Henry de Lima Carvalho

Relatório de Estágio - NXP Semiconductors

Campina Grande, Brasil

Dezembro de 2018

Henry de Lima Carvalho

Relatório de Estágio - NXP Semiconductors

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 obtenção do título de Graduado em Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Orientador: Gutemberg Gonçalves dos Santos Júnior, D.Sc.

Campina Grande, Brasil

Dezembro de 2018

Henry de Lima Carvalho

Relatório de Estágio - NXP Semiconductors

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 obtenção do título de Graduado em Engenharia Elétrica.

Trabalho aprovado em: 20/12/2018

Gutemberg Gonçalves dos Santos Júnior, D.Sc.
Orientador

Marcos Ricardo Alcântara Morais, D.Sc.
Convidado

Campina Grande, Brasil
Dezembro de 2018

Agradecimentos

Agradeço a minha família por ter me dado todo o apoio em um momento de mudança tão drástica, me dando força e confiança para seguir meu rumo profissional. Em especial, agradeço meus pais por terem me fornecido a infraestrutura e recursos necessários para poder traçar meu caminho.

Agradeceço imensamente à todos os excelentes profissionais que compõem o BSTC, com foco para a equive *SoC Verif-B*, em especial ao meu gerente Reginaldo Gabarrão pelo voto de confiança e pela oportunidade de ter vivenciado tudo isso, e ao meu tutor Humberto Carvalho por se mostrar extremamente prestativo na minha iniciação dentro da empresa, sempre sanando qualquer dúvida necessária.

Agradeço também à minha Gabrielle pelo apoio emocional diário por ela prestado sempre com muito carinho e amor, e como sempre, me fazendo acreditar no meu mérito e capacidade de superar sempre novos desafios.

Agradeço aos meus amigos que como eu, também se aventuraram nessa nova jornada em Campinas, pelo constante companheirismo e por toda diversão vivenciada, sendo fundamentais para amenizar a saudade de casa.

Agradeço aos professores que compõem o laboratório XMEN, em especial ao meu professor orientador Gutemberg, coordenador do projeto, pela visão e sensibilidade de decidir investir e focar em uma área de grande expasão no momento, possibilitando que tivéssemos uma vasta experiência na área, o que sem dúvidas contou bastante no sucesso do estágio.

Ao pessoal da coordenação de Engenharia Elétrica, em especial o grande Tchaikovsky pela grande ajuda e boa vontade nos processos que precederam o início do estágio.

Resumo

Este relatório consiste em uma descrição das atividades de estágio realizadas no Brazil Semiconductor Technology Center - BSTC, sede da NXP Semicondutores em Campinas - SP. Durante o período de um ano, tarefas no segmento da microeletrônica (projeto de circuito integrado) foram desempenhadas com foco na área de verificação de SoC, incluindo planejamento e criação de testes direcionados, estratégias de verificação em RTL e *gate-level*, e técnicas de *debug* de *designs* digitais. Desta forma, relatando a experiência pessoal e profissional nas atribuições dessas atividades.

Palavras-chaves: SoC; Verificação.

Abstract

This report consist of a brief description of the internship activities carried in the Brazil Semiconductor Technology Center - BSTC, NXP site in Campinas-SP. During a period of one year, tasks regarding microelectronics segment have been performed focusing in the SoC verification field, including planning and creation of oriented testcases, RTL and gate level verification strategies and digital design debug techniques. Therefore, reporting the personal and professional experience in the competency of those activities.

Key-words: SoC; Verification.

Lista de ilustrações

Lista de abreviaturas e siglas

EDA	Electronic Design Automation
HDL	Hardware Desription Language
RTL	Controlador Lógico Programável
SDF	Standart Delay Format
SoC	System on Chip
SVA	SystemVerilog Assertions

Sumário

1 Introdução

Este trabalho dercreve as atividades realizadas no âmbito profissional durante o Estágio Integrado, como parte indispensável para a formação acadêmica em Engenharia Elétrica.

O estágio foi realizado durante o período de 19/01/2018 a 17/12/2018 na NXP Semicondutores, precisamente no BSTC (Brazil Semiconductor Technologicy Center), cidade de Campinas, São Paulo, totalizando uma carga horária de 1522 horas e atendendo aos requisitos previstos na Resolução 01/2012 do Colegiado do Curso de Graduação de Engenharia Elétrica e em concordância com a Lei do Estágio (Lei 11.788/2008).

1.1 Objetivos

O ambiente acadêmico proporcionou, ao longo do período de curso, a possibilidade de aprendizado de novas habilidades e conhecimentos que capacitassem o aluno para o ambiente de trabalho, entretanto o estágio é fundamental para que possa ser adquirida a experiência necessária na a formação de um profissional completo.

Com a oportunidade de por em prática os conhecimentos técnicos obtidos com a graduação e contribuir de forma ativa para a construção de algo realmente significativo, o aprimoramento das competências técnicas é conquistado naturalmente, reiterando a importância da inserção do concluinte em um ambiente profissional para crescimento profissional e pessoal.

Como objetivos principais do programa de estágio, destacam-se:

- Aplicar na realidade industrial as habilidades desenvolvidas durante a graduação, tanto em aulas, como em atividades extra-curriculares.
- Aprimorar e expandir os conhecimentos na área.
- Desenvolver habilidades interpessoais no ambiente profissional em equipe.
- Especializar-se na área de ênfase do estágio.

1.2 A empresa

A NXP semicondutores é uma empresa de semicondutores multinacional com sede em Eindhoven, Holanda e emprega hoje cerca de 31.000 pessoas em mais de 35 países,

sendo líder no fornecimento de semicondutores para indústrias de identificação e segurança, automotivo e redes digitais.

Fundada em 1953 como Phillips Semiconductors, a empresa foi vendida em um consórcio de investidores em 2006, quando mudou seu nome para NXP. Em 2015 foi anunciada a fusão com a também desenvolvedora e fabricante de chips Freescale Semiconductors em um negócio de $40 bilhões de dólares.

