Sara Pinheiro de Brito

# Relatório Final de Estágio
# Estágio Integrado na Idea! Sistemas Eletrônicos

**Campina Grande, Paraíba**

**26 de fevereiro de 2019**

Sara Pinheiro de Brito

# Relatório Final de Estágio
# Estágio Integrado na Idea! Sistemas Eletrônicos

Relatório de Estágio Integrado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Centro de Engenharia Elétrica e Informática – CEEI

Unidade Acadêmica de Engenharia Elétrica – UAEE

Orientador: Prof. Dr. Gutemberg Gonçalves Júnior

Supervisor: Jacklyn Dias Reis, PhD

Campina Grande, Paraíba

26 de fevereiro de 2019

Sara Pinheiro de Brito

# Relatório Final de Estágio
# Estágio Integrado na Idea! Sistemas Eletrônicos

Relatório de Estágio Integrado submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Trabalho aprovado. Campina Grande, Paraíba, 25 de fevereiro de 2019:

---

**Prof. Dr. Gutemberg Gonçalves Júnior**
Orientador

---

**Prof. Dr. Marcos Ricardo Alcântara Morais**
Convidado

Campina Grande, Paraíba
26 de fevereiro de 2019

# Agradecimentos

Agradeço, primeiramente, aos meus pais, que sempre se esforçaram para que eu pudesse ter as melhores oportunidades de ensino. Agradeço também ao meu irmão, por estar presente sempre que precisei.

Agradeço a Lucas, que me acompanhou em praticamente toda a jornada que foi esse curso. Muito obrigada pela (enorme) paciência e pelo companheirismo de sempre.

Agradeço aos professores de Engenharia Elétrica da UFCG, que se esforçam diariamente para entregar a sociedade os melhores engenheiros. Agradeço principalmente aos professores orientadores (prof. Marcos e prof. Gutemberg) do Laboratório em Excêlencia em Microeletrônica no Nordeste (XMEN) por fazerem a diferença em nosso meio acadêmico e buscarem sempre mais - o que, por consequência, fez o laboratório nascer. Agradeço também pela oportunidade de trabalhar no laboratório, sem dúvida, foi a melhor oportunidade que pude ter durante a graduação.

Agradeço ao Ramo Estudantil IEEE UFCG e todos seus voluntários. Com vocês aprendi que a gente pode buscar sempre mais, há sempre o que melhorar e a enxergar que as diferenças podem nos levar muito além.

Agradeço aos todos membros do XMEN, com os quais aprendi muito. Agradeço especialmente a Felipe, que nos ensinou muito mais do que o necessário com entusiasmo e paciência. Microeletrônica é muito lindo, mas com certeza a experiência dentro do laboratório foi mais legal pelo ambiente que criamos, que só foi possível porque todo mundo era meio doido.

Agradeço a todos que compõe a Idea!, especialmente a Jacklyn e a equipe de *backend*, Davi, Schneider, Leo e João, por proporcionarem um ambiente de aprendizado contínuo e estarem sempre disponíveis para me orientar em todas dúvidas.

*"Any form of human creativity is a process*
*of doing it and getting better at it.*
*You become a writer by writing.*
*There is no other way. So do it, do it more.*
*Do it better. Fail. Fail better."*
*(Margaret Atwood)*

# Lista de ilustrações

# Sumário

# 1 Introdução

Este relatório descreve as atividades desenvolvidas durante a execução da disciplina curricular Estágio Integrado. O estágio foi realizado na empresa *Idea! Electronic Systems*, no setor de microeletrônica e este relatório se refere as atividades realizadas entre 10 de Setembro de 2018 e 22 de Fevereiro de 2019, no total de 960 horas.

## 1.1 Microeletrônica

Os avanços progressivos nos processos de fabricação de circuitos integrados (CIs) vêm permitindo a concepção de circuitos eletrônicos cada vez mais densos. Em 1958, por exemplo, Jack Kilby desenvolveu pela *Texas Instruments* o primeiro *flip-flop* em CI contendo dois transistores. Já em 2008, o microprocessador Itanium da Intel já continha 2 bilhões de transistores.

Este processo de miniaturização dos dispositivos torna impraticável o desenvolvimento manual destes circuitos, por isto foram desenvolvidas ferramentas de automação de desenvolvimento eletrônico (*Electronic Design Automation* - EDA), que hoje em dia são dominadas por empresas como a *Cadence* e a *Synopsys*. Além disto, desenvolveram-se diversos modelos de produção neste setor, como aqueles que são especializados no projeto dos circuitos eletrônicos, que são conhecidos como casas de desenvolvimento ("design houses") como a ARM; aqueles que são responsáveis pela concepção física dos projetos, conhecidas como fábricas ("foundries"), como a XFab; aqueles que lidam com interação com cliente e desenvolvimento de projetos, denominados fabricação sem fábrica ("fabless foundry"), como a NXP; e aqueles responsáveis por todo o projeto, desde contato com cliente até desenvolvimento do produto final, denominados fabricantes de dispositivos integrados ("Integrated Device Manufacturer" - IDM), como a Intel.

As foundries apresentam diferentes processos de fabricação dos dispositivos, e fornecem aos seus clientes um conjunto de arquivos contendo células pré-definidas e regras de projeto que precisam ser respeitadas, que é chamado kit de projeto físico ("Process Design Kit" - PDK). O projeto de circuitos integrados então é realizado utilizando as ferramentas de EDA em conjunto com o PDK fornecido pela indústria.

## 1.2 A Idea! Electronic Systems

A Idea! Electronic Systems, localizada em Campinas - SP, é uma empresa de serviços de alta tecnologia focada na evolução do estado da arte da fotônica e da microeletrônica.

A empresa vem alcançando as condições para atingir suas metas investindo em um time de engenheiros altamente capacitados, construindo parcerias com empresas de alto nível (Figura 1) e investindo em salas limpas, equipamentos e ferramentas de desenvolvimento (IDEA!, 2019).

Seus projetos, focados na transmissão digital de informação, colaboram para suprir a crescente demanda no tráfego de dados digitais. Estes avanços estimulam o que há de mais inovador no campo de processamento de sinais. Além disso, o mercado de comunicações por fibra óptica revolucionou a indústria de telecomunicações principalmente por proporcionar transmissões de dados a uma elevada taxa, por distâncias bem maiores e com menos perdas.

