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

Uma Ferramenta de Geração de Código VHDL a partir do Modelo de Atores utilizando o Framework Ptolemy

RAMON LEONN VICTOR MEDEIROS

JOÃO PESSOA-PB
AGOSTO – 2012

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

Uma Ferramenta de Geração de Código VHDL a partir do Modelo de Atores utilizando o Framework Ptolemy

RAMON LEONN VICTOR MEDEIROS

JOÃO PESSOA-PB
AGOSTO – 2012

RAMON LEONN VICTOR MEDEIROS

Uma Ferramenta de Geração de Código VHDL a partir do Modelo de Atores utilizando o Framework Ptolemy

DISSERTAÇÃO APRESENTADA AO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA, COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE EM INFORMÁTICA (SISTEMAS DE COMPUTAÇÃO).

Orientador: Prof. Dr. Alisson Vasconcelos Brito

JOÃO PESSOA-PB
AGOSTO – 2012

M488u Medeiros, Ramon Leonn Victor
Uma ferramenta de geração de Código VHDL a partir do Modelo de Atores utilizando o Framework Ptolemy / Ramon Leonn Victor Medeiros.--João Pessoa, 2012.
107f : il.
Orientador: Alisson Vasconcelos Brito
Dissertação (Mestrado) – UFPB/CCEN
1. Sistemas de Computação. 2. Ptolemy. 3. Geração de Código. 4. Modelo de Atores. 5. VHDL.

.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **RAMON LEONN VICTOR MEDEIROS**, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 24 de agosto de 2012.

Ao vigésimo quarto dia do mês de agosto do ano dois mil e doze, às dez horas, no auditório do CCEN - da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Sinais, Sistemas Digitais e Gráficos*", o Sr. **RAMON LEONN VICTOR MEDEIROS**. A comissão examinadora foi composta pelos professores doutores: ALISSON VASCONCELOS DE BRITO (PPGI-UFPB), Orientador e Presidente da Banca, JOSÉ ANTÔNIO GOMES DE LIMA (PPGI-UFPB), como examinador interno e MÁRCIO EDUARDO KREUTZ (UFRN) como examinador externo. Dando início aos trabalhos, o professor ALISSON VASCONCELOS DE BRITO, cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*Uma Ferramenta de Geração de Código VHDL a partir do Modelo de Atores utilizando o Framework Ptolemy*". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, eu, professora Liliane dos Santos Machado, vice-coordenadora deste programa, servindo de secretária, lavrei a presente ata que vai assinada por mim mesma e pelos membros da Banca Examinadora. João Pessoa, 24 de agosto de 2012.

Liliane dos Santos Machado

Prof. Dr. ALISSON VASCONCELOS DE BRITO
Orientador (PPGI-UFPB)

Prof. Dr. JOSÉ ANTÔNIO GOMES DE LIMA
Examinador Interno (PPGI-UFPB)

Prof. Dr. MÁRCIO EDUARDO KREUTZ
Examinador Externo (UFRN)

Agradecimentos

Primeiramente, agradeço a Deus por estar ao meu lado, guiando todos os meus passos, abrindo tantas portas e colocando pessoas maravilhosas no meu caminho.

À minhas mães - Luciene Victo Belarmino, mãe, e Felícia Fernandes Belarmino, avó-; À minhas tias - Cleide, Célia, Ninha e Tetê –; Enfim, a toda minha família agradeço todo o amor, dedicação e confiança. Por terem me mostrado o melhor caminho e proporcionado condições para que eu o seguisse.

Aos professores José Antônio, Alexandre Duarte, Tatiana, Antonio Carlos e Alisson Brito, por terem compartilhado seu conhecimento e terem compreendido e ajudado em momentos especiais.

Aos amigos, André Lucena, Judson Mesquisa, Carlão e José Claúdio pela parceria e os momentos de alegria e ‘aperreio’.

Enfim, todos que colaboraram direta ou indiretamente para a concretização deste sonho.

Para vocês, ofereço esta página. De coração, obrigado a todos!

Resumo

Na área de Microeletrônica uma das maiores dificuldades trata-se de encontrar profissionais capacitados. Para a formação destes há uma demanda por recursos – humanos e materiais - que, muitas vezes, tem alto custo. Assim, se faz necessária uma solução tão acessível a instituições e pesquisadores, quanto adaptável às suas necessidades, propiciando escalabilidade e disseminação à formação, sem deixar de focar a qualidade. A partir do levantamento bibliográfico, baseando-se e apresentando o framework Ptolemy, desenvolvido em Java e que possibilita a simulação de sistemas concorrentes e heterogêneos, e no modelo de atores foram direcionados os esforços para o desenvolvimento de uma ferramenta de geração de código *open-source* e multiplataforma. Sendo assim, o objetivo deste trabalho é o desenvolvimento de uma ferramenta de geração de código VHDL baseada em Atores utilizando o Framework Ptolemy. Onde, além de projetar e simular graficamente há um diferencial na possibilidade de modelagem em níveis diferentes de abstração e geração de código VHDL de forma hierárquica. Apresentado o processo de desenvolvimento da ferramenta e resultados de sua utilização, bem como as contribuições.

Palavras-chave: Ptolemy, Geração de Código, Modelo de Atores.

Abstract

In the area of Microelectronics, one of the most difficult it is to find qualified professionals. For the training of these there is a demand for resources - human and material - which most of the time have high cost. Thus, a solution is needed as accessible to institutions and researchers, as adaptable to your needs, providing scalability and dissemination to training, without focusing on quality. From the literature, based and presenting the Ptolemy framework, developed in Java and that allows the simulation of concurrent and heterogeneous systems, and in the Actor Model were directed efforts for the development of a tool for code generation open-source and multiplatform. Thus, the aim of this work is to develop a tool to generate VHDL code based on Ptolemy Actors using the Ptolemy framework. Where, in addition to design and simulate graphically there is a difference in the possibility of modeling at different levels of abstraction and generation of the VHDL code in a hierarchical manner, presenting the process of development of the tool and the results of its use, as well as contributions.