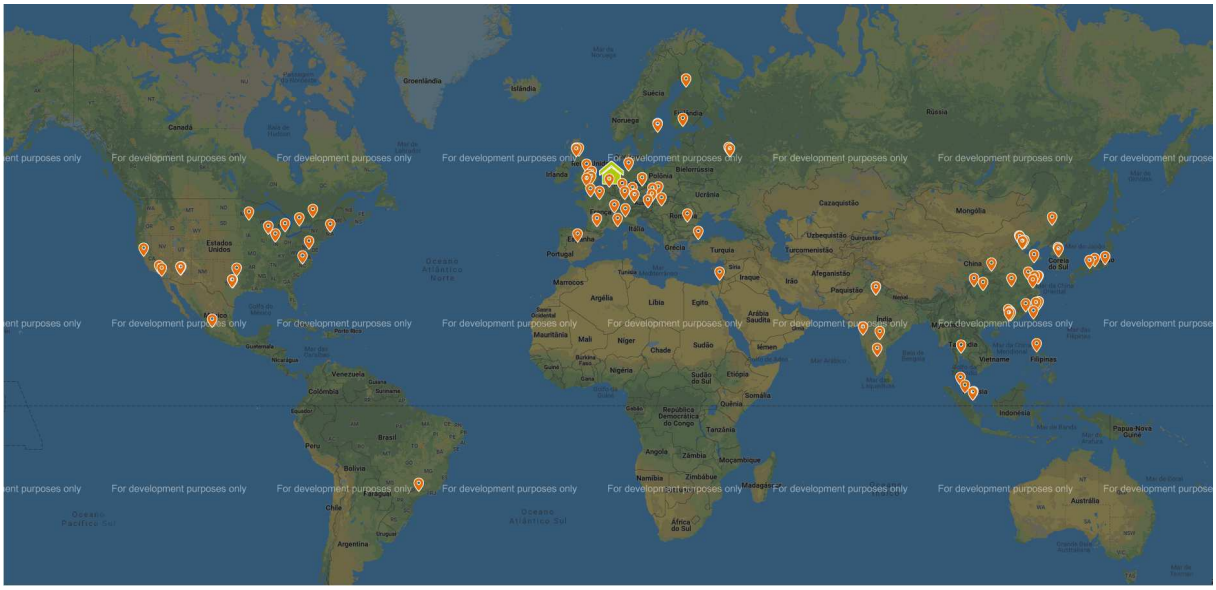

Figura 1 – Sedes da NXP espalhadas pelo mundo

1.2.1 BSTC - NXP no Brasil

A NXP Brasil iniciou suas operações em 1967 quando ainda atuava sobre a direção da Motorola Inc. que apostou na capacidade intelectual e de inovação dos engenheiros locais. A região de Campinas foi escolhida por sua localização estratégica e pela disponibilidade de mão de obra altamente especializada além da proximidade das melhores universidades do país.

O BSTC (Brazil Semiconductor Technology Center) iniciou suas operações em dezembro de 1997 com 8 experientes engenheiros e ao final do ano 2000 esse número já ultrapassava 60. Atualmente o BSTC conta com mais de 130 funcionários (95% engenheiros) contabilizando mais de 100 projetos entregues, incluindo dezenas de microcontroladores e IP chave em áreas como gerenciamento de potência, redes automotivas, processamento digital de sinais e aceleradores criptográficos e de temporização.

A área comercial da empresa também tem presença significativa na América do Sul, tendo o Brasil como base de operações. Apresentando uma estrutura enxuta e altamente competente, a equipe comercial brasileira conta com gerentes de vendas e engenheiros de aplicação que juntos, oferecem aos seus clientes soluções completas de suporte e acompa-

nhamento que se inicia no conceito do projeto, passa pelo desenvolvimento propriamente dito e culmina com o fornecimento dos dispositivos semicondutores via canais franqueados de distribuição ou venda direta.

A NXP desenvolve e fabrica dispositivos para os segmentos Automobilístico, Industrial, de Consumo e de Transações/Acesso seguros. As principais indústrias que desenvolvem e fabricam produtos nestes segmentos no Brasil são clientes da NXP.

Figura 2 – Site da NXP no Brasil em Campinas - SP

2 Embasamento teórico

2.1 SoC

Uma das aplicações possibilitadas desde o desenvolvimento da tecnologia de transistores foi a de chips capazes de computar dados e os distribuir através de suas conexões periféricas. Esses produtos, em geral, comportam de milhares a milhões de transistores dentro de seus circuitos.

Os produtos chamados SoC (*System on a Chip*) são componentes eletrônicos complexos que, envolvem processadores, memórias e interconexões criadas para um certo campo de aplicação. A gama de aplicações que eles podem suportar é virtualmente infinita, dependendo apenas do seu processo de fabricação e para o quê o chip foi projetado. Aplicações mais utilizadas costumam ser digitais, analógicas, mistas analógico-digitais e em rádio frequência. Eles também são vastamente utilizados no mercado de embarcados, sendo, usualmente, o componente principal da aplicação, agindo como controlador ou o processador de outros componentes.

O desenvolvimento de um SoC passa por várias etapas antes de chegar até o consumidor final, desde a análise financeira do projeto, diversas etapas de criação do chip até análises de qualidade. Para que seja possível analisar esse processo, é necessário primeiro estudar os vários conceitos importantes que ditam seu processo de criação.

2.2 Fluxo de projeto

O desenvolvimento de um sistema eletrônico complexo como um SoC a nível industrial requer uma cadeia de etapas especializadas e divididas de forma que otimizem a produção e agilizem o tempo de projeto. Profissionais capacitados em cada área promovem um fluxo de trabalho que segue um padrão pré-definido até alcançar um produto final aceitável.

A Figura 3 ilustra de forma básica a sequência de etapas para a concepção de um novo SoC. Dependendo do projeto, todo esse o fluxo pode levar vários meses e as vezes mais de ano.