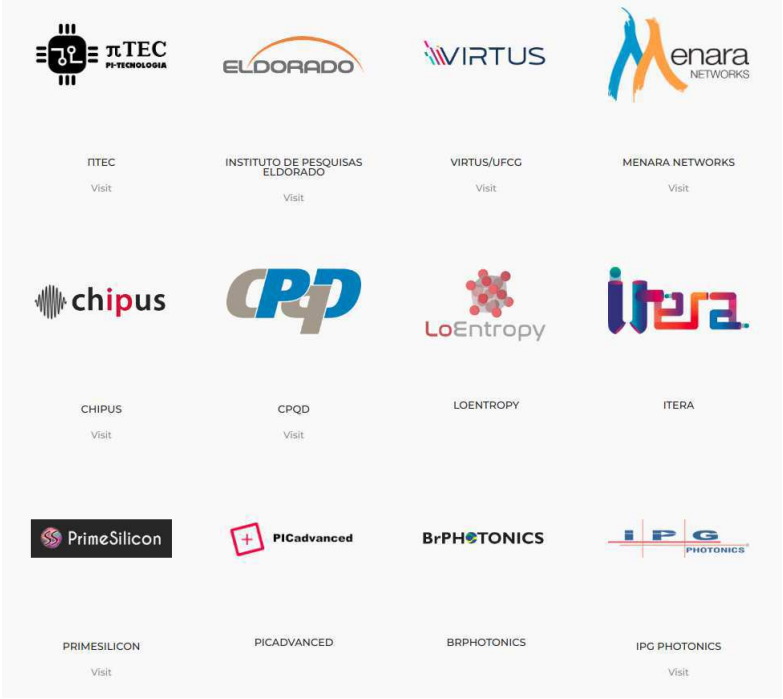



Figura 1 – Empresas parceiras da Idea!.

Com a comunicação óptica como um motivador, os mercados globais de telecomunicações e de comunicação de dados (*data-centers*) são os principais alvos das soluções da Idea!. A empresa é organizada em três divisões de desenvolvimento: microeletrônica, fotônica e engenharia de produto. Dentre as tecnologias desenvolvidas por essas divisões, destacam-se: *Application Specific Integrated Circuits* (ASICs), *Photonic Integrated Circuits* (PICs), laser sintonizáveis e amplificadores ópticos.

## 1.3 Objetivos

Este estágio tem como objetivo que a aluna desempenhe as seguintes tarefas:

- Estudo e ajustes de melhorias/adaptação do algoritmo implementado em nível de RTL, execução preliminar do fluxo de síntese lógica na tecnologia CMOS usada para implementação do circuito;
- Estudar documentação referência do fluxo de implementação física de DSP-ASICs focando em tecnologias deep-submicron abaixo de 28 nm;
- Estudar os manuais de referência do processo de fabricação de circuito digital e ferramentas de EDA usados durante o projeto;
- Executar síntese lógica e física dos blocos de RTL que serão implementados fisicamente em tecnologia deep-submicron;
- Executar testes e experimentos de implementação física do bloco de DSP no processo de fabricação e usando as ferramentas de EDA estudas na tarefa anterior;
- Analisar os resultados obtidos, como relatórios de temporização, regras de projeto, equivalência lógica/física, contagem de células, área do circuito, integridade de potência, entre outros resultados obtidos durante a tarefa de implementação física.

Essas atividades fazem parte de uma capacitação tanto no fluxo de microeletrônica quanto nas ferramentas de EDA.

# 2 Embasamento Teórico

O design de circuitos integrados (CIs) é dividido em várias etapas, um diagrama dessas etapas pode ser visto na Figura 2. Inicialmente tem-se a especificação do circuito a ser desenvolvido que é seguida pela sua implementação em linguagem de descrição de hardware. Essas etapas são classificadas como *frontend* e possuem um nível de abstração mais alto, isto é, não há preocupação com detalhes de implementação física (*Physical Design*), como os transistores a serem utilizados, capacitâncias e resistências parasitas e atraso na propagação do sinal. É nas etapas seguintes, classificadas como *backend*, em que estes detalhes são tratados.

Após a implementação física do circuito, é necessário passar por algumas etapas de verificação física do projeto. O próximo passo após o projeto físico verificado é o processo de fabricação que é feito nas *Wafer Fabrication Houses* (ou *foundries*). São as *foundries* que fornecem as bibliotecas de tecnologia, que contém informações como o tipo de wafer de silício usado, as células padrão usadas e regras de layout.

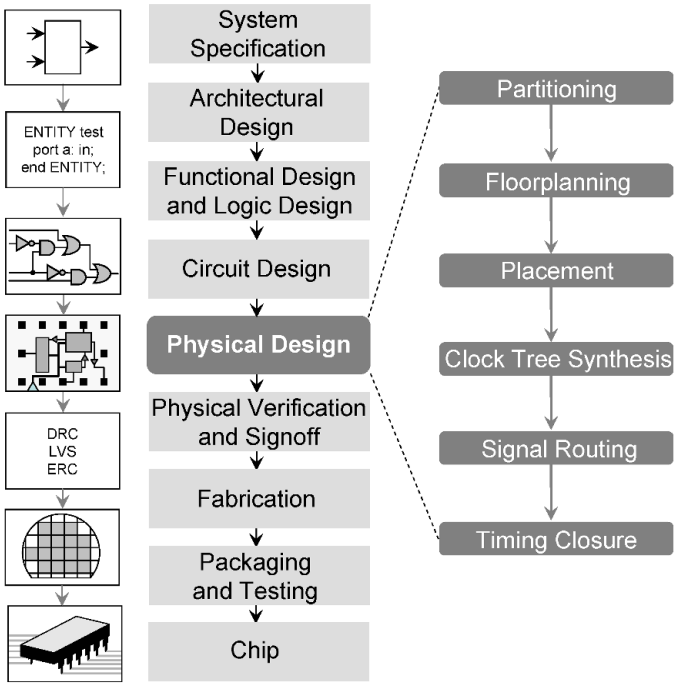



Figura 2 – Fluxo de Projeto de Microeletrônica. Fonte: (WIKIPEDIA, 2019)

## 2.1 Visão Geral do Fluxo de Microeletrônica Digital

O primeiro passo para a realização do projeto é a especificação do sistema. Nela são definidas as aspectos físicos, como entradas e saídas, frequência de operação, potência,

área, etc, e aspectos funcionais, isto é, a finalidade do sistema. Estas características podem ser definidas por uma demanda de um cliente específico ou pela própria empresa que visa atingir um mercado em potencial, ou por um arcordo entre os dois.

Após a especificação, são planejados os detalhes da implementação do circuito, tais como os blocos constituintes, largura dos barramentos, lógica a ser inserida, tamanho da memória, etc. Desta forma, é possível a codificação do sistema em Linguagem de Descrição de Hardware (*Hardware Description Language - HDL*). O nível de abstração do circuito após essa codificação é chamado de *Register Transfer Level* (RTL).

