UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

PadsTool: uma Ferramenta Gráfica para Mapeamento e Posicionamento dos Pads

JOÃO JANDUY BRASILEIRO PRIMO

João Pessoa
2013

JOÃO JANDUY BRASILEIRO PRIMO

PadsTool: uma Ferramenta Gráfica para Mapeamento e Posicionamento dos Pads

Dissertação apresentada ao Programa de Pós-Graduação em informática do Departamento de Informática da Universidade Federal da Paraíba como parte dos requisitos para obtenção do título de Mestre em Informática

Área de concentração: Processamento de Sinais e Sistemas Gráficos

Orientador: Prof. Dr. José Antônio Gomes de Lima

João Pessoa
2013

Dedico esse trabalho

A Deus, a minha esposa
aos meus pais e familiares
e aos meus amigos...
companheiros de todas as horas...

Agradecimentos

A UFPB, que me deu a oportunidade de realizar meu sonho.

A todos os companheiros de curso, que me deram força e me incentivaram nos momentos difíceis.

A todos meus professores, em especial Antônio Carlos, Hamilton Soares e Leonardo Batista, pelos ensinamentos que levarei por toda vida.

Aos amigos do laboratório LASID, onde passei excelentes momentos de minha vida.

Ao meu orientador José Antônio Gomes de Lima por acreditar em mim e investir no meu trabalho.

Por fim, agradeço a todos os meus familiares e amigos que fazem de mim uma pessoa melhor.

Resumo

As ferramentas EDA (Electronic Design Automation) são utilizadas para facilitar o projeto e desenho de circuitos integrados (CI). O Floorplaning é uma importante etapa na fase de design do layout no desenvolvimento de um CI. Nesta etapa, os macroblocos são posicionados no chip, além de serem decididas: a localização dos pads de entrada e saída, a localização dos pads de alimentação e as estratégias de distribuição da alimentação e do sinal de clock pelo núcleo. Comumente, é feito um wrapper em HDL que mapeia as portas de entrada e saída do projeto em instâncias de Pads, com seus diferentes tipos, definidos pelo desenvolvedor e um arquivo que indica a posição de cada Pad no circuito. Dessa maneira, tanto esse mapeamento quanto tal posicionamento, em geral, são feitos manualmente por meio de scripts, gerando uma dificuldade para os desenvolvedores, pois para um CI com uma quantidade razoável de entradas e saídas esses procedimentos são susceptíveis a falhas. Esses arquivos, em geral, são utilizados em todas as ferramentas EDA e também pelos fornecedores de Design Kits, além disso, as ferramentas possuem sintaxes diferentes para os arquivos. Este trabalho propõe a construção de uma ferramenta com interface gráfica capaz de fornecer aos desenvolvedores uma maneira mais fácil e intuitiva de gerenciar tanto o mapeamento quanto o posicionamento dos pads, tornando o processo mais rápido e menos susceptível a falhas humanas. Para validar o trabalho, a ferramenta é testada em projetos de CI’s.

Palavras chave: Floorplanning, Pad, EDA, mapeamento, posicionamento, script.

Abstract

EDA Tools (Electronic Design Automation) are used to facilitate the project and layout of Integrated Circuits (IC). Floorplanning is an important step in the layout design phase of the development of an IC. In this step the macroblocks are positioned on the chip, and the following properties are determined: the location of input and output pads, the location of the power pads and the strategies of distribution of the power and clock signal by the core. Commonly a wrapper in HDL that maps the input and output ports of the project in instances of pads is done, with the different types, defined by the developer and a file that indicates the position of each pad on the circuit. Thus, both the mapping and positioning are usually manually done through scripts, generating a great difficulty for developers, because an IC with a reasonable amount of inputs and outputs becomes extremely susceptible to human failure. These files are generally used by all EDA tools as well as by the Design kits suppliers, moreover, the tools have different syntaxes for the files. This work shows a tool with a GUI (Graphical User Interface) able to provide to the developers an easy and intuitive way to manage both the mapping and positioning of the pads, making the process faster and less susceptive to human failure. To validate the work, the tool is tested on some IC projects.

Keywords: Floorplanning, Pad, EDA, mapping, positioning, script.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo I: Introdução

1.1. Motivação

A evolução da eletrônica permitiu que sistemas que utilizavam válvulas, relés e conectores e que, além disso, ocupavam até uma sala inteira, fossem concebidos em pequenos circuitos que ocupam uma pequena área menor que a palma de uma mão e acumulam várias funções.

Esta evolução só foi possível após a integração de milhões de componentes, denominados transistores, em apenas um circuito. A estes se dá o nome de Circuitos Integrados (CI's) e são eles os responsáveis por grande parte dos avanços tecnológicos da eletrônica digital. Atualmente os CIs são um elemento onipresente em praticamente todos os componentes de sistemas comerciais, científico e militar. Eles têm permitido grandes avanços em áreas como computadores, eletrônica de entretenimento, câmeras de vídeo, sistemas de posicionamento global, e em inumeráveis outras aplicações (Chatterjee, et al., 2002).

O ritmo desses avanços da tecnologia de fabricação tem se mantido exponencial nas últimas décadas, como atesta a Lei de Moore. Esta lei é devida a Gordon E. Moore, que em 1965 observou que a densidade de componentes em circuitos integrados dobrava a intervalos regulares, inferindo que este comportamento perduraria por muito tempo ainda. O intervalo medido por Moore para que a densidade média de CIs dobrasse foi de 18 meses, o que ainda hoje permanece uma taxa estável.

Atualmente, para construção de um circuito digital, é possível utilizar uma linguagem de descrição de hardware (HDL). Com ela o desenvolvedor pode partir de uma especificação de entradas, saídas e funções a serem executadas e construir um módulo que realize o que foi especificado.

Esta forma de descrever circuitos digitais permite uma maior versatilidade em seu projeto além de permitir o reuso, uma propriedade desejável em um projeto seja ele de software ou de hardware. Tal propriedade permite que um módulo, uma vez implementado e bem documentado, seja reutilizado em outro projeto, otimizando a relação custo x benefício x prazo.

Ferramentas de CAD (Computer Aided Design) tem sido utilizadas desde há muito para o projeto automatizado de sistemas eletrônicos, sendo aplicadas em

diferentes etapas de sua concepção. O uso destas ferramentas permite que os projetistas trabalhem com informações mais significativas, evitando a necessidade de conhecimento e intervenção nas etapas de projeto mais próximas da tecnologia de fabricação. Estas etapas tratam com um número elevado de elementos, e são mais susceptíveis a erros. Quando é possível modelar e solucionar os problemas computacionalmente, realiza-se a tarefa de projeto automaticamente em tempo reduzido.

O crescimento na complexidade destes problemas provocou o surgimento de uma considerável indústria de software para automação de projetos eletrônicos, ou EDA (Electronic Design Automation). A utilização de um sistema EDA é essencial para a concepção de novos circuitos integrados. De acordo com Avenir (Avenier, 1979), a principal melhoria esperada por todo sistema EDA é a redução do tempo de design, o que naturalmente conduz à redução dos custos, por meio de um aumento da produtividade e uma melhor captura do mercado. Segundo Vanzi (Vanzi, 1990), fabricantes de IC consideram a tecnologia EDA tão importante quanto o processo tecnológico, devido ao impacto da ferramenta EDA e da automação do projeto tanto sobre a produtividade da comunidade de projetistas quanto na capacidade de exploração daquilo que a tecnologia de silício que pode oferecer.

O Projeto Brazil-IP (Brazil-Ip, 2012) é um esforço colaborativo de universidades brasileiras para criar uma rede distribuída de centros de desenvolvimento de Circuitos Integrados capazes de atender a demanda por Propriedades Intelectuais em Semicondutores (semiconductor Intellectual Property – IP-cores). O Brazil-IP tem duas missões. Em curto prazo, o projeto almeja aumentar o nível de especialização no desenvolvimento em nível mundial de CI’s.

Isto é feito através da exposição dos estudantes às melhores práticas mundiais de desenvolvimento e verificação. Em longo prazo, o projeto contribuirá para trazer as condições necessárias para estabelecer Design Houses (DH) capazes de competir no mercado internacional de IP’s.

O grupo da UFPB, formado pelos laboratórios LASID (Laboratório de Sistemas Digitais) e LASIC (Laboratório de Arquiteturas, Sistemas Integráveis e Circuitos) do Departamento de Informática, de acordo com a chamada do Programa Brazil-IP 2008, ficou responsável pelo IP-core intitulado BSMILC (Biological Signals and Medical Images Lossless Compressor) – Compressor Sem Perdas de Sinais Biológicos e

Imagens Médicas (MARQUES, et al., 2011). As ferramentas EDA utilizadas no desenvolvimento desse projeto foram fornecidas pela Synopsys (Synopsys, 2012).

Durante o desenvolvimento do BSMILC, mais precisamente na etapa de Floorplanning (etapa importante na fase de design do layout no desenvolvimento de CI), a equipe percebeu que os arquivos utilizados para o mapeamento e o posicionamento dos Pads no chip eram gerados manualmente, acarretando problemas para os desenvolvedores, pois os arquivos são extensos tornando-se bastante suscetível a falhas humanas e a dificuldade na localização de erros.

Tentando resolver esse problema foi realizada uma pesquisa bibliográfica procurando outra forma de criação dos scripts, para os principais fornecedores de ferramentas EDA que não fosse manual, porém não foi encontrado nenhum outro mecanismo que substituísse a criação de tais scripts de forma manual.

Além disso, percebeu-se que esses scripts são utilizados pelas diferentes ferramentas EDA, modificando apenas a sintaxe de mapeamento e posicionamento.

1.2. Objetivos

O objetivo geral desse trabalho consiste em desenvolver uma ferramenta, denominada PadsTool, para a criação de scripts de mapeamento e posicionamento dos pads no núcleo automaticamente, de forma a possibilitar aos desenvolvedores gerenciar tanto o mapeamento quanto o posicionamento dos Pads por uma interface gráfica.

Para tanto, é possível subdividir o trabalho nos seguintes objetivos específicos:

- Análise dos scripts de instanciação (mapeamento) e posicionamento, considerando seus requisitos;
- Projetar uma arquitetura capaz de possibilitar a criação de regras de script para diversos fornecedores de ferramenta EDA.
- Criar geração automática de scripts para as ferramentas EDA da Synopsys e Cadence.
- Realização de testes com projetos de CIs;
- Utilização de metodologia para verificação da ferramenta;
- Análise do desempenho da ferramenta utilizada por usuários;

1.3. Organização

Este trabalho está organizado da seguinte forma. No Capítulo II é realizada a fundamentação teórica para introduzir os principais conceitos que são utilizados neste trabalho e servem como base para entendimento dessa proposta. O Capítulo III discursa sobre a metodologia de trabalho utilizada para fins de elaboração da arquitetura, implementação, teste e verificação da ferramenta proposta. Os resultados de validação com projetos de CI são apresentados no Capítulo IV. Por fim, o Capítulo V contém a conclusão de tudo que foi apresentado, além de arquivos gerados nos testes da ferramenta, apresentados na forma de Apêndices.

Capítulo II: Fundamentação Teórica

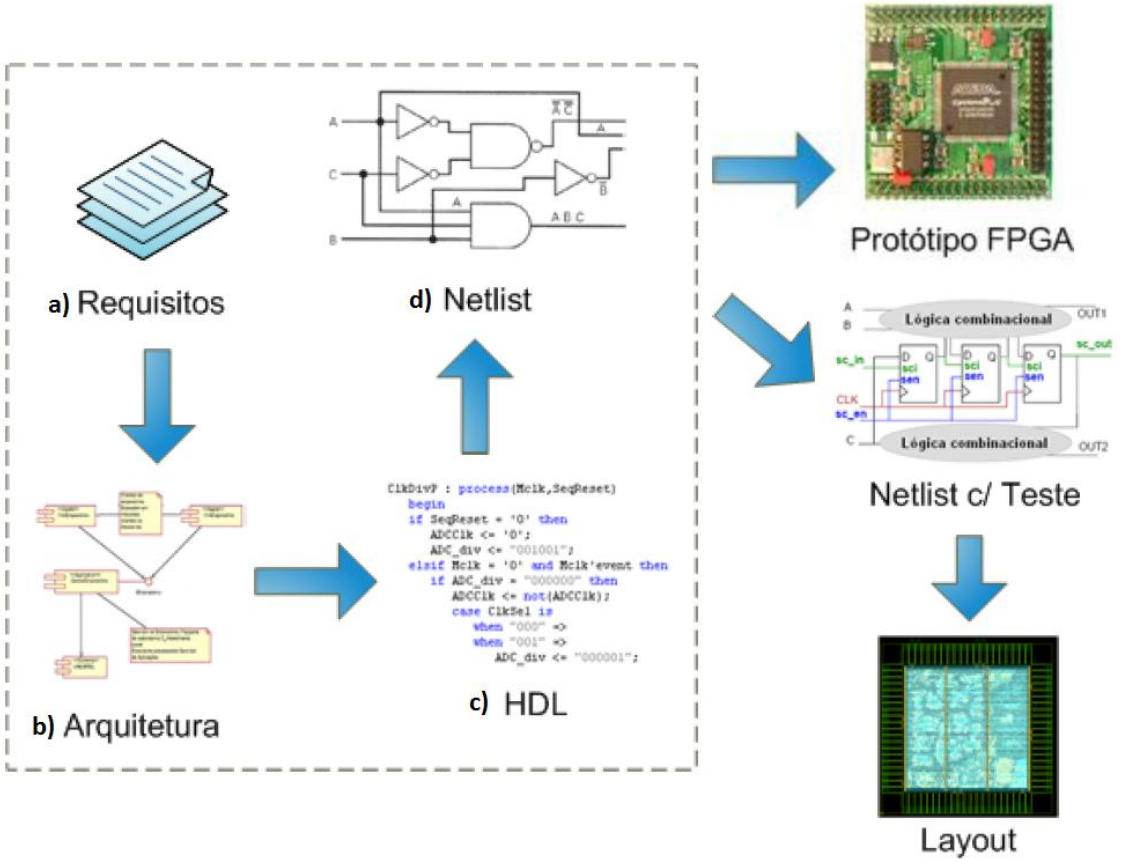
2.1. Fluxo de desenvolvimento de CI

Para o desenvolvimento bem definido de um módulo de hardware, assim como nos processos definidos pela Engenharia de Software, é necessário vencer algumas etapas que vão variar de acordo com a metodologia adotada. Em geral, esse fluxo divide-se em duas grandes etapas: Front-End e Back-End.

