposium on Integrated Circuits and Systems Design*, page 2. ACM, 2015.

[45] EE Times, 2015.

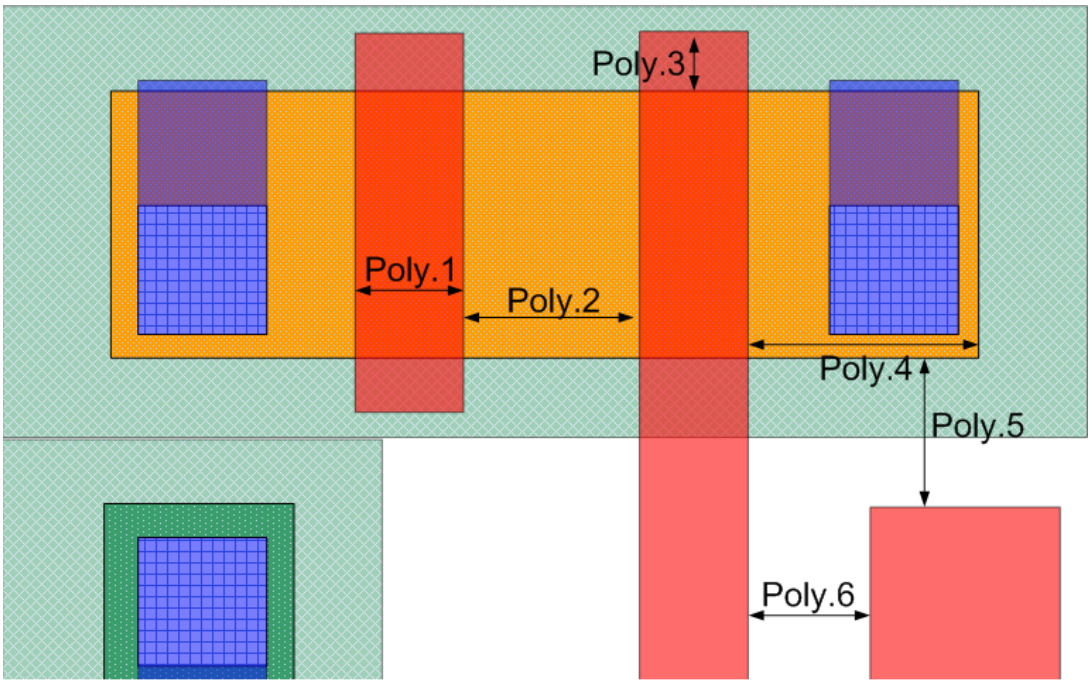
[46] FreePDK45. http://www.eda.ncsu.edu/wiki/FreePDK45:Contents. Seção: Design Rules. Acessado: 14/11/2015.

ANEXO A – Regras de *design* da FreePDK45.

Este anexo contém todas as regras de *design* da FreePDK45 e também a arquitetura das células da NanGate-FreePDK. Esse documento serve de base para a criação da biblioteca proposta neste trabalho.

