UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Érico Ramalho de Freitas

ESTÁGIO INTEGRADO NA IDEA! ELECTRONIC SYSTEMS

CAMPINA GRANDE
2019

ÉRICO RAMALHO DE FREITAS

ESTÁGIO INTEGRADO NA IDEA! ELECTRONIC SYSTEMS

Relatório de Estágio Integrado submetido à Universidade Federal de Campina Grande, como requisito necessário para obtenção do grau de Bacharel em Engenharia Elétrica

Campina Grande, dezembro de 2019

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

ÉRICO RAMALHO DE FREITAS

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Engenharia de Elétrica, sendo aprovada em sua forma final pelo orientador:

Prof. Dr. Marcos Ricardo Alcântara Morais
Universidade Federal de Campina Grande

Campina Grande, 9 de dezembro de 2019

Lista de ilustrações

Sumário

1 Introdução

Este relatório tem como objetivo descrever as atividades desenvolvidas durante o período de Estágio Integrado realizado na empresa Idea! Electronic Systems do dia 03 de junho de 2019 a 03 de dezembro de 2019 totalizando uma carga horária de 960 horas.

O estágio em questão foi realizado no setor de Microeletrônica da referida empresa, especificamente na área de design físico (backend), na qual foram atribuídas ao estagiário as seguintes atividades:

- Estudo e familiarização com o fluxo de referência da Idea! para projeto físico dos circuitos digitais e estratégias de *backend*;
- Estudo, através de vídeo-aulas e documentações, sobre o fluxo de desenvolvimento físico de um bloco digital;
- Aplicação dos estudos no desenvolvimento de um bloco digital;
- Realização e análise de temporização do *ASIC* a nível de topo;
- Desenvolvimento de scripts para inserção de paredes isoladores em blocos digitais que possuem interface com a parte analógica do *ASIC*;

Essas atividades foram desenvolvidas sob supervisão dos coordenadores da área de *backend* Davi Castro e Eduardo Schneider e do diretor de Microeletrônica Jacklyn Dias Reis, sob orientação do professor doutor Marcos Ricardo Alcântara Morais.

1.1 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 das áreas de microeletrônica e fotônica, focando na fusão entre essas duas áreas (Idea! 2019).

A empresa vem alcançando as condições para atingir suas metas investindo em um time de engenheiros altamente capacitados (Figura 1), firmando parcerias com empresas de tecnologia de ponta (Figura 2) e investindo fortemente em infraestrutura, com salas limpas para confecção de lasers, equipamentos, servidores e ferramentas de automação eletrônica (*electronic design automation* - EDA).

Figura 1 – Time Idea! Fonte: https://www.idea-ip.com/about-us/, acesso em nov. 2019

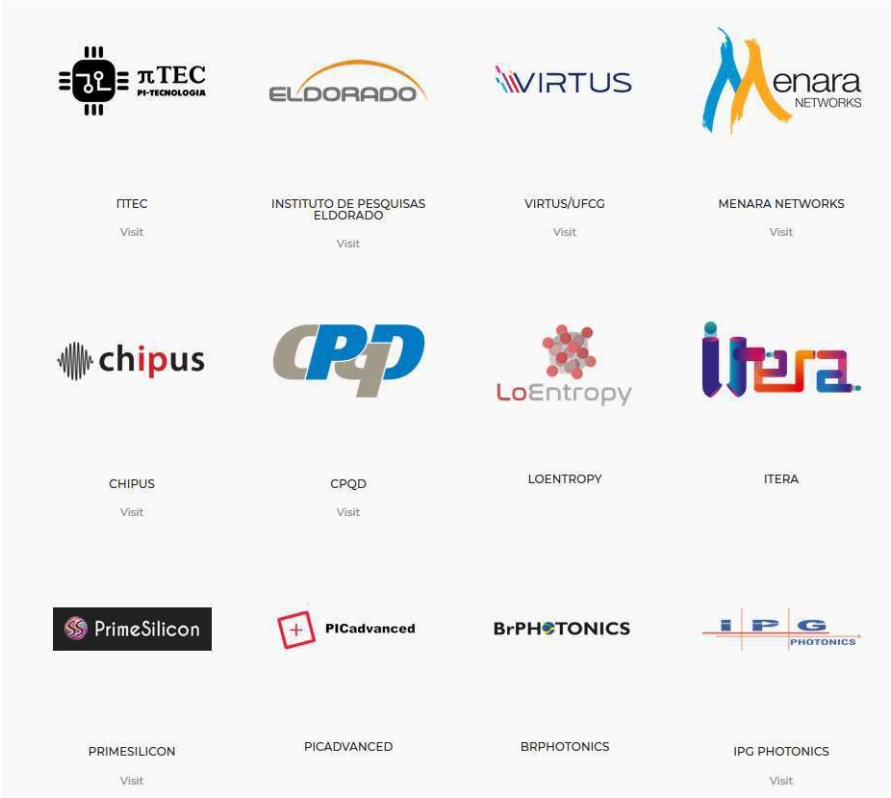


Figura 2 – Parceiros Idea! Fonte: https://www.idea-ip.com/partners/, acesso em nov. 2019

Seus projetos, focados na área de transmissão digital da informação, colaboram para suprir a crescente demanda no tráfego de dados digitais. Esses avanços estimulam o que há de mais inovador no campo de processamento de sinais, sendo um desafio motivador para as equipes.

Neste contexto, a empresa desenvolve tecnologias voltadas para as comunicações ópticas, como processadores digitais de sinais (DSPs), lasers sintonizáveis e amplificadores ópticos.

2 Embasamento Teórico

2.1 Fluxo de Projeto de um Physical Block

O projeto físico de um bloco lógico que faz parte de um *ASIC*(Application-specific Integrated Circuit) consiste na execução das seguintes etapas em sequência, porém muita vez havendo reiteração das etapas.

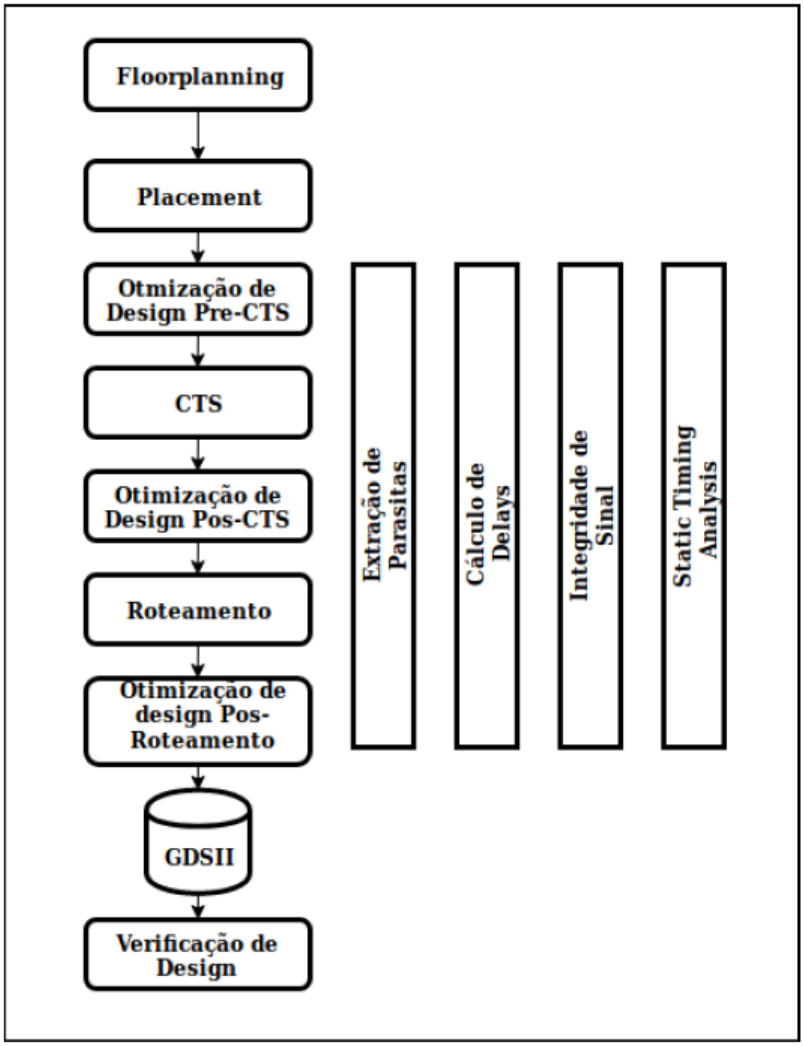


Figura 3 – Etapas de Projeto de um *Physical Block*. Fonte: Medeiros 2018

2.1.1 Floorplan

A primeira fase do fluxo consiste na definição da geometria e posição dos elementos de um projeto. São definidas a área total do bloco (ou *die*) e a área onde serão colocadas as células (ou *core*). Elementos importantes, como memórias, dependem muito de um bom posicionamento para que o projeto funcione adequadamente.