2.1.1. Front-End

A primeira etapa do Front-End consiste de uma análise ampla do que se deseja desenvolver. A Figura 1 mostra os quatro importantes passos seguidos na etapa de Front-End: Requisitos, Arquitetura, Desenvolvimento em HDL e por fim a Geração da Netlist.

Figura 1: Etapas iniciais para o desenvolvimento de um módulo de hardware.



A etapa de requisitos, vista na Figura 1.a, visa definir requisitos funcionais e não funcionais. Em um projeto comercial esta etapa passaria por conversas com o cliente

para que sejam entendidas as suas necessidades. Após o bom entendimento do que se deseja atingir, é formalizada a especificação de requisitos através de um documento de especificação e/ou documento de visão.

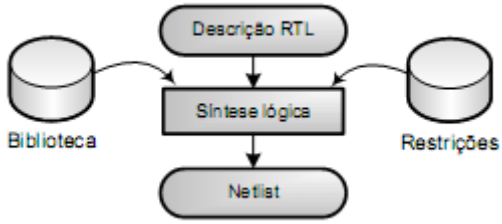
A partir desses requisitos é iniciada fase de desenvolvimento da arquitetura, conforme ilustra a Figura 1.b. Nela é definida a arquitetura do projeto (utilização de módulos) e como será estruturada a implementação dos requisitos.

A segunda etapa é a de Desenvolvimento RTL (Register Transfer Level - descrição comportamental do módulo), onde os documentos são transcritos em uma HDL,visto na Figura 1.c, que podem ser escritas, dentre outras, em Verilog ou VHDL.

A síntese de circuitos integrados é o processo de transformar uma descrição de mais alto nível – de maior abstração - para uma descrição de um nível mais baixo – mais próximo do nível físico -, mantendo a funcionalidade, conforme a Figura 1.d.

Continuando com o processo de construção do CI, a próxima etapa é a de síntese lógica (Figura 2), que faz uso de uma biblioteca de tecnologia e de equações booleanas para produzir um conjunto de células interconectadas.

Figura 2: Síntese lógica.



Para que esta conversão seja possível é necessária uma biblioteca que contém a definição das unidades lógicas previamente definidas. Estas informações podem ser obtidas através de design kits fornecidas pela foundry. Ainda com base nas informações contidas no design kit é possível fazer otimizações no projeto. Chamamos de otimização o passo no processo de síntese que procura combinar a biblioteca de células com restrições de tempo, área, funcionalidade e potência definidas pelo projetista, gerando a netlist de saída.

Depois de gerada a netlist é realizado o processo de verificação funcional, que é responsável por validar os pressupostos feitos na etapa de especificação através de uma demonstração concreta, bem como validar se os requisitos foram implementados corretamente e se o CI está funcionando como desejado.

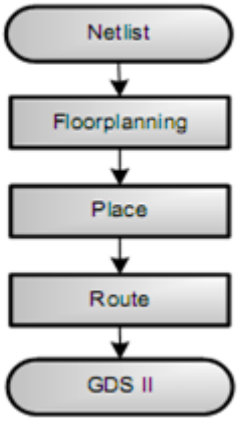
2.1.2. Back-End

Após a síntese lógica concluída e devidamente verificada, inicia-se a etapa de layout. Nessa etapa devem ser especificadas as características geométricas dos componentes do circuito (Macros, células lógicas, Pads, entre outros), que serão utilizadas para a construção do seu layout, para que o circuito integrado possa ser fabricado. Sendo assim, o layout apresenta uma superposição de máscaras do processo CMOS e pode ser confeccionada com a ajuda de sistemas computadorizados.

Para que seja formulada essa estrutura geométrica é necessária uma série de passos para ordenar e interconectar os componentes do circuito de maneira que sua funcionalidade não seja prejudicada. Um projeto de layout mal feito pode, por exemplo, posicionar elementos tão distantes um do outro que o atraso de propagação do sinal entre eles insira falhas na funcionalidade desejada do circuito.

As etapas do layout consistem essencialmente do Floorplaning, Placement (posicionamento) e Route (roteamento), como esta indicada na Figura 3.

Figura 3: Fluxo de projeto de layout



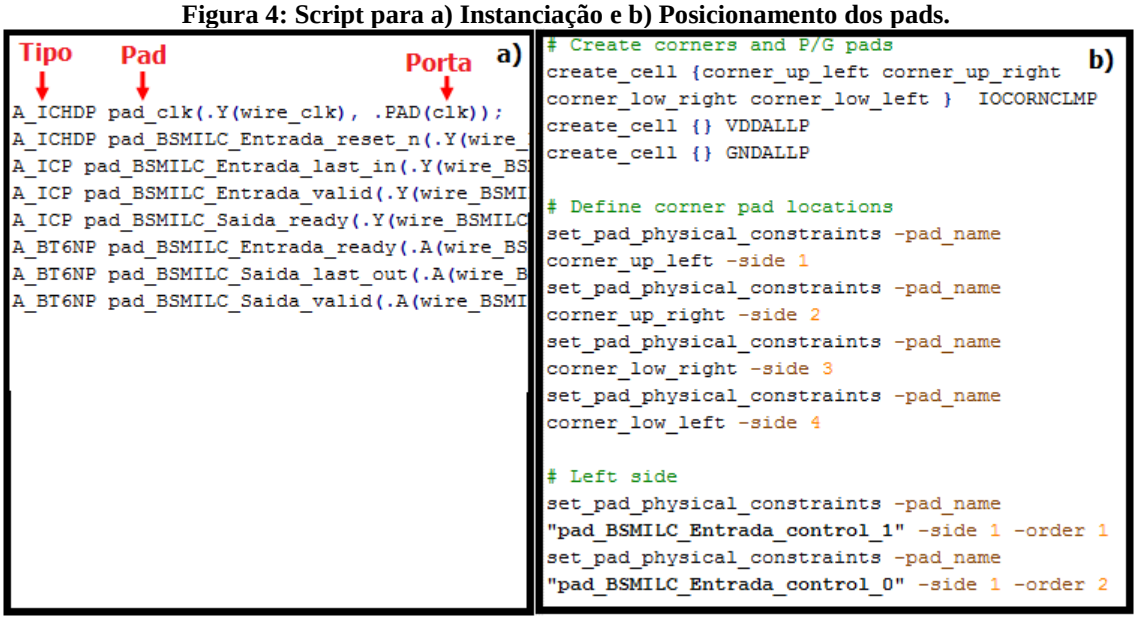
2.1.2.1. Floorplanning

Essa etapa consiste em minimizar a área do chip e os problemas com relação a atrasos. Durante a etapa de floorplaning são realizadas as seguintes atividades (Kommuru, et al., 2011):

- Posicionar os macroblocos no chip;
- Decidir a localização dos PADs de entrada e saída;
- Decidir a localização e o tipo de distribuição dos PADs de alimentação;
- Decidir a localização e o tipo da distribuição de clock.

Para auxiliar essa atividade, comumente, é feito um wrapper (camada de abstração) em HDL que mapeia as portas de entrada e saída do projeto em instâncias de Pads, ilustrada na Figura 4.a, com seus diferentes tipos, definidos pelo desenvolvedor e um arquivo que indica a posição de cada Pad no circuito, como mostra a Figura 4.b.

Figura 4: Script para a) Instanciação e b) Posicionamento dos pads.



Além disso, na etapa de Floorplanning é definida a limitação física de tamanho do CI, se é limitado pelo núcleo (*core limited*) ou pelo Pad (*pad limited*).

Um CI é do tipo *Core Limited* quando o tamanho do núcleo é que define o tamanho do CI, ou seja, o perímetro do núcleo é maior que a soma da largura de todos os pads inseridos no CI.

Já no tipo *Pad Limited*, o que define o tamanho do CI é a soma da largura de todos os pads, ou seja, a soma da largura de todos os pads inseridos no CI é maior que o tamanho do núcleo.

2.1.2.2. Place

Após completar o Floorplanning é possível iniciar a etapa de posicionamento das células lógicas do projeto dentro do núcleo do chip. De acordo com (Lira, 2009) Posicionamento é a tarefa de encontrar uma posição física para cada bloco ou célula que compõe o circuito, sem que nenhuma sobreposição ocorra, e com o objetivo de minimizar o tamanho das conexões. A etapa de posicionamento é muito mais passível de processos de automatização do que a etapa de floorplaning.

2.1.2.4. Route

Após as etapas *floorplanning* e *placement* no projeto é feito o roteamento (*route*) das células lógicas. Route é a ligação dos vários blocos do chip que estão colocados no chip.

De acordo com (Kommuru, et al., 2011), o roteamento é dividido em duas etapas:

- **Roteamento Global**: planeja as conexões globais entre todos os blocos e as redes. Seu principal objetivo é minimizar o comprimento total de interconexão, minimizar o atraso do caminho crítico. Ele determina as atribuições de faixas para cada interconexão. O chip é dividido em blocos pequenos. Esses pequenos blocos são chamados de caixas de roteamento. O tamanho do bloco de roteamento depende do algoritmo utilizado na ferramenta EDA.
- **Roteamento detalhado**: Nesta etapa ocorre a ligação efetiva entre todos os blocos. Ela cria a via real e ligações metálicas. O principal objetivo do roteamento detalhado é minimizar a área, o comprimento total do fio, e atraso nos caminhos críticos. Ele especifica as faixas específicas para a interligação, cada camada tem sua grade de roteamento próprio. Durante esse processo a largura, a camada e o local exato da interconexão são decididas.

2.2. Projeto BSMILC

Desenvolvido pelo grupo formado pelos laboratórios LASID e LASIC, o BSMILC é CI que faz a compressão sem perdas de sinais biológicos e imagens médicas.

Esse módulo é capaz de comprimir sinais biológicos e imagens médicas sem perdas através de um modelo adaptativo e contextual baseado no algoritmo PPM (Prediction by Partial Matching) (CLEARY, et al., 2006).

As ferramentas EDA utilizadas no desenvolvimento desse projeto foram fornecidas pela Synopsys (Synopsys, 2012).

2.3. Digital Modulator

A modulação corresponde a um processo de conversão de sinais para fins de transmissão, sendo definido como um sistema que recebe duas entradas (informação e portadora) e fornece um sinal de saída que será utilizado no transporte da informação.

Com o intuito de criar um modulador mais eficiente o LINCS (Laboratório para Integração de Circuitos e Sistemas) do CETENE (Centro de Tecnólogias Estratégicas

do Nordeste) situado em Recife – Pernambuco, desenvolveu o projeto Digital Modulator.

O Digital Modulator é constituído por uma cadeia de transmissão com dois canais (I e Q), defasados de 90° entre si, e uma cadeia de recepção com dois canais (I e Q), também defasados de 90° entre si.

O sistema possui ainda um módulo NCO, para gerar as ondas senoidais na frequência intermediária, um seletor de canais que define o valor do incremento de fase correspondente a cada canal, um multiplexador para alternar os canais I e Q na saída para o DAC e um demultiplexador para alternar os canais I e Q da entrada vinda do ADC.

Para o desenvolvimento do projeto, foi utilizada a metodologia IpProcess e para verificação do sistema foi utilizada a metodologia OVM_tpi.

Finalizada a implementação do RTL e verificação foi dado inicio a criação do layout. Para isto, foram utilizadas as ferramentas e scripts da Cadence®.

2.4. Linguagem Java

Java é uma linguagem de programação desenvolvida pela Sun Microsystems (Java Technology, 2012). A linguagem Java foi utilizada durante todo o desenvolvimento do projeto BSMILC, além disso, Java possui algumas características nativas que auxiliaram no desenvolvimento da ferramenta.

2.4.1. Classe Abstrata Java

Uma classe abstrata possui características que devem ser implementadas por classes filhas que vão especializar suas características e comportamento, permitindo instanciar o objeto indiretamente através de seu filho (Java Tutorials, 2012).

Além de não produzir instância, uma classe abstrata permite a declaração de métodos abstratos. Os métodos abstratos são obrigatoriamente implementados pelas classes filhas concretas. Ou seja, uma classe abstrata serve apenas como modelo para uma classe concreta.

2.4.2. Reflexão Java

A reflexão é uma característica da linguagem de programação Java (Richmond, 2001). A API (Application Programmer Interface) de reflexão Java permite que um

programa, em tempo de execução, recupere informações sobre si e sobre o sistema de execução (máquina virtual) em que ele executa.

Usando a reflexão, um programa pode obter informações sobre a estrutura e o ambiente de execução. Com essa informação o programa pode instanciar classes arbitrárias e invocar métodos em tempo de execução (McCluskey, 1998).

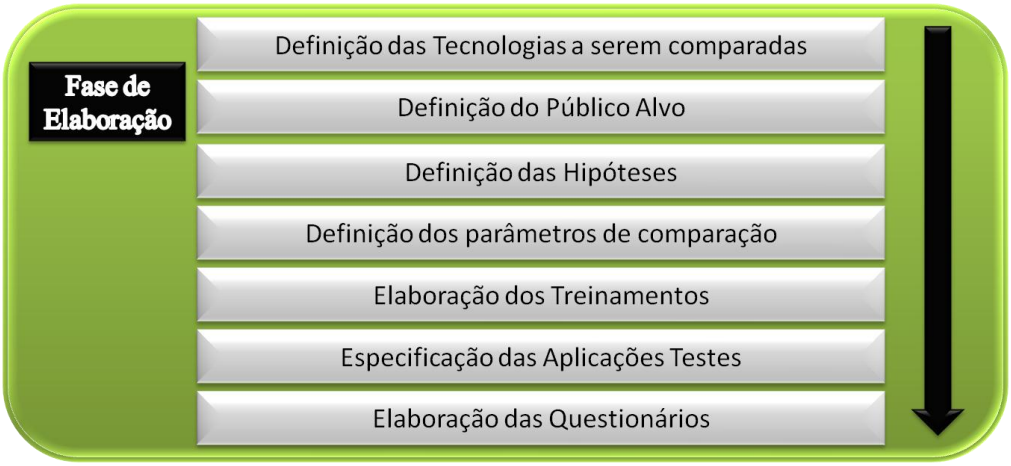
2.5. Metodologia Para Validação

Para validar frameworks através de testes com usuários, SEGUNDO (SEGUNDO, 2011) definiu uma metodologia capaz de avaliar o uso de frameworks por desenvolvedores. Sua metodologia dividiu-se em três etapas de execução: a Fase I consiste na elaboração, a Fase II na execução e, por fim, a Fase III é a etapa de análise dos resultados.