2.2.1 Requisitos do Cliente

Os clientes de uma empresa de semicondutores geralmentes são outras empresas que planejam usar o chip em seus sistemas ou produtos. Desta forma, as exigências do cliente representam um papel importante no processo de decisão de como o chip deve ser

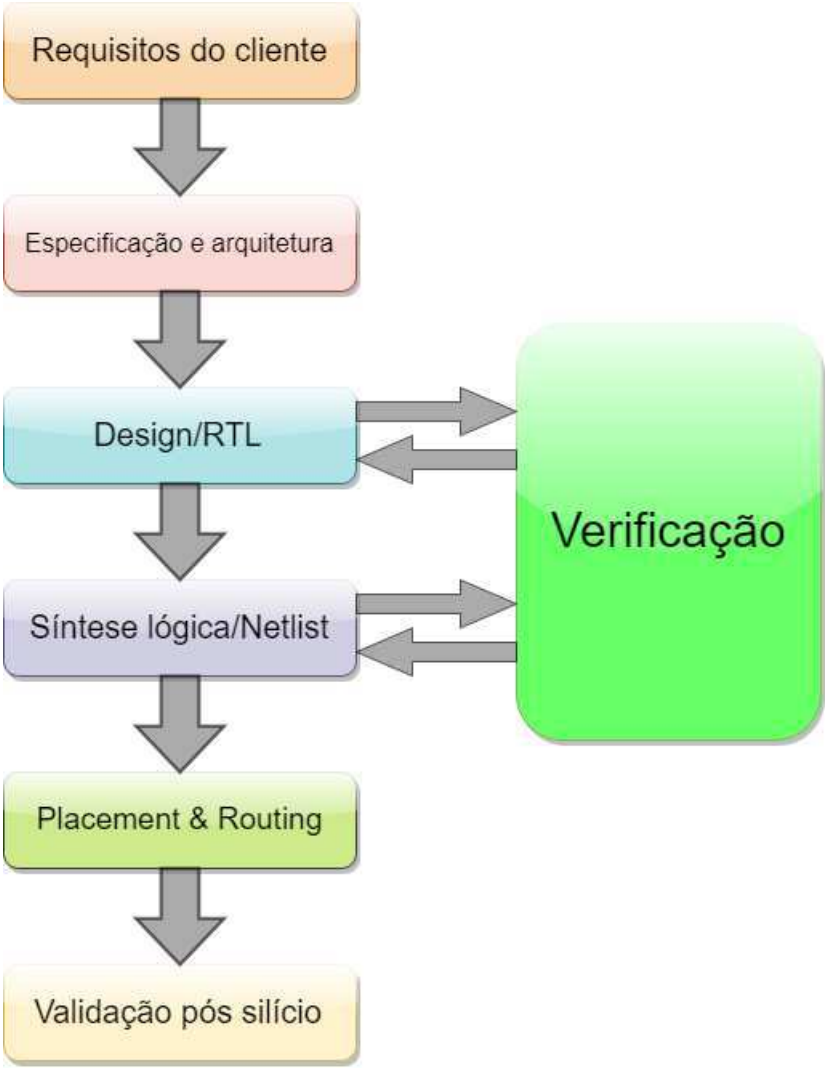

Figura 3 – Diagrama representando as etapas no projeto de um SoC

projetado. Naturalmente, o primeiro passo deve ser coletar os requisitos, estimar o valor de mercado do produto, e então avaliar os recursos necessários para a execução do projeto.

2.2.2 Especificação e Arquitetura

O próximo passo seria coletar as especificações que descrevem de forma abstrata a funcionalidade, interface e arquitetura geral do chip a ser projetado. O documento deve conter informações do tipo:

1. Requer dois processadores ARM A53 interconectados que devem funcionar a uma frequência de 600 MHz;
2. Requer interfaces USB 3.0, Bluetooth e PCIe 2ª geração;
3. Controlador de display com resolução de 1920x1080 *pixels*.

Em seguida, os arquitetos elaboram um plano a nível de sistema de como o chip deve operar como mostrado na Figura 4. Eles decidirão quais outros componentes serão necessários, as frequências de *clock* presentes, e como os requisitos de consumo serão atingidos. Também é definido como os dados devem fluir entre os blocos no chip.

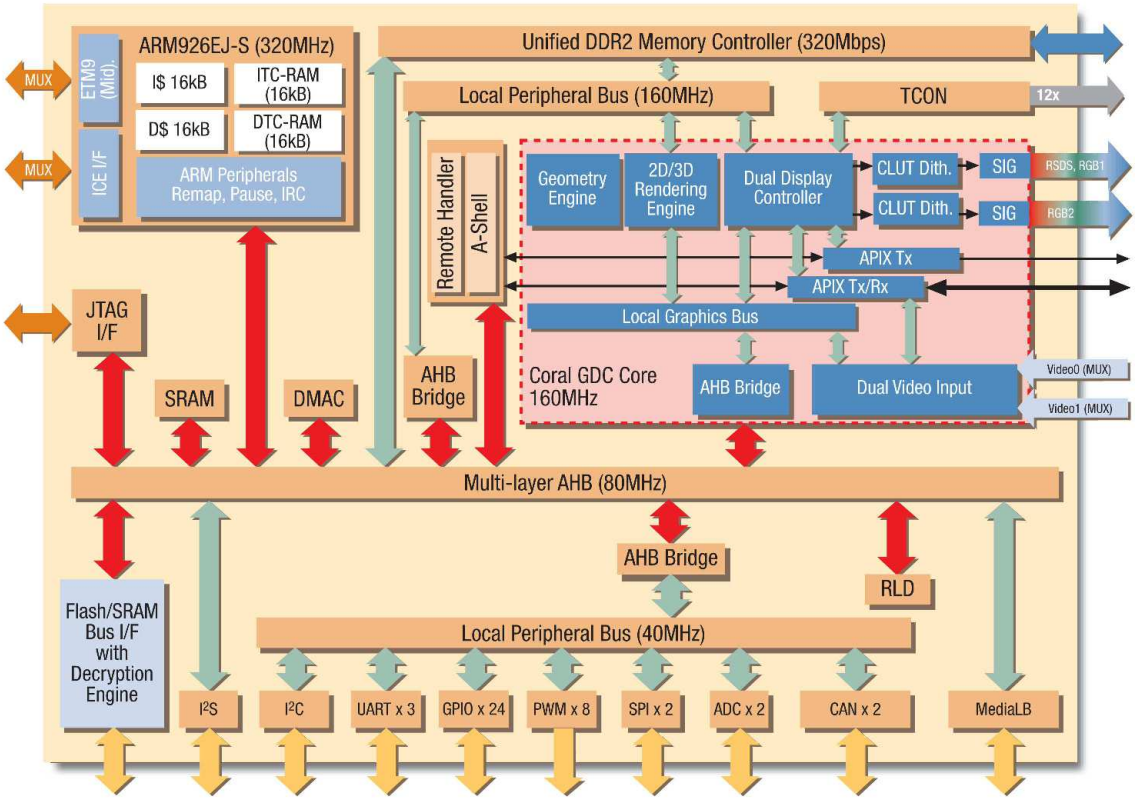

Figura 4 – Exemplo de uma arquitetura de SoC.
Fonte: Fujitsu Semiconductor America Inc.