É também na etapa de floorplan onde são feitas as definições de roteamento de energia do bloco. Há dois tipos de roteamento mais usados:

- **Power Stripes :** Trançado alternado de linhas de VDD e VSS sobre o bloco, geralmente nas camadas mais altas de metal.

- **Power Ring :** Linhas de VDD e VSS formando um perímetro ao redor do bloco.

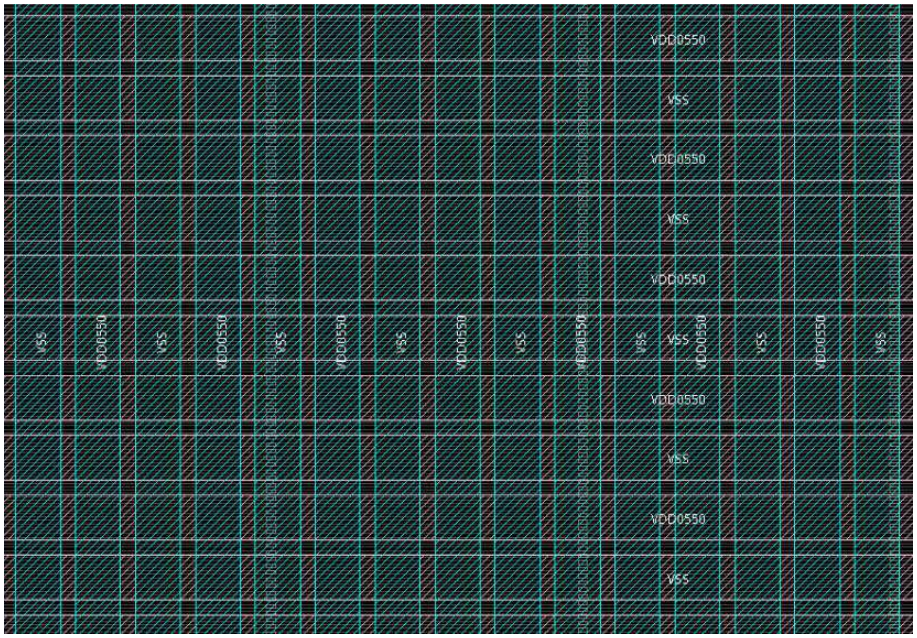


Figura 4 – Power Stripes. Fonte: próprio autor

Ambas as técnicas tem o objetivo de diminuir a queda de tensão ao longo do bloco, visto que as *standard cells* não irão se comportar como previsto na biblioteca se a tensão em seus terminais não estiverem dentro dos limites caracterizados.

O elemento mais importante do roteamento de energia são os followpins. Os followpins definem os espaços ondem poderão ser colocadas as células. Linhas horizontais de metal, geralmente das camadas mais baixas, com espaçamento fixo entre elas e alternando entre VDD e VSS. Isto conecta todos os pinos de VDD e VSS das células a mesma *net*. As linhas alternadas implicam que as células também podem ser colocadas invertidas em torno do eixo x.

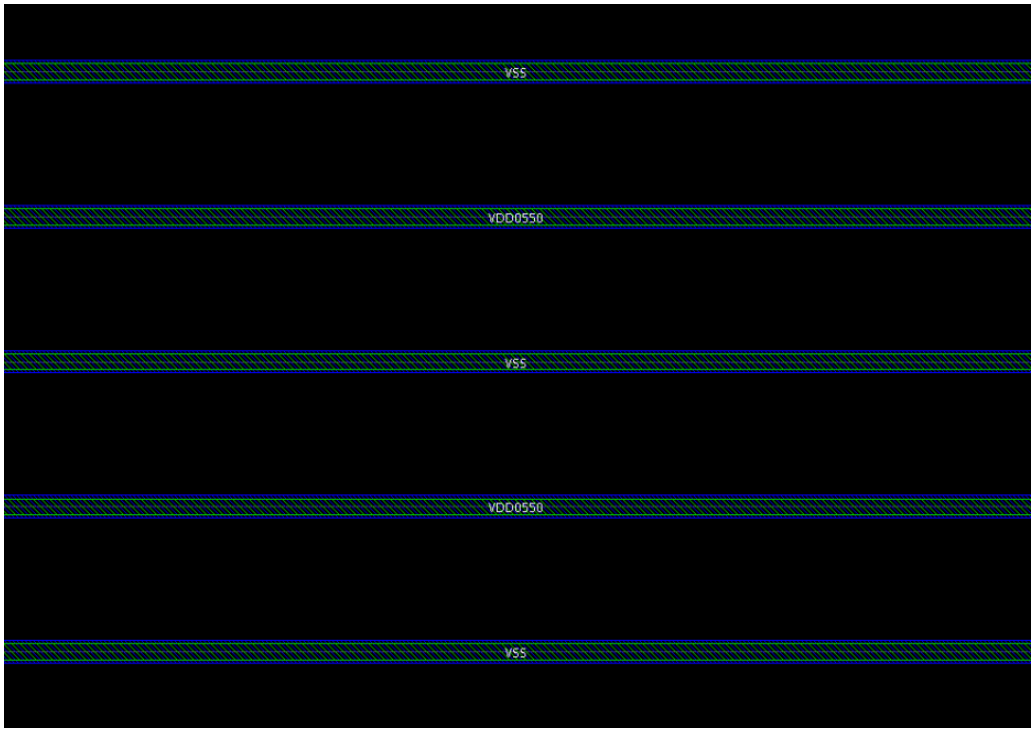
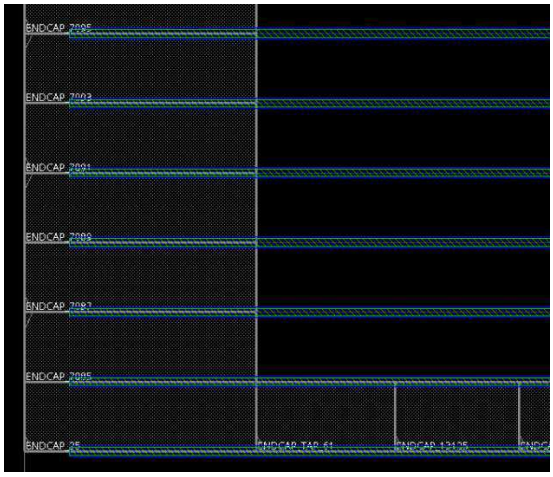


Figura 5 – Followpin. Fonte: próprio autor

Antes de passar para o etapa de *placement* da células lógicas, é preciso primeiro colocar células exclusivamente físicas, chamadas de *Endcap cells* e *Welltap cells*. Essas células não influenciam no comportamento lógico do bloco. Elas garantem que o potencial no poço n e substrato se mantenham constante.