2.5.1. Elaboração

De acordo com SEGUNDO, na fase de elaboração são definidos os aspectos específicos de cada teste, tais como os formulários a serem utilizados na pesquisa e os parâmetros específicos do domínio do framework, conforme descrito na Figura 5.

Figura 5: Etapas da fase de elaboração.



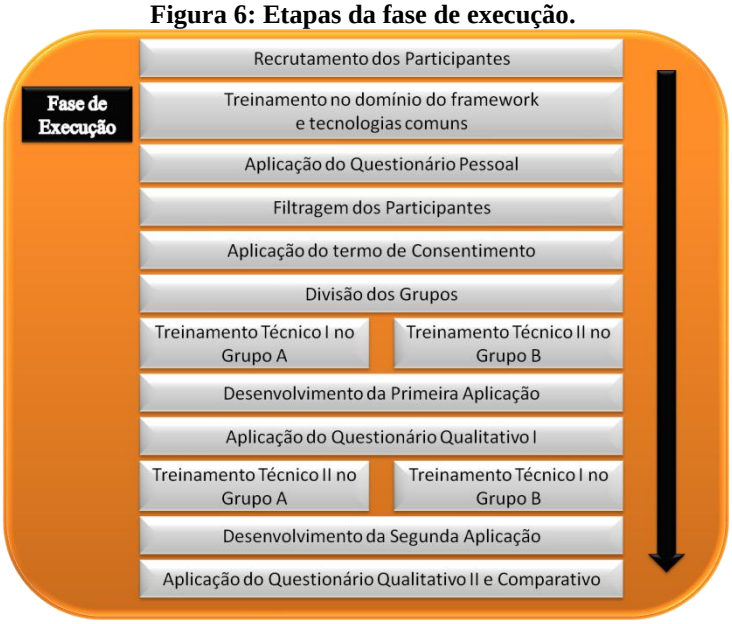
(SEGUNDO, 2011)

2.5.2. Execução

A fase de execução corresponde à fase de aplicação dos testes, utilizando os artefatos gerados na fase de elaboração. Suas etapas são descritas na Figura 6.

De acordo com a quantidade de testes a ser feito, as fases de Treinamento Técnico, Desenvolvimento da Aplicação e Aplicação do Questionário são repetidas sequencialmente.

Figura 6: Etapas da fase de execução.

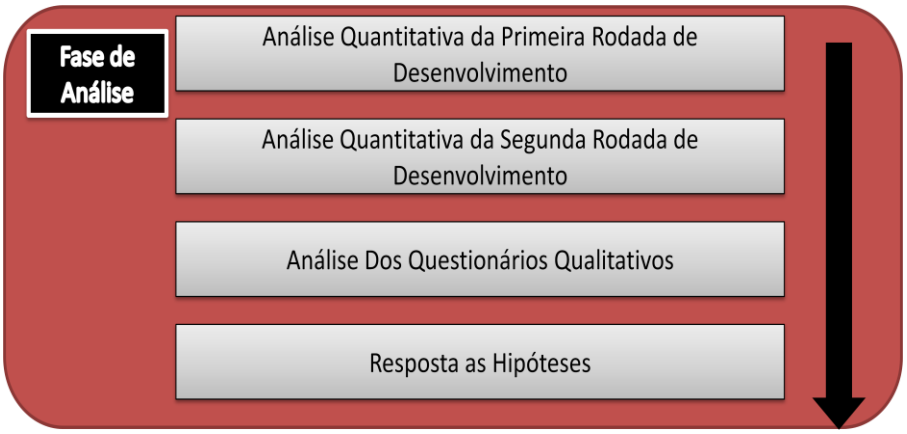


(SEGUNDO, 2011)

2.5.3. Análise dos Resultados

SEGUNDO definiu essa fase para a coleta e avaliação dos dados obtidos pelos testes realizados pelos participantes com o objetivo de averiguar se as hipóteses especificadas na primeira fase estão conforme o planejado. As etapas da fase estão descrita na Figura 7.

Figura 7: Etapas da fase de Análise.



(SEGUNDO, 2011)

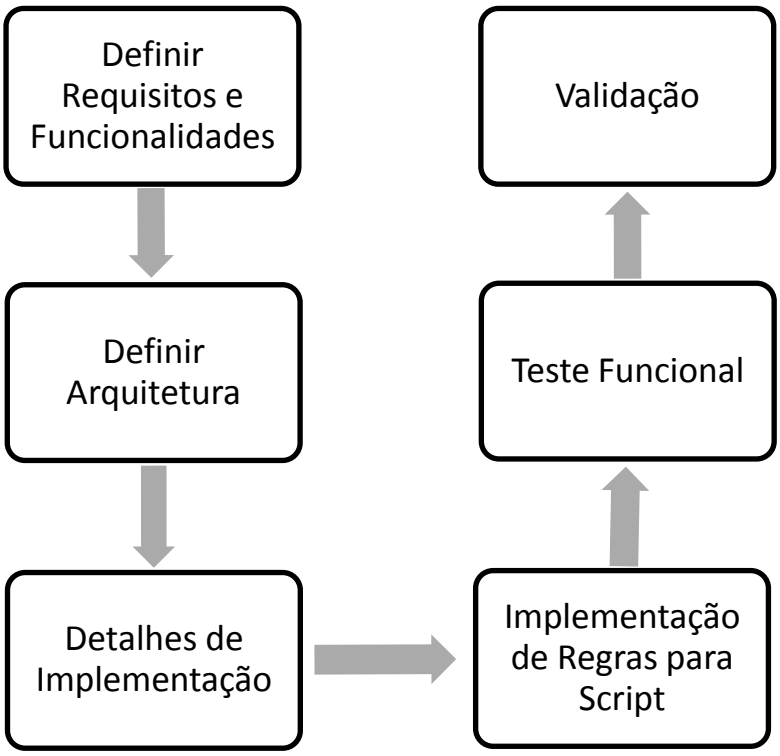
As quantidades das fases da Análise Quantitativa variam de acordo com o número de casos a serem testados.

Capítulo III: Sistema Proposto e Metodologia

3.1. Fluxo de Atividades

De acordo com os objetivos desse trabalho, é possível definir uma sequência lógica de atividades necessárias à sua conclusão e a obtenção dos resultados esperados, conforme a figura a seguir.

Figura 8: Fluxo de atividades empregado no trabalho.



3.2. Definição dos Requisitos e Funcionalidades

A principal funcionalidade do sistema proposto é a geração automática de scripts com o mapeamento e posicionamento dos Pads ao redor do núcleo a partir de um arquivo de netlist gerado pela síntese lógica. Para isso, o programa deve abrir a netlist escolhida e reconhecer os módulos e os pinos nela definidos.

Além disso, o programa deverá reconhecer automaticamente para quais fornecedores de ferramentas EDA (Synopsys, Cadence, Menthor) está disponível a geração de script.

Outras características e funcionalidades desejáveis no sistema é ser multiplataforma, ser capaz de detectar erros como sobreposicionamento dos Pads, não mapeamento ou posicionamento dos Pads, além da não criação dos Pads de alimentação.

3.3. Definição da Arquitetura

A ferramenta PadsTool, conforme ilustra a Figura 9, tem como entrada a netlist, no formato verilog (*.v) e tem como saída dois arquivos:

- Arquivo contendo a instancia dos pads do módulo escolhido, estruturado de acordo com a ferramenta escolhida.

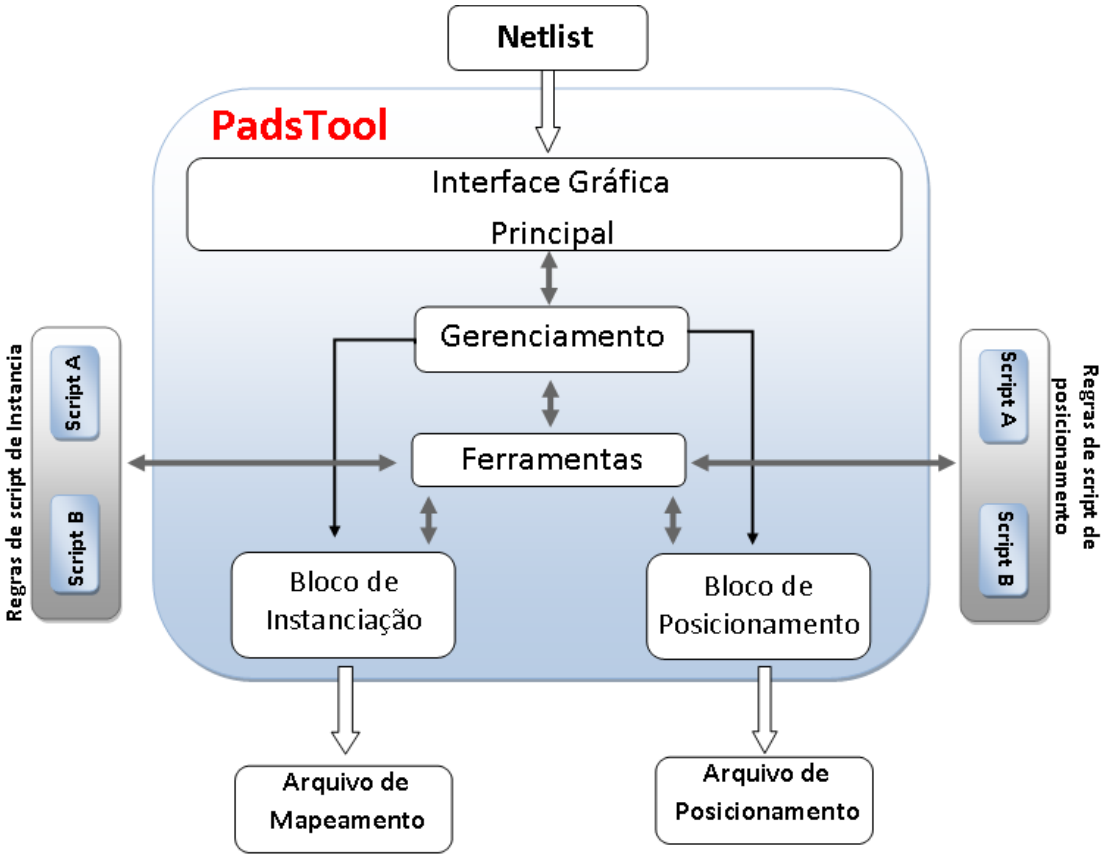
- Arquivo com as informações da estrutura de posicionamento dos pads ao redor do núcleo do chip.

Figura 9: Arquivos de entrada e saída da ferramenta.

Internamente a arquitetura divide-se em cinco blocos, conforme apresentado na Figura 10:

- Interface Gráfica Principal (GUI - Graphical User Interface);
- Gerenciamento;
- Ferramentas;
- Bloco de Instanciação;
- Bloco de Posicionamento;

Figura 10: Arquitetura interna da ferramenta.



Além disso, há uma comunicação com dois blocos externos que contêm as regras de criação dos scripts de instanciação e posicionamento para as ferramentas EDA disponíveis.

3.3.1. Interface gráfica principal

O primeiro bloco é responsável pela visualização gráfica da ferramenta. Nela o desenvolvedor gerencia o mapeamento e o posicionamento dos Pads. A tela é alternada entre mapeamento e posicionamento de acordo com o usuário.

Esse bloco comunica-se diretamente com o bloco de gerenciamento e com ele o usuário poderá executar todas as atividades definidas na ferramenta.

3.3.2. Gerenciamento

Tem a finalidade de fazer a comunicação entre a interface gráfica e os blocos de mapeamento e posicionamento. Além disso, ele captura informações do bloco ferramentas sobre quais ferramentas de EDA estão disponíveis para gerar os scripts.

Este bloco também é responsável por abrir o arquivo de entrada bem como acionar os blocos de instanciação e posicionamento para salvar os scripts com os pads mapeados e posicionados, respectivamente.

3.3.3. Ferramentas

O bloco ferramentas é responsável por explorar o arquivo de entrada que é carregado pelo bloco de Gerenciamento e coletar informações sobre os módulos nele contido armazenando os pinos de entrada e saída de cada um. Essas informações são passadas para a GUI através do bloco de gerenciamento e com elas o desenvolvedor poderá realizar o mapeamento e o posicionamento de cada pad.

Esse bloco também comunica-se com os blocos externos (Regras de scripts de Instância e Posicionamento) e verifica, em tempo de execução, as classes que estão disponíveis para a geração dos scripts e as fornecem para os blocos de instanciação e posicionamento. Essas classes devem ser compatíveis com os templates fornecidos pelos blocos de instanciação e posicionamento.

3.3.4. Blocos de instanciação e posicionamento

São responsáveis, respectivamente, pela criação dos scripts de mapeamento e posicionamento dos pads. Esses blocos recebem do bloco ferramentas os arquivos

responsáveis pela criação dos scripts das várias ferramentas EDA e os executam de acordo com a escolha do usuário.

Além disso, esses blocos contêm os templates, com os parâmetros de entrada e saída, que deverão ser adotados pelos desenvolvedores na implementação das regras de scripts das diversas ferramentas EDA.

3.3.5. Regras de script para instanciação e posicionamento

Os blocos de regras de script de instanciação e de posicionamento são externos a ferramenta. Neles estão contidos os arquivos que geram os scripts com a sintaxe dos diversos fornecedores de ferramentas EDA.