2.2.3 Design

Devido à complexidade dos chips modernos, não é possível construir tudo do zero, e em muitos casos, muitos componentes serão reusados. Digamos que uma empresa **X** requer um módulo *FlexCAN* que interaja com outros módulos em um automóvel. Eles podem comprar o módulo desejado de outra empresa, economizando tempo e esforço, ou utilizar-se de recursos para produzir o seu próprio.

Não é nada prático projetar sistemas digitais a partir de blocos básicos tais como *flip-flops*, portas lógicas e transistores CMOS. Diante disso, uma descrição comportamental é desenvolvida para analisar o *design* em termos de funcionalidade, performance e outros aspectos de alto nível, usando uma linguagem de descrição de hardware como Verilog ou VHDL. Isso é feito pelo time de *design* e o processo é similar a programação computacional de alto nível, adicionada de competências em eletrônica digital.

2.2.4 Verificação

Tão logo que o RTL esteja pronto, ele precisa ser verificado quanto a sua exatidão, isto é, garantir que o *design* executa suas funcionalidades corretamente. Isto é feito com a ajuda de simuladores EDA (*Electronic Design Automation*) que tem a capacidade de modelar o *design* e aplicar estímulos. Este é o trabalho do engenheiro de verificação.

Para economizar tempo e alcançar o máximo de funcionalidade, ambos os times de *design* e verificação atuam em paralelo, onde os times de *design* liberam uma versão de RTL e o time de verificação desenvolve ambientes e casos de teste (*testbenches* e *testcases*) para testar a funcionalidade da versão do RTL. Se algum teste falhar, pode ser indicativo de algum problema com o *design* e um *"bug"* será sinalizado no elemento de *design*. Este *bug* deve ser consertado na próxima versão do RTL pelo time de *design*. Esse processo se repete até se alcançar um nível de confiablidade na precisão funcional do *design*.

2.2.5 Síntese lógica

Com um *design* satisfatório em mãos, é hora de convertê-lo em um esquemático de hardware com elementos reais como portas lógicas combinacionais e *flip-flops*. Esse passo é chamado de síntese. Ferramentas de síntese lógica permitem a conversão da descrição HDL do RTL em uma *netlist* de portas lógicas. Essa *netlist* nada mais é do que uma descrição do circuito em termos de portas e conexões entre elas. A Figura 5 mostra isso de uma forma intuitiva.

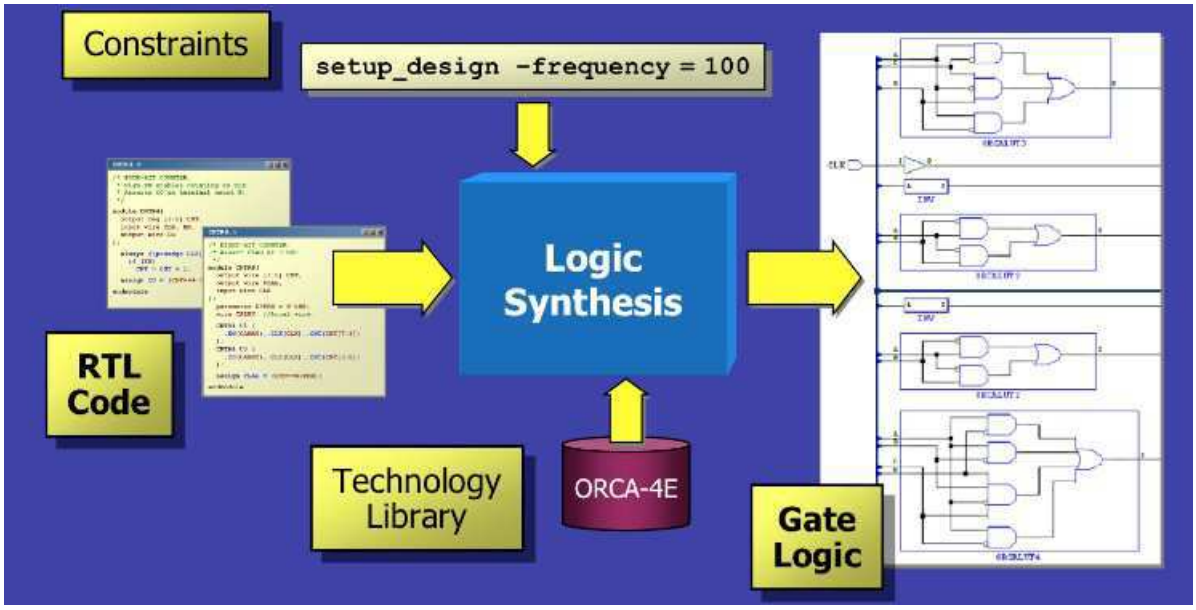

Figura 5 – Esquema ilustrativo do processo de síntese lógica.

As ferramentas de síntese lógica garantem que a *netlist* obedeça as especificações de *timing*, área e consumo. Normalmente elas tem acesso a diferentes bibliotecas de tecnologias de elementos digitais e podem fazer cálculos inteligentes para satisfazer diversos

critérios. Estas bibliotecas são obtidas com as fábricas de semicondutores que fornecem dados característicos para diferentes componentes como tempos de subida e descida em *flip-flops*, tempo de entrada e saída para portas combinacionais, *etc.*

É conferida a equivalencia entre a *netlist* em *gate-level* e o RTL, e algumas vezes uma verificação *gate-level* é realizada, onde a verificação de certos elementos e sua funcionalidade é feita mais uma vez. A diferença é que desta vez a verificação é a nível de portas lógicas em menor nível de abstração. Simulações tendem a ser mais demoradas devido ao grande numero de elementos envolvidos no *design* nesse ponto, além dos atrasos nas células.