(a) Endcap

(b) Welltap

Figura 6 – Células Físicas. Fonte: próprio autor

2.1.2 Placement

Tendo os *followpins* devidamente posicionados pode-se então começar a etapa de *placement*. Uma primeira rodada de *placement* é feita de maneira grosseira, tentando primeiramente validar a possibilidade de roteamento da região. As células são posicionadas de uma maneira relativamente lógica, em que elas fiquem perto dos pinos de saída ao quais estão relacionadas. Além do congestiomento a ferramenta também leva em consideração violações de *timing*. Nessa etapa do projeto não há árvore de *clock* nem roteamento de dados, então a ferramenta usa aproximações de atraso dos fios e um *clock* ideal.

Após essa primeira rodada, ocorre uma rodada de refinamente do *placement*. Nesse momento ocorrem correções de sobreposição e afinamento das distâncias, afim de melhorar os atrasos calculados na primeira iteração.

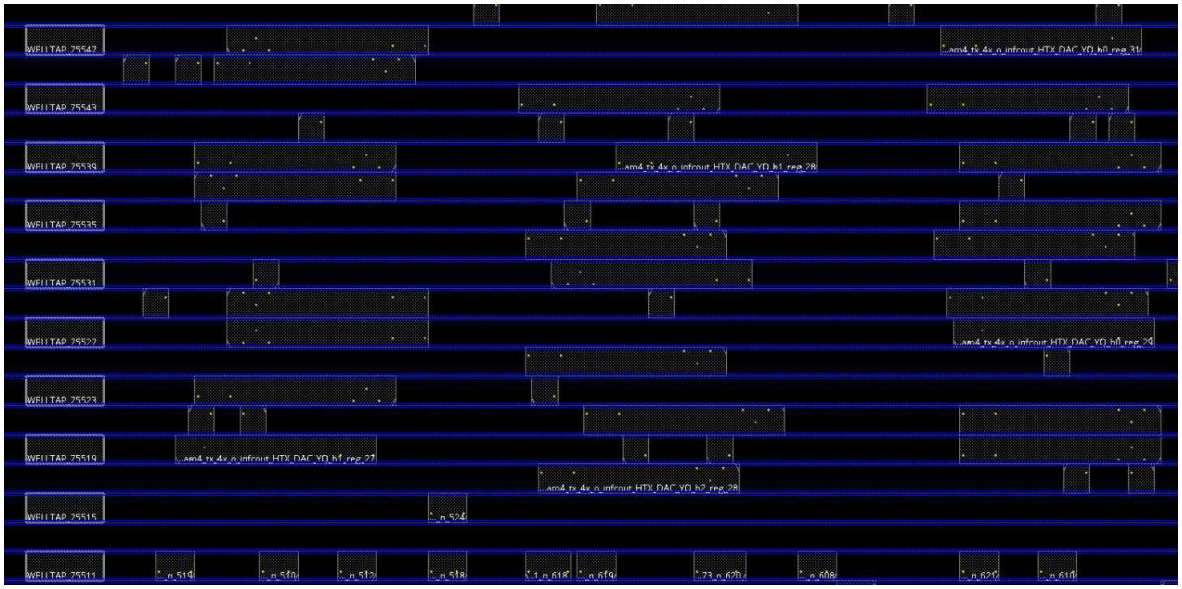


Figura 7 – Standart Cell Placement. Fonte: próprio autor

2.1.3 Clock Tree Synthesis

O *clock* foi até agora calculado e utilizado de uma forma bem simplificada. Células sequenciais do circuito, como diversos *flip-flops*, necessitam que uma sinal de relógio(*clock)* chegue até seus devidos pinos para que elas funcionem. Esse sinal também deve chegar ao mesmo tempo em todos os pinos do circuito.

O roteamento do *clock* traz propriedades físicas reais ao sinal, como atraso nos fios e necessidade de balancemento desses atrasos. Propriedades como latência e *skew* agora são calculadas. A latência se refere ao tempo que o sinal de clock leva para chegar, desde de sua fonte, até o registrador mais longe.

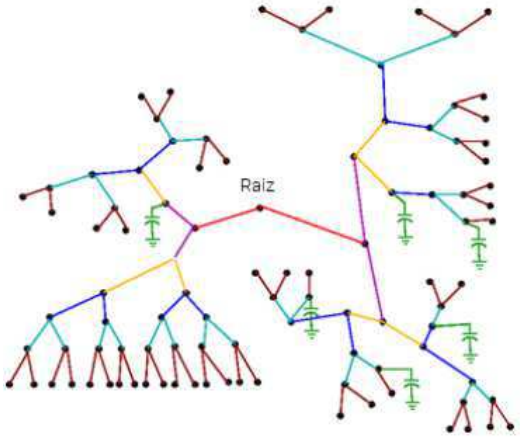


Figura 8 – Balanceamento da CTS. Fonte: Medeiros 2018

O *clock skew* refere-se a diferença de latência entre registradores no mesmo circuito. Isso significa que a borda do *clock* está acontecendo em tempos diferentes e se isso ocorrer no mesmo caminho de dados pode ocasionar uma violação de *timing*. A estratégia inicial, chamada de *Zero Skew*, tem como objetivo zerar o *skew* da *clock tree*, o que nem sempre é possível. Restando *skew* é possível ainda usar técnicas de *Useful Skew* para ajudar no fechamento do *timing*.

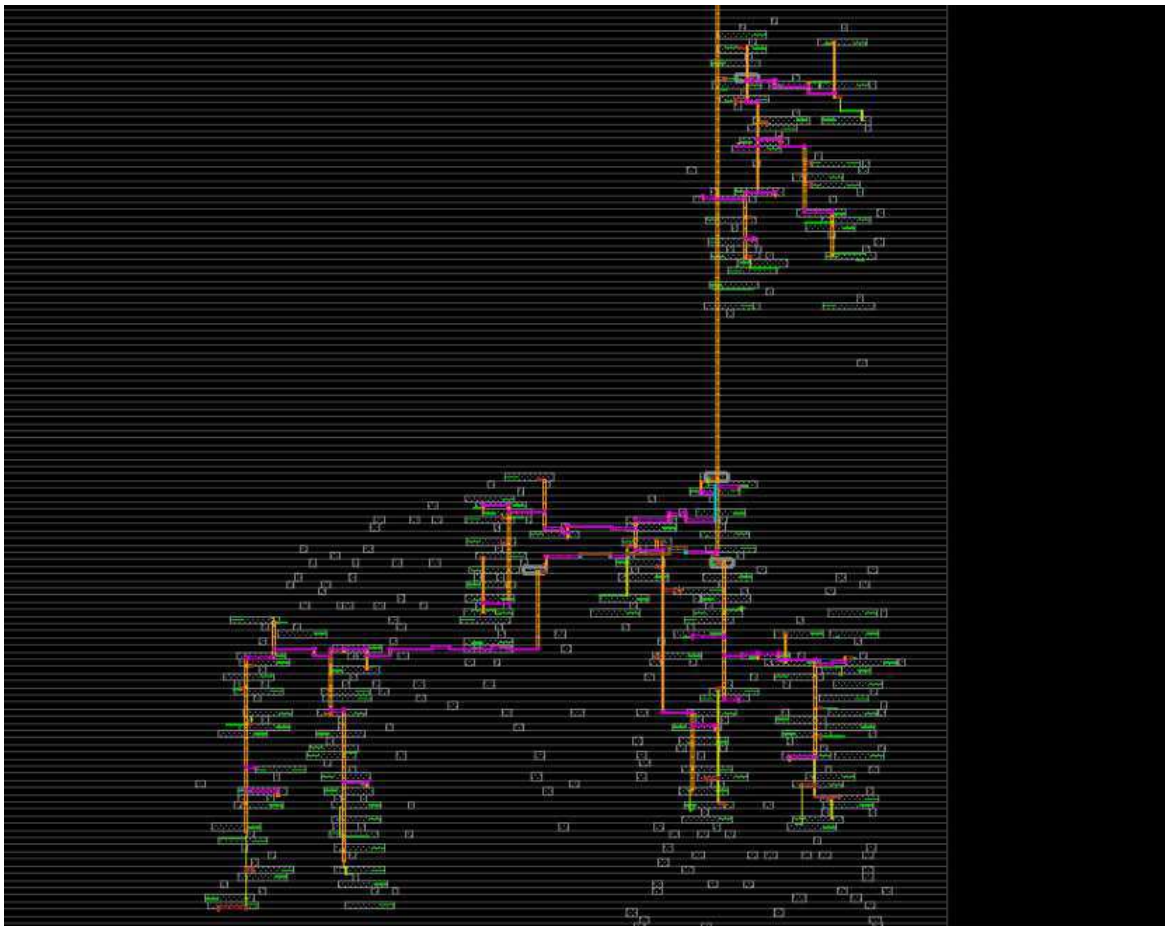


Figura 9 – CTS. Fonte: próprio autor

2.1.4 Routing

O roteamento é a etapa do fluxo que irá inserir os fios que transportarão os dados entre as diversas portas lógicas do circuito. Esses fios irão também inserir novos atrasos que serão contabilizados no cálculo de fechamento de *timing*. A ferramenta de *Place and Route* utiliza de vários tipos de algoritmos para rotear o sinais entre as diversas camadas de metais, procurando o caminho mais curto em busca do *Zero Slack*.