Dependendo da implementação escolhida, os blocos tem a opção de serem unificados com os arquivos implementados para a instanciação e o posicionamento, conforme descrito na Figura 11.

Figura 11: Bloco com as regras de scripts de instanciação e posicionamento.

3.4. Detalhes de Implementação

A ferramenta foi subdividida em cinco módulos principais, como mostrado na Figura 10: GUI, Gerenciamento, Ferramentas, Bloco de Instanciação e Bloco de Posicionamento.

3.4.1. Linguagem de Programação e Ambiente de Desenvolvimento

Definida a arquitetura da ferramenta proposta, o próximo passo foi escolher a linguagem de programação que melhor se encaixe na arquitetura.

A linguagem de programação Java foi utilizada para a implementação da ferramenta. O uso da técnica de interface Java, descrita no capítulo anterior, ajuda na

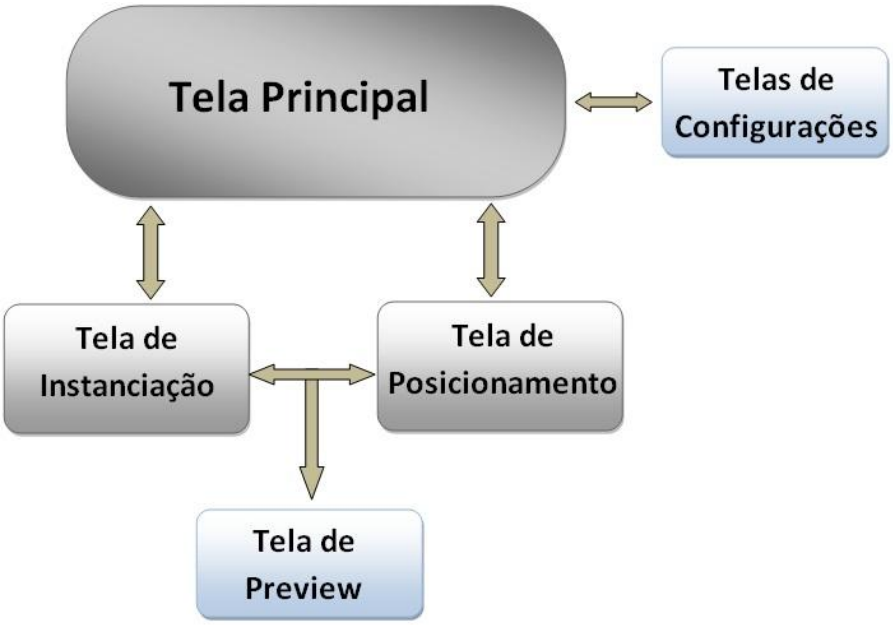
criação dos templates e a reflexão é bastante eficiente no carregamento dos arquivos em tempo de execução.

Além disso, o fato da linguagem ser independente de plataforma torna o programa compilado e os scripts implementados pelo desenvolvedor utilizável nos diversos sistemas operacionais sem precisarem ser recompilados. O ambiente de desenvolvimento utilizado foi o NetBeans IDE 6.9.1 (NetBeans, 2012).

3.4.2. Interface gráfica

A interface gráfica foi subdividida em cinco módulos, como mostrado na Figura 12: Tela principal, Telas de configurações, Tela de instanciação, Tela de posicionamento e Tela de preview.

Figura 12: Divisão da interface gráfica.

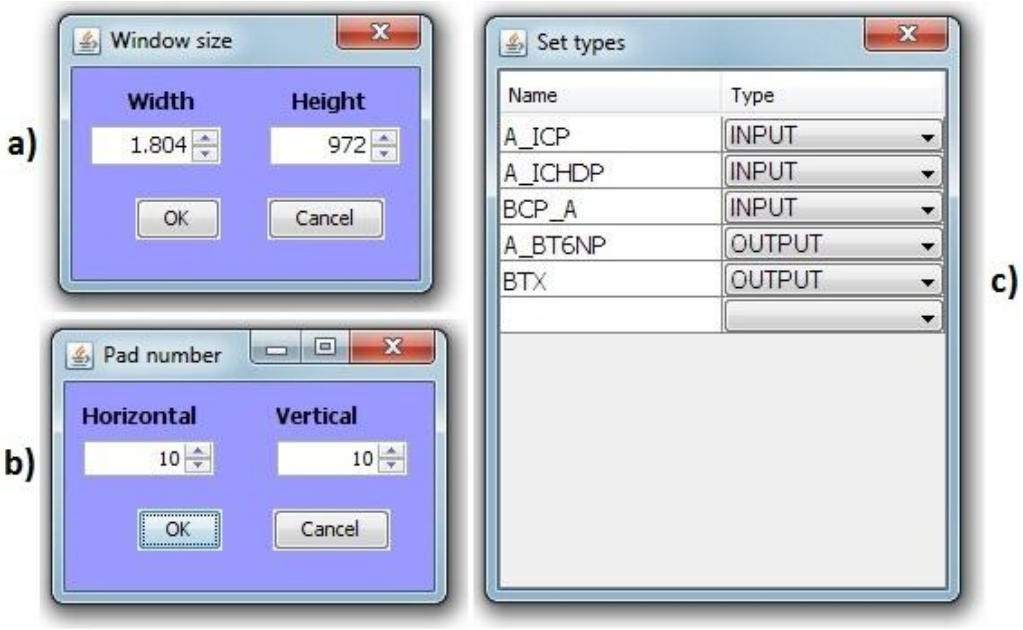


A Tela principal, conforme é apresentada na Figura 13, é responsável pela comunicação com o bloco de Gerenciamento. Nela o usuário escolhe o arquivo a ser trabalhado, faz o chaveamento entre as telas de instanciação e posicionamento, aciona as telas de configurações do sistema e seleciona para qual fornecedor de ferramenta EDA serão gerados os scripts.

Figura 13: Tela principal.

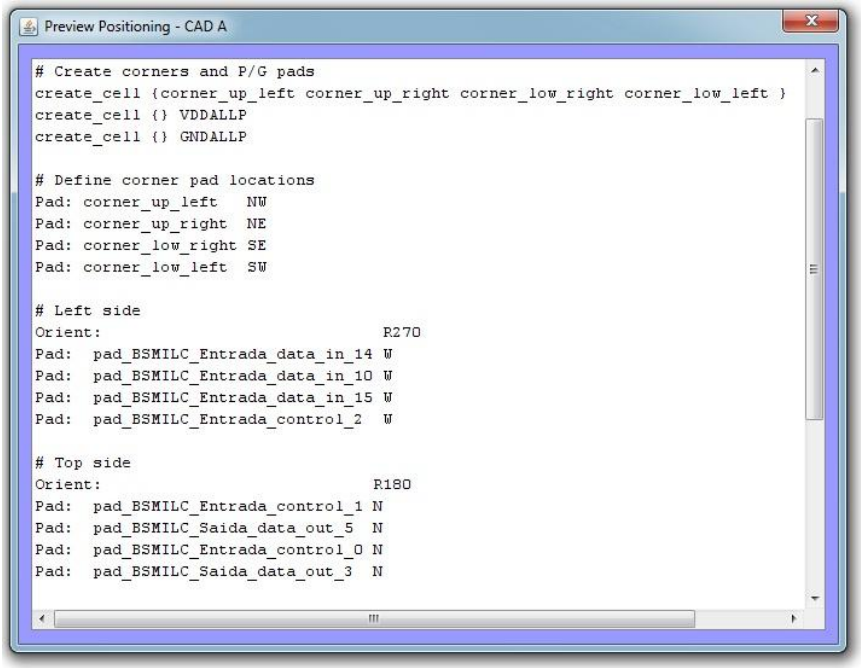
Nas telas de configurações, ilustrada na Figura 14, o usuário pode fazer todas as configurações do sistema: ajustar tamanho da tela, gerenciar a quantidade de pads ao redor do core do chip, listar os tipos de pads para instanciação.

Figura 14: Telas de configuração a) Tamanho da janela, b) Número de Pads, c) Tipos de Pads.



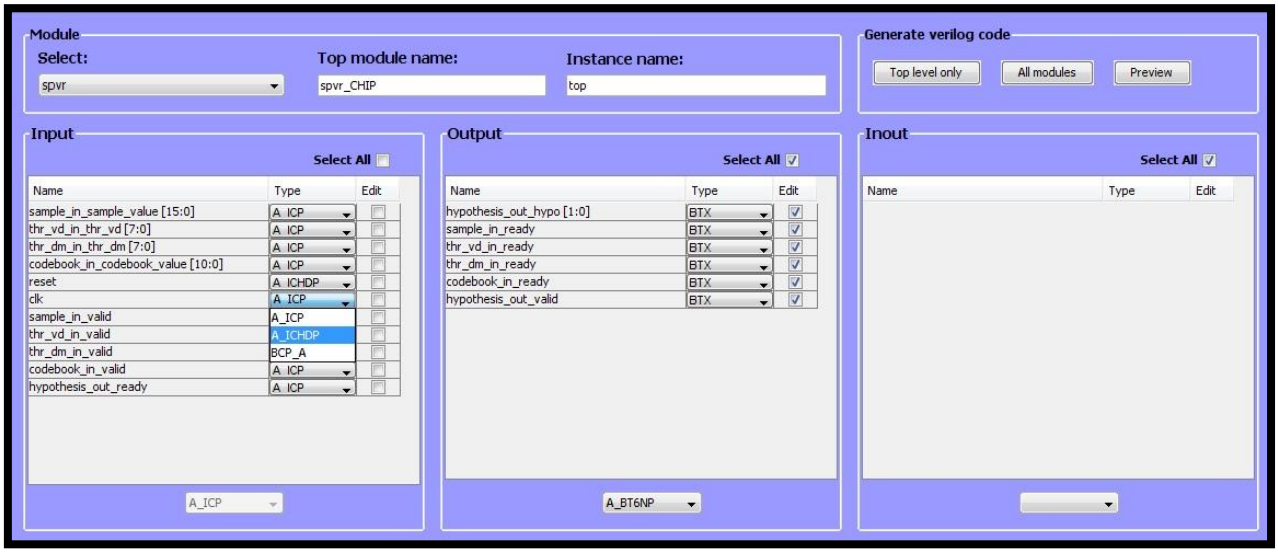
Na tela de preview é apresentada uma prévia do script gerado pela ferramenta antes que ele seja salvo em arquivo, ver Figura 15.

Figura 15: Tela de preview.



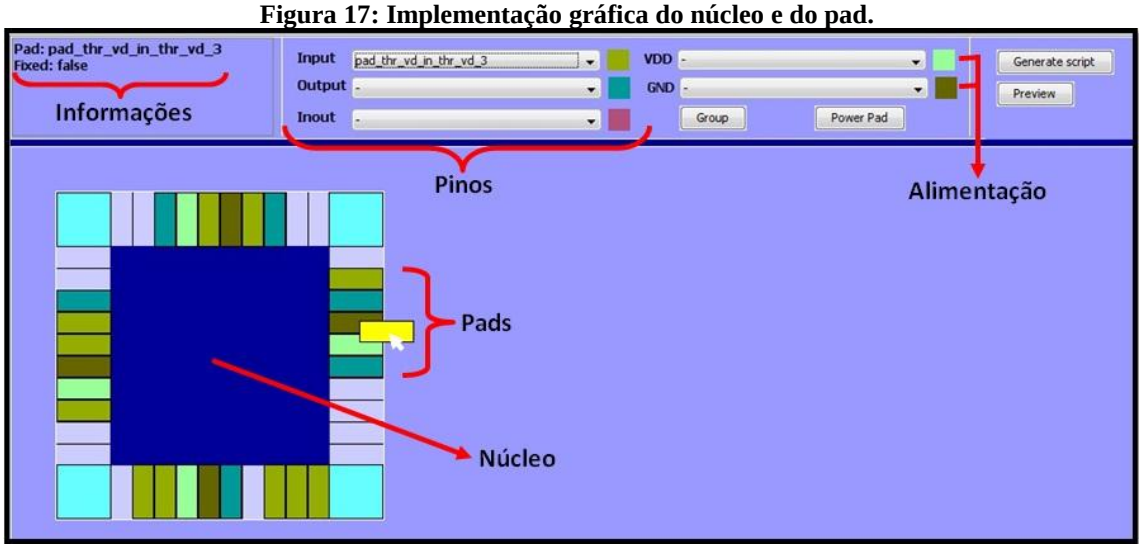
Na tela de instanciação, apresentada na Figura 16, são listados todos os módulos encontrados no arquivo carregado e para cada módulo são mostrados todos os pinos separados pelo tipo (input, output e inout). O usuário tem a opção de renomear o nome da instância ou do módulo top, além disso, ele pode ativar a tela de preview (Figura 15) ou então salvar o arquivo instanciado.

Figura 16: Tela de instanciação.



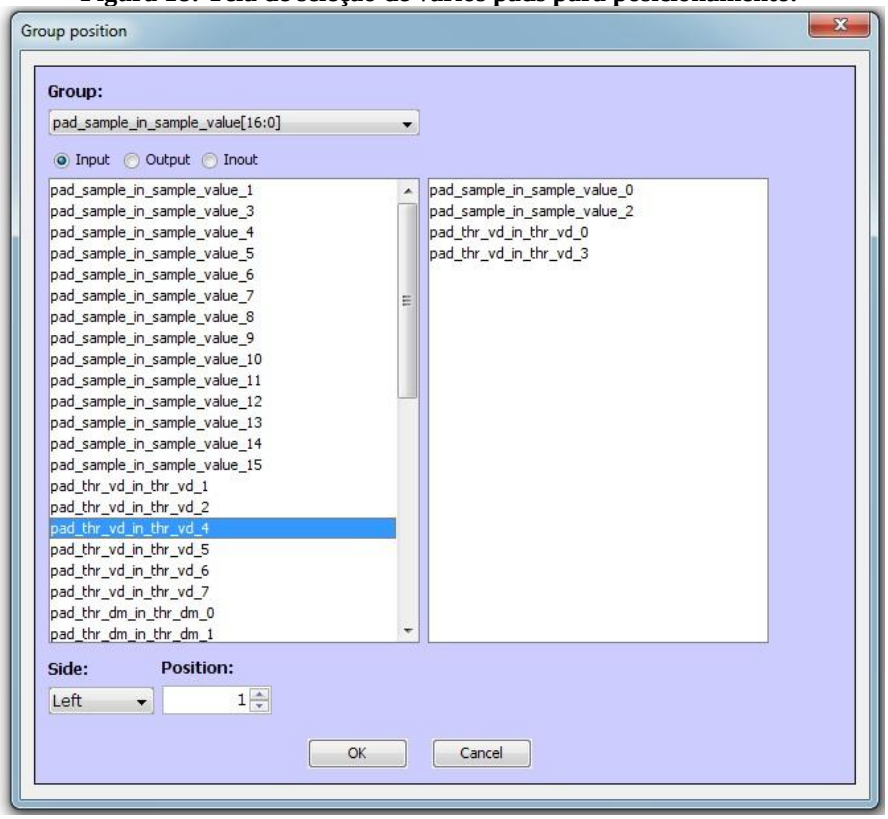
Na tela de posicionamento são listados todos os pads, que já foram previamente mapeados, e ao selecioná-los o usuário pode mover para qualquer lugar ao redor do núcleo do chip, veja Figura 17.