2.2.6 Placement e routing

A *netlist* é inserida no fluxo de *design* físico, onde o *placement* e *routing* é feito automaticamente através de ferramentas de EDA. Esse processo irá selecionar e alocar células padrão em linhas, definir o mapeamento dos pinos de entrada e saída, criar diferentes camadas de metal, e alocar *buffers* para fechar o *timming*. Uma vez que esse processo é concluído, um *layout* é gerado e enviado para fabricação. Este estágio geralmente é conduzido pela equipe de *design* físico que é familiarizada com o nó de tecnologia e detalhes da implementação física. A Figura 6 mostra a visualização do resultado obtido desse processo.

2.2.7 Validação

Uma amostra do chip é fabricada seja pela própria empresa de semicondutores, ou enviada para uma *foundry* terceira como TMSC ou Global Foundries. Essa amostra passa por um processo de validação pós-silício onde outro time de engenheiros executa diversos padrões em uma testadora. É muito mais difícil o *debug* na validação pós-silício do que na verificação pré-silício devido ao grau de visibilidade dos nós internos de um chip ser drasticamente reduzido. Se houver algum problema real encontrado nessa fase, terá de ser consertado em RTL, re-verificado e todos os passos subsequentes serão repetidos.

2.3 Setup e Hold

Entradas síncronas em componentes lógicos possuem uma especificação de tempo chamadas *setup* e *hold* no que diz respeito ao sincronismo com a entrada de *clock*. Essas condições especificam que o dado de entrada deve permanecer estável por um determinado intervalo de tempo antes e depois da entrada de clock variar.

Tempo de setup O intervalo de tempo que o dado da entrada síncrona (D) deve permanecer estável antes do pulso de *clock*.

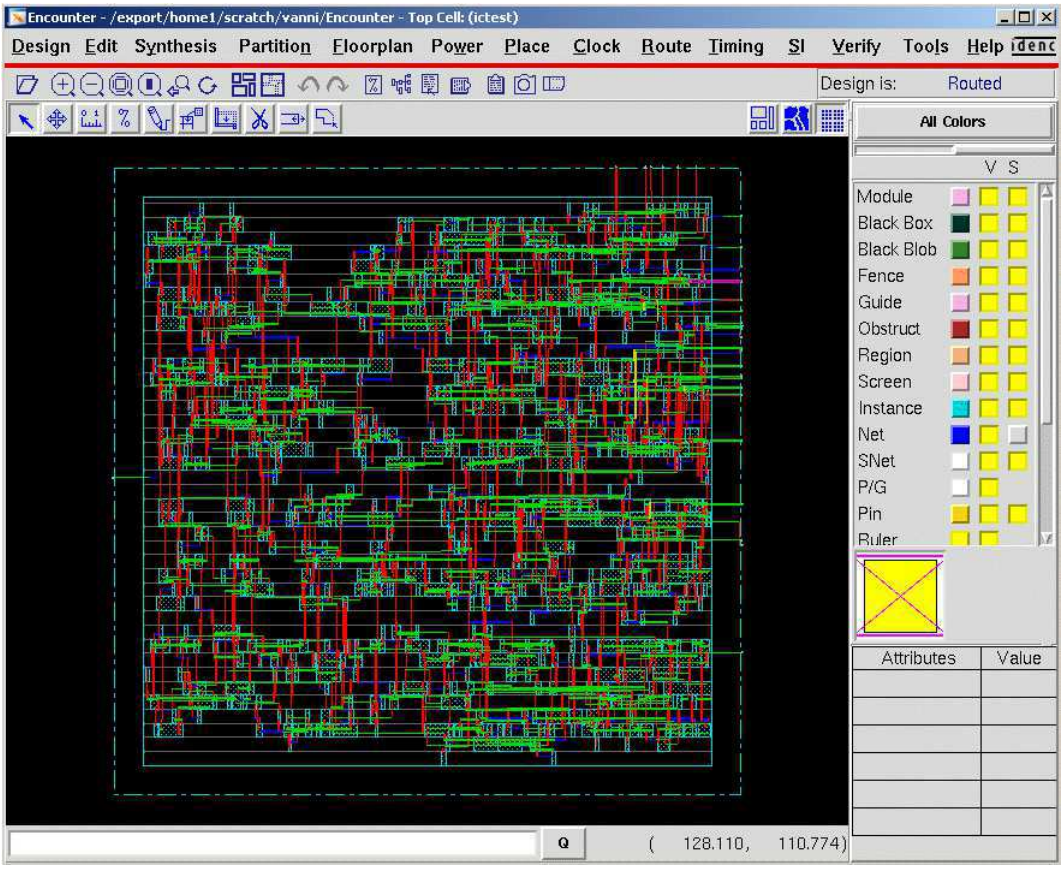

Figura 6 – Resultado do processo de placement e routing.
Fonte: Georgia Institute of Technology

Tempo de hold O intervalo de tempo que o dado da entrada síncrona (D) deve permanecer estável depois do pulso de *clock*.

A Figura 7 ilustra as condições de *setup* e *hold* em um registrador simples. Alterações do dado (D) que ocorram dentro do intervalo de *setup* e *hold* marcados como t_{su} e t_h respectivamente, geram uma violação de *timing*, propagando assim um valor indeterminado na simulação.

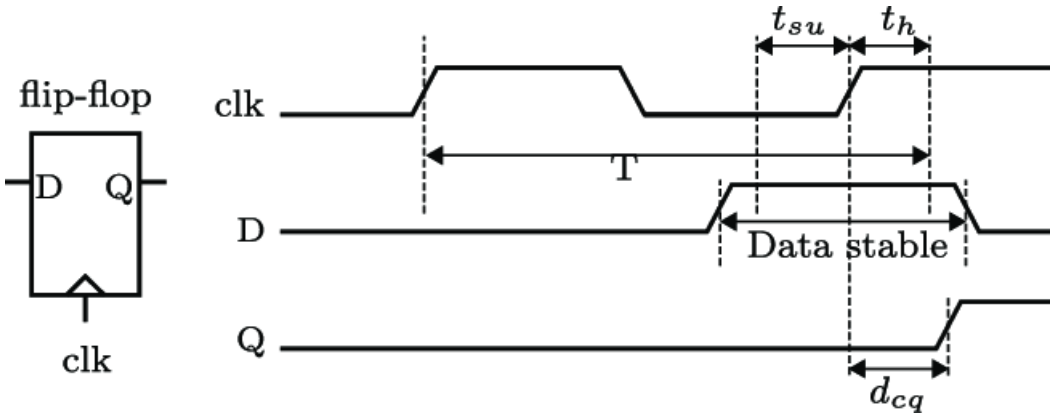

Figura 7 – Ilustração dos intervalos de setup e hold.