O *slack* referece-se à divergência entre tempo que um dado levou para chegar até a porta de entrada de um registrador e o tempo em que ele precisava chegar. Isso tudo levando em consideração verificações de *Setup* e *Hold*. Um dado para ser capturado corretamente por um registrador precisa estar estável uma quantidade de tempo antes da borda de subida do *clock*. Esse tempo é o atraso máximo que o dado pode ter e é chamado de tempo de *setup*. Além disso, o dado precisa se manter estável um certo tempo após o *clock* para que ele seja adequadamente transferido para a saída do registrador. Este é o atraso mínimo do dado, chamado de tempo de *hold*.

Torna-se natural que, com isso, a ferramenta busque um ponto em que o *slack* seja zero, para mais facilmente suprir os dois requisitos. Há muito tempo que o maior ofensor no quesito atraso dentro de circuitos integrados é decorrente dos fios, ou *nets*, conectando as portas lógicas. Além do comprimento do fio adicionar resistência ao sinal, vários fios de metal correndo em paralelo acabam gerando um efeito chamado capacitâncias parasitas. Todos esses elementos afetam o tempo de transição que um sinal em uma determinada *net* leva para mudar de estado.

Atrasos nas *nets* roteadas são resolvidas por dois métodos principais:

- **Adicionar *buffers*** - Fios muito longo são quebrados para adicionar *buffers* no caminho e com isso reforçar a transição do sinal.
- **Aumentar *drive strenght* das células** - Bibliotecas de células geralmente possuem várias versões de uma mesma porta lógica com diferentes capacidades de dispor corrente de saída. A troca de uma porta de força X1 por uma X2 ou maior pode compensar o tempo de transição na sua saída se ela tiver que alimentar diversas outras portas. Essas outras portas são vistas como capacitores a serem carregados pela saída porta.

2.2 STA(Static Timing Analysis)

A primeira validação para o encaminhamento de um bloco para as etapas seguintes de produção é a validação de temporização, ou *timing*. Isto é, cumprir com as exigências de *timing* definidas pelas *constraints* do bloco. Um arquivo de *constraints* deve conter

informações como o ponto de criação do *clock*, o seu período e forma de onda, valores de incerteza e de transição máxima além de inúmeros outros parâmetros que podem ser definidos. Esses parâmetros devem estar de acordo com as especificações técnicas de funcionamento definidas no projeto para aquele bloco. Os valores definidos serão usados pela ferramenta para calcular as margens de *setup* e *hold* que irão validar o *timing closure*.

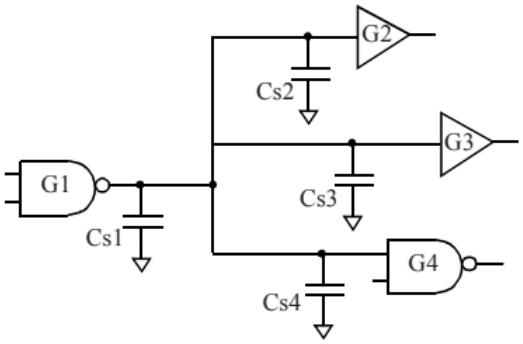


Figura 10 – Capacitâncias Parasitas. Fonte: Bhasker e Chadha 2009

Um processo chamado de extração de capacitâncias é realizado para extrair as informações de capacitância de todas as *nets* do *design*. Essas informações são utilizadas para calcular a carga de saída das células lógicas e retirar das *look_up_tables* na lib os valores de atrasos nesses caminhos.

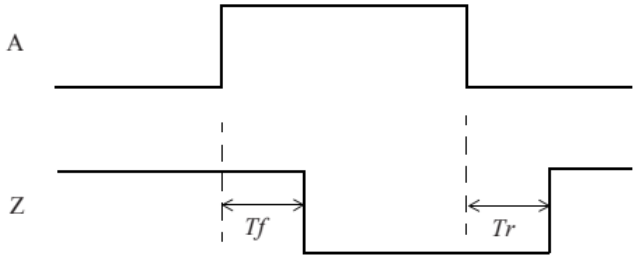


Figura 11 – Latência da Árvore de *Clock*. Fonte: Bhasker e Chadha 2009

Da mesma forma os atrasos são calculados na árvore de *clock* para definir a a latência e o *clock skew*. Juntando todas as informações a ferramenta pode fazer uma análise de temporização ponto a ponto em 3 principais grupos:

- **in2reg:** Temporização entre as portas de entrada do bloco e registradores de interface.
- **reg2reg:** Temporização entre registradores dentro do bloco.
- **reg2out:** Temporização entre registradores de interface e portas de saída.

Os grupos **in2reg** e **reg2out** são altamente dependente de parâmetros de *input delay* e *output delay* definidos nas *constraints*. Valores de máximo e mínimo são definidos para simular o atraso sofrido pelo sinal fora do bloco.

2.2.1 Pontos de Operação (PVT)

Um característica importante que influencia nos cáculos de atraso de um circuito integrado são os pontos de operação do sistema. A tensão e temperatura previstas de operação são definidas no ínicio do projeto e dependem da disponibilidade de bibliotecas de células caracterizadas nessas condições. Além disso é preciso levar em consideração variações em torno do ponto de operação ideal definido.

Se as especificações do *design* estão previstas para operar em tensão nominal de 0.75V e 25°C é previso prever variações nessas condições. Geralmente são usadas variações de 10% para mais e para menos na tensão e valores de temperatura como -20°C e 85°C. Esses fatores afetam os atrasos no circuito para mais ou para menos, afetando então as violações de *Setup* e *Hold*.

O terceiro fator está relacionado ao processo de fabricação. Variações de parâmetros durante a fabricação do *wafer* podem gerar transistores mais rápidos ou mais lentos do que os em condições normais. O transistor pode ser avaliado no quesito processo como T(*Typic*), S(*Slow* e F(*Fast*). Além disso, os transistores P-MOS e N-MOS podem ser afetados de forma diferente, o que define 5 pontos de operação: TT, SS, FF, FS e SF. A primeira letra refere-se ao transitor N-MOS e a segunda ao transistor P-MOS.

2.3 Fluxo Hierárquico de um Projeto Físico de ASIC

É comum que em um projeto grande, com vários módulos com funções bem definidas seja trabalhado de forma hierárquica. Isso quer dizer que cada um desses módulos é desenvolvido como um bloco físico a parte e são então integralizados em um módulo de topo que funciona como um emcapsulamento. Essa metodologia torna os blocos físicos relativamente independentes.