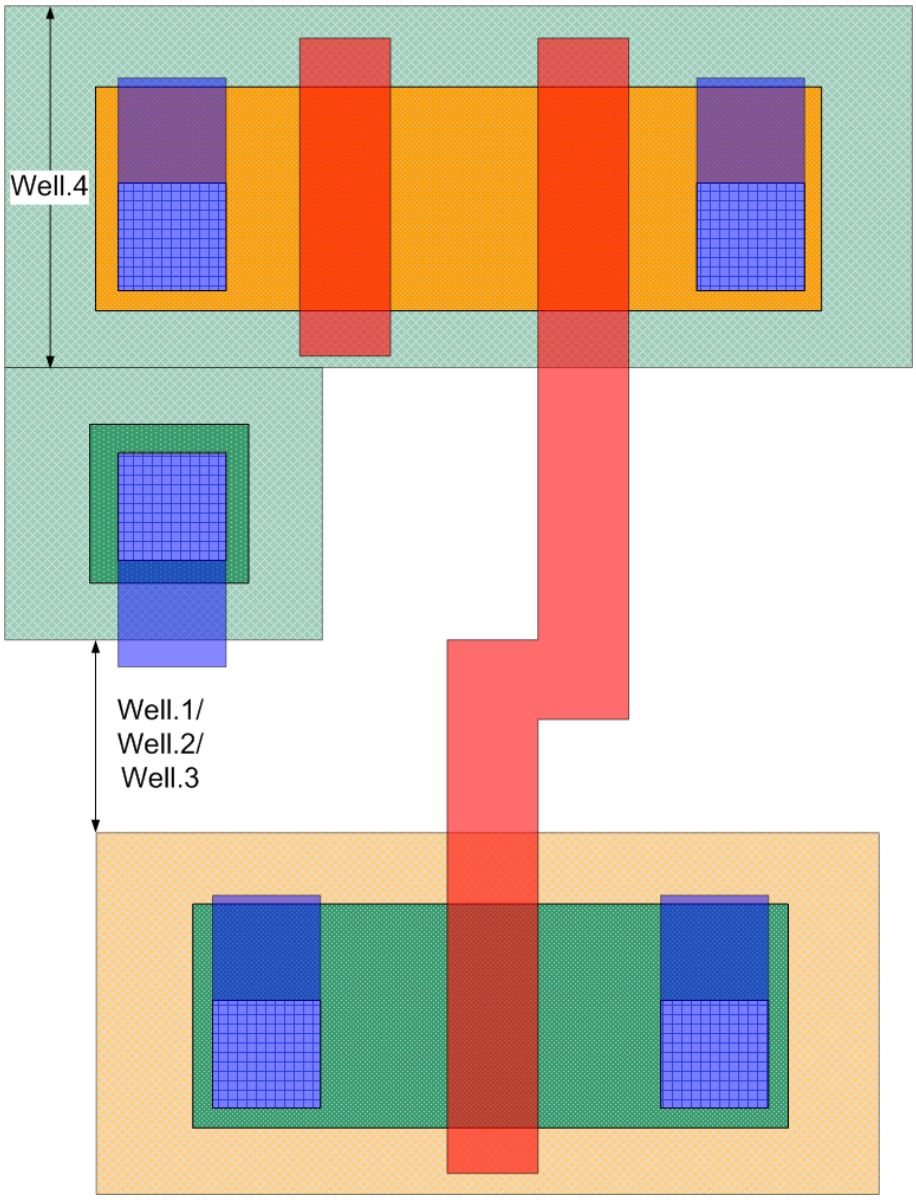
Design Rules
for
FreePDK45

Poly Rules



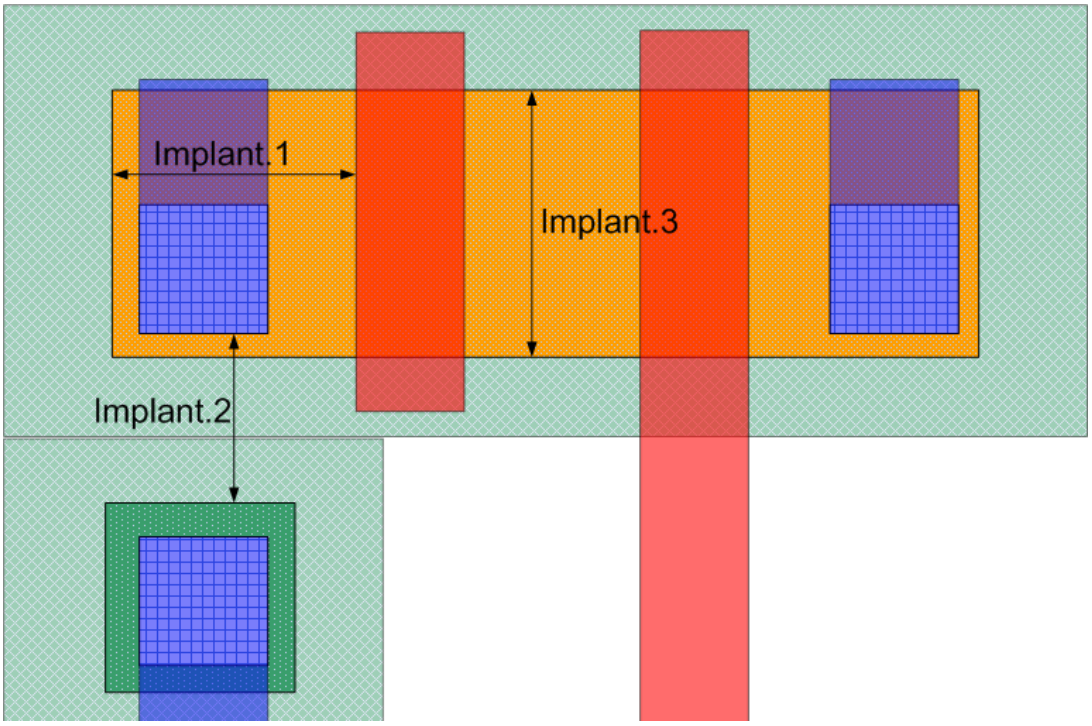
Rule	Value	Description
POLY.1	50 nm	Minimum width of poly
POLY.2	140 nm	Minimum spacing of poly AND active
POLY.3	55 nm	Minimum poly extension beyond active
POLY.4	70 nm	Minimum enclosure of active around gate
POLY.5	50 nm	Minimum spacing of field poly to active
POLY.6	75 nm	Minimum Minimum spacing of field poly