3 Atividades desenvolvidas

A atuação junto ao time de SoC verification ao longo do ano de 2018 envolveu uma atuação ativa no trabalho de verificação de projetos, precedido de um extenso treinamento e familiarização do ambiente de projeto. O estagiário teve a oportunidade de participar de forma integral nas atividades designadas a um engenheiro de verificação, estanto totalmente envolvido no fluxo do projeto vigente, assumindo responsabilidades, prazos e tarefas a cumprir.

Dentre suas aribuições, o estagiário realizou atividades nas seguintes competências:

- Desenvolver e corrigir testes de verificação;
- Detectar e sinalizar falhas de integração no *design* de projeto;
- Apresentar soluções para possíveis erros de *design*;
- Analisar relatórios de cobertura.

Reuniões semanais discutiam o andamento geral do projeto do ponto de vista da equipe, onde abordava-se o status atual no projeto, bem como os próximos passos a serem tomados. Pequenas apresentações eram recorrentes e eram incentivadas como forma de compartilhar conhecimentos entre membros não só internamente na equipe, mas na empresa como um todo.

3.1 Treinamentos

Na fase inicial de estágio, a empresa se dispôs a disponibilizar vários treinamentos para os novos estagiários, que variaram desde conteúdos mais básicos, até assuntos mais avançados no que diz respeito ao fluxo de projeto, em especial a verificação funcional. Os treinamentos aconteceram tanto de forma presencial, tutoriado por funcionários mais experientes da empresa, como também com o fornecimento de material para *self-training*.

As sessões de treinamento foram extremamente úteis para relembrar velhos conceitos aprendidos na graduação, mas muitas vezes esquecidos, bem como uma forma de gerar uma integração extremamente saudável entre os estagiários, bem como entre os estagiários e os engenheiros mais experientes.

Alguns módulos de treinamentos merecem destaque e servem como bons exemplos para ilustrar a metodologia de treinamento

3.1.1 Verilog

Apesar de ser considerado um pré-requisito para o ingresso na vaga de estágio, os estagiários passaram por um detalhado treinamento de Verilog presencial com um experiente designer, onde foram abordados desde conceitos mias elementares, até técnicas de descrição de hardware voltadas para a otimização de síntese.

Durante este treinamento pequenos exercícios eram passados com o intuito de praticar as habilidades desenvolvidas, com a construção de pequenos módulos em Verilog que desempenhassem determinadas funcionalidades pré-definidas de forma otimizada e eficaz.

O treinamento durou cerca de uma semana de intensivas aulas teóricas e práticas e foi bastante útil tanto para refrescar o assunto, como para desenvolver habilidades na linguagem não conhecidas até então.

3.1.2 SystemVerilog Assertions - SVA

O treinamento de SVA (*SystemVerilog Assertions*) foi um exepmlo de *self-training* experienciado pelo estagiário. Nesse treinamento, módulos de laboratório de SVA foram disponibilizados, bem como material teórico para que o estagiário desenvolvesse habilidades no que diz respeito à criação de *assertions* no contexto de verificação funcional para monitoramento comportalmental de sinais.

Nos módulos de treinamento o estagiário desenvolvia *assertions* para satisfazer os desafios propostos pelo módulo de treinamento. Dessa forma foi possível compreender a mecânica e sintaxe do recurso, sem contar com a contribuição pessoal em aprender algo que até então não se tinha conhecimento.

3.1.3 Certitude

Certitude é uma ferramena desenvolvida pela Synopsys que permite que seja medida a qualidade do ambiente de verificação identificando em que partes o ambiente precisa ser melhorado. Certitude fornece informações detalhadas sobre a capacidade de propagação e detecção dos ambientes de verificação. Basicamente, o Certitude funciona injetando erros de forma proposital no *design* e indica se a verificação feita no bloco é capaz de detectar ou não essa falha.

Durante esse treinamento, o estagiário teve o primeiro contato com um projeto real da empresa. O treinamento consistiu em estudar e implementar a ferramenta em determinados blocos do SoC, e inserindo falhas em suas interfaces, observar se a verificação atual conseguia detectá-las.

Foi observado que em alguns blocos, nem todas as falhas simuladas pela ferramenta

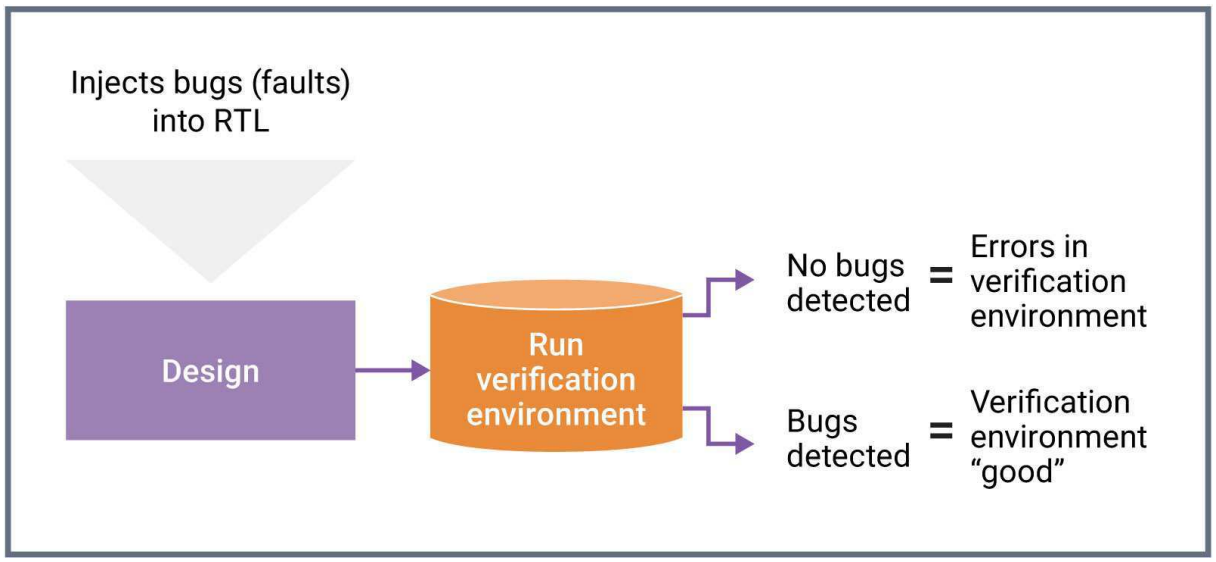

Figura 8 – Esquema de funcionamento básico da ferramenta Certitude
Fonte: Synopsys