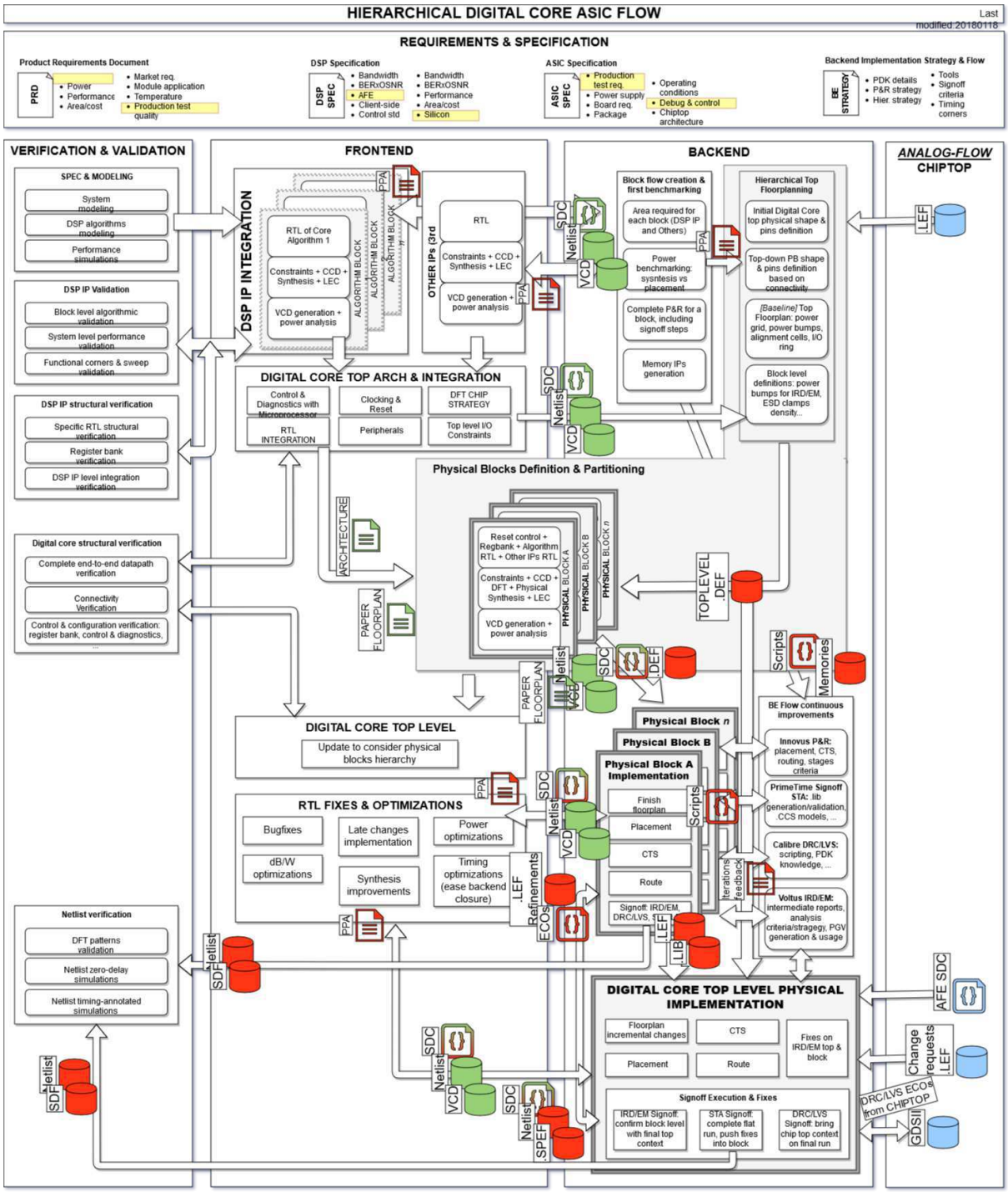


Figura 12 – *ASIC Flow*. Fonte: IDEA! 2019

Um fluxo de trabalho é planejado entre as equipes de *frontend*, *backend* e verificação de forma que a iteração entre as partes no quesito fluxo de informação e dados ocorra de forma organizada e correta, fazendo com que o projeto possa ser validado em suas partes e por completo.

A nível de topo existem informações como o tamanho que o *ASIC* pode ter, sua conexões de interface com o exterior e a disposição dos blocos para que o *datapath* esteja coerente. O projetista de topo então define o formato e as posições de cada bloco no *floorplan*. A distribuição do *powerplan* também pode ser feita nesse nível. Ele então gera partições do *ASIC*, em formato *.DEF*, que são passados para os projetistas de cada bloco em particular. Cada *block owner* então irá trabalhar no seu bloco a partir do *floorplan* definido no *.DEF*, junto com a *netlist* recebida do *frontend*. Quando o fluxo de *design* físico dos blocos é finalizado eles realimentam o topo com as *netlists* e *.DEF* atualizados. O projetista de topo se encarrega então de carregar os blocos nos devidos lugares e se encarrega de fazer as conexões entre eles.

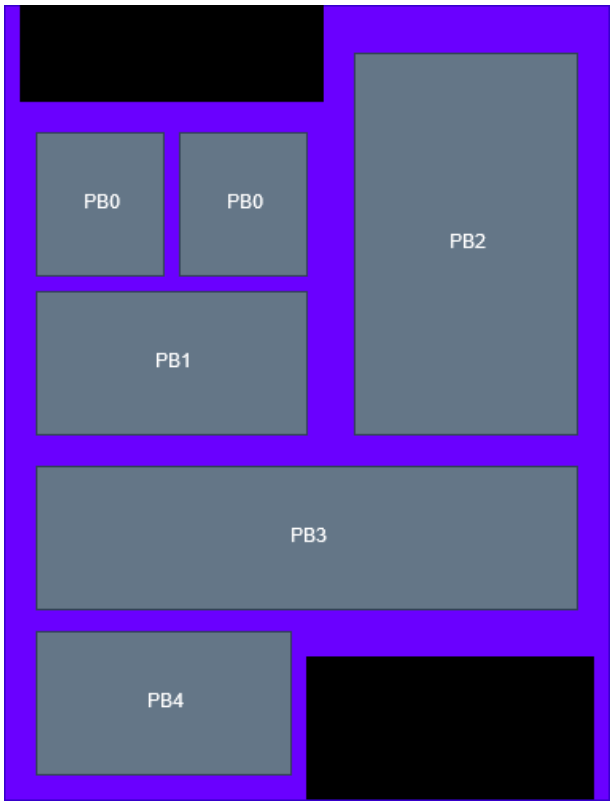


Figura 13 – *Floorplan de topo*. Fonte: próprio autor

2.3.1 STA Flat a Nível de Topo

A análize de temporização a nível de topo pode ser feita de duas maneiras: hierárquica ou *"flat"*. Hierarquicamente, cada projetista gera arquivos de *'.lef'* e *'.lib'* do seu bloco e passa para o topo. O projetista de topo pode então carregar os blocos e a ferramenta irá tratar os blocos como *standard cells*.

A análise *flat* trará todo o projeto ao mesmo nível. Ele irá 'colar' os blocos no *floorplan* de topo e realizar o STA como um bloco único. Para isso os blocos devem fornecer as *netlists* atualizadas e os *.spefs*, ou arquivos de capacitâncias parasitas.