Os arquivos do circuito descrito em HDL passam por uma transformação denominada síntese lógica, que mapeia o RTL para transistores e portas lógicas da tecnologia a ser utilizada no processo de fabricação. O resultado da síntese é um arquivo chamado *netlist*, cujo nível de abstração é denominado *Gate Transfer Level* (GTL). Esse é o arquivo de entrada para a etapa de implementação física que será explicada na próxima seção.

O processo de descrição de *hardware*, síntese lógica e implementação física é ilustrado na Figura 3. Nela, é possível notar os diferentes níveis de abstração presentes no fluxo de microeletrônica: começando pelo código em HDL que apresenta expressões condicionais e atribuições sequenciais de forma mais clara sem se preocupar com detalhes de implementação (HDL *View*); passando pela *netlist*, onde percebe-se a utilização de portas lógicas (NOR3_4, INVERT_A) e *flip-flops* (D_F_LPH0001_4) - estes, por sua vez, já foram anteriormente caracterizados pela *foundry* (*Netlist View*); finalmente, na implementação física são especificadas as posições dos blocos, os caminhos dos fios, as posições dos pinos de entrada e saída, etc (*Physical View*).

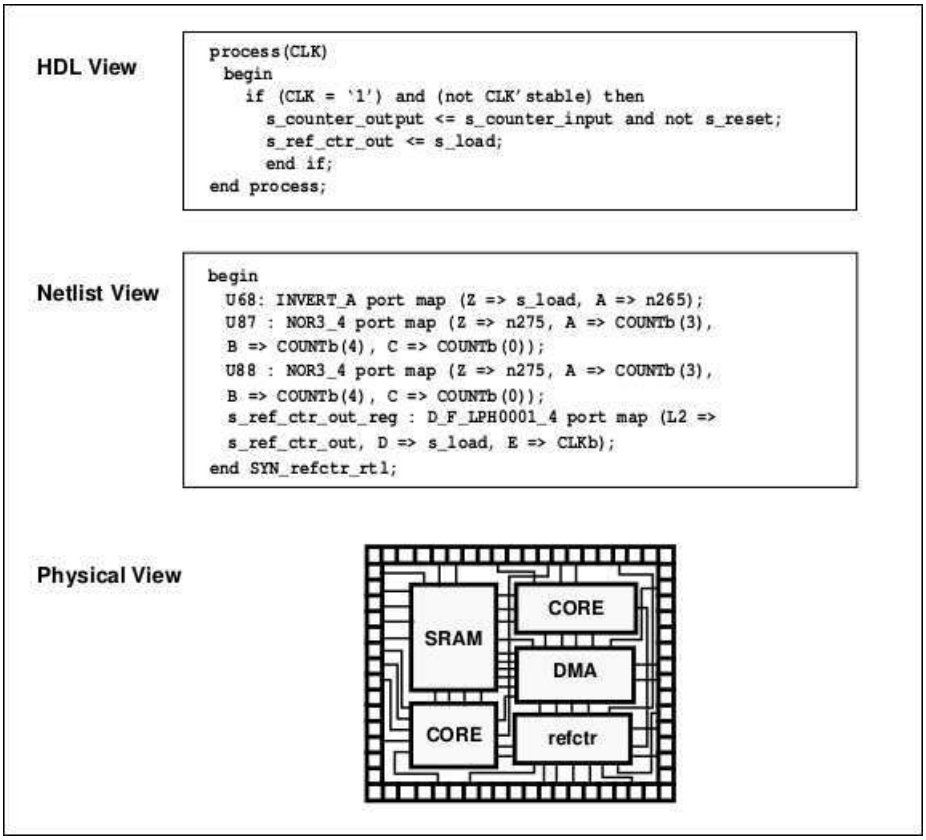



Figura 3 – Visão do *design* da descrição à implementação física. Fonte: (IBM, 1998)

## 2.2 Implementação Física (*Backend*)

O processo de produção de um *chip* é bastante minucioso, processando o semicondutor a ser utilizado (normalmente o silício) por várias etapas envolvendo máscaras que protegem-no ou revelam-no de modo a introduzir ou remover camadas. Na Figura 4 é possível observar a ilustração didática da implementação física das camadas de um inversor, constituída apenas cinco camadas sobre as quais são depositados diferentes tipos de materiais. Para a fabricação, é necessário pelo menos uma máscara por camada para definir as regiões que terão determinado material.

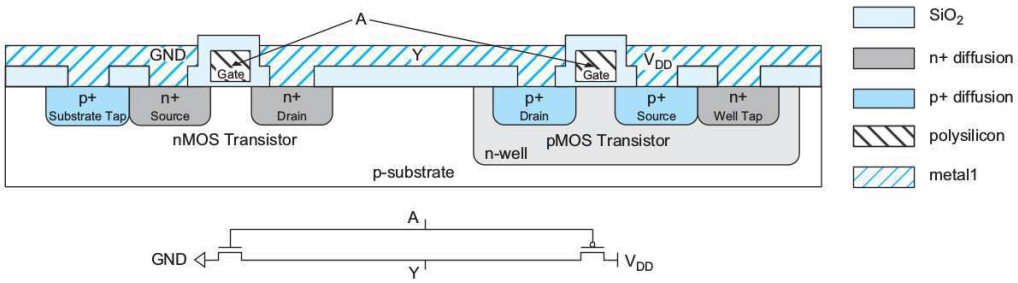



Figura 4 – Corte transversal de um inversor. Fonte: (WESTE; HARRIS, 2011)

Os *chips* fabricados atualmente são classificados como integração em larga escala (*Very Large Scale Integration* - VLSI), nos quais milhares de transistores são integrados,

consequentemente o processo de produção também necessita de mais máscaras. Devido ao grande número de componentes, seria inviável projetá-los manualmente, sendo necessário o uso de ferramentas de *Electronic Design Automation* (EDA) e do *Process Design Kit* (PDK) fornecido pela *foundry*.

As etapas da implementação física (*backend*) tomam como base a *netlist*, que descreve as portas lógicas e conexões a serem implementadas, associada com a automação permitida pelas ferramentas de EDA e com o conhecimento físico do PDK para confeccionar as máscaras constituintes do *chip* a ser fabricado. Essas etapas são denominadas: *Floorplan*, *Placement*, *Clock Tree Synthesis* (CTS), Roteamento e *Static Time Analysis* (STA). Após essas etapas, o projeto deve passar por um processo de verificação da implementação física (*Physical Verification*), as principais verificações são a de *Design Rule Check* (DRC) e *Layout versus Schematic* (LVS) - a primeira checa regras de design e a segunda checa se o *layout* (máscaras de litografia) correspondem ao circuito que foi projetado.