puderam ser detectadas na verificação, abrindo espaço então para a primeira contribuição real no projeto que seria aprimorar o ambiente de verificação até que os furos fossem consertados. Desta forma, o treinamento foi extremamente eficaz tanto para a familiarização com o ambiente de projeto real, como para a primeira atuação prática como profissional de verificação funcional.

3.2 SoC Verification

O estagiário atuou junto à equipe *SoC Verification* na verificação funcional sob o ponto de vista de interface e integração entre os blocos em nível de SoC. A verificação funcional do SoC se difere da verificação de IP no escopo de observação, pois não visa verificar as funcionalidades e sinais internos de cada bloco, e sim sua correta interação com os demais IPs quando desempenhando tarefas que estimulem múltiplos componentes do chip. Geralmente cada verificador ficava responsável por alguns blocos, e tinha por objetivo cobrir todas as interações em suas interfaces, bem como as possibilidades desse bloco interagir com demais blocos ao seu alcance.

Cada membro do time de *SoC Verification* era designado para uma certa quantidade de blocos que ficavam sob seu cuidado, por assim dizer, onde ele devia verificar a interconectividade desse bloco dentro do sistema como um todo, observando suas interações com os demais blocos. O estagiário foi designado para verificar vários blocos de *timers* diferentes, sendo necessário estudar suas documentações e especificações de forma a compreender totalmente o funcionamento do bloco, sendo possível assim verificá-lo com sensatez.

O procedimento de verificação a nível de SoC se dava com a criação de rotinas

chamadas de *testcases* objetivando estimular os blocos e interfaces desejadas. Esses *testcases* eram escritos em C, e se assemelham à programação de um microcontrolador, onde em seu código o verificador procura implementar as funcionalidades associadas ao bloco. Dessa forma era possível observar se seu bloco se comunicava e realizava corretamente as interações com o restante do chip em que ele tem acesso funcional.

3.2.1 Testcases

A implementação dos testcases em C deve ser cuidadosamente planejada para que haja uma quantidade suficiente de cenários gerados e não haja furos na cobertura. Uma das principais atividades do estagiário de verificação de SoC ocorre sobre esses *testcases*, seja criando novos testes ou realizando manutenção nos já existentes de forma que atenda a funcionalidade esperada. Quando um teste em RTL detecta um erro de *design* ele gera um *fail*, Caso contrário gera um *pass*.

Para a implementação de um *testcase* é de suma importânia que o verificador tenha pleno conhecimento das funiconalidades e arquitetura do bloco sobre o qual ele estará gerando seus estímulos. Uma análise detalhada na documentação do bloco permite que ele tenha o conhecimento necessário para saber que valores deve atribuir a quais registradores para desepenhar determinada tarefa de seu interesse.

Também no *testcase* se faz necessária muitas vezes realizar a conexão entre os blocos no SoC para direcionar os sinais de um bloco para outro, seja este sinal uma interrupção, um *trigger*, ou qualquer outro. Desta forma o verificador precisa saber que sinais cada bloco pode exportar ou receber e assim, realizar de forma coerente as conexões desejadas.

Macros desenvolvidas pela equipe são de extrema importância para facilitar a programação de um *testcase*, e também deixar o código mais intuitivo e compreensivo para futuras manutenções, além claro, de garantir uma confiabilidade na operação.

3.2.2 Verificação RTL

A verificação em nível de RTL corresponde ao processo de verificação em um maior nível de abstração lógica, ou seja, com foco nas simulações executadas sobre os módulos descritos em HDL. Os *testcases* desenvolvidos são codificados para rodarem primeiramente a nível de RTL, onde a simulação executa rigorosamente o que foi descrito na implementação.

Essa etapa da verificação visa encontrar erros lógicos implementados pelo time de *desgin* na integração do SoC. O estagiário atuou nessa etapa, inclusive detectando erros consideráveis e sinalizando-os para que pudessem ser corrigidos. A seção 3.3 aborda o procedimento de identificação de erros e procedimentos a serem tomados.

3.2.3 Verificação Gate-level

Chegado o ponto em que a grande maioria de *testcases* em RTL apresentam status *pass*, o design pode ser sintetizado com segurança, gerando assim sua *netlist*. A partir da *netlist* gerada, pode-se iniciar a verificação *gate-level*, que passa a considerar agora as portas lógicas geradas pela síntese. Alguns *testcases* devem receber algumas modificações para funcionarem corretamente nessa etapa.

Inicialmente as simulações em gate-level são realizadas em *zero delay*, isso quer dizer que as células lógicas estão sendo simuladas sem seu atraso inerente à implementação física de qualquer componente lógico. Uma vez que novamente tenhamos uma quantidade razoável de testes passando em *zero delay*, são introduzidos os arquivos SDF (*Standart Delay Format*), que permitem simular os atrasos de cada célula da netlist.

Com a introdução do SDF, o verificador precisa identificar violações de *timing* geradas uma vez que atrasos estão agora sendo simulados e condições de violação de *setup* e *hold* agora podem ser observados na simulação, que por muitas vezes acaba propagando valores indeterminados. Para consertar esta situação, a equipe de verificação identifica os pontos de violação no design e sugerem a adição de sincronizadores para que seja possível fechar o *timming* do circuito. O estagiário atuou também nessas etapas, contribuindo inclusive na avaliação da necessidade de inserção ou não de sincronizadores nos blocos em que era responsável.