Figura 17: Implementação gráfica do núcleo e do pad.



Além disso, o usuário pode selecionar vários pads para posicionamento, conforme ilustra a Figura 18, pode ativar a tela de preview, vista na Figura 15, ou então salvar o arquivo posicionado.

Figura 18: Tela de seleção de vários pads para posicionamento.



3.4.3. Gerenciamento

Este módulo é responsável pela comunicação entre a GUI e os módulos de acesso a arquivos e criação dos scripts. Ele também gerencia a troca das telas de instanciação e posicionamento, efetuada pela janela principal da GUI.

3.4.4. Ferramenta

O módulo Ferramenta carrega todas as informações necessárias para o mapeamento e posicionamento dos pads. Ela abre a netlist selecionada, captura os pinos de entrada e saída e envia para a interface principal, por meio do módulo de gerenciamento.

Além disso, ela carrega, por meio de reflexão, todas as classes com as regras de scripts para mapeamento e posicionamento, verifica se elas são instancias da classe abstrata definida no template e as envia para os blocos de instanciação e posicionamento, respectivamente.

3.4.5. Blocos de Instanciação e Posicionamento

Esses blocos são responsáveis por gerar os scripts de instanciação e posicionamento. Neles estão contidas todas as classes com as regras de geração de scripts das diferentes ferramentas, carregada pelo bloco Ferramentas.

Além disso, está definida a classe abstrata com os métodos que devem ser implementados o para mapeamento e posicionamento, descrito na Figura 19.

Figura 19: Template para mapeamento e posicionamento.

```
package template;

/**
 *
 * @author Janduy
 */
public abstract class ScriptInstantiationAndPosition {


    public abstract void initPosition(String[] padsLeft, String[] padsTop,
                                String[] padsRight, String[] padsBottom,
                                String[] corners, String[] pads_vdd, String[] pads_gnd);

    public abstract String getScriptPosition();


    public abstract void initInstantiation(String[] input, String[] output, String[] inout,
                                           String[] tipoInput, String[] tipoOutput, String[] tipoInout,
                                           int[] nInput, int[] nOutput, int[] nInout,
                                           String modulo, String instancia, String top);

    public abstract String getScriptInstantiation();

    public abstract String getNomeFerramenta();
}
```

3.4.6. Regras para Criação de Script

Para que o programa gere scripts automaticamente é necessário que as regras de criação de scripts estejam implementadas. O programa foi projetado de tal forma que as regras ficam independente do sistema, facilitando o desenvolvedor, pois ele não necessitar entender o código do programa (nem mesmo do código fonte) para criar as regras. Basta extender o template de criação (que é uma classe abstrata e fica separada do programa) e implementar os métodos nela contidos.

Após implementar e compilar a classe, o usuário precisa apenas colocar o código (extensão *.class) na pasta referente as classes com as regras para as diferentes ferramentas EDA.

3.4.7. Implementação de Funcionalidades

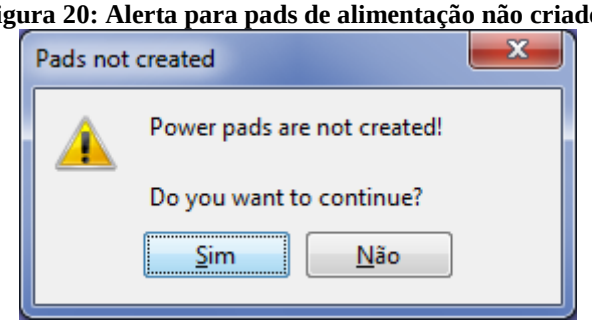
O programa é capaz de detectar e corrigir erros que podem ser gerados na criação dos scripts de mapeamento e posicionamento dos Pads.

A interface gráfica de posicionamento foi implementada de forma que não há possibilidade de algum Pad ficar sobreposto a outro. Quando o usuário desloca um Pad

para uma posição onde já exista um Pad a ferramenta, automaticamente, desloca Pad (o que já estava ou o que deseja ser posicionado) para a posição imediatamente vizinha ou para a posição vazia mais próxima, dependendo da configuração definida pelo usuário.

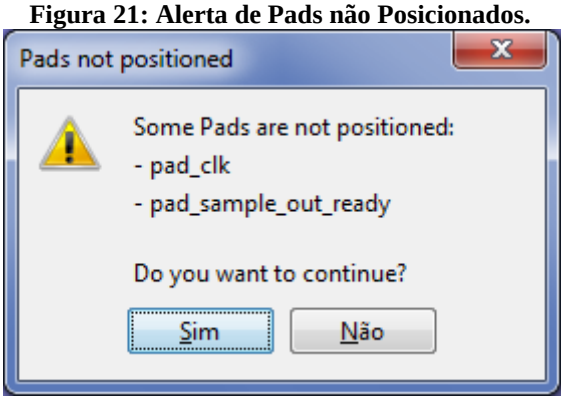
Quando o usuário não cria os Pads de alimentação e tenta gerar o script, a ferramenta faz um alerta para a não criação dos Pads de alimentação, como ilustra a Figura 20. Isso ajuda a evitar que o usuário crie os scripts de posicionamento e esqueça-se de mapear e instanciar os Pads de alimentação.

Figura 20: Alerta para pads de alimentação não criados



Além disso, a ferramenta faz o alerta caso o usuário tenha se esquecido de posicionar todos os Pads e tente gerar o script de posicionamento, como mostra a Figura 21. Isso evita que o script seja criado com alguns Pads não posicionados.

Figura 21: Alerta de Pads não Posicionados.



Outra funcionalidade implementada na ferramenta é a detecção de pinos não mapeados. A interface gráfica da ferramenta foi desenvolvida de forma que todos os pinos de entrada e saída, definidos pelo usuário, são mapeados e criados suas instâncias de Pads. Isso garante que o usuário crie o script de mapeamento sem que nenhum pino seja mapeado, conforme mostra a Figura 22.

Figura 22: Mapeamento dos pinos.

3.5. Verificação Funcional

O objetivo da verificação é validar a implementação de acordo com o que se espera ser o seu funcionamento correto. No teste funcional, os arquivos gerados pela ferramenta são confrontados com os scripts gerados manualmente e seus valores de saída são comparados para que o teste seja finalizado, ou então, os arquivos gerados automaticamente pela ferramenta são substituídos pelos scripts gerados manualmente dando continuidade no processo de Floorplanning. Caso não haja diferença, o teste é validado.

Para realização do teste funcional bem como validar a ferramenta foi utilizado o projeto de BSMILC e Digital Modulator.

3.5.1. Teste com o Projeto BSMILC

O projeto foi desenvolvido com as ferramentas EDA da Synopsys e possui 60 pinos de entrada, saída e alimentação, veja Tabela 1.

Tabela 1: Lista de pinos do projeto BSMILC.

Pinos	Input	Output	VDD	GND	Total
Quantidade	25	11	12	12	60

No script de instanciação, o pino de clock e de reset do BSMILC foi mapeado em PAD do tipo A_ICHDP, e as demais portas de entrada e de saída em A_ICP e A_BT6NP respectivamente.

O projeto do BSMILC possui 24 pads de alimentação; sendo eles 12 de VDD e 12 de GND, distribuídos de forma a alimentar o núcleo e os pads. Os pads de VDD são do tipo VDDALLP de maneira semelhante para os pads VSS – do tipo GNDALLPD. Além disso, existem quatro corner pads do tipo IOCORNCLMP.

3.5.2. Teste com o Projeto Digital Modulator

O projeto foi desenvolvido com as ferramentas EDA da Cadence e possui 75 pinos de entrada, saída e alimentação, veja Tabela 2.

Tabela 2: Lista de pinos do projeto Digital Modulator.

Pinos	Input	Output	VDD	GND	Total
Quantidade	31	36	4	4	75

No script de instanciação, o pino de clock e de reset do Digital Modulator foi mapeado em PAD do tipo A_ICHDP, e as demais portas de entrada e de saída em A_ICP e A_BT6NP respectivamente.

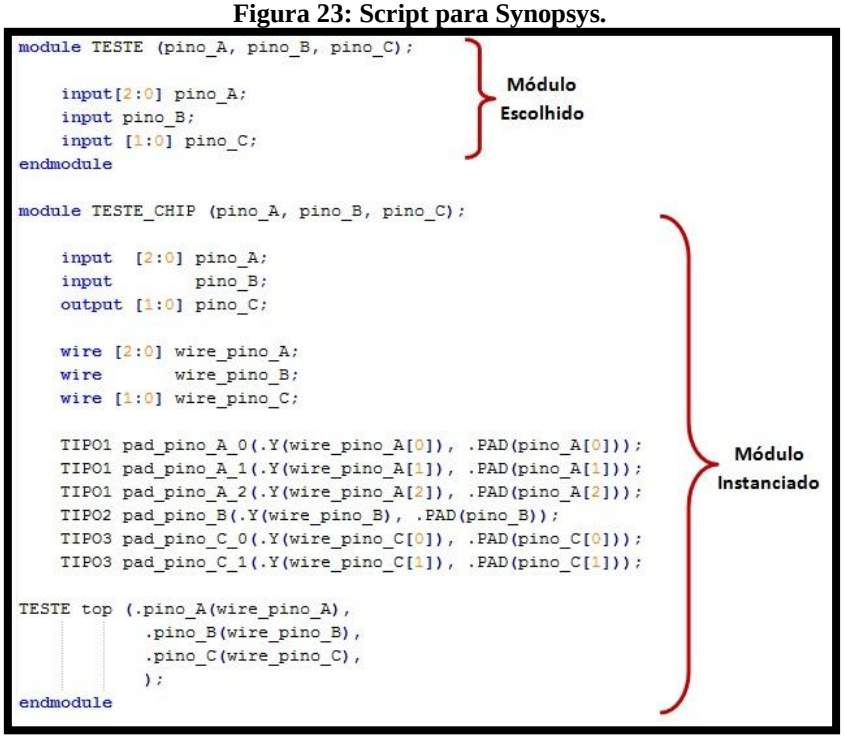
O projeto do Digital Modulator possui 8 pads de alimentação; sendo eles 4 de VDD e 4 de GND, distribuídos de forma a alimentar o núcleo e os pads separadamente. Os pads de VDD são do tipo VDDALLP de maneira semelhante para os pads VSS – do tipo GNDALLPD. Além disso, existem quatro corner pads do tipo IOCORNCLMP.

3.5.3. Criação de Regras para Script Synopsys

Para validar a criação dos scripts para o projeto BSMILC, foi implementada uma classe com as regras de criação de script para ferramentas EDA fornecidas pela Synopsys.

O script de instanciação é um arquivo com a extensão verilog (*.v) e ele pode ser anexado à netlist. O módulo selecionado é transcrito no script com todos os pinos de entrada e saída acrescentado de declaração de fios para todos os pinos, veja Figura 23
. Além disso, é feito a ligação entre os pinos e o pad, pelos fios criados, com os tipos fornecidos pelo usuário.

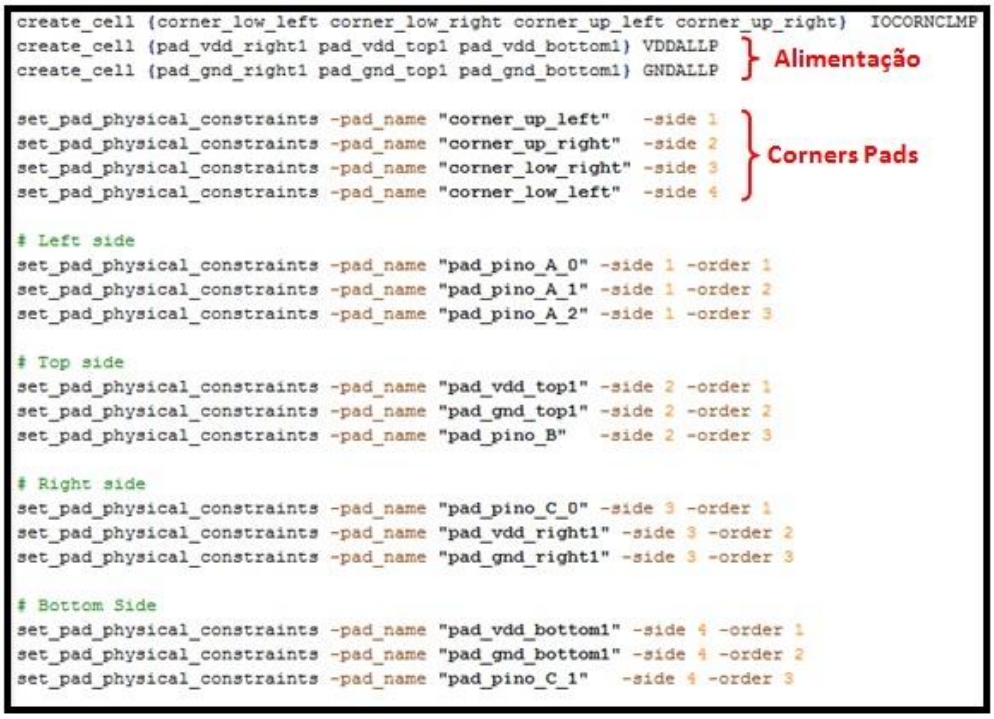
Figura 23: Script para Synopsys.




```
module TESTE (pino_A, pino_B, pino_C);

    input[2:0] pino_A;
    input pino_B;
    input [1:0] pino_C;
endmodule

module TESTE_CHIP (pino_A, pino_B, pino_C);

    input  [2:0] pino_A;
    input        pino_B;
    output [1:0] pino_C;

    wire [2:0] wire_pino_A;
    wire       wire_pino_B;
    wire [1:0] wire_pino_C;

    TIPO1 pad_pino_A_0(.Y(wire_pino_A[0]), .PAD(pino_A[0]));
    TIPO1 pad_pino_A_1(.Y(wire_pino_A[1]), .PAD(pino_A[1]));
    TIPO1 pad_pino_A_2(.Y(wire_pino_A[2]), .PAD(pino_A[2]));
    TIPO2 pad_pino_B(.Y(wire_pino_B), .PAD(pino_B));
    TIPO3 pad_pino_C_0(.Y(wire_pino_C[0]), .PAD(pino_C[0]));
    TIPO3 pad_pino_C_1(.Y(wire_pino_C[1]), .PAD(pino_C[1]));

TESTE top (.pino_A(wire_pino_A),
           .pino_B(wire_pino_B),
           .pino_C(wire_pino_C),
           );
endmodule
```

No script de posicionamento são declarados os pads de alimentação e os corners pads, de acordo com os tipos definidos pela ferramenta EDA escolhida. Depois, cada pad é posicionado e sua localização é determinada pelo lado (Left – 1, Top – 2, Right – 3 e Bottom – 4) e pela ordem (sentido bottom-top e left-Right) conforme na Figura 24.