Keywords *— Ptolemy, Actor-based models, Code Generator*

Lista de Ilustrações

Lista de Códigos Fonte

Sumário

1 Introdução

A necessidade de qualificação de capital humano na área de Microeletrônica foi preponderante para a escolha do tema deste trabalho. Uma demanda por soluções que propiciem escalabilidade e disseminação, sem deixar de focar a qualidade na formação do capital humano.

Percebendo esta demanda, escolheu-se o desenvolvimento de uma ferramenta de geração de código de Linguagem de Descrição de Hardware (HDL) para ensino de Microeletrônica utilizando o *Framework* Ptolemy.

Esse *framework* sendo um projeto criado e mantido na Universidade de Berkeley apresenta um suporte a modelagem, projeto e simulação de quaisquer sistemas baseados em atores, focando na concorrência, na execução em tempo real e em sistemas embarcados. Ele é um *framework* de software escrito na linguagem Java com uma interface gráfica chamada Vergil. Seu principal diferencial em relação a outras ferramentas é o suporte a modelos heterogêneos (SCHOEBERL, BROOKS, LEE, 2009).

Assim, este trabalho propõe o desenvolvimento de uma ferramenta de geração de código VHDL que auxilie o ensino de Microeletrônica, baseada no Modelo de Atores e desenvolvida a partir do *Framework* Ptolemy.

1.1 Motivação

No processo de ensino-aprendizagem em desenvolvimento de sistemas digitais, atualmente, o primeiro contato do aluno é com a linguagem HDL. Sob o ponto de vista técnico, foram introduzidos desde as primeiras aulas da disciplina, o uso de Linguagens de Descrição de Hardware (HDL) e o de ferramentas de automação de projetos (SOBREIRA *et al*, 2007). Fato que, se analisado com bastante atenção, evidência a carência/ausência de um contato introdutório, onde o estudante vivencie – de forma gráfica – o sistema e seja capaz de simular seus modelos.

Nessa perspectiva, Kolb apud Murari (2008) defende que as pessoas aprendem por meio de suas interações com o ambiente e das escolhas envolvidas nessas interações que, consequentemente, cada uma desenvolve características próprias de pensamento, adotando diferentes estilos de decisão e resolução de problemas como respostas aos desafios impostos pelo ambiente.

Seria muito importante se fosse possível aprender Microeletrônica se utilizando de recursos gráficos, sem, a princípio, saber qualquer linguagem de programação, apenas identificando e inserindo componentes e os conectando. O Brasil é um país com grande diversidade regional, cultural e com grandes desigualdades sociais; portanto, não é possível pensar em um modelo único para incorporação de recursos tecnológicos na educação. É necessário pensar em propostas que atendam aos interesses e necessidades de cada região ou comunidade (BRASIL, 1998).

Há demanda por uma ferramenta de ensino que seja portável, de fácil instalação, acesso e manutenção, capaz de funcionar nos mais variados Sistemas Operacionais. Na verdade, uma solução tão acessível a instituições e pesquisadores, quanto adaptável à suas necessidades, possibilitando simular o modelo criado graficamente e, após a conclusão, programar um dispositivo para funcionar de acordo com o modelo criado e simulado em software.

Por fim, poder ampliar o poder de funcionamento da ferramenta colaborando com seu desenvolvimento, propondo atualizações, novas funcionalidades e correções para possíveis erros, por ser uma ferramenta *open-source.*

1.2 Objetivos

Este trabalho visa desenvolver uma ferramenta para a geração de código em linguagem de descrição de hardware (HDL), possibilitando uma visualização gráfica e simulação a partir da utilização de modelo de atores.

1.2.1 Objetivos Específicos

- Utilizar o Framework Ptolemy para criação e simulação de modelos de Circuitos Integrados;
- Criar bibliotecas de atores no Ptolemy que sejam condizentes com as bibliotecas do Alliance CAD System;
- Desenvolver uma ferramenta que traduza o modelo do Ptolemy para Linguagem de Descrição de Hardware condizente com o Alliance CAD System;
- Validar o modelo utilizando o Alliance CAD System.

1.3 Metodologia

Para a Bertasi (2002) a característica de obter precisamente qual modelo que será abstraído permite capturar precisamente o comportamento do sistema, governado por certo modelo computacional particular. Tal afirmação justifica a possibilidade de utilizar o Ptolemy e sua capacidade para modelar, simular e ensinar circuitos eletrônicos.

A ferramenta proposta neste trabalho pretende facilitar o ensino de Microeletrônica a partir da facilidade de modelar e simular graficamente o circuito desejado. Para isso está sendo utilizado o framework Ptolemy, cujo é desenvolvido em Linguagem Java e utiliza, como já foi dito, o modelo de atores, além de possibilitar a simulação concorrente de modelos heterogêneos. O Fato de ser um *framework open-source* faz do Ptolemy um ótimo candidato para o que estamos nos propondo a fazer: desenvolver uma ferramenta de geração de código para ensino de Microeletrônica, que seja de baixo custo e acessível as instituições, pesquisadores e estudantes que precisem de subsídio para seus estudos.

Inicialmente é necessário conhecer os atores disponíveis e os modelos de computação (MoCs), analisando seus comportamentos. Diante da inexistência de atores, o Ptolemy permite a criação destes a partir da utilização da linguagem Java e do conhecimento da dinâmica do *framework.* Tendo os atores e o diretor necessário, é possível modelar e simular o circuito desejado.

Obtendo êxito na simulação do circuito estudado é chegada a hora de gerar o código em linguagem de descrição de *hardware*, no caso desta ferramenta, a linguagem *VHSIC Hardware Description Language* (VHDL). Para isso, a ferramenta mapeia as entidades – atores – do modelo construído no Ptolemy para itens da biblioteca de *standard cells* utilizada.

Neste momento entra em ação o Alliance CAD System, conjunto de ferramentas de desenvolvimento de circuitos integrados, onde um dos recursos é a SXLib, biblioteca de *standard cells*. Esta, a princípio foi utilizada como fonte de componentes a serem relacionados aos atores dos modelos simulados.

A geração de código em linguagem VHDL acontece através de um gerador de código incorporado ao Ptolemy, sendo assim, também desenvolvido em Linguagem Java. Este gerador de código identifica os atores e busca, no diretório previamente indicado, por arquivos VHDL correspondente.