Well Rules



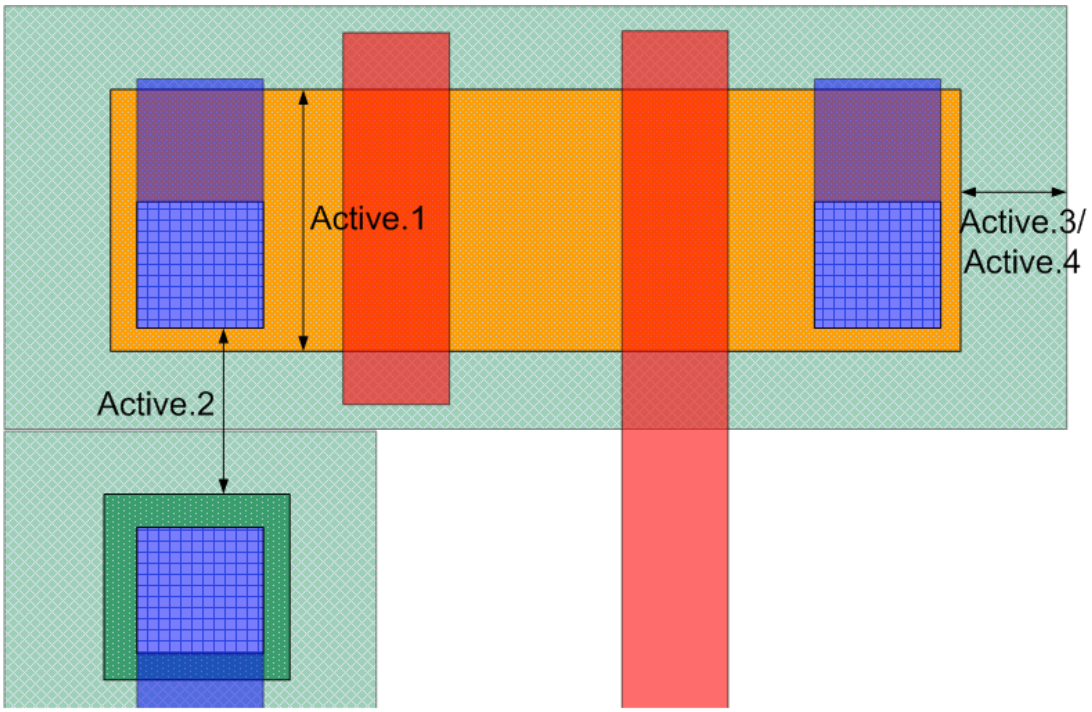
Rule	Value	Description
WELL.1	none	saveDerived: nwell/pwell must not overlap
WELL.2	225 nm	Minimum spacing of nwell/pwell at different potential
WELL.3	135 nm	Minimum spacing of nwell/pwell at the same potential
WELL.4	200 nm	Minimum width of nwell/pwell