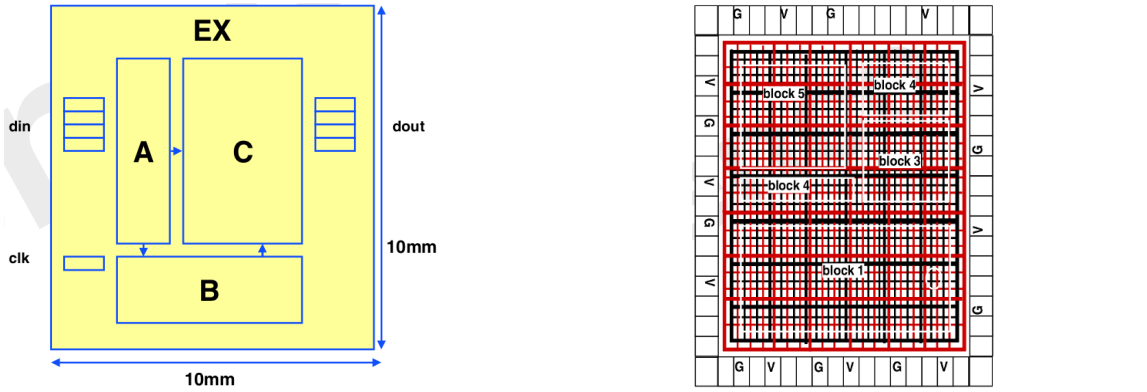
### 2.2.1 *Floorplan*

O *floorplan* é a primeira etapa do *backend*, é nele em que se organiza os componentes do *chip*, posicionando-os com uma visão geral do circuito. É o processo de identificar estruturas que devem ser colocadas próximas e alocar espaço para elas de uma maneira que a área ocupada esteja de acordo com os requisitos do produto, visando também melhorar desempenho, visto que a organização dos blocos pode facilitar ou prejudicar a interconexão entre eles.

O *floorplan* posiciona todos os componentes do *design*: memórias, blocos lógicos (circuitos) e os pinos de entrada e saída, levando em consideração a necessidade de fixar um bloco em uma determinada área e a viabilidade de roteamento do *chip* final, isto é, blocos que possuem muitas conexões entre si devem ser posicionados próximos um do outro. Já para o posicionamento dos pinos o *floorplan* tem como objetivo identificar os caminhos mais críticos e tentar minimizá-los, além de tentar minimizar o tamanho de todas as conexões fazendo rotas mais diretas, resultando em um bom *timing* (quando o sinal percorre seu caminho dentro do tempo previsto, normalmente um período de *clock*) e menor congestionamento de fios.

Uma das principais decisões de engenharia do *floorplan* é realizar alocar os componentes do *chip* de forma a conseguir conectá-los e não desperdiçar área e nem congestionar o roteamento de alguma área. É necessário fazer um bom *trade-off* entre área e performance, utilizando a menor área possível para realizar corretamente todas as conexões sem problemas de integridade de sinal e de *timing*. Otimizar o *design* para usar uma área menor permite que seja usado menos recursos de roteamento e, ao mesmo tempo, que os blocos fiquem mais próximos uns dos outros. Isso leva a distâncias menores de interconexão e caminhos de sinal mais curtos. A Figura 5a ilustra um exemplo de *floorplan*.

Além disso, também é no *floorplan* que é decidido onde ficarão as fontes de alimentação do *chip*. Normalmente, é feito um plano de alimentação (*power plan*), com o objetivo de cobrir todo o *chip* e fornecer energia necessária para todas as instâncias. É nessa fase que é decidido o tamanho dos fios de alimentação (*power stripes*), os espaços entre eles, de forma que obedeça às larguras mínimas e espaçamentos mínimos para a tecnologia e de forma que evite efeitos de queda de tensão (*IR drop*), deslocamento de terra (*ground bounce*) e eletromigração (*electromigration*). A Figura 5b ilustra um *floorplan* após a inserção do plano de alimentação.




(a) Ilustração de *floorplan* (b) Ilustração de *floorplan* depois do *power plan*.

Figura 5 – Ilustrações de *floorplan*.

### 2.2.2 *Placement*

Depois do *floorplan*, é necessário realizar o *placement*, que é o processo de posicionar as *standard cells* do *design* planejado. As *standard cells* são células de portas lógicas e *flip-flops* encontradas no PDK, elas possuem a mesma altura e são posicionadas em linhas, de forma a compartilharem as mesmas tensões de alimentação, seja VDD ou VSS (Figura 6). É possível ver o processo de *placement* de forma parecida com o de *floorplan*, pois ele tem como objetivo posicionar as *standard cells* dentro dos blocos de uma maneira que o roteamento seja viável, visando diminuir congestionamento e caminhos críticos.

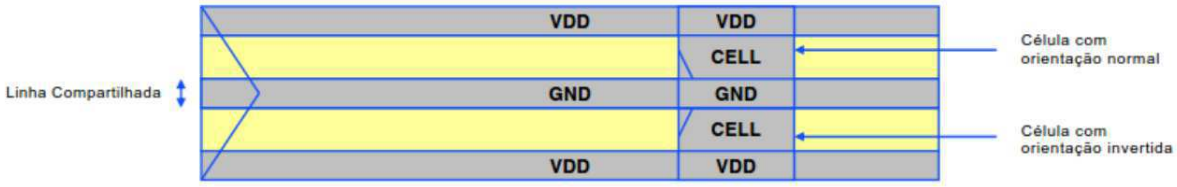



Figura 6 – *Standard cells* compartilhando a mesma alimentação. Fonte: (MEDEIROS, 2018)

O primeiro passo da ferramenta no *placement* é chamado de *global placement*. Nessa etapa, as células são espalhadas de forma um pouco mais grosseira, havendo muitas vezes

sobreposição entre células.

O objetivo maior nesse passo é ter um posicionamento global. Após o *global placement*, é feita uma etapa de legalização, onde as sobreposições são corrigidas, e as células também são corrigidas para ficarem com a mesma altura.

A terceira etapa é chamada de *detailed placement*. É nessa etapa onde são feitas melhorias, pois até agora, o *placement* só foi distribuído de forma grosseira. A ferramenta de *placement* irá tentar minimizar a distância de células que se conectam, com objetivo de diminuir quantidade de fios. Visa também diminuir congestionamento, pensando na viabilidade do roteamento. Essas mudanças são feitas a partir do reposicionamento das células no *design*. Pode ser visto na Figura 7 um esboço de um *design* após *placement*.