Assim, para utilizar a ferramenta pode-se pontuar a seguinte sequência:

- Utilização dos atores do Ptolemy
  - Desenvolvimento dos atores caso não existam os necessários
- Escolha do Diretor (modelo de computação – MoC)
- Modelagem e Simulação no Ptolemy
- Geração de Código VHDL
- Validação com Alliance CAD System

Vale ressaltar que a partir do desenvolvimento de novos atores, quando necessários, permite o desenvolvimento incremental de novos atores possibilitando a ampliação dos circuitos objetos de estudo.

1.4 Organização do Trabalho

Este trabalho se apresenta com a seguinte estrutura:

Capítulo 2 apresenta o *framework* Ptolemy com seu potencial. Levantando assim, bases para a efetiva construção da ferramenta proposta, elencando trabalhos relacionados.

O Capítulo 3 traz uma introdução à Síntese de Alto Nível, aborda o fluxo de projeto e contextualiza o Alliance CAD System neste cenário. Neste é possível conhecer os níveis de abstração para a concepção de circuitos integrados e as ferramentas necessárias, entre outros.

Por fim, os Capítulos 4 e 5 apresentam a ferramenta proposta neste trabalho. O Capítulo 4 enfoca a estrutura da ferramenta, as etapas para a geração de código VHDL. Enquanto o Capítulo 5 mostra a prova de conceito, os resultados da aplicação da ferramenta. As considerações finais, contribuições e trabalhos futuros, são aplicados no Capítulo 6.

2 O Ptolemy e o Modelo de Atores

O Ptolemy segundo Filiba, Leung e Nagpal (2008) é um *framework* desenvolvido em Java, mantido na Universidade de Berkeley que apresenta a possibilidade de modelagem, projeto e simulação de quaisquer sistemas baseados em atores, focando na concorrência, na execução em tempo real e com suporte a modelos heterogêneos, através de representações gráficas.

Consiste basicamente em pacotes Java. Alguns dos pacotes são:

- *Kernel*: sintaxe (entidades com portas e interconexões)
- *Data*: dados transportados entre atores
- *Actor*: semântica de execução
- *Domains*: Modelos de Computação
- *Vergil*: ambiente de edição visual
- *Moml*: persistência no formato de arquivo XML

Além da vantagem de possuir a interface gráfica, denominada *Vergil,* o que possibilita a interação visual com o modelo estudado e a capacidade de simulação heterogênea e concorrente de alto nível, que chamam atenção para as possibilidades que o Ptolemy fornece, este framework é *open-source*, facilitando o seu estudo, uso e disseminação. Possibilitando, ainda, a geração de código em Linguagem Java ou C – e a implementação da geração de código em outras linguagens – dos modelos estudados, fato que é objeto de estudo neste trabalho.

O Ptolemy II possibilita a modelagem de sistemas embarcados, particularmente aqueles que mesclam tecnologias (SCHOEBERL, BROOKS, LEE, 2010). Tais como dispositivos eletrônicos analógicos e digitais, eletromecânicos, hardwares dedicados e com complexidade nos vários níveis de abstração.

Existem muitas maneiras de utilizar o Ptolemy II. Ele pode ser utilizado como uma estrutura para montagem de componentes de software, como modelagem e ferramenta de simulação, como um editor de diagrama de blocos, como uma aplicação de nível de sistema de prototipagem rápida, como um kit de ferramentas baseado em design apoiando à investigação de componentes, ou como um conjunto de ferramentas para construção de aplicações em Java. Esse

trabalho apresenta sua utilização como ferramenta de modelagem e simulação, se utilizando de sua capacidade de heterogeneidade e simulação concorrente.

Inicialmente, há algumas dificuldades no manuseio do framework, logo superadas pelas vantagens oferecidas. Tais como:

- Um grande conjunto de mecanismos de interação entre os domínios heterogêneos, forçando os desenvolvedores de componentes a pensarem no padrão como os outros componentes do domínio estão interagindo;
- Seus componentes são polimorfos, no campo do domínio, isso significa que eles podem interagir com outros componentes mesmo esses sendo de domínios diferentes;
- Possui ferramentas para a simulação dos modelos, permitindo a execução das mesmas como aplicações locais ou *web*, podendo ser utilizadas para apresentações à distância. É importante ressaltar tal ponto, em virtude da crescente utilização de ferramentas virtuais para ensino.
- Possui uma linguagem própria de marcação (*Markup Language*), baseada em XML (denominada MoML), além de possuir uma ferramenta visual (*Vergil*), que facilita a realização e o entendimento de modelos mais complexos;
- Seu código é aberto e gratuito, além de possuir milhares de usuários em todo o mundo.

A seção a seguir apresenta mais detalhes acerca do Modelo de Atores e o funcionamento do Ptolemy II sobre esta perspectiva.

2.1 Projeto e modelagem orientados a Atores

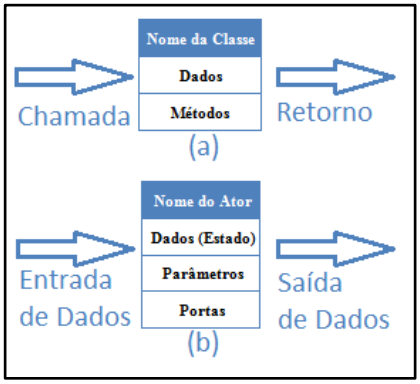
Ptolemy utiliza o papradigma orientado a atores para modelagem dos sistemas, onde o bloco básico de construção de um sistema é um ator (FILIBA, LEUNG E NAGPAL, 2006). Atores são componentes concorrentes (simultâneos) que se comunicam através de interfaces chamadas portas. Relações definem as interconexões entre essas portas, e assim, a estrutura de comunicação entre os atores.

Para Brito (2009):

> Um Modelo de Atores é um modelo matemático de computação concorrente que trata "atores" como sendo os elementos primitivos da computação concorrente digital. Em resposta a uma mensagem recebida por um ator, ele toma uma decisão local, cria mais atores, envia mais mensagens e determina como responder à próxima mensagem recebida.