Implant Rules



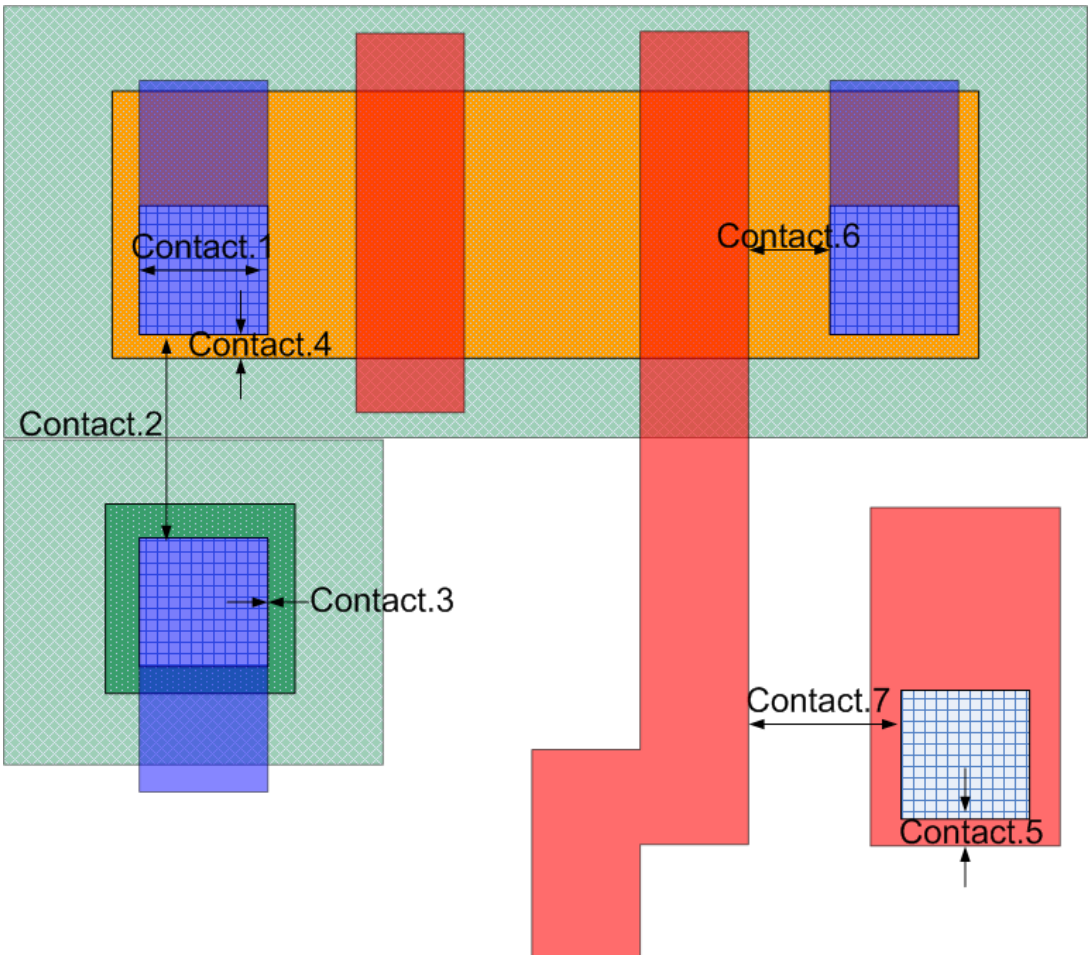
Rule	Value	Description
IMPLANT.1	70 nm	Minimum spacing of nimplant/ pimplant to channel
IMPLANT.2	25 nm	Minimum spacing of nimplant/ pimplant to contact
IMPLANT.3/4	45 nm	Minimum width/ spacing of nimplant/ pimplant
IMPLANT.5	none	Nimplant and pimplant must not overlap

Active Rules



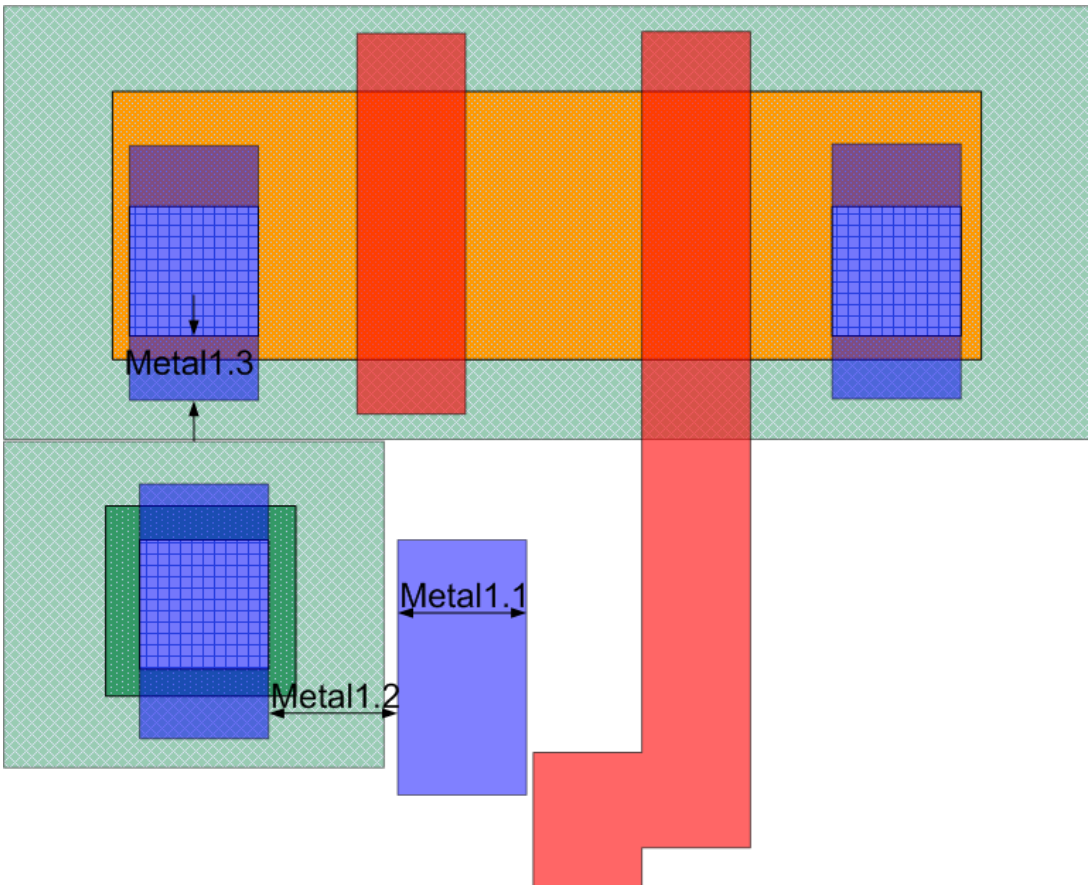
Rule	Value	Description
ACTIVE.1	90 nm	Minimum width of active
ACTIVE.2	80 nm	Minimum spacing of active
ACTIVE.3	55 nm	Minimum enclosure/spacing of nwell/pwell to active
ACTIVE.4	none	saveDerived: active must be inside nwell or pwell