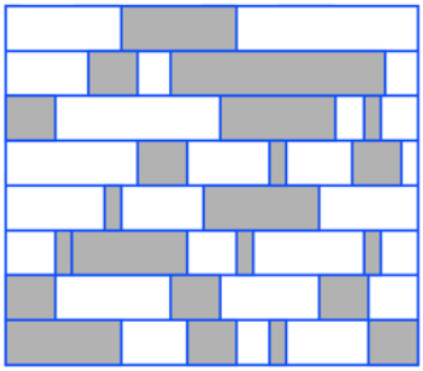


Figura 7 – Esboço de *placement*. Fonte: (MEDEIROS, 2018)

### 2.2.3 *Clock Tree Synthesis*

Na microeletrônica digital, o sinal de *clock* é um dos mais importantes do circuito, já que ele é usado para coordenar as ações do circuito completo. A mesma rede de *clock* conecta todos os elementos síncronos no *design*, independentemente da quantidade e das distâncias até esses elementos. A dificuldade para fazer essa distribuição aumenta quando temos um *chip* grande, com distribuição não uniforme, e o *clock* precisa chegar ao mesmo tempo para todas as instâncias. Uma visão geral de uma árvore de clock (clock tree) pode ser vista na Figura 8.

Em um *design* muito extenso e uma *clock tree* grande, também haverá uma grande quantidade de fios para poder fazer todas as ligações. Fios longos significam um atraso no tempo de chegada do *clock*. O desempenho pode ser impactado pelo *skew* associado grande *fanout* da *clock tree*, que causa atraso de sinal e também prejuízo no tempo de transição. O processo de corrigir algumas dessas dificuldades inclui a inserção de *buffers* para adiantar o sinal de *clock* e assim reduzir o atraso (*delay*) entre a fonte de *clock* e os registradores de destino.

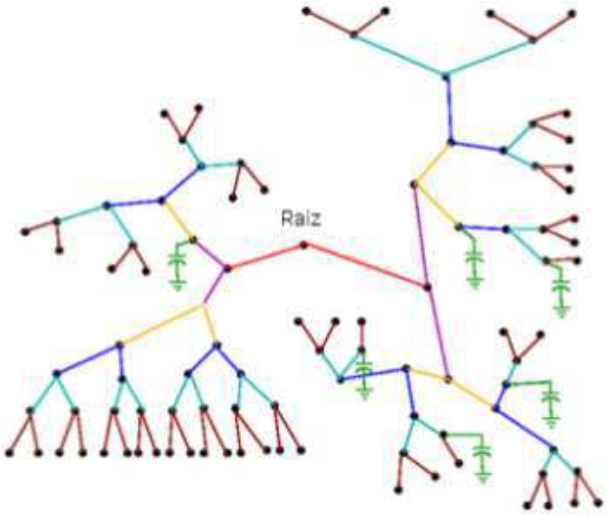



Figura 8 – Ilustração de uma *Clock Tree*. Fonte: (MEDEIROS, 2018)

O objetivo da *clock tree synthesis* é assegurar que o sistema funcione corretamente com a frequência de *clock* planejada, inserindo buffers para minimizar *skew* e latência, otimizando *timing*. Ao final da síntese, a árvore de *clock* tem que entregar o *clock* com uma transição aceitável, minimizando consumo de energia, já que o *clock* está sempre comutando.

Para que a ferramenta usada consiga fazer a *clock tree*, o projetista deve informar alguns parâmetros que ele quer que seja obedecido. Deve ser definido quais as camadas que a *clock tree* deve ocupar, *constraints* (regras limitantes) que devem ser satisfeitas, como máximo *insertion delay*, máximo *skew*, definir tipos de *buffers* que devem ser colocados, máximo *fanout* desses *buffers*, entre outros parâmetros. Após os buffers serem inseridos a CTS é roteada, além de passar por otimizações para atingir *timing*.

### 2.2.4 Roteamento

Após o *placement* das células e das macros, as conexões entre do *chip* precisam ser formadas usando metal e vias, caracterizando a etapa de roteamento, como pode ser visto na Figura 9. Após esta etapa, as medidas de *timing* ficam mais exatas, já que o *design* já possui um *clock* real e já está todo conectado. Todas essas conexões precisam obedecer às regras de *design*, levando em consideração fatores como: largura dos fios, regras de área mínima, espaçamento entre fios, integridade de sinal.

A ferramenta de roteamento enxerga o *design* como uma malha, dividida em linhas horizontais e verticais, onde essa malha serve como um guia para roteamento, tendo uma área como referência nessa malha e algumas vizinhas a ela para rotear. Isso é feito com objetivo de simplificar a complexidade dos algoritmos matemáticos responsáveis pelo *routing*.

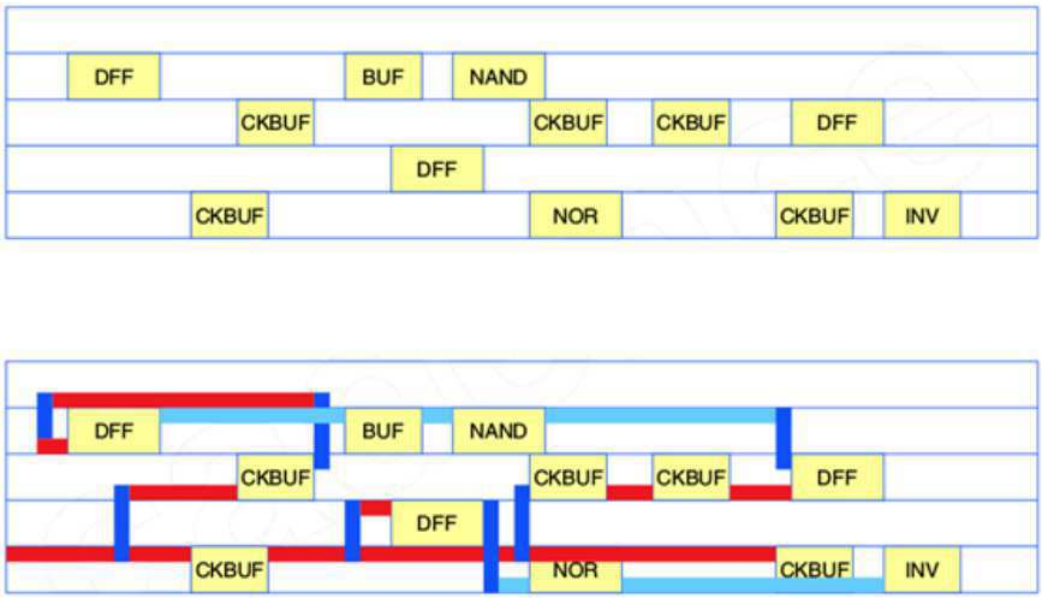



Figura 9 – Antes e depois do roteamento. Fonte: (MEDEIROS, 2018)

As linhas de metal precisam ter uma largura mínima, e devem ser espaçados por um intervalo mínimo, a depender da tecnologia de fabricação. Essas larguras e espaçamentos existem, pois, deve-se evitar curto circuito ou circuito aberto durante a fabricação. Congestionamento acontece quando existem mais fios a ser passados do que fios disponíveis para roteamento.