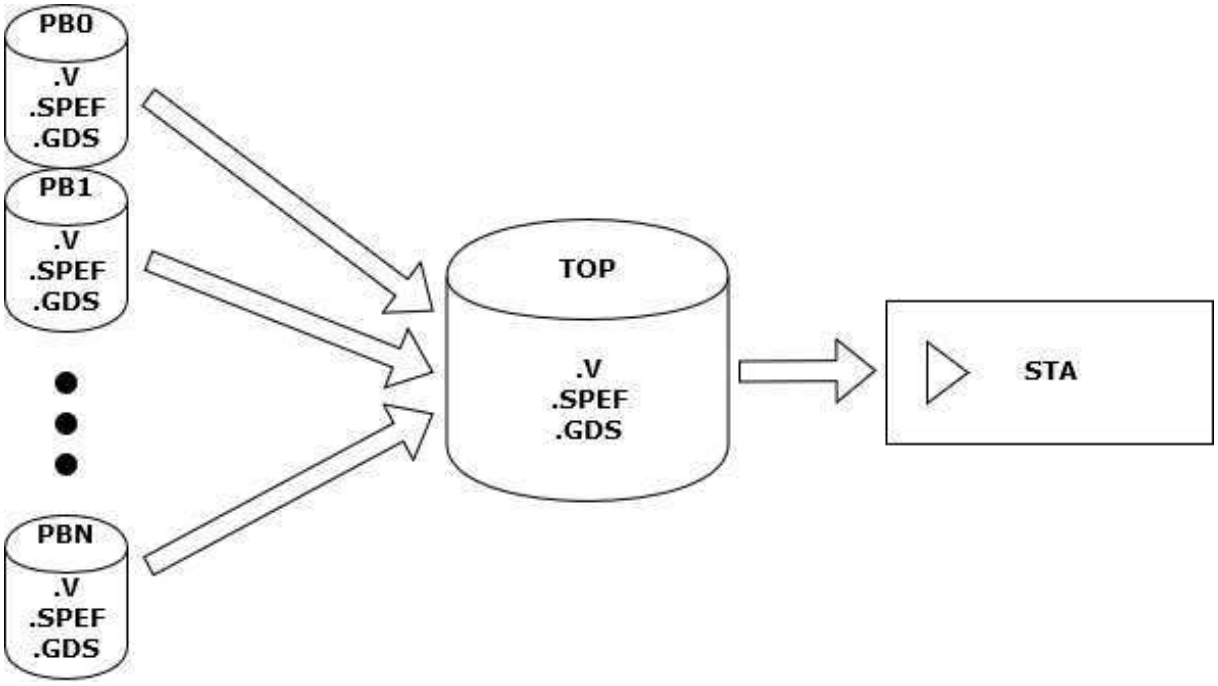


Figura 14 – *STA em Topo*. Fonte: próprio autor

3 Atividades Desenvolvidas

Durante o primeiro mês o aluno teve que estudar e aprender o fluxo de trabalho da empresa, como o padrão no uso de ferramentas de gestão de versões e o fluxo de dados entre as equipes. Foram assistidas vídeo-aulas com cursos básicos de desenvolvimento de físico de blocos digitais e foram lidos documentações que auxiliavam esses cursos. Semanalmente houveram sessões de discussão e tira-dúvidas com o coordenador Davi Castro.

3.1 STA Flat de Topo do DSP

A primeira atividade realizada foi a análise de temporização do módulo de topo do DSP. O progresso do desenvolvimento do *ASIC* era registrado através de entregas datadas. Em cada uma dessas datas o projetista de topo entregava a *netlist*, o *.DEF* e o *.GDS* atual do módulo. Com o *.GDS* era possível então fazer a extração das capacitâncias parasitas das *nets* de topo. Era, então, preciso agrupar todas as *netlists* e *.spefs* dos blocos relacionados aquela entraga do topo e realizar o *STA*.

Ao finalizar o *STA* o trabalho passa a ser a revisão dos relatórios gerados. Primeiramente verificava-se se a anotação dos parasitas estava coerente. É ideal que 100% das *nets* reais estejam anotadas. Isso implica que as *nets* apresentadas nas *.spefs* são as mesmas na *netlists* e não há nenhuma descorrelação.

Net Type	Count	Annotated (%)	Not Annotated (%)
total	19542332	19083804 97.7%	458528 2.3%
assign (*)	73	0 0.0%	73 100.0%
0-term:floating (*)	441231	0 0.0%	441231 100.0%
no drive (*)	21	0 0.0%	21 100.0%
1-term:no load	40864	23852 58.4%	17012 41.6%
real net (complete)	19013203	19013012 100.0%	191 0.0%
real net (broken)	46940	46940 100.0%	0 0.0%

Figura 15 – Anotação de Parasitas. Fonte: próprio autor

A partir dos relatórios das análises era gerado uma planilha com um histograma de violações. Essas violações eram divididas em duas categorias: violações internas aos blocos e violações entre blocos. Cada *block owner* deve então analisar essas violações e verificar que estão coerentes com as violações vistas a nível de bloco. O projetista de topo por sua vez analisa as violações entre blocos que é sua jurisdição.

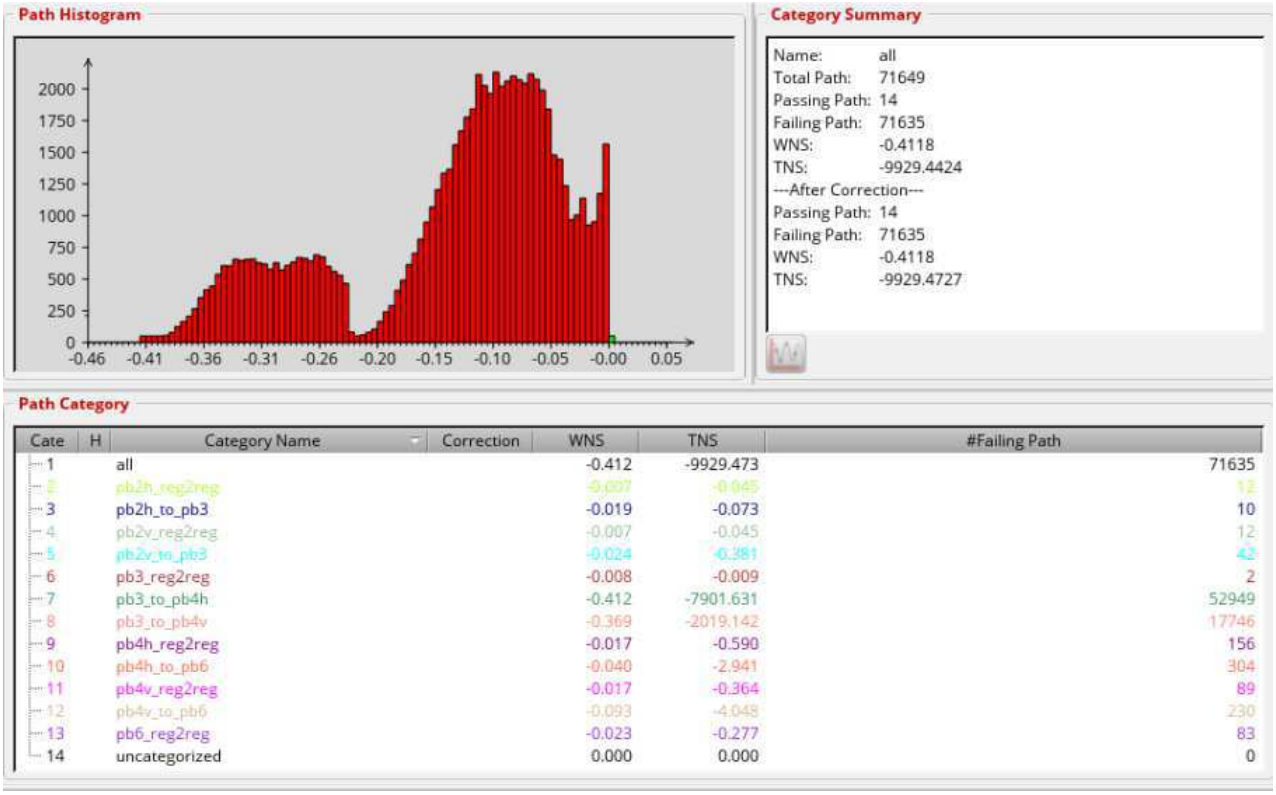


Figura 16 – Histograma do STA. Fonte: próprio autor

3.2 Fluxo de Backend em um Bloco de Interface com o Analógico

Foi então atribuído um bloco para ser trabalhado, após a entrega de uma *netlist* funcional por parte do *frontend*. O objetivo então era seguir o fluxo de projeto físico de um bloco digital neste bloco, começando pela entrega do *.DEF* pelo *designer* de topo. Uma particularidade é que esse era um dos blocos que faria interface com a parte analógica do *ASIC*.