Orientação a ator é uma maneira de estruturar e conceituar um sistema que complementa as técnicas orientadas a objetos que hoje são amplamente utilizadas para estruturar sistemas de software. Visualizando um sistema como uma estrutura de atores destaca sua estrutura causal e suas atividades simultâneas, juntamente com a sua comunicação e dependências de dados. Uma perspectiva orientada a objetos vê o estado de um sistema como uma estrutura de objetos que estão relacionados por referências um ao outro. A Figura 1 mostra a diferença na estrutura entre estes dois tipos.

Figura 1 Diferença entre o modelo orientado a Objetos (a) e o modelo orientado a Atores (b)



Fonte: Autor

Quando se utiliza a orientação a objetos, onde os objetos são construídos a partir de classes que possuem dados (atributos) e métodos. A comunicação acontece apenas chamando os métodos uns dos outros, assim, efetivamente, cada objeto transfere o controle de seus códigos.

Já a visão de um sistema só a perspectiva da orientação à atores é uma dissociação da transmissão de dados a partir do controle de transferência, onde cada ator tem seus dados (estado) e tanto o transfere para o próximo ator atravez de sua porta de saída, quanto recebe um novo dado de outro ator através da sua porta de entrada, seguindo um comportamento guiado por parâmetros.

Desta forma, para facilitar, pode-se afirmar que na orientação a atores se adota a filosofia tudo é um ator, onde cada entidade (ator) pode agir de maneira concorrente, sendo gerenciado por um MoC (*Model of Computation*). Um ator é uma entidade computacional assíncrona que, em resposta a uma mensagem que ele receba, pode simultaneamente: emitir um número finito de mensagens a outros atores, criar um número finito de novos atores, designar o comportamento a ser usado para a mensagem seguinte que recebe.

No projeto Ptolemy II, onde os princípios a orientação a atores é a base do desenvolvimento, apresenta-se os seguintes componentes:

- **Diretor:** componente que controla a execução do *workflow*, gerenciando outros componentes (atores, portas etc.). Define o modelo computacional a ser utilizado (síncrono, paralelo, distribuído, etc.), sendo obrigatória a sua presença, também denominado domínio.
- **Ator:** componente do *workflow* que representa um dado ou serviço. Podem ser conectados com outros atores por meio de portas e ter parâmetros configuráveis.
- **Porta:** cada ator pode conter uma ou mais portas, utilizadas no consumo e na produção de dados assim como na comunicação com outros atores envolvidos no *workflow*. Atores são conectados entre si por meio de portas. A conexão que representa o fluxo de dados entre um ator e outro é chamada de canal. As portas podem ser divididas em três tipos diferentes:
    - **Entrada:** usada para consumo de dados;
    - **Saída:** destino dos dados produzidos pelo ator;
    - **Entrada/Saída:** para consumo e saída de dados.

Além disso, as portas podem ser configuradas como simples ou múltiplas (multiportas). Uma porta de entrada simples pode estar conectada a um único canal. Já uma porta de entrada múltipla pode estar conectada a vários canais simultaneamente.

- **Relação (*Relations*):** permitem replicar fluxos de dados. Assim, o mesmo dado pode ser mandado para vários lugares do *workflow*.
- **Parâmetro:** são valores configuráveis que podem fazer parte de um *workflow*, diretor ou ator.

Quando Lee et. Al (2003) descreve o framework Ptolemy, afirma que

> "Este é baseado no Modelo de Atores. Este que consiste na definição de entidades denominadas "atores", que por sua vez processam os dados presentes nas suas portas de entrada - ou os criam - e enviam dados para outras entidades por meio de suas portas de saída. Estas portas contidas nos atores podem ser usadas para entrada, saída ou ambos. As portas podem ter um número variável de canais que podem ser conectados aos canais de outros portos. Todos os dados enviados através de portas são encapsulados por *tokens*, dos quais existem vários tipos".

A semântica exata de comunicação é determinada pelo domínio em que um sistema executa. Por exemplo, no domínio CSP (*Communicating Sequential Processes*), cada ator executa em uma *thread* separada, e a comunicação é feita através de encontros (um ator enviando dados a uma porta deve bloquear até que outro ator tenta receber dados da porta, e um ator recebendo dados de uma porta deve bloquear até que outro ator tenta enviar dados para a porta, momento em que os dados podem ser trocados). Outro domínio, que é o foco deste projeto, é o *Synchronous Dataflow* (SDF), em que todos os atores podem ser executados seqüencialmente de acordo com uma programação estática.

Atores também podem ter parâmetros, o que pode ser configurado em tempo de execução para mais flexibilidade. Parâmetros podem ser consultados para e com os valores de *token*. Uma vez que os parâmetros são campos públicos de atores, que podem ser consultadas e definir a partir de fora da classe ator.

Um sistema executável em Ptolemy consiste em instâncias de atores que estão contidos em uma instância de um ator composto. Um ator composto contém um diretor, que invoca os seguintes métodos de execução para cada um dos atores:

- **preinitialize**( ), que é invocado exatamente uma vez antes da simulação começa.
- **initialize**( ), que é invocado exatamente uma vez após o método *preinitialize()* é chamado.
- **prefire**( ), que é chamada uma vez por iteração.

- **fire**( ), que é chamado várias vezes por iteração, depois que o método *prefire()* é chamado.
- **postfire**( ), que é chamada uma vez por iteração, após a última chamada do *prefire()*.
- **wrapup**( ), que é invocado exatamente uma vez após a simulação começa.

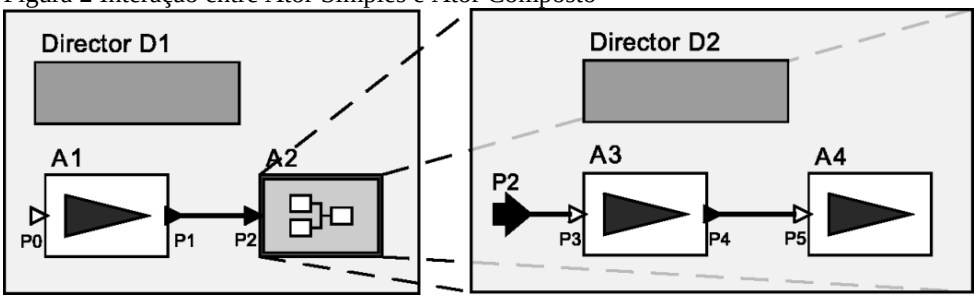
Para Lee et. al. (2005), cada ator consiste em um código fonte escrito em Java e compilado no arquivo *makefile* do seu diretório. Assim, ao criar um novo ator, é necessário acrescentar o seu nome para o *makefile* local.