3.3 Debug

Quando um teste falha, é necessário saber o que impediu o correto funiconamento da simulação, se foi algum problema no padrão ou se foi algum erro de *design*. Nesse momento entra a capacidade do verificador de realizar um debug detalhado para identificar a fonte desse erro. Essa é uma das principais atribuições do engenheiro de verificação e que foi posta em prática exaustivamente pelo estagiário.

Seja na simulação RTL ou gate-level, o Debug é inevitável para detecção de erros. Muitos recursos podem ser utilizados para auxiliar no debug como *logfiles*, formas de onda, esquemáticos, entre outros. A análise detalhada desses meios revelam pistas do que pode ser a causa do teste falhar. Ao observar o comportamento de uma forma de onda, por exemplo, o verificador ao conhecer o bloco que está a verificar, deve ter a capacidade de reconhecer um comportamento anormal e a partir disso investigar a fonte da anormalidade.

No trabalho de *debug*, além de identificar as causas do erro, é uma boa prática que o verificador, se possível, proponha uma solução para o problema. Uma vez identificado o causador de tal anormalidade, ele de antemão implementa uma possível solução e já

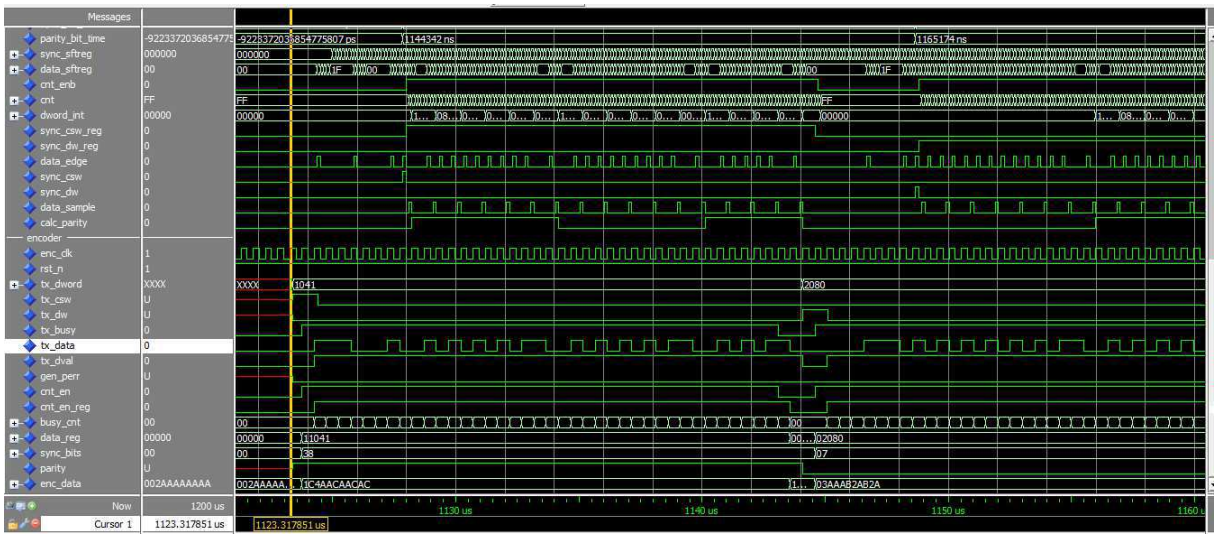

Figura 9 – Exemplo de forma de onda usada para debug

testa sua funcionalidade. Essa é uma prática que pode agilizar o andamento do projeto, pois diminui a chance do time de *design* implementar uma correção ainda não funcional. O estagiário pôde se deparar com a chance de sinalizar erros e apontar possíveis soluções que posteriormente foram de fato implementadas pelo time de *design*.

4 Considerações Finais

Durante o preríodo de um ano, o estagiário obteve êxito na execução de suas atividades, cuprindo assim suas metas e auxiliando de forma direta a equipe no fechamento da verificação do SoC em desenvolvimento no site da NXP no Brasil. O estagiário teve a oportunidade de atuar ativamente no trabalho desenvolvido de forma totalmente integrada à equipe.

A NXP proporcionou um excelente ambiente de trabalho oferencendo sempre os recursos necessários e tratando o estagiário de forma digna sem nenhuma distinção para com os demais funcionários. Com a oportunidade de estar inserido em um ambiente referência em microeletrônica como a NXP e cercados mais mais brilhantes mentes da área no Brasil e até no mundo, o aprendizado ocorre de forma excepcional.

O programa de estágio se mostrou extremamente necessário para a formação completa do aluno, que pôde passar por um processo contínuo de aprendizado e aquisição de experiência. Apesar de pensamentos opostos e decisões arbitrárias sobre o estágio ser realizado no período de um ano, esse tempo é absolutamente necessário quando se trata de uma área de atividade bastante complexa, muitas vezes não compreendidas por pessoas que, inseridas dentro de uma bolha acadêmica, podem desconhecer a realidade profissional do mercado de trabalho, alimentando-se apenas de burocracia e prejudicando assim alunos que querem contribuir de forma real para a sociedade no campo da ciência e tecnologia.

Referências

MEYER, A. *Principles of Functional Verification*. [S.l.]: Elsevier Science, 2003.

PIZIALI, A. *Functional Verification Coverage Measurement and Analysis*. [S.l.]: Springer US, 2007.

SALEMI, R. *The Uvm Primer: A Step-By-Step Introduction to the Universal Verification Methodology*. [S.l.]: Boston Light Press, 2013.

SPEAR, C.; TUMBUSH, G. *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. [S.l.]: Springer US, 2012.

VASUDEVAN, S. *Effective Functional Verification: Principles and Processes*. [S.l.]: Springer US, 2006.

WIEMANN, A. *Standardized Functional Verification*. [S.l.]: Springer US, 2007.

WILCOX, P. *Professional Verification: A Guide to Advanced Functional Verification*. [S.l.]: Springer US, 2007. (Ifip Series).

WILE, B.; GOSS, J.; ROESNER, W. *Comprehensive Functional Verification: The Complete Industry Cycle*. [S.l.]: Elsevier Science, 2005. (Systems on Silicon).