Contact Rules



Rule	Value	Description
CONTACT.1	65 nm	Minimum width of contact
CONTACT.2	75 nm	Minimum spacing of contact
CONTACT.3	none	saveDerived: contact must be inside active or poly or metal1
CONTACT.4	5 nm	Minimum enclosure of active around contact
CONTACT.5	5 nm	Minimum enclosure of poly around contact
CONTACT.6	35 nm	Minimum spacing of contact and gate
CONTACT.7	90 nm	Minimum spacing of contact and poly

Metal1 Rules

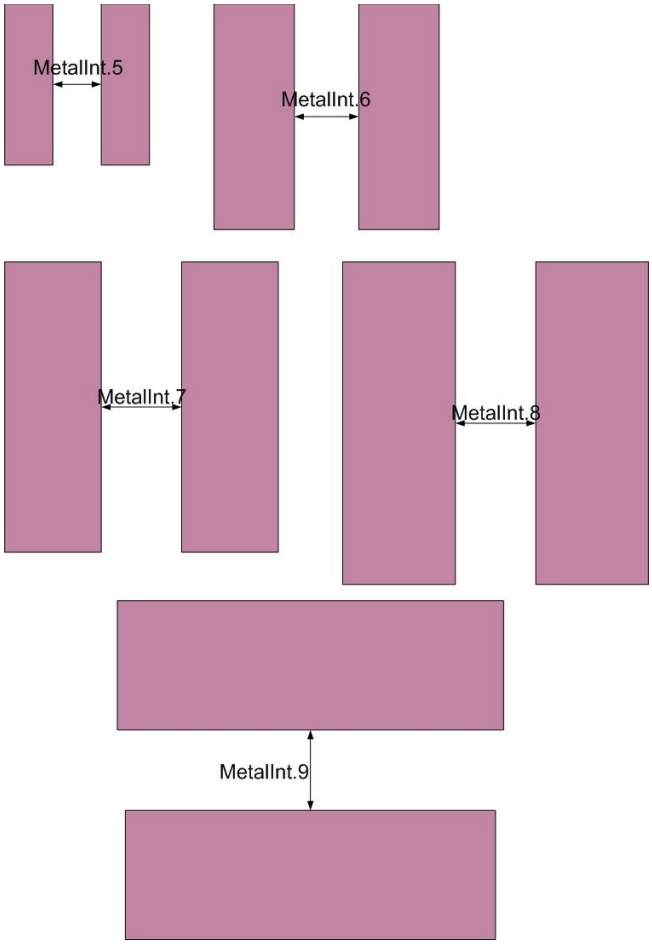


Rule	Value	Description
METAL1.1	65 nm	Minimum width of metal1
METAL1.2	65 nm	Minimum spacing of metal1
METAL1.3	35 nm	Minimum enclosure around contact on two opposite sides
METAL1.4	35 nm	Minimum enclosure around via1 on two opposite sides
METAL1.5	90 nm	Minimum spacing of metal wider than 90 nm and longer than 900 nm
METAL1.6	270 nm	Minimum spacing of metal wider than 270 nm and longer than 300 nm
METAL1.7	500 nm	Minimum spacing of metal wider than 500 nm and longer than 1.8um
METAL1.8	900 nm	Minimum spacing of metal wider than 900 nm and longer than 2.7 um
METAL1.9	1500 nm	Minimum spacing of metal wider than 1500 nm and longer than 4.0 um