2.1.1 Os atores

O ator sendo o componente básico de funcionamento para o Ptolemy pode ser de dois tipos: *AtomicActor* e *CompositeActor* (LEE, 2003). Atores que pertencem à classe *AtomicActor* representam entidades simples de processamento de dados, a parte mais baixa da hierarquia, enquanto que atores do tipo *CompositeActor*, como mostra a Figura 2, são entidades compostas por vários outros atores, podendo estes ser compostos ou não.

Como já foi dito, atores têm portas, que são suas interfaces de comunicação. A porta pode ser uma porta de entrada, de saída, ou ambos, onde são estabelecidos canais de comunicação para conectar portas. A porta para um ator composto pode ter ambas as conexões para o interior e para o exterior, ligando assim a comunicação dos atores de dentro para os atores de fora.

Figura 2 Interação entre Ator Simples e Ator Composto



Fonte: Lee, 2003.

Na Figura 2, o ator composto de alto nível (lado esquerdo da Figura) contém atores A1 e A2. A2 também é um ator composto, como sugerido pelo ícone diferente, que contém atores A3 e A4 (lado direito da Figura). Atores A1, A3 e A4 são atores do tipo *AtomicActor*, ou seja, atores simples. A porta P2 da A2 é uma porta externa e se conecta a porta do P1 por fora e a porta P3 por dentro. Note-se que o ator composto de alto nível não tem portas externas, implicando que encapsula totalmente o projeto.

Atores, tanto simples quantos compostos, são executáveis. A execução de um ator de composto é uma execução controlada dos atores que ele contém. Este composicionalidade de execução é a chave para gerir a heterogeneidade hierarquicamente.

2.1.2 Domínio

No Ptolemy, o domínio é a implementação de um MoC específico associado a um ator composto. Define a organização da comunicação e a ordem de execução entre os atores.

O mecanismo de comunicação é implementado utilizando receptores que estão contidos nas portas de entradas dos atores, onde há um receptor para cada canal de comunicação.

No modelo da Figura 2, há um receptor nas portas de entrada P2, P3 e P5. Porta P0 não tem receptor, porque não é ligado a um canal de entrada. Atores, quando residente em um domínio, adquirem receptores específicos do domínio. Desta forma, ao separar computação e comunicação, muitos atores (chamado atores de domínio polimórfico) podem ser reutilizados em diferentes domínios.

A ordem de execução dos atores contidos em um ator composto é controlada por um diretor. Desde que os receptores e os diretores trabalhem juntos, o diretor também é responsável pela criação de receptores. Quando um ator composto é executado, o diretor executa os atores contidos neste.

A Figura 2 mostra um modelo hierárquico utilizando dois domínios diferentes. No modelo, o ator composto de alto nível contém um D1, diretor e A2 é um ator composto com diretor D2. Assim, o diretor de D1 controla a ordem de execução de atores A1 e A2, já o diretor D2 controla a execução de A3 e A4 sempre que A2 é executado. O receptor na porta P1, criado pelo diretor D1, mediando a comunicação entre as portas P1 e P2. Da mesma forma, o diretor D2 para mediar a comunicação entre as portas P2 e P3, e as portas P4 e P5, criou receptores em P3 e P5, respectivamente.

Vários dominios já foram e, outros, estão sendo criados. Vejamos alguns a seguir:

- Domínios que apresentam certo grau de maturidade:
    - Continuous: continuous-time modeling
    - DDF: dynamic dataflow

  - DE: discrete-event modeling
  - Modal: finite state machines and modal model
  - PN: process networks with asynchronous message passing
  - Rendezvous: process networks with synchronous message passing
  - SDF: synchronous dataflow
  - SR: synchronous reactive
  - Wireless: wireless
- Domínios em fase experimental:
  - CI: component interaction (push/pull)
  - DDE: distributed discrete events
  - DT: discrete time
  - Giotto: periodic time-driven
  - GR: 3-D graphics
  - HDF: heterogeneous dataflow
  - PSDF: parameterized synchronous dataflow
  - TM: timed multitasking

O domínio mais maduro e objeto de estudo deste trabalho é Synchronous Dataflow (SDF), muito utilizado para processamento de sinais e aplicações multimídia. Quando um ator é executado neste modelo, ele consome um número fixo de tokens de cada porta de entrada e produz um número fixo de tokens para cada porta de saída. Uma propriedade importante de modelos SDF é que deadlocks e boundedness podem ser estaticamente analisados. Receptores neste domínio representam filas FIFO com capacidade fixa finita, e a ordem de execução de componentes é programada estaticamente.

2.2 Geração de Código

A construção do suporte à geração de código no Ptolemy II (PTII) tem como objetivo automatizar o processo de segmentação em modelos de simulação para plataformas de aplicação real (WILLIAMSON, 1998). O *framework* de geração de código do PTII tem uma abordagem *helper-based*, em que um conjunto de componentes auxiliares é usados para gerar o código para os atores PTII. Cada *helper* tem uma correspondência *one-to-one* para um ator. Como uma

separação lógica entre os seus domínios de relevância, *helpers* são componentes de geração de código, enquanto os atores são componentes de simulação.

A lógica por trás de um *framework helper-based* é baseada em separação de interesses para reduzir generalidade. Isto significa ter um *kernel* leve e uma biblioteca de classes auxiliares que se minimiza a complexidade. A arquitetura permite o desenvolvimento incremental e rápido, porque as classes auxiliares contêm lógicas simples e blocos de código que são mutuamente exclusivos. Como outra teoria do *framework* auxiliar, um conjunto diferente de *helpers* implica a geração de código de uma linguagem algo diferente. Assim é possivel adicionar uma nova linguagem à geração de código, diferente das linguagens suportadas atualmente, as lingugens C e Java.