O primeiro passo foi carregar *.def*. Esse arquivo é criado como uma partição do topo e define o tamanho e a forma que o bloco deve ter para que ele se encaixe no *floorplan* do topo.

Figura 17 – Floorplan. Fonte: próprio autor

A distribuição de linhas de VDD e VSS, ou *powerplan* é feita globalmente no topo. Então a partição já possui esses elementos, bastando então seguir com o fluxo. Os pinos do bloco já vêm distribuídos e fixados.

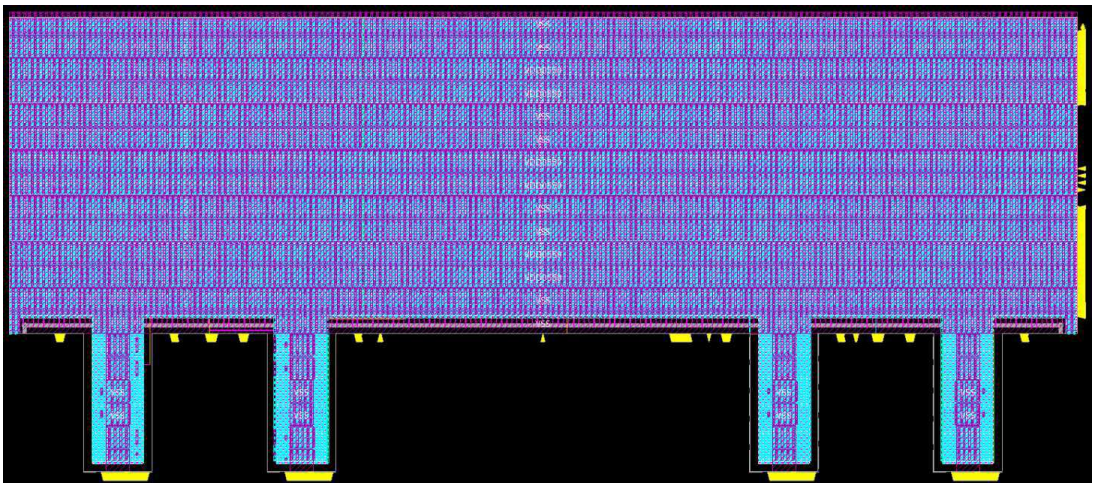


Figura 18 – Powerplan. Fonte: próprio autor

Após algumas iterações do fluxo, constatou-se que a etapa de *placement* feita pela ferramenta estava colocando os registradores da interface de saída mais a esquerda muito longe dos pinos. Supôs-se que isso foi devido a esses registradores terem ligações com caminhos de dados da interface à direita do bloco e isso fez com que a ferramenta quisesse encurtar o caminho. Para resolver esse problema foi criado uma região de *placement* perto da interface e atribuido essa região aos registradores de interface.

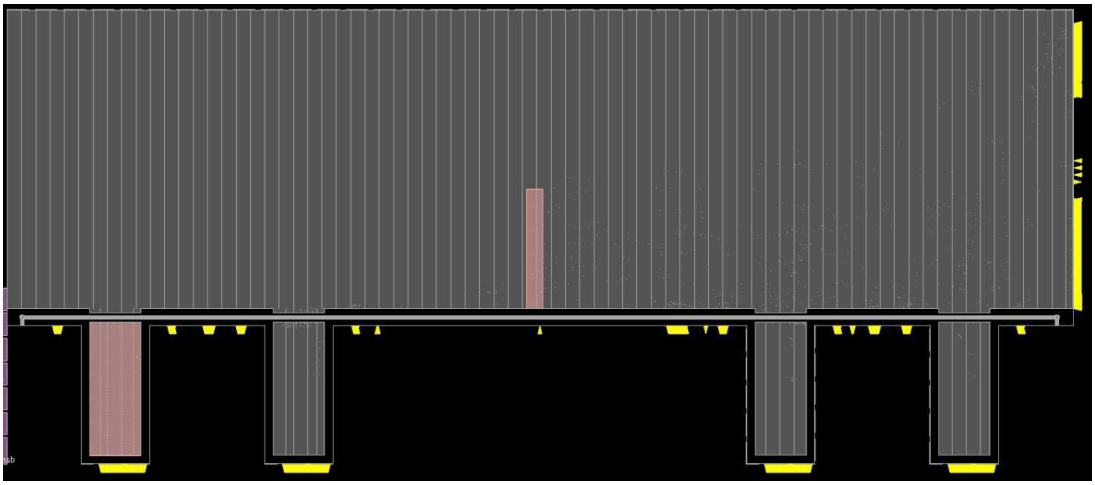


Figura 19 – Placement Region. Fonte: próprio autor

Realizou-se então a etapa de *placement* normalmente e verificou-se a posição mais favorável dos registradores.

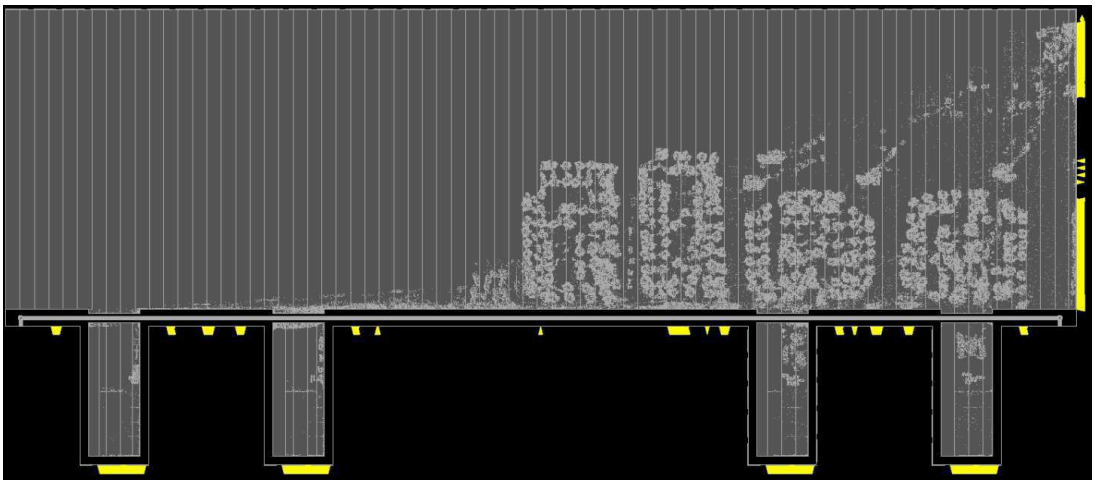


Figura 20 – Placement. Fonte: próprio autor

Logo após a etapa foi realizada a síntese da árvore de *clock*. Nesta etapa alguns ajuste tiveram que ser feitos no balanceamento da árvore. O mesmo *clock* que ia para os registradores de interface com o analógico também ia para uma parte do caminho de dados mais longe. Por isso, mesmo o pino de *clock* ter sido posicionado bem no meio dos pinos de interface, ele tinha uma latência grande referente ao balanceamento feito com os outros registradores. Isso causou violações de *timing* na interface. Para corrigir esse problema foi criado um *skew group* com os registradores de interface. Assim, a ferramenta iria balancear esses registradores em separado dos outros, mesmo eles tendo a mesma fonte de *clock*.