VIA1 Rules

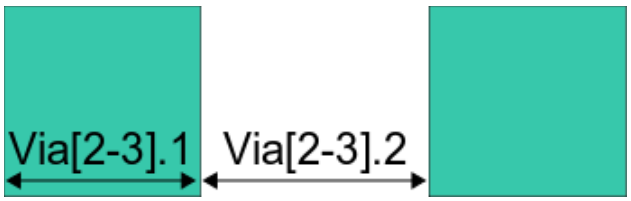
Rule	Value	Description
VIA1.1	65 nm	Minimum width of Via1
VIA1.2	75 nm	Minimum spacing of Via1
VIA1.3	none	saveDerived: Via1 must be inside metal1
VIA1.4	none	saveDerived: Via1 must be inside metal2

Metal Int Rules



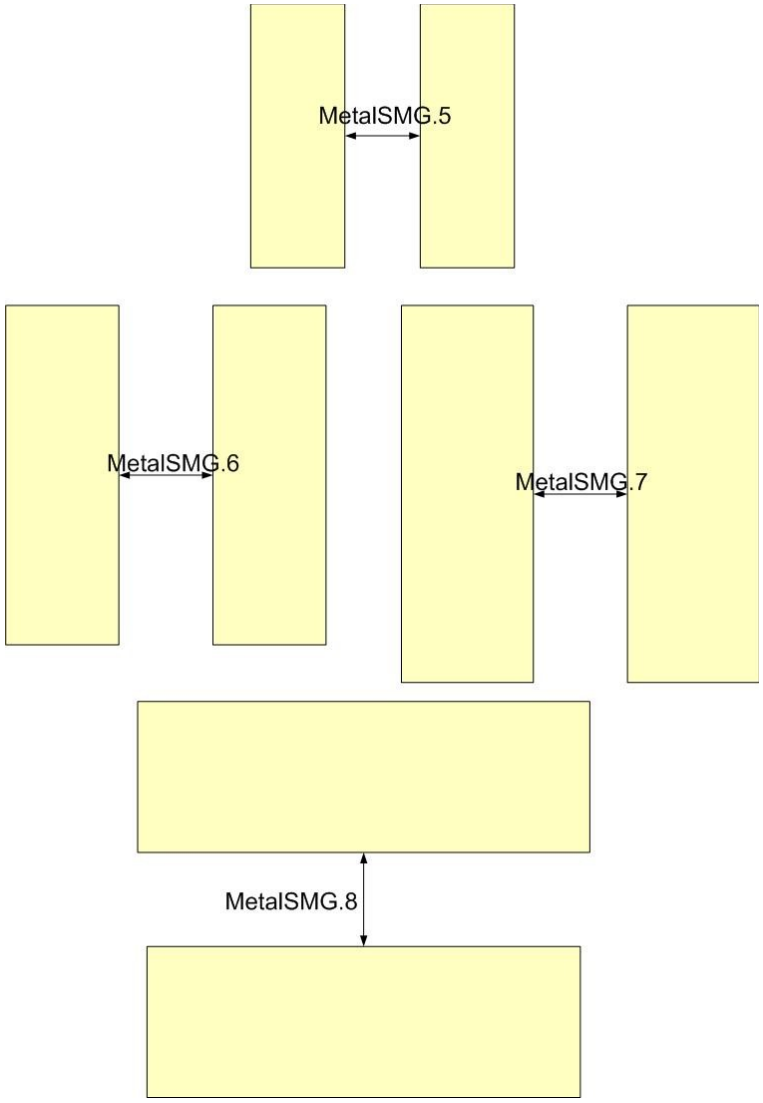
Rule	Value	Description
METALINT.1	70 nm	Minimum width of intermediate metal
METALINT.2	70 nm	Minimum spacing of intermediate metal
METALINT.3	35 nm	Minimum enclosure around via1 on two opposite sides
METALINT.4	35 nm	Minimum enclosure around via[2-3] on two opposite sides
METALINT.5	90 nm	Minimum spacing of metal wider than 90 nm and longer than 900 nm
METALINT.6	270 nm	Minimum spacing of metal wider than 270 nm and longer than 300 nm
METALINT.7	500 nm	Minimum spacing of metal wider than 500 nm and longer than 1.8um
METALINT.8	900 nm	Minimum spacing of metal wider than 900 nm and longer than 2.7 um
METALINT.9	1500 nm	Minimum spacing of metal wider than 1500 nm and longer than 4.0 um