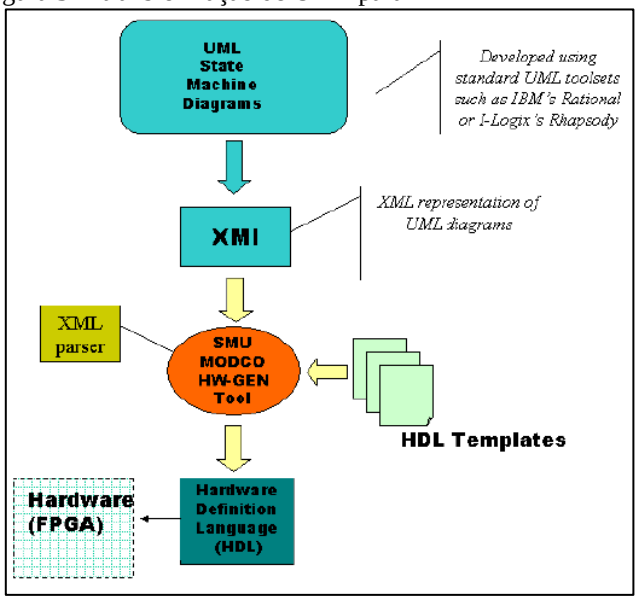
2.3 Trabalhos relacionados

Os seguintes trabalhos foram estudados com a finalidade de se obter subsidio para o desenvolvimento da ferramenta proposta neste trabalho.

Brito (2009) apresenta o desenvolvimento de uma ferramenta para simulação de sistemas concorrentes. A utilização do Ptolemy e o modelo de atores são similaridades com o trabalho aqui proposto, no entanto o autor não propõe a utilização de bibliotecas de *standard cells* nem a geração de código VHDL. Um ponto interessante, e aplicável a este trabalho, é a metodologia baseada na escolha de níveis de abstração, desenvolvimento e testes de atores para simulação, bem como a ideia da implementação de roteiros de estudo.

O *SMU Co-Design Project* (COYLE E THORNTON, 2005), propõe a ferramenta MODCO, uma ferramenta de transformação que parte do *Unified Modelling Language* (UML) e gera código HDL para ser utilizado em *Field Programmable Gate Arrays* (FPGA).

Figura 3 A transformação de UML para HDL



Fonte: Coyle e Thornton, 2005.

O destaque da ferramenta MODCO é utilização de um *parser*, como mostra a Figura 3, para interpretar *o XML Metadata Interchange* (XMI) – arquivo XML que representa o diagrama UML – e, assim, mapear os componentes UML para *HDL Templates* pré-definidos. Esta é a principal contribuição para a ferramenta proposta neste trabalho.

Para geração de código VHDL utilizando o Ptolemy, foi desenvolvido o Projeto PeaCE (Ptolemy Extension as Codesign Environment) proposto por Ha e Oh (1998) e Sung (1997). É um ambiente para desenvolvimento rápido de sistemas heterogêneos digitais, bem como projetos de sistema-em-chip (*System-on-Chip* – *SoC*) para projeto de sistemas distribuídos heterogêneos. Diferencia da ferramenta proposta neste trabalho opor ser um ambiente de *codesign*, ou seja, se propõe a simular e gerar sistemas em *hardware* e *software*, gerando assim, código em linguagem VHDL e C. A contribuição do projeto PeaCE seria a utilização do domínio *Synchronous Data Flow* (SDF) para a simulação e a ideia de testar em FPGA após a conclusão das devidas validações.

Já Filiba, Leung e Nagpal (2008) propõe a geração de código VDHL a partir da utilização do Ptolemy. Este traz como contribuição a escolha do domínio SDF e a utilização do tipo de dados *Fix-Point* que possibilita o tratamento de valores binários e a escolha do tamanho do valor em *bits.* Para os atores no Ptolemy, porém não trabalha com uma biblioteca de componentes pré-definida, o que pode implicar em restrições na geração de VHDL sintetizável.

Tanto Filiba, Leung e Nagpal (2008) quanto Zhou, Leung e Lee (2007) abordam a geração de código *helper-based*, onde, para cada ator que vai ser gerado um código na linguagem alvo existe um arquivo auxiliar que guarda um código pré-definido para aquele ator. Este ainda contribui ao detalhar o processo de geração de código, mesmo sem especificar qual seria a linguagem alvo.

O levantamento de trabalhos relacionados é importante para auxiliar no dimensionamento das contribuições, subsidiar ideias e alertar para possíveis problemas que venham a surgir durante o projeto a ser desenvolvido.

3 Projeto de sistemas digitais e o Alliance CAD System

Este capítulo trata do fluxo de projeto de sistemas digitais, apresentando o *Alliance CAD System*. Ao tratar deste fluxo e apresentar a ferramenta, é traçada uma breve apresentação sobre a semelhança entre a semântica do Ptolemy e do VHDL, fato importante para a justificativa desse trabalho.

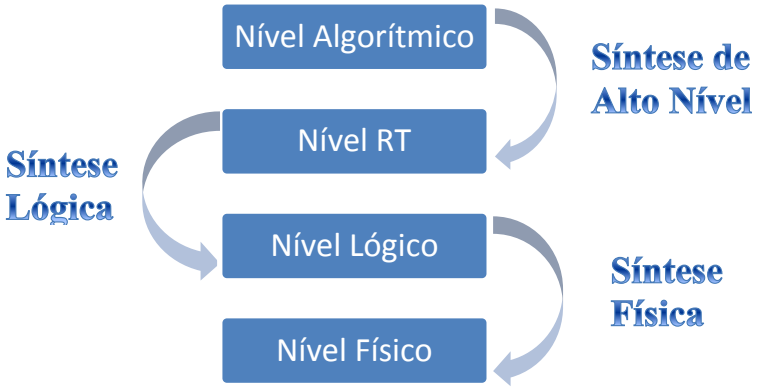
3.1 Síntese de Sistemas Digitais

Tanto a necessidade de redução no tempo quanto a de se enfrentar a crescente complexidade dos sistemas, motivaram o desenvolvimento de ferramentas para o projeto de sistemas. Tais ferramentas auxiliam no projeto, consideradas essenciais, através de técnicas CAD *Computer Aided Design* (CAD), permitem a automação de várias etapas do projeto eletrônico, daí a denominação clássica *Eletronic Design Automation* (EDA) (SENTURIA, 1998).