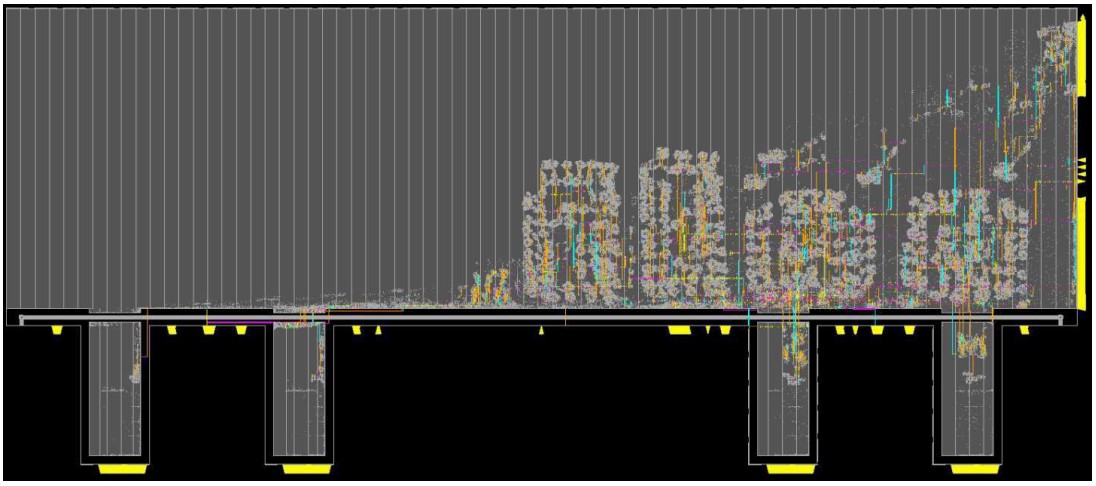


Figura 21 – Clock Tree. Fonte: próprio autor

O roteamento foi realizado sem muitos problemas e com algumas rodadas de otimização foi alcançado o estado de *timing closure*.

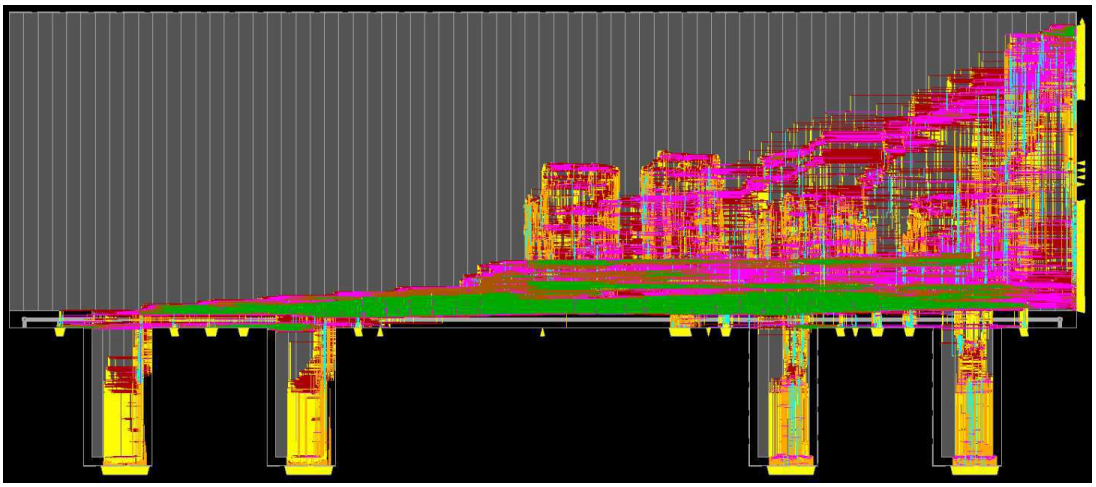


Figura 22 – Roteamento. Fonte: próprio autor

3.2.1 Inserção de Triple Wall

Uma preocupação surgiu durante o desenvolvimento do *ASIC* devido as suscetibilidade da parte analógica a ruídos. Sendo assim foi necessário isolar o substrato da parte

digital com a parte analógica. Isso foi feito através da inserção de uma barreira chamada de *"triple wall"*. Esse isolador consiste de duas paredes de *n-well* ao redor de um fosso de BF_2, um material de grande impedância. Os dois poços-n são então ligados ao VSS.

Foi decidido que essa parede seria inserida a nível de bloco e depois continuada no topo. Um script teve que ser desenvolvido para cortar e inserir a *triple wall* nos blocos de interface com o analógico.

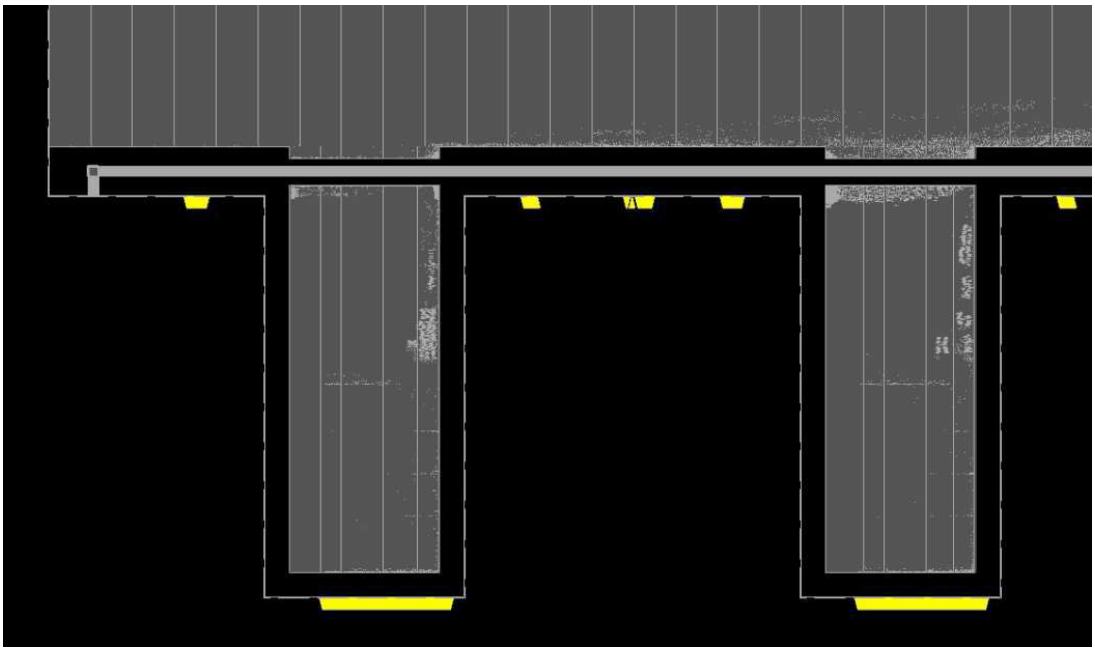


Figura 23 – Histograma do STA. Fonte: próprio autor

4 Considerações Finais

As atividades que foram designadas para o período de estágio foram realizadas com êxito e durante esse tempo, foi possível adquirir bastante conhecimento na área de Microeletrônica, com ênfase na área de Backend, seja a nível de bloco ou a nível de topo.

A realização bem sucedida das atividades foi em grande parte devido ao ambiente propício proporcionado pela empresa. Desde o planejamento e organização das atividades pelos coordenadores até o incentivo a constante troca de idéias e informações entre os colaboradores, todos com o objetivo comum de elevar o nível da empresa. Tudo isso contribuindo para o bem estar e a motivação do colaborador.

Durante o estágio foram abordados assuntos vistos nas disciplinas de Arquitetura de Sistema Digitais e Projeto de Circuito Integrado. A experiência obtida neste estágio foi um fator fundalmental na formação do aluno, visto que houveram conceitos adiquiridos durante o estágio que não haviam sido abordado por completo ou até parcialmente durante a graduação. A oportunidade recebida pelo aluno para o aprendizado nessa área de atuação no mercado de trabalho se mostrou de grande valor.

Referências

BHASKER, J.; CHADHA, R. *Static Timing Analysis for Nanometer Designs: A Pratical Approach*. [S.l.]: Springer, 2009. Citado na página 17.

IDEA! *Idea! Converging Photonics & Microelectronics*. 2019. Disponível em: <www.idea-ip.com>. Acesso em: 18 fev. 2019. Citado na página 9.

MEDEIROS, J. L. P. Estágio integrado na idea! eletronic systems. *Relatório de Estágio (Engenharia Elétrica), UFCG, Campina Grande Brasil*, 2018. Citado 2 vezes nas páginas 11 e 15.