### 2.2.5 *Static Time Analysis* (STA)

O objetivo de fazer uma análise de tempo (*Static Time Analysis* - STA) é verificar se o *design* está cumprindo os requerimentos de *timing*, dado certas *constraints* de *timing*. Essa etapa pode ser executada em qualquer estágio no fluxo de *backend*. Essa etapa permite determinar o quão rápido um design pode operar sem violar *timing* e pode ser executada em qualquer estágio no fluxo de *backend*.

A etapa de STA calcula os *delays* de todos os caminhos possíveis do circuito e compara com as *constraints* de *timing*, analisando todos os possíveis caminhos. STA ignora funcionalidade do circuito, analisando caminhos que logicamente podem não ser percorridos. Esses caminhos devem ser definidos como *false path* de *timing* pelo projetista.

As *constraints* de *timing* representam os objetivos do seu *design*, como definição de *clock*, *delay* de entrada, *delay* de saída e condições de operação. As *constraints* são verificadas antes de entrar no processo de STA, para verificar se foram definidas todos os *delays* de entrada e saída, ou se existem erros, como múltiplos *clocks* chegando em uma célula.

### 2.2.6 Verificação Física

A última etapa antes de entregar o GDSII (arquivo final da implementação física) é a verificação física. Nessa etapa, o *design* final passa por ferramentas de EDA as quais verificam se o *chip* está apto a ser fabricado. As duas principais verificações são as a *Design Rule Check* (DRC) e *Layout versus Schematic* (LVS), a primeira checa regra de *designs* da *foundry* e a segunda checa se o *layout* (máscaras de litografia) correspondem ao circuito que foi projetado.

#### 2.2.6.1 Design Rule Check

O *Design Rule Check* verifica se o *layout* está de acordo com as regras de *foundry*, ou seja, se o *layout* será fabricável. Essas regras envolvem, por exemplo:

→ Geometria: Verifica medidas como largura, comprimento, espaçamento, área e extensão dos fios.

→ Antena check: Evita deixar muito metal numa camada só, prevenindo acumulação de cargas nesse metal.

→ Manufaturabilidade (*Yield*): Verifica se há o uso de vias redundantes, células mais robustas e uso de fios mais grossos, com o objetivo de, mesmo com uma falha de fabricação do *wafer*, o *design* ainda atinjam um maior rendimento (maior probabilidade de *chips* funcionais em um mesmo *wafer*).

#### 2.2.6.2 Layout versus Schematic

O Layout Versus Schematic é a etapa de verificação de *design* que determina se um determinado *layout* de um *chip* corresponde ao esquemático original ou diagrama de circuito do projeto.

Uma verificação de DRC bem-sucedida garante que o *layout* esteja de acordo com as regras exigidas para fabricação sem falhas, no entanto não garante se representa realmente o circuito que deseja fabricar. É por isso que também é necessário uma verificação LVS.

LVS compara o *netlist* com o GDSII para assegurar funcionalidade. Na maioria dos casos, o teste de LVS aponta falhas no *layout* ao ser executado pela primeira vez, exigindo que o engenheiro responsável examine os relatórios gerados e faça alterações no *layout*. Erros típicos encontrados durante o LVS incluem:

→ Curto-circuitos: Dois ou mais fios que não devem ser conectados em contato entre si.

→ Circuito aberto: Os fios ou componentes que devem ser conectados ficam desconectados ou parcialmente conectados.

→ Incompatibilidade de células: Um tipo incorreto de células foi utilizado.

→ Componentes ausentes: Um componente esperado foi deixado de fora do *layout*.

# 3 Atividades desenvolvidas

## 3.1 Análise de Potência

No processo de especificação de projeto de microeletrônica três aspectos importantes são considerados na análise, são eles: desempenho, área e consumo de energia. O feedback inicial sobre os requisitos de energia de um projeto permite que os projetistas façam trocas de projetos básicos oportunos para atingir os objetivos de energia.

O cálculo de potência de um circuito é determinado por: $P = \alpha f C V^2$, onde $\alpha$ é o fator de atividade daquele circuito, é relacionado com a funcionalidade daquele circuito, isto é, em termos de sinais digitais (binários) quantas vezes seus sinais mudaram de 0 para 1 ou ao contrário. Já os outros fatores $f$, $C$ e $V^2$ são, respectivamente, frequência, capacitância e tensão do circuito, são valores mais fixos, apesar da capacitância depender da implementação física do circuito também.

Quando fala-se de circuitos integrados digitais, a dissipação de energia para um circuito consiste em:

1. Dissipação de Energia de Fuga (*Leakage*);
2. Dissipação de Energia nos caminhos (*nets*) que os sinais percorrem;
3. Dissipação de Energia Interna.

A ideia desta atividade é fazer uma estimativa de potência dos blocos antes da implementação física deles, então, o foco eram as potências de *leakage* e interna dos blocos, já que a potência das *nets* não são fáceis de estimar.

No caso da atividade desenvolvida, o objetivo era fazer a análise de potência de todos os blocos do projeto para diferentes tensões e diferentes células da tecnologia. O foco principal era fazer a comparação entre algumas características intrínsecas da tecnologia, como o impacto de células multibit e também o impacto de células DDB (*Double Diffusion Break*) ou MDB (*Mixed Diffusion Break*), todas estas características se referem a detalhes do processo de fabricação.

Esta atividade consistia na realização da síntese lógica dos circuitos, depois simulação para posteriormente fazer a análise de potência. No total, o projeto conta com 18 blocos de circuitos eletrônicos, então, para começar foi realizada a atividade em apenas 2 blocos, um relativamente pequeno, como forma de treinamento e outro maior, para analisar melhor o impacto no projeto como o todo.

Para a síntese foi utilizada a ferramenta *Genus*, da *Cadence*. A síntese consiste na transformação do código descrito em linguagem de *hardware* (HDL - *Hardware Description Language*) em circuitos lógicos. Isto é, a síntese lê o código *RTL - Register Transfer Level* (arquivos .v ou .sv) junto com bibliotecas físicas das células padrão que podem conter informações de atraso (arquivos .lib), dimensões físicas e informações da camada de metal dentro da célula (arquivos .lef) e outras arquivos de restrição para converter o código comportamental ou de fluxo de dados em portas de células padrão físicas reais. Na Figura 10, pode-se ver como a ferramenta realiza uma síntese: primeiro, a ferramenta traduz o circuito para células padrão e depois mapeia para a tecnologia a ser utilizada.