Figura 24: Sintaxe de posicionamento Synopsys.




```
create_cell {corner_low_left corner_low_right corner_up_left corner_up_right}  IOCORNCLMP
create_cell {pad_vdd_right1 pad_vdd_top1 pad_vdd_bottom1} VDDALLP
create_cell {pad_gnd_right1 pad_gnd_top1 pad_gnd_bottom1} GNDALLP

set_pad_physical_constraints -pad_name "corner_up_left"   -side 1
set_pad_physical_constraints -pad_name "corner_up_right"  -side 2
set_pad_physical_constraints -pad_name "corner_low_right" -side 3
set_pad_physical_constraints -pad_name "corner_low_left"  -side 4

# Left side
set_pad_physical_constraints -pad_name "pad_pino_A_0" -side 1 -order 1
set_pad_physical_constraints -pad_name "pad_pino_A_1" -side 1 -order 2
set_pad_physical_constraints -pad_name "pad_pino_A_2" -side 1 -order 3

# Top side
set_pad_physical_constraints -pad_name "pad_vdd_top1" -side 2 -order 1
set_pad_physical_constraints -pad_name "pad_gnd_top1" -side 2 -order 2
set_pad_physical_constraints -pad_name "pad_pino_B"   -side 2 -order 3

# Right side
set_pad_physical_constraints -pad_name "pad_pino_C_0" -side 3 -order 1
set_pad_physical_constraints -pad_name "pad_vdd_right1" -side 3 -order 2
set_pad_physical_constraints -pad_name "pad_gnd_right1" -side 3 -order 3

# Bottom Side
set_pad_physical_constraints -pad_name "pad_vdd_bottom1" -side 4 -order 1
set_pad_physical_constraints -pad_name "pad_gnd_bottom1" -side 4 -order 2
set_pad_physical_constraints -pad_name "pad_pino_C_1"   -side 4 -order 3
```

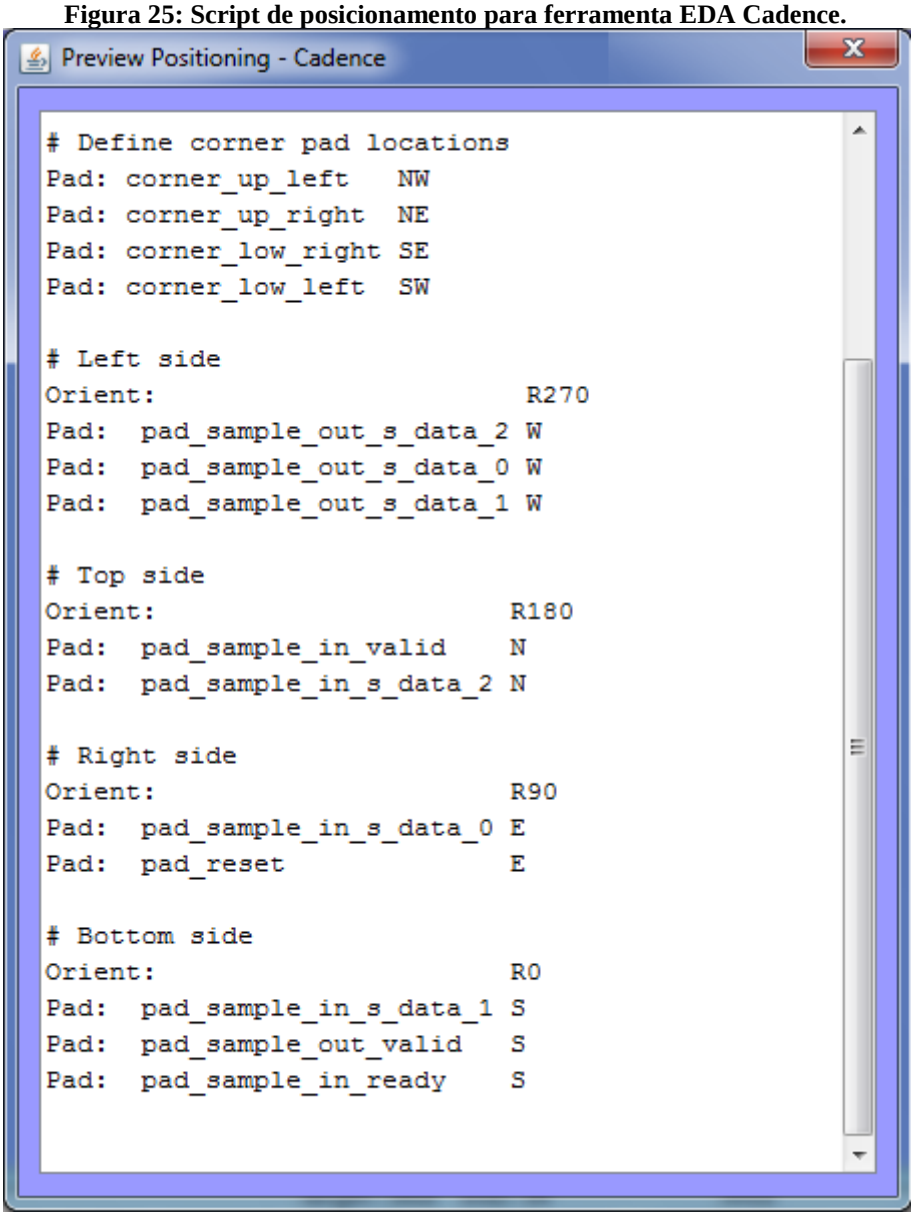
3.5.3. Criação de Regras para Script Cadence

Para validar a criação dos scripts para o projeto Digital Modulator, foi implementada uma classe com as regras de criação de script para ferramentas EDA fornecidas pela Cadence.

O processo de criação do script de mapeamento é idêntico ao processo de geração para ferramentas da Synopsys, descrita no tópico anterior (cap. 3.5.2).

No script de posicionamento são declarados os pads de alimentação e os corners pads, de acordo com os tipos definidos pela ferramenta EDA escolhida. Depois, cada pad é posicionado e sua localização é determinada pelo lado (West – W, North – N, East – E e South – S) e pela ordem (sentido south-north e east-west), veja Figura 25.

Figura 25: Script de posicionamento para ferramenta EDA Cadence.



3.6. Validação do PadTools com a Metodologia Proposta

A validação tem o propósito de avaliar o uso do PadsTool pelos desenvolvedores. Para definição da validação, foi utilizada a metodologia proposta por SEGUNDO, descrita no capítulo II.

3.6.1. Elaboração

Dois métodos de desenvolvimento de scripts para o mapeamento e posicionamento dos pads foram utilizados para o teste:

- A criação dos scripts manualmente de forma textual;
- A criação dos scripts automaticamente por meio de uma interface gráfica utilizando a ferramenta PadsTool;

O público alvo da ferramenta é desenvolvedor de circuitos integrados.

As seguintes hipóteses foram consideradas na fase:

- O uso do PadsTool facilitou a criação de scripts em comparação aos criados manualmente;
- O uso do PadsTool reduziu o tempo de desenvolvimento da criação de scripts em comparação as criados manualmente;
- O programadores preferiram o utilizar o PadsTool, em comparação ao desenvolvimento manual;

E também as hipóteses alternativas:

- O uso do PadsTool dificultou a criação de scripts em comparação as criados manualmente;
- O uso do PadsTool aumentou o tempo de desenvolvimento da criação de scripts em comparação as criados manualmente;
- O programadores preferiram o utilizar a criação manual dos scripts em comparação ao PadsTool;

Como parâmetros qualitativos foram utilizados:

- Facilidade de aprendizado: avaliar a facilidade dos participantes em aprender a utilizar as tecnologias avaliadas;
- Facilidade de uso: avaliar a facilidade dos participantes em utilizar as tecnologias avaliadas;
- Documentação: verificar se a documentação das tecnologias é suficiente para seu uso, ou se melhoras devem ser feitas;

Já os parâmetros quantitativos foram:

- Tempo de desenvolvimentos dos scripts gerados;
- Linhas de código geradas;

Três treinamentos foram elaborados para serem realizados com os participantes:

- O primeiro consistiu em treinar os participantes no desenvolvimento de circuitos integrados.
- O segundo treinamento abordou a criação de scripts manualmente, seguido do treinamento prático onde os participantes utilizaram os conhecimentos adquiridos para gerar os scripts.
- O terceiro treinamento abordou o PadsTool, e semelhante ao segundo foi seguido da prática dos participantes.

Para elaboração das práticas, foram definidos dois documentos de especificação de criação dos scripts para serem utilizados como referências pelos participantes na etapa de desenvolvimento. Esses documentos baseiam-se nos scripts de mapeamento e posicionamento do projeto de um modulador DPCM. Os documentos de mapeamento e posicionamento encontram-se nos anexos.

Foi criado um formulário para análise qualitativa das tecnologias. As perguntas abordadas foram:

1. Você já conhecia a lógica do desenvolvimento da ferramenta?
2. Como você avalia o seu aprendizado da tecnologia utilizada?
3. Como você avalia a prática de uso da tecnologia utilizada na criação dos scripts?
4. Quais os pontos positivos no uso da tecnologia?
5. Quais os pontos negativos no uso da tecnologia?
6. Comente sua experiência no desenvolvimento.

Também foi criado um questionário comparativo para avaliar qual das tecnologias mais agradou aos participantes. As questões abordadas foram:

1. Qual foi a melhor tecnologia utilizada na criação dos scripts?
2. Justifique sua resposta.

Capítulo IV: Resultados e Discussão

Nesse capítulo são relacionados os resultados.

4.1. Validação Funcional

Os scripts gerados pela ferramenta com o projeto BSMILC e com o projeto Digital Modulator foram comparados com os arquivos feitos manualmente e se mostraram idênticos. Os arquivos criados automaticamente foram substituídos pelos feitos manualmente no projeto e as atividades de Floorplanning foram realizadas sem erros.

Os arquivos gerados pela ferramenta para o projeto BSMILC estão contidos nos Apêndices (BSMILC_CHIP.v e BSMILC.tcl).

4.2. Comparação com a Criação Manual de Scripts

A Tabela 3 mostra a comparação entre a criação dos scripts de mapeamento e posicionamento através da ferramenta PadTools e a criação manual.

Tabela 3: Comparação do PadTools com a criação manual.

Características	PadsTool	Feito Manualmente
Corrigir pads alocados na mesma posição	X	-
Detecção de pads de alimentação não criados	X	-
Alerta de pads não posicionados	X	-
Detecção de pads não mapeados	X	-
Geração automática dos scripts para outras ferramentas EDA	X	-
Multiplataforma	X	X
Independência de Linguagem de Programação	-	X

Alguns possíveis erros encontrados na criação dos scripts como a não instanciação de algum pino, não posicionamento de algum pad ou alocação de mais de um pad no mesmo espaço foram identificados pela ferramenta, impedindo que o usuário gerasse script com falha. Esses erros são mais difíceis de serem encontrados e corrigidos com a criação manual.

Contudo a ferramenta proposta possui uma dependência da linguagem de programação, no caso Java que foi escolhida para a implementação da ferramenta, tendo em vista que a criação de novas regras para geração de scripts para outras ferramentas EDA depende da linguagem utilizada no desenvolvimento do software.

4.2. Análise dos Resultados da Validação com Usuário

Com todos os testes realizados, códigos coletados e questionários respondidos, foi possível iniciar a fase de análise dos resultados. O primeiro passo, como previsto na metodologia, é a análise dos códigos gerados pelos desenvolvedores, na primeira e na segunda fase.

A Tabela 4 apresenta os resultados médios do desenvolvimento.

Tabela 4: Resultados do desenvolvimento

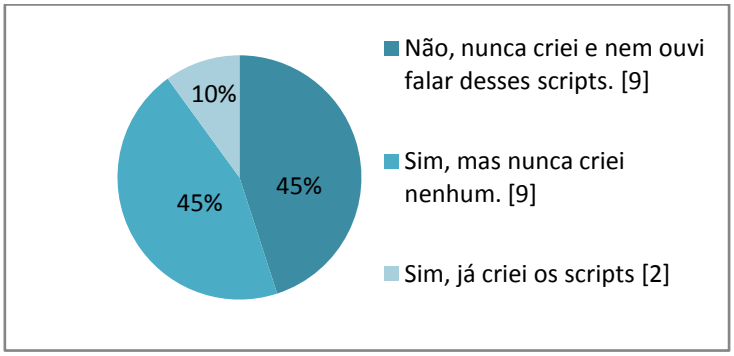
Parâmetro	Criação Manual	PadsTool
Tempo de desenvolvimento	01h 01min	17min
Número de linhas escritas	74 linhas	74 linhas
Porcentagem de criação dos scripts	100%	100%

Fazendo um comparativo entre as tecnologias, podemos perceber que no o uso do PadsTool permitiu que o grupo criasse os scripts em um menor tempo. A porcentagem de criação dos scripts foi a mesma, porém o uso da ferramenta PadsTool reduziu o tempo de desenvolvimento em 72%.