Para Bertasi (2002) a síntese automática de Sistemas Digitais é a transformação de uma especificação realizada em um determinado nível de abstração em outra descrição, a um nível mais próximo da realização física, através da aplicação de um conjunto de ferramentas que acrescentam detalhes estruturais e/ou geométricos à especificação inicial.

Síntese de alto nível é o processo pelo qual uma descrição de um algoritmo ou uma função é convertida em uma descrição do hardware necessário para executar o algoritmo ou função. Consiste, ainda, a partir da especificação do comportamento requerido e de um conjunto de restrições e objetivos de um sistema digital, encontrar uma estrutura que implemente o comportamento especificado e satisfaça as restrições e objetivos.

Figura 4 Os níveis de abstração do projeto de sistemas



Fonte: Santos, 2000.

O Projeto de sistemas é exemplificado por Santos (2000) como uma sucessão de etapas que envolvem diferentes níveis de abstração, tais como nível Algorítmico, nível de transferência de registradores ou *register-transfer* (RT), nível Lógico e nível Físico, conforme ilustra a Figura 4.

O primeiro nível, chamado nível algorítmico. Nele acontece a concepção e organização da ideia, a efetiva estruturação da função de um sistema, descrita na forma de um algoritmo que calcula os valores das saídas do sistema a partir dos valores em suas entradas, abstraindo-se a forma em que o sistema será implementado.

Em seguida, vem o nível de transferência entre registradores ou R*egister Transfer Level* (RTL). No nível RTL, a estrutura do sistema digital é descrita como um circuito composto de unidades funcionais, elementos de memória e de interconexão, ou seja, neste nível se utiliza somadores, unidades lógicas e aritméticas, multiplexadores, entre outros, para montar o sistema pensado anteriormente se utilizando de elementos como registradores e bancos de memória para oferecer persistência, e os conectando através de barramentos e multiplexadores, por exemplo. Denominando cada componente do sistema descrito no nível RTL de módulo.

O nível Lógico corresponde a uma descrição de cada módulo do nível RTL que compõe o sistema em termos de componentes básicos como portas lógicas e flip-flops. Já no nível Físico, este mesmo sistema que antes era uma ideia (nível Algorítmico), passou a ser representados por módulos (nível RTL) e, posteriormente, por portas lógicas e flip-flops (nível Lógico), passa neste nível, o Físico, a ser descrito como a interconexão de transistores, resistores e capacitores.

É, justamente, a tradução de uma descrição em dado nível para uma descrição no nível imediatamente inferior que é denominada síntese. Ainda de acordo com a Figura 4, esta síntese pode ser de Alto Nível, Lógica ou Física, de acordo com os níveis que são utilizados com início e fim no processo de síntese.

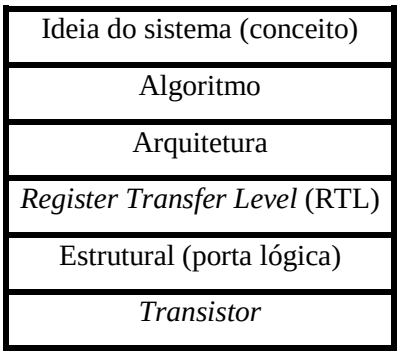
3.2 Fluxo de projeto

Um fluxo de projeto ou fluxo de concepção pode ser entendido como um conjunto de operações que são realizadas em sequência para a concepção de circuitos integrados, permitindo ao projetista progredir de uma especificação até sua implementação final em um caminho “livre” de erros (WESTE E HARRIS, 2005).

É comum partir de uma ideia ou conceito, fazendo uma descrição em alto nível de abstração, e seguir executando etapas com funções e ferramentas distintas a fim de chegar à implementação física do circuito.

Para Grout (2008) a Figura 5 pontua a abstração de descrição de um projeto em seis níveis.

Figura 5 Níveis de abstração de descrição de projeto



Fonte: Grout, 2008.

É a partir do conceito de um novo sistema – o mais alto nível de abstração – que se obtém uma especificação do projeto. No nível de algoritmo esta especificação de projeto é transcrita em uma descrição comportamental de alto nível, descrevendo matematicamente as interações no projeto. A partir do momento que este algoritmo é estruturado em *hardware* é identificada uma arquitetura, identificando quais blocos funcionais serão utilizados para alcançar os objetivos ao trabalharem em conjunto.

O nível RTL descreve o armazenamento dos dados, utilizando registradores, e o fluxo dos dados no projeto e as operações lógicas existentes. Neste, normalmente, são utilizadas ferramentas de síntese para converter a descrição do projeto numa descrição estrutural, "uma *netlist* do projeto em termos de portas lógicas e a interconexão entre elas" (GROUT, 2008). Portas estas implementadas usando *transistor*.

É perceptível que a Figura 5 transcende três perspectivas. Kaeslin (2008) as define como:

- Perspectiva Comportamental: Que é interessada no que o circuito faz, não como ele é concebido. O projeto é visto como uma "caixa preta" que produz algum resultado de saída em reposta a um estímulo de entrada;
- Perspectiva Estrutural: Aborda a conectividade do circuito eletrônico, quais blocos funcionais serão necessários e como cada um está conectado;
- Perspectiva Física: Pontua a organização física de cada componente do circuito, onde ficará cada bloco, por onde passarão os fios, etc.

Conhecendo as definições de Kaeslin (2008) e em consonância com Grout (2008), é possível inferir que o projeto se dá em duas macro etapas:

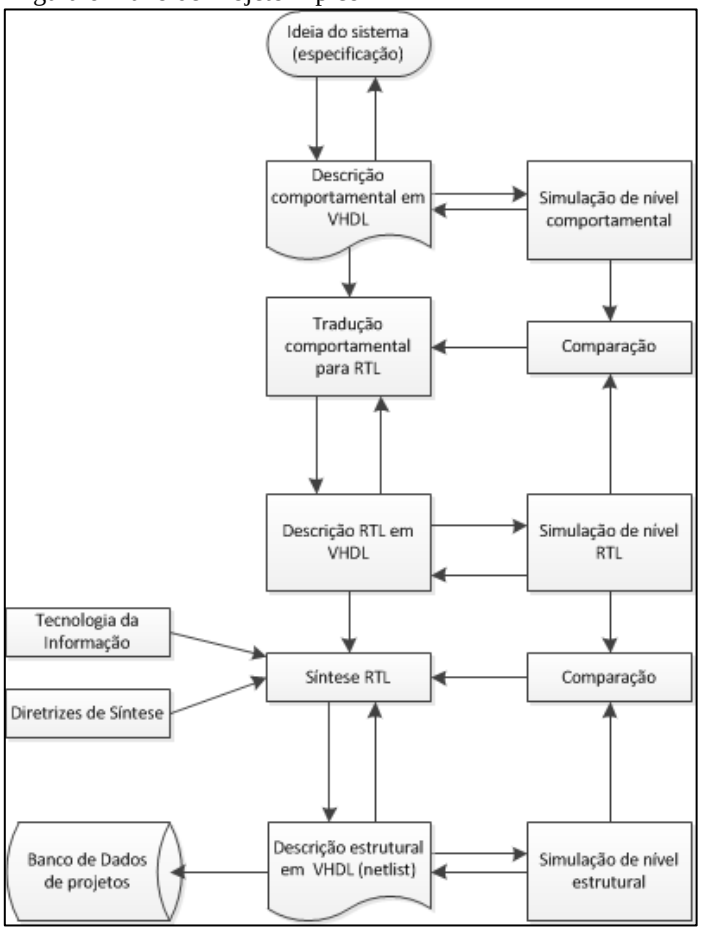
- Especificação Comportamental;
- Síntese RTL.

3.2.1 Especificação Comportamental

Esta etapa contempla, tomando como base a Figura 5, os níveis de algoritmo e arquitetura. Em nível comportamental, a operação de uma ferramenta é capturar, sem precisar especificar a implementação. Este nível é marcado por representações comportamentais que descrevem uma função do circuito integrado. Então é descrito, através de uma linguagem de descrição de hardware (HDL), o comportamento de cada bloco funcional.

Nesta etapa apresentam-se as ferramentas de simulação e comparação comportamental. A Figura 6 mostra um típico fluxo de projeto para um circuito digital, relacionando os níveis de abstração mostrados na Figura 5 com as etapas do fluxo de projeto, dando o devido destaque a etapa de especificação comportamental.

Figura 6 Fluxo de Projeto Típico



Fonte: Autor

Grout (2008) diz que a partir da ideia do sistema é escrita uma descrição comportamental em VHDL. Já nesta etapa é realizada uma simulação para verificar se a descrição comportamental está em conformidade com a ideia inicial. Para finalizar esta etapa, após êxito na comparação, a descrição pode ser traduzida – automática ou manualmente – para código RTL.

A descrição comportamental precisa ser validada, com a finalidade de realizar uma checagem entre os resultados das simulações realizadas, para isso existe o processo de comparação.

3.2.2 Síntese RTL

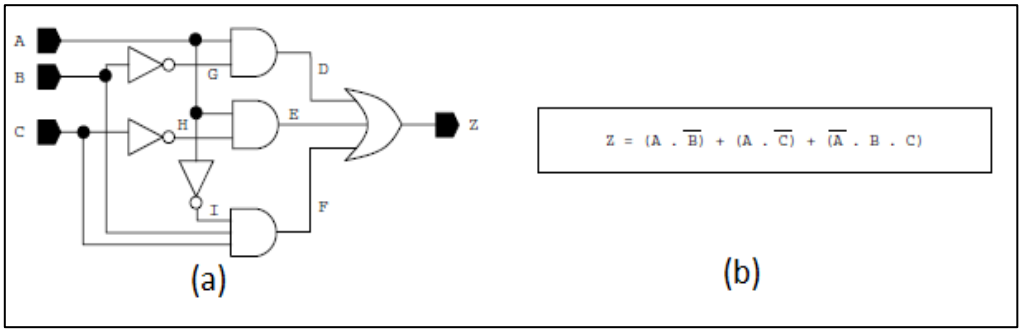
Esta etapa consiste em uma transformação de uma descrição comportamental para um *netlist* (lista de conexões) otimizado de portas lógicas (células padrões, células genéricas, caminho de dados, entre outros).

Como as descrições são normalmente hierárquicas, um componente em um nível pode ser decomposto tanto em termo de seus componentes constituintes como em termo de seus componentes elementares. São exemplos de descrição estruturais:

1) Desenhos esquemáticos: representação gráfica – Figura 7.a – de um projeto utilizando símbolos de portas lógicas interconectados por fios;

2) Expressões *Booleanas*: descrição textual – Figura 7.b – de projetos de lógica combinacional utilizando Álgebra Booleana[1];

Figura 7 Exemplos de descrições estruturais



Fonte:

3) Projeto HDL: representação textual de operações de um sistema ou um circuito logico digital utilizando uma linguagem;

4) Diagramas de Estado de Transição: apresenta visualização gráfica de máquinas de estado que identificam os estados e as transições entre estados do projeto.

Outro ponto importante é que o projetista pode dar parâmetros de otimização e restrições a serem satisfeitas, tais como:

- Tempo de atraso / tempo de chegada;
- Superfície;

[1] Uma álgebra Booleana pode ser definida com um conjunto de operadores e um conjunto de axiomas, que são assumidos verdadeiros sem necessidade de prova.

- Consumo de energia; entre outros.

A etapa de síntese RTL, assim como a Especificação Comportamental, tem de ser validada.

3.2.3 Etapa Place & Route

Mesmo não constando na enumeração das etapas do fluxo de projeto feita anteriormente, Esta etapa é importante e consiste, sob a perspectiva física, em uma transformação de uma descrição *netlist* para um *layout* físico.

O projetista pode dar alguns parâmetros e as restrições a serem satisfeitas, tais como:

- O posicionamento inicial das células;
- O posicionamento para cada conector físico;
- Número de camadas de metal a serem utilizadas, entre outros.

Da mesma forma que a Especificação Comportamental e a Síntese RTL, a etapa Place & *Route* tem de ser validada

3.2.4 Linguagem de Descrição de Hardware (HDL)

Uma linguagem de descrição de circuito padronizada possibilita o intercâmbio de informações referentes ao comportamento de um circuito entre fabricantes, fornecedores de sistemas e empresas (d'Amore, 0000).

Uma Linguagem de Descrição de