VIA[2-3] Rules



Rule	Value	Description
VIA[2-3].1	70 nm	Minimum width of Via[2-3]
VIA[2-3].2	85 nm	Minimum spacing of Via[2-3]
VIA[2-3].3	none	saveDerived: Via[2-3] must be inside metal[2-3]
VIA[2-3].4	none	saveDerived: Via[2-3] must be inside metal[3-4]

Metal SMG Rules



Rule	Value	Description
METALSMG.1	140 nm	Minimum width of semi-global metal
METALSMG.2	140 nm	Minimum spacing of semi-global metal
METALSMG.3	0 nm	Minimum enclosure around via[3-6] on two opposite sides
METALSMG.6	270 nm	Minimum spacing of metal wider than 270 nm and longer than 300 nm
METALSMG.7	500 nm	Minimum spacing of metal wider than 500 nm and longer than 1.8um
METALSMG.8	900 nm	Minimum spacing of metal wider than 900 nm and longer than 2.7 um

Via[4-6] Rules

Rule	Value	Description
VIA[4-6].1	140 nm	Minimum width of Via[4-6]
VIA[4-6].2	160nm	Minimum spacing of Via[4-6]
VIA[4-6].3	none	saveDerived: Via[4-6] must be inside metal[4-6]
VIA[4-6].4	none	saveDerived: Via[4-6] must be inside metal[5-7]

METAL TNG Rules

Rule	Value	Description
METALTNG.1	400 nm	Minimum width of metalTNG
METALTNG.2	400 nm	Minimum spacing of metalTNG
METALTNG.3	0 nm	Minimum enclosure around via[6-8] on two opposite sides
METALTNG.4	(20000 nm)²	Minimum area of metalTNG straddling via[6-8]
METALTNG.5	90 nm	Minimum spacing of metal wider than 90 nm and longer than 900 nm
METALTNG.6	270 nm	Minimum spacing of metal wider than 270 nm and longer than 300 nm

VIA[7-8] Rules

Rule	Value	Description
Via[7-8].1	400 nm	Minimum width of Via[7-8]
Via[7-8].2	440 nm	Minimum spacing of Via[7-8]
Via[7-8].3	none	saveDerived: Via[7-8] must be inside metal[7-8]
Via[7-8].4	none	saveDerived: Via[7-8] must be inside metal[8-9]