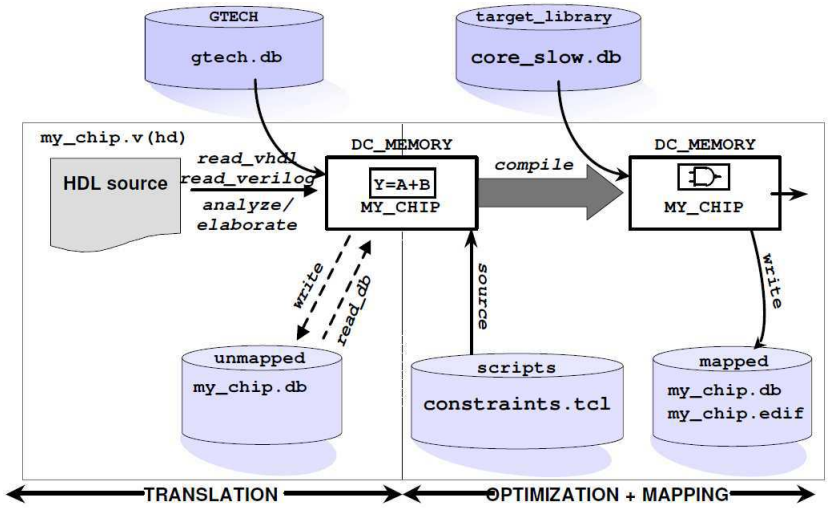



Figura 10 – Etapas da Síntese Lógica. (SYNOPSYS, 2017)

Ao final da síntese, tem-se dois arquivo de saída: um deles também em HDL mas agora contém informações do circuito mapeado para tecnologia - esse arquivo é chamado de *netlist* e outro com a anotação de tempo, é um arquivo SDF *Standard Delay Format*, que a ferramenta consegue gerar extraindo as informações dos arquivos .lib e .lef fornecidos pela *foundry*.

Depois da síntese, foi realizada a simulação com a ferramenta *Xcelium*, também da *Cadence*. A simulação tinha como arquivos de entrada a *netlist* gerada pela síntese, o modelo arquitetural desenvolvido em *SystemC* pela equipe *frontend* e arquivos com estímulos de entrada para o bloco e gerava arquivos de saída com os resultados do bloco (do modelo arquitetural e da netlist) e um arquivo que registra todas as variações dos sinais ao longo da simulação, o formato desse arquivo é *VCD - Value Change Dump*.

Esta simulação era o ponto base da atividade, os resultados do modelo arquitetural tinha que coincidir com os resultados da *netlist*, para garantir a funcionalidade do circuito e a simulação gerava o arquivo VCD do caso teste para que possa ser calculado o fator de atividade daquele bloco.

Após a simulação, era necessário voltar a ferramenta *Genus* com o arquivo VCD

para finalmente realizar a análise de potência. Então, de volta ao *Genus*, é necessário carregar um arquivo *database* que ele mesmo gerou no processo de síntese e o arquivo gerado pela simulação. Durante a síntese, a ferramenta já extraído dos arquivos de entrada (*constraints* e os arquivos da tecnologia) as informações de tensão e frequência e também já havia feito cálculos para estimar a capacitância do circuito (inclusive para gerar o SDF). Então, com todas essas informações juntamente com o arquivo VCD gerado pela simulação, a ferramenta consegue estimar a dissipação de potência no circuito.

Como citado anteriormente, esta atividade foi feita para dois blocos do projeto, e os testes realizados foram:

1. Células DDB, com tensão 0.50 V;
2. Células DDB, com tensão 0.55 V;
3. Células DDB e multibit, com tensão 0.55 V;
4. Células MDB, com tensão 0.55 V;
5. Células MDB e multibit, com tensão 0.55 V;

Com esses testes, foi possível notar que as células multibit ajudaram a diminuir a potência interna principalmente para lógica sequencial. Já entre células DDB versus MDB não foi observado tanta diferença, apenas que o MDB juntamente com multibit tinha a tendência de trazer melhores resutaldos. Em seguida, foi realizado mais dois testes elevando as tensões desses blocos para 0.65V e 0.75V, apesar da potência de *leakage* praticamente desaparecer com essas tensões, a potência total ficou maior em ambos os casos.

Então, com os resultados desses blocos, foi escolhido fazer a estimativa de potência para todos os blocos com células MDB e multibit com tensão 0.55V. Esta análise pôde constatar que as células multibit em blocos com muita lógica combinacional não fazia muita diferença, as vezes aumentava o consumo de potência, mas em blocos com muita lógica sequencial era possível ganhos de até 24%.

## 3.2 Equivalência Lógica

Para fabricação de um chip é necessário passar por muitas etapas e muitas ferramentas diferentes. Mas para conseguir confiar que o que vai ser projetado pela *foundry* é realmente aquilo que foi especificado precisa-se passar por várias etapas de "checagem". Uma dessas etapas é a checagem da equivalência lógica, este teste checa a equivalência entre o RTL desenvolvido pelo *design* e a *netlist* que a ferramenta de síntese gerou.

Essa atividade foi realizada com a ferramenta *Conformal Equivalency Check*, da *Cadence*. Novamente, o objetivo era checar o efeito das células multibit e principalmente

garantir que a *netlist* gerada pela ferramenta era equivalente ao RTL que havia sido projetado. Também foi feito primeiro um teste com um bloco que havia sido sintetizado na atividade anterior como forma de treinamento.

A ferramenta *Conformal* realiza a checagem em duas etapas: primeiro entre o RTL e uma *netlist* intermediária que a ferramenta de síntese gera e posteriormente entre essa *netlist* intermediária com a *netlist* final.

Durante o próprio fluxo de síntese, a ferramenta da síntese gera um *script* para a ferramenta *Conformal*. Então, foi necessário apenas analisar os relatórios gerados em cada etapa da verificação de equivalência lógica.

No caso do primeiro bloco a ser testado, não houve nenhum erro e a equivalência lógica foi comprovada. Em seguida, foi feito o teste de equivalência lógica do projeto completo. Este último teste obteve algumas partes do circuito com não-equivalência lógicas. O resultado foi analisado e reportado para a equipe de *frontend*, responsáveis pelo projeto do *design*.

## 3.3 Treinamento em Implementação Física

Normalmente, o fluxo de *backend* não é amplamente explorado na academia, além das licenças das ferramentas de EDA serem muito caras, não tornando o seu uso viável para as universidades. Para suprir essa lacuna, todos os estagiários da área de *backend* da Idea passam por uma capacitação tanto teórica quanto prática. Para esta atividade, o líder da equipe de *backend* transmite seu conhecimento teórico para a estagiária em forma de pequenas "aulas" e recomenda alguma atividade teórica a ser realizada logo em seguida.