O próximo passo a ser realizado foi a análise dos pontos qualitativos avaliados pelos participantes. Com os dados foi possível avaliar a satisfação dos participantes em relação ao uso da ferramenta PadsTool e do uso da criação manual dos scripts.

A primeira informação retirada dos questionários foi que apenas dois participantes já haviam criados os scripts, e todos os outros desconheciam a lógica de criação ou conheciam, mas nunca os criou, segundo indicado na Figura 26.

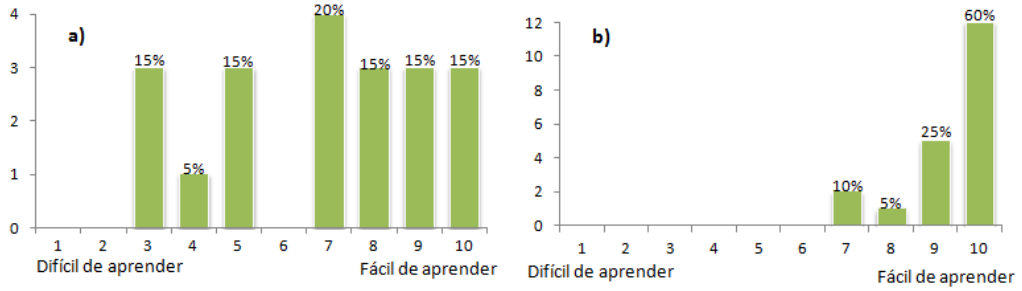
Figura 26: Conhecimento sobre a lógica de criação dos scripts.



A Figura 27 mostra gráficos com as avaliações das tecnologias em relação à facilidade de aprendizado. Segundo o gráfico de satisfação a criação manual obteve, em média, nota 6,85 enquanto que na criação manual a média foi de 9,35. Analisando a Figura 27 observa-se que apenas seis pessoas (equivalente a 30%) deram notas 9 ou 10 em relação a criação manual, enquanto que dezessete pessoas (85%) deram notas 9 ou 10 para a facilidade de criação dos scripts pela ferramenta PadsTool.

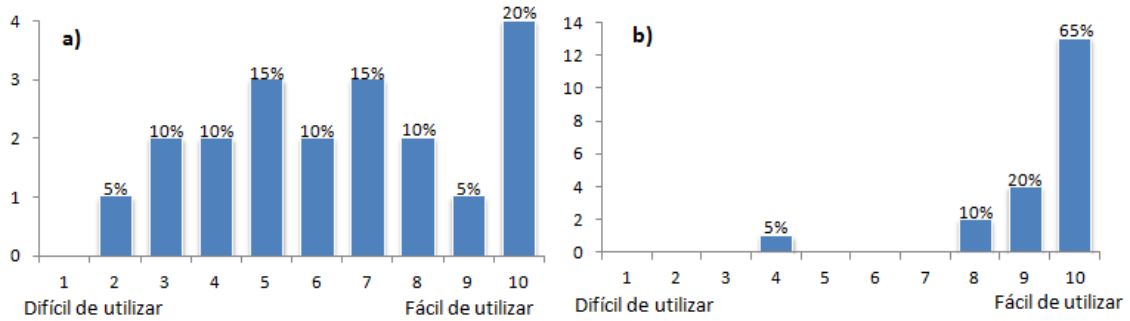
Podemos ver que a facilidade de aprendizado da ferramenta é melhor que para aprender a criar os scripts manualmente.

Figura 27: Notas dadas para facilidade de aprendizado: a) Criação Manual, b) Ferramenta PadsTool.



A Figura 28 mostra gráficos semelhantes, só que relacionados à facilidade de uso das tecnologias. De acordo com o gráfico a média das notas da criação manual foi 6,45 enquanto a ferramenta PadsTool obteve 9,3. Novamente podemos ver que a ferramenta mostrou-se mais fácil de ser utilizada.

Figura 28: Notas dadas para facilidade de uso: a) Criação Manual, b) Ferramenta PadsTool.



A última parte do questionário permitia que os participantes escrevessem, de forma aberta, sobre os pontos positivos e negativos das tecnologias e suas experiências durante o desenvolvimento.

Os principais problemas apontados sobre a ferramenta PadsTool foram relativos à interface gráfica. Alguns sentiram falta da enumeração dos pads quando posicionados no núcleo, outros acharam a interface pouco intuitiva, mas que se torna bastante simples após o primeiro uso efetivo.

Pontos positivos também foram destacados como: a praticidade, rapidez e a automatização na geração dos scripts, a não necessidade de muito conhecimento estruturado ou associação com a formalidade dos termos. Outro ponto destacado foi uma maior compreensão do processo de criação dos scripts, dado que a ferramenta tem uma simulação do posicionamento real. Outro ponto positivo é a resistência a erros, já que não será mais preciso digitar linha por linha cada posicionamento de cada componente. De acordo com um dos participantes: *"A ferramenta permite focar-se no design do CHIP, esquecendo-se dos detalhes de sintaxe dos scripts. Assim, é mais fácil criá-los sem erros."*.

Sobre a experiência do desenvolvimento utilizando a ferramenta PadsTool a maioria dos comentários ressaltaram a facilidade e rapidez na criação dos scripts. Muitos destacaram a grande utilidade da ferramenta no processo de desenvolvimento dos scripts.

Os pontos negativos do uso da criação dos scripts manualmente foi o processo ser muito repetitivo tornando a geração bastante lenta, cansativa e altamente propensa a erros. Alguns dos participantes indicaram que houve certa exaustão em repetir várias vezes a mesma sentença da sintaxe. Segundo um dos participantes: *"Toma muito tempo,*

é cansativo, grande chance de se cometer erros. Não há auxilio da ferramenta para solução de erros eventuais.”.

Pontos positivos do uso da criação manual dos scripts foi o melhor entendimento sobre o funcionamento do mapeamento e posicionamento dos pinos do circuito. Também foi destacado a não necessidade de um ambiente de desenvolvimento complexo, com apenas um editor de texto é possível criar os scripts manualmente.

Sobre a experiência no desenvolvimento usando a criação manual, o maior número de comentários foi sobre a demora na construção dos scripts. Além disso, a repetição e semelhança dos códigos torna difícil a visualização de erros.

De acordo com o questionário comparativo todos os participantes indicaram a ferramenta PadsTool melhor do que a criação manual dos scripts. De acordo com um dos participantes: *“A Ferramenta PadsTool facilita muito a criação de scripts, quando comparada à criação manual. Ao usar o PadsTool, nós sabemos o posicionamento real, quando na criação manual são definidos por números. Com a Ferramenta PadsTool também tem uma resistência a erros, já que é carregado um arquivo e é gerado automaticamente. Com a criação manual existe uma grande vulnerabilidade a erros na digitação.”*.

Capítulo V: Conclusão e Trabalhos Futuros

Apesar do Floorplanning não ser um tema novo na construção de um CI, ainda é muito importante o estudo sobre o processo na busca de melhorá-lo e deixá-lo mais simples para o usuário.

O presente trabalho apresenta uma possível solução para o problema com a criação manual de scripts para instanciação e posicionamento dos pads no núcleo. A ferramenta gráfica apresentada é capaz de proporcionar uma forma mais simples e intuitiva para o usuário, além de gerar os scripts automaticamente e evitar alguns erros na construção dos scripts.

O fato de a ferramenta ter capacidade de ser executada em diversas plataformas e ser capaz de geração de scripts para diversos fornecedores de ferramentas EDA torna-a bastante completa e reutilizável.

A ferramenta foi validada em dois projetos de CI que utilizaram ferramentas EDA de diferentes fornecedores (Synopsys e Cadence) e seus resultados foram testados e validados, bem como a identificação de possíveis falhas colocadas pelo usuário.

Comparado com a criação manual, a ferramenta PadTools mostrou-se mais eficiente em relação a detecção e correção de erros provocados na geração dos scripts de mapeamento e posicionamento.

Para validar o desempenho com o usuário, foi realizado um trabalho de validação. Essa validação trouxe pontos positivos na utilização da ferramenta PadTools em comparação com a criação manual de scripts.

De acordo com as análises feitas com os códigos e formulários, é possível dizer que todas as hipóteses levantadas foram confirmadas, e são elas:

- O uso do PadsTool facilitou a criação de scripts em comparação aos criados manualmente;
- O uso do PadsTool reduziu o tempo de desenvolvimento da criação de scripts em comparação as criados manualmente;
- O uso do PadsTool reduziu os erros inseridos na criação dos scripts em comparação as criados manualmente (Colocar média de erros inseridos);

Além disso, os projetistas preferiram o utilizar o PadsTool, em comparação ao desenvolvimento manual.

A confirmação da validade do trabalho aqui exposto é reforçada pela publicação do artigo intitulado "*Graphical tool for mapping and positioning of the Pads*" aceito para apresentação no WCAS 2013 (*Second Workshop on Circuits and Systems Design*), um dos eventos paralelos que ocorreu durante o *Chip in Brasília* (http://cecci.sbc.org.br/index.php?option=com_content&view=article&id=91) entre 30 de Agosto e 2 de Setembro de 2012 em Brasília – DF.

Para trabalhos futuros, pode ser adicionada a função de salvar e abrir o projeto no meio da criação, possibilitando o usuário salvar durante o processo e poder dar continuidade depois.

Além disso, pode ser implementado a leitura automática do arquivo que contém as informações dos pinos de entrada, saída, clock, reset e alimentação. Esses arquivos são fornecidos pela foundry e pode auxiliar o usuário com a listagem de todos os tipos de Pads, bem como as informações de tamanho dos Pads. Pode ser criada outras regra de geração de scripts para outros fornecedores de ferramentas EDA, como a Aliance e a Mentor Graphics.

Outra funcionalidade a ser inserida na ferramenta é implantar um algoritmo para melhor posicionamento dos Pads em torno do core. Isso, aliado a criação automática dos scripts, torna a ferramenta bem mais robusta e automatiza todo o processo de pré-posicionamento, feitos na etapa de Floorplanning.

Referências Bibliográficas

Avenier, J. P. 1979. circuits, Current aspects of CAD for integrated. 1979, Vols. IEE Publn.178, 43.

Brazil-Ip. 2012. BRAZIL IP Network. *Brazil-Ip.* [Online] 2012. [Cited: Junho 29, 2012.] http://www.brazilip.org.br/.

Chatterjee, Pallab K., Malhi, Satwinder D.S. and Mark G. 2002. 20 - Integrated Circuits. [book auth.] M. Middleton Wendy and E. Valkenburg Mac. *Reference Data for Engineers (Ninth Edition).* Woburn : s.n., 2002.

CLEARY, J. and WITTEN, I. 2006. Data compression using adaptive coding and partial string. *IEEE Transactions on Communications.* 2006, Vol. 32, 4.

Java Technology. 2012. Learn About Java Technology. *JAVA.* [Online] Oracle, 2012. [Cited: Maio 2, 2012.] http://www.java.com/en/about/.

Java Tutorials. 2012. Abstract Methods and Classes. *The Java Tutorials.* [Online] Oracle, 2012. [Cited: Junho 16, 2012.] http://docs.oracle.com/javase/tutorial/java/IandI/abstract.html.

Kommuru, H. B. and Mahmoodi, H. 2011. ASIC Design Flow Tutorial. [Online] 2011. [Cited: Junho 13, 2011.] http://userwww.sfsu.edu/~necrc/files/synopsys%20tutorials/ASIC%20Design%20Flow%20Tutorial.pdf.

Lira, Patrícia Freire. 2009. *IPPROCESS 3.0: INCLUSÃO DO FLUXO DE DESENVOLVIMENTO DE HARD IP-CORES.* 2009.

MARQUES, D. S., et al. 2011. A Biological Signals and Medical Images Lossless Compressor IP-Core Design. *Chip on the Cliffs, Proceedings of First Workshop on Circuits and Systems Design.* 2011.

McCluskey, G. 1998. Sun Developer Network. *Using Java Reflection.* [Online] Janeiro 1998. [Cited: Junho 13, 2012.] http://java.sun.com/developer/technicalArticles/ALT/Reflection/.

NetBeans. 2012. Informação da versão do NetBeans IDE 6.9.1. *NetBeans.* [Online] Oracle, 2012. [Cited: Julho 02, 2012.] http://netbeans.org/community/releases/69/index_pt_BR.html.

Richmond, M., Noble, J. 2001. Reflections on remote reflection. *Computer Science Conference.* 2001, doi: 10.1109/ACSC.2001.906638.

Sabharwal, C.L. **Java, Java, Java. 1998.** 3, s.l. : Potentials, IEEE, 1998, Vol. 17. doi: 10.1109/45.714612.

SEGUNDO, R. M. C. 2011. *Athus: um framework para desenvolvimento de jogos para TV Digital utilizando Ginga.* João Pessoa, 2011 : Dissertação (Mestrado em Informática) - UFPB, 2011.

Synopsys. 2012. Helping You Design the Chip Inside. *Synopsys.* [Online] SYNOPSYS, 2012. [Cited: Julho 02, 2012.] http://www.synopsys.com/Tools/Pages/default.aspx.

Vanzi, M. 1990. CAD Tools Evolution. *Solid-State Circuits Conference.* s.l. : ESSCIRC '90. Sixteenth European, 1990.

Glossário

BSMILC: Compressor Sem Perdas de Sinais Biológicos e Imagens Médicas (Biological Signals and Medical Images Lossless Compressor).

CI: Circuito Integrado.