Metal G Rules

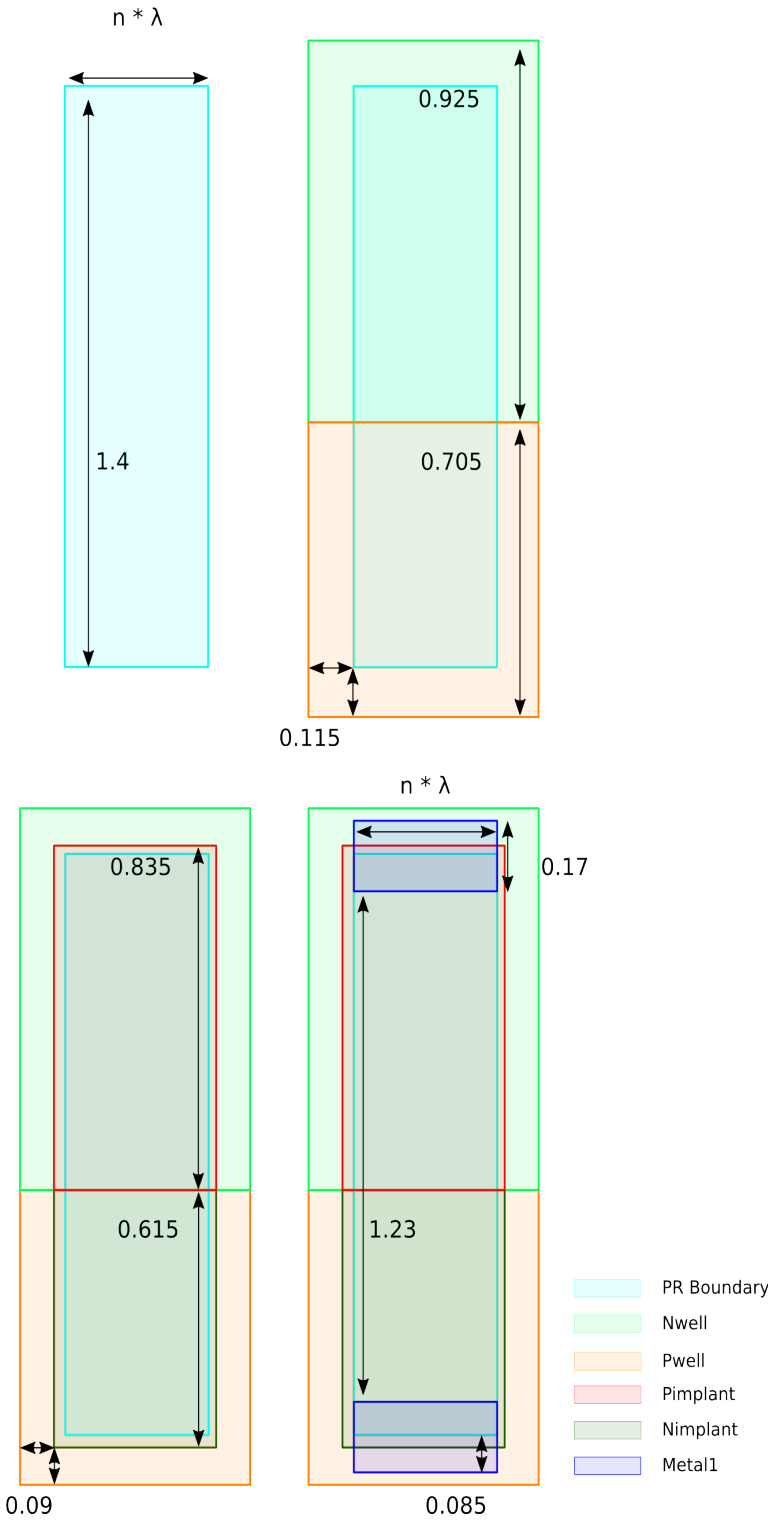
Rule	Value	Description
METALG.1	800 nm	Minimum width of global metal
METALG.2	800 nm	Minimum spacing of global metal
METALG.3	0 nm	Minimum enclosure around via[8-9] on two opposite sides
METALG.8	900 nm	Minimum spacing of metal wider than 900 nm and longer than 2.7 um
METALG.9	1500 nm	Minimum spacing of metal wider than 1500 nm and longer than 4.0 um

Via 9 Rules

Rule	Value	Description
Via[9].1	800 nm	Minimum width of Via9
Via[9].2	880 nm	Minimum spacing of Via9
Via[9].3	none	saveDerived: Via9 must be inside metal9
Via[9].4	none	saveDerived: Via9 must be inside metal10

NanGate Standard Cell Constrains

PITCH = λ = 0.19



Definitions

Different Net

The Different-Net modifier is the opposite of the Same Net modifier.

Enclousure

An enclosure rule checks the distance between the outside edges of the enclosed shapes to the inside edges of the enclosing shapes. This is commonly used for vias to specify the enclosure of metal around the via cut shape.
Enclosure does not imply that one shape must be completely contained within another. Rather, it merely checks the spacing of the edges.
Enclosure is often confused with Overlap, which refers to the distance between inside edges of overlaping shapes.

Overlap

Overlap of two shapes refers to the distance between inside edges of overlaping shapes.
Overlap is often confused with Enclosure, which refers to the case when one shape must be completely contained within another.

Same Net

The Same-Net modifier alters a design-rule check to apply only to shapes that are electrically connected (i.e. on the same net). The analagous Different Net modifier checks only shapes that are NOT electrically connected. One common use for this rule is to check spacing between well-shapes, which may have a resistance-path through the substrate. Same-net wells need not be spaced as far apart as different-net wells.

Spacing

Spacing refers to distance between outside edges of shapes on a layer. Typically, design-rule checkers will merge all shapes on a layer before performing this check. The most common form of this rule is a minimum spacing check, which ensures that the merged shapes on the same layer or different layers are spaced by a minimum amount. This rule is typically used to ensure that there will be no short-circuits in the design.
Spacing rules are often used with the Same Net or Different Net modifiers. Spacing rules are also often dependent on the width of the shapes.

Width

Width refers to distance between inside edges of shapes on a layer. Typically, design-rule checkers will merge all shapes on a layer before performing this check. The most common form of this rule is a minimum width check, which ensures that the merged shapes never show a width smaller than a certain amount. This rule is typically used to ensure that there will be no open circuits in the design.

Reference

http://www.eda.ncsu.edu/wiki/FreePDK45