A Idea! possui um fluxo de *backend* bem consolidado, no qual foi possível que a estagiária utilizasse todos os *scripts* para realizar a implementação física de um bloco digital, com a tecnologia deep-submicron abaixo de 28nm, como forma de treinamento prático. Este bloco foi integrado em um topo analógico e foi enviado para *foundry* fabricar em um *Multi Project Wafer* (MPW).

### 3.3.1 Testchip - MPW

Como os custos de fabricação de um *chip* são extremamente altos, a fabricação só se torna viável quando se é fabricado milhares de *chip*. Visando a produção em baixas quantidates, existe o MPW, onde os recursos de máscara e *wafer* são compartilhados. Um MPW integra diversos *chips* em diferentes *wafers* de várias equipes, incluindo projetos de empresas privadas, projetos de estudantes e pesquisadores de universidades.

No cenário da Idea!, que está desenvolvendo o primeiro projeto no nó tecnológico atual, a oportunidade poder fazer um ou mais *testchips* tem grande valor para amadurecer

o fluxo de projeto interno da equipe. No geral, é uma oportunidade tanto para as empresas quanto para a *foundry* que também está amadurecendo seu processo tecnológico.

Um *testchip* normalmente usa uma lógica mais simplificada do projeto final, porém, procura preservar elementos importantes, como comunicação com memórias e interfaces. A interação entre a *design house* e a *foundry* nessa etapa pode evidenciar problemas que não foram pensados, influenciando diretamente na qualidade do *chip* final.

A estagiária teve a oportunidade por em prática o fluxo de *backend* da Idea!, com um bloco digital de uma *Phase Locked Loop* (PLL). Este bloco foi feito para ser integrado à parte analógica do *chip*, por isso suas definições de *floorplan* vieram considerando o topo analógico (Figura 11).

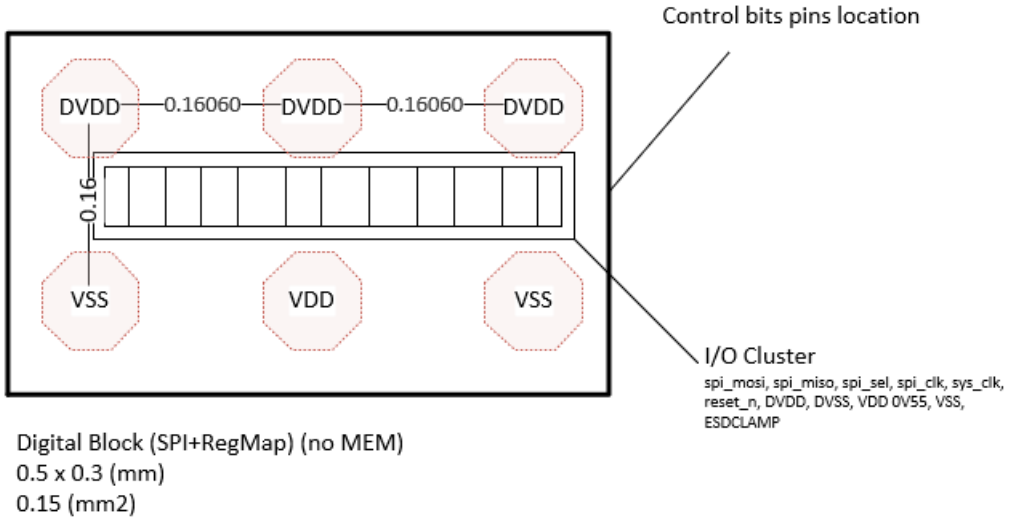



Figura 11 – *Floorplan* da PLL.

As etapas de *backend* foram seguidas utilizando os *scripts* de referência da empresa, realizando as adaptações adequadas ao bloco, principalmente na etapa de *floorplanning*, que apresenta características como pinagem e área intrínsecas aos blocos.

O bloco em questão é relativamente pequeno, contendo aproximadamente 3600 células, então, as etapas de *place and routing* foram executadas sem grandes dificuldades, sem problemas de densidade, congestionamento ou *timing*.

Por fim, foi possível realizar a verificação física do bloco. No LVS não foram encontrados problemas, porém no DRC indicou erro de densidade máxima entre as *stripes* de alimentação. Então, foi necessário voltar à etapa de *floorplan* e refazer as *stripes*, diminuindo a largura e aumentando o espaçamento entre elas. Depois dessa correção, o DRC não encontrou mais problemas.

O fluxo se mostrou bastante consolidado permitindo uma rápida familiarização com a tecnologia e uma alta flexibilidade para resolução de problemas e para implementação de blocos variados.

# Considerações finais

O estágio integrado realizado na Idea! Electronic Systems se mostrou extremamente importante para a formação da aluna, possibilitando o contato com o dia-a-dia de uma empresa que trabalha com o estado da arte da tecnologia de microeletrônica.

A empresa concedente teve papel determinante durante o período de estágio, tanto na definição das atividades quanto no apoio técnico e estrutural, resultando no êxito da execução das atividades, mesmo com a grande curva de aprendizado existente na área de *backend*.

Particularmente relevantes às atividades de estágio foram os conhecimentos adquiridos na disciplina de Arquitetura de Sistemas Digitais, Projeto de Circuitos Integrados e Estrutura e Concepção de Circuitos Intregados e em projetos realizados em paralelo durante o curso, mais especificamente, os projetos desenvolvidos no Laboratório em Excelência em Microeletrônica do Nordeste (XMEN), fonte dos conhecimentos prévios na área de microeletrônica.

Por fim, foi bastante enriquecedora a experiência de trabalhar na empresa, podendo contribuir com os resultados das análises de potência e equivalência lógica e, principalmente, podendo contribuir com a implementação física de um bloco para o *testchip*.

# Referências

IBM. *ASIC Design Methodology Primer - ASIC Products Application Note*. [S.l.]: IBM, 1998. Citado na página 19.

IDEA! *Idea! Converging Photonics & Microelectronics*. 2019. Disponível em: <www.idea-ip.com>. Acesso em: 18 fev. 2019. Citado na página 14.

MEDEIROS, J. L. P. *Relatório de Estágio - Idea! Electronic Systems*. Universidade Federal de Campina Grande, 2018. Citado 4 vezes nas páginas 21, 22, 23 e 24.

SYNOPSYS. *Introduction to Synthesis*. Synopsys Chip Synthesis Workshop, 2017. Citado na página 28.

WESTE, N. H. E.; HARRIS, D. M. *CMOS VLSI Design: A Circuits and Systems Perspective*. 4. ed. [S.l.]: PEARSON, 2011. Citado na página 19.

WIKIPEDIA. *Physical design (electronics)*. 2019. Disponível em: <https://en.wikipedia.org/wiki/Physical_design_(electronics)>. Acesso em: 18 fev. 2019. Citado na página 17.