CMOS: Tipo de tecnologia empregada na fabricação de circuitos integrados (Complementary Metal-Oxide-Semiconductor).

DH: Design Houses.

DPCM: Differential pulse-code modulation.

EDA: Electronic Design Automation.

Foundry: Empresa que produz o CI.

GUI: Interface Gráfica do Usuário (Graphical User Interface).

HDL: Linguagem de Descrição de Hardware (Linguagem de Descrição de Hardware).

RTL: descrição comportamental do módulo de um circuito digital (Register Transfer Level).

VHDL: linguagem usada para o design (projeto/concepção) de circuitos digitais (VHSIC Hardware Description Language).

Verilog: é uma linguagem de descrição de hardware usada para modelar sistemas eletrônicos.

Wrapper: Camada de abstração para uma classe ou objeto

Apêndices

BSMILC_CHIP.v

Arquivo de mapeamento gerado pela ferramenta PadsTool para utilização no projeto de CI, desenvolvido na UFPB, denominado BSMILC.

```
module BSMILC_CHIP( clk, BSMILC_Entrada_data_in, BSMILC_Entrada_control,
        BSMILC_Entrada_reset_n, BSMILC_Entrada_last_in, BSMILC_Entrada_valid,
        BSMILC_Entrada_ready, BSMILC_Saida_data_out, BSMILC_Saida_last_out,
        BSMILC_Saida_valid, BSMILC_Saida_ready );

  input [15:0] BSMILC_Entrada_data_in;
  input [3:0] BSMILC_Entrada_control;
  output [7:0] BSMILC_Saida_data_out;
  input clk, BSMILC_Entrada_reset_n, BSMILC_Entrada_last_in,
  BSMILC_Entrada_valid, BSMILC_Saida_ready;
  output BSMILC_Entrada_ready, BSMILC_Saida_last_out, BSMILC_Saida_valid;

  wire [15:0] wire_BSMILC_Entrada_data_in;
  wire [3:0] wire_BSMILC_Entrada_control;
  wire [7:0] wire_BSMILC_Saida_data_out;
  wire wire_clk, wire_BSMILC_Entrada_reset_n, wire_BSMILC_Entrada_last_in, wire_BSMILC_Entrada_valid,
wire_BSMILC_Saida_ready;
  wire  wire_BSMILC_Entrada_ready, wire_BSMILC_Saida_last_out, wire_BSMILC_Saida_valid;

  A_ICP pad_BSMILC_Entrada_data_in_0(.Y(wire_BSMILC_Entrada_data_in[0]), .PAD(BSMILC_Entrada_data_in[0]));
  A_ICP pad_BSMILC_Entrada_data_in_1(.Y(wire_BSMILC_Entrada_data_in[1]), .PAD(BSMILC_Entrada_data_in[1]));
  A_ICP pad_BSMILC_Entrada_data_in_2(.Y(wire_BSMILC_Entrada_data_in[2]), .PAD(BSMILC_Entrada_data_in[2]));
  A_ICP pad_BSMILC_Entrada_data_in_3(.Y(wire_BSMILC_Entrada_data_in[3]), .PAD(BSMILC_Entrada_data_in[3]));
  A_ICP pad_BSMILC_Entrada_data_in_4(.Y(wire_BSMILC_Entrada_data_in[4]), .PAD(BSMILC_Entrada_data_in[4]));
  A_ICP pad_BSMILC_Entrada_data_in_5(.Y(wire_BSMILC_Entrada_data_in[5]), .PAD(BSMILC_Entrada_data_in[5]));
  A_ICP pad_BSMILC_Entrada_data_in_6(.Y(wire_BSMILC_Entrada_data_in[6]), .PAD(BSMILC_Entrada_data_in[6]));
  A_ICP pad_BSMILC_Entrada_data_in_7(.Y(wire_BSMILC_Entrada_data_in[7]), .PAD(BSMILC_Entrada_data_in[7]));
  A_ICP pad_BSMILC_Entrada_data_in_8(.Y(wire_BSMILC_Entrada_data_in[8]), .PAD(BSMILC_Entrada_data_in[8]));
  A_ICP pad_BSMILC_Entrada_data_in_9(.Y(wire_BSMILC_Entrada_data_in[9]), .PAD(BSMILC_Entrada_data_in[9]));
  A_ICP pad_BSMILC_Entrada_data_in_10(.Y(wire_BSMILC_Entrada_data_in[10]), .PAD(BSMILC_Entrada_data_in[10]));
  A_ICP pad_BSMILC_Entrada_data_in_11(.Y(wire_BSMILC_Entrada_data_in[11]), .PAD(BSMILC_Entrada_data_in[11]));
  A_ICP pad_BSMILC_Entrada_data_in_12(.Y(wire_BSMILC_Entrada_data_in[12]), .PAD(BSMILC_Entrada_data_in[12]));
  A_ICP pad_BSMILC_Entrada_data_in_13(.Y(wire_BSMILC_Entrada_data_in[13]), .PAD(BSMILC_Entrada_data_in[13]));
  A_ICP pad_BSMILC_Entrada_data_in_14(.Y(wire_BSMILC_Entrada_data_in[14]), .PAD(BSMILC_Entrada_data_in[14]));
  A_ICP pad_BSMILC_Entrada_data_in_15(.Y(wire_BSMILC_Entrada_data_in[15]), .PAD(BSMILC_Entrada_data_in[15]));
  A_ICP pad_BSMILC_Entrada_control_0(.Y(wire_BSMILC_Entrada_control[0]), .PAD(BSMILC_Entrada_control[0]));
  A_ICP pad_BSMILC_Entrada_control_1(.Y(wire_BSMILC_Entrada_control[1]), .PAD(BSMILC_Entrada_control[1]));
  A_ICP pad_BSMILC_Entrada_control_2(.Y(wire_BSMILC_Entrada_control[2]), .PAD(BSMILC_Entrada_control[2]));
  A_ICP pad_BSMILC_Entrada_control_3(.Y(wire_BSMILC_Entrada_control[3]), .PAD(BSMILC_Entrada_control[3]));
```

```
    A_BT6NP pad_BSMILC_Saida_data_out_1(.A(wire_BSMILC_Saida_data_out[1]), .PAD(BSMILC_Saida_data_out[1]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_2(.A(wire_BSMILC_Saida_data_out[2]), .PAD(BSMILC_Saida_data_out[2]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_3(.A(wire_BSMILC_Saida_data_out[3]), .PAD(BSMILC_Saida_data_out[3]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_4(.A(wire_BSMILC_Saida_data_out[4]), .PAD(BSMILC_Saida_data_out[4]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_5(.A(wire_BSMILC_Saida_data_out[5]), .PAD(BSMILC_Saida_data_out[5]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_6(.A(wire_BSMILC_Saida_data_out[6]), .PAD(BSMILC_Saida_data_out[6]), .EN(0));
    A_BT6NP pad_BSMILC_Saida_data_out_7(.A(wire_BSMILC_Saida_data_out[7]), .PAD(BSMILC_Saida_data_out[7]), .EN(0));
    A_ICHDP pad_clk(.Y(wire_clk), .PAD(clk));
    A_ICHDP pad_BSMILC_Entrada_reset_n(.Y(wire_BSMILC_Entrada_reset_n), .PAD(BSMILC_Entrada_reset_n));
    A_ICP pad_BSMILC_Entrada_last_in(.Y(wire_BSMILC_Entrada_last_in), .PAD(BSMILC_Entrada_last_in));
    A_ICP pad_BSMILC_Entrada_valid(.Y(wire_BSMILC_Entrada_valid), .PAD(BSMILC_Entrada_valid));
    A_ICP pad_BSMILC_Saida_ready(.Y(wire_BSMILC_Saida_ready), .PAD(BSMILC_Saida_ready));
    A_BT6NP pad_BSMILC_Entrada_ready(.A(wire_BSMILC_Entrada_ready), .PAD(BSMILC_Entrada_ready), .EN(0));
    A_BT6NP pad_BSMILC_Saida_last_out(.A(wire_BSMILC_Saida_last_out), .PAD(BSMILC_Saida_last_out), .EN(0));
    A_BT6NP pad_BSMILC_Saida_valid(.A(wire_BSMILC_Saida_valid), .PAD(BSMILC_Saida_valid), .EN(0));

BSMILC bsmilc_i(.BSMILC_Entrada_data_in(wire_BSMILC_Entrada_data_in),
      .BSMILC_Entrada_control(wire_BSMILC_Entrada_control),
      .BSMILC_Saida_data_out(wire_BSMILC_Saida_data_out),
      .clk(wire_clk),
      .BSMILC_Entrada_reset_n(wire_BSMILC_Entrada_reset_n),
      .BSMILC_Entrada_last_in(wire_BSMILC_Entrada_last_in),
      .BSMILC_Entrada_valid(wire_BSMILC_Entrada_valid),
      .BSMILC_Saida_ready(wire_BSMILC_Saida_ready),
      .BSMILC_Entrada_ready(wire_BSMILC_Entrada_ready),
      .BSMILC_Saida_last_out(wire_BSMILC_Saida_last_out),
      .BSMILC_Saida_valid(wire_BSMILC_Saida_valid));
endmodule
```

BSMILC.tcl

Arquivo de posicionamento gerado pela ferramenta PadsTool para utilização no projeto de CI, desenvolvido na UFPB, denominado BSMILC.

```
# Create corners and P/G pads
create_cell {corner_low_left corner_low_right corner_up_left corner_up_right}  IOCORNCLMP
        create_cell {pad_vdd_left1 pad_vdd_right1 pad_vdd_top1 pad_vdd_bottom1 pad_vdd_left2 pad_vdd_right2 pad_vdd_top2
        pad_vdd_bottom2 pad_vdd_left3 pad_vdd_right3 pad_vdd_top3 pad_vdd_bottom3} VDDALLP

create_cell {pad_gnd_left1 pad_gnd_right1 pad_gnd_top1 pad_gnd_bottom1 pad_gnd_left2 pad_gnd_right2 pad_gnd_top2
         pad_gnd_bottom2 pad_gnd_left3 pad_gnd_right3 pad_gnd_top3 pad_gnd_bottom3} GNDALLP

# Define corner pad locations
set_pad_physical_constraints -pad_name "corner_up_left"   -side 1
set_pad_physical_constraints -pad_name "corner_up_right"  -side 2
set_pad_physical_constraints -pad_name "corner_low_right" -side 3
set_pad_physical_constraints -pad_name "corner_low_left"  -side 4

# Left side
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_3" -side 1 -order 1
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_4" -side 1 -order 2
set_pad_physical_constraints -pad_name "pad_vdd_left1" -side 1 -order 3
set_pad_physical_constraints -pad_name "pad_gnd_left1" -side 1 -order 4
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_5" -side 1 -order 5
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_6" -side 1 -order 6
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_7" -side 1 -order 7
set_pad_physical_constraints -pad_name "pad_vdd_left2" -side 1 -order 8
set_pad_physical_constraints -pad_name "pad_gnd_left2" -side 1 -order 9
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_8" -side 1 -order 10
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_9" -side 1 -order 11
set_pad_physical_constraints -pad_name "pad_vdd_left3" -side 1 -order 12
set_pad_physical_constraints -pad_name "pad_gnd_left3" -side 1 -order 13
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_10" -side 1 -order 14
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_11" -side 1 -order 15

# Top side
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_12" -side 2 -order 1
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_13" -side 2 -order 2
set_pad_physical_constraints -pad_name "pad_vdd_top1" -side 2 -order 3
set_pad_physical_constraints -pad_name "pad_gnd_top1" -side 2 -order 4
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_14" -side 2 -order 5
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_15" -side 2 -order 6
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_ready" -side 2 -order 7
set_pad_physical_constraints -pad_name "pad_vdd_top2" -side 2 -order 8
set_pad_physical_constraints -pad_name "pad_gnd_top2" -side 2 -order 9
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_valid" -side 2 -order 10
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_last_in" -side 2 -order 11
set_pad_physical_constraints -pad_name "pad_vdd_top3" -side 2 -order 12
set_pad_physical_constraints -pad_name "pad_gnd_top3" -side 2 -order 13
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_ready" -side 2 -order 14
set_pad_physical_constraints -pad_name "pad_clk" -side 2 -order 15
```

```
# Right side
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_valid" -side 3 -order 1
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_last_out" -side 3 -order 2
set_pad_physical_constraints -pad_name "pad_vdd_right1" -side 3 -order 3
set_pad_physical_constraints -pad_name "pad_gnd_right1" -side 3 -order 4
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_0" -side 3 -order 5
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_1" -side 3 -order 6
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_2" -side 3 -order 7
set_pad_physical_constraints -pad_name "pad_vdd_right2" -side 3 -order 8
set_pad_physical_constraints -pad_name "pad_gnd_right2" -side 3 -order 9
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_3" -side 3 -order 10
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_4" -side 3 -order 11
set_pad_physical_constraints -pad_name "pad_vdd_right3" -side 3 -order 12
set_pad_physical_constraints -pad_name "pad_gnd_right3" -side 3 -order 13
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_5" -side 3 -order 14
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_6" -side 3 -order 15

# Bottom side
set_pad_physical_constraints -pad_name "pad_BSMILC_Saida_data_out_7" -side 4 -order 1
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_reset_n" -side 4 -order 2
set_pad_physical_constraints -pad_name "pad_vdd_bottom1" -side 4 -order 3
set_pad_physical_constraints -pad_name "pad_gnd_bottom1" -side 4 -order 4
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_control_0" -side 4 -order 5
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_control_1" -side 4 -order 6
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_control_2" -side 4 -order 7
set_pad_physical_constraints -pad_name "pad_vdd_bottom2" -side 4 -order 8
set_pad_physical_constraints -pad_name "pad_gnd_bottom2" -side 4 -order 9
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_control_3" -side 4 -order 10
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_0" -side 4 -order 11
set_pad_physical_constraints -pad_name "pad_vdd_bottom3" -side 4 -order 12
set_pad_physical_constraints -pad_name "pad_gnd_bottom3" -side 4 -order 13
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_1" -side 4 -order 14
set_pad_physical_constraints -pad_name "pad_BSMILC_Entrada_data_in_2" -side 4 -order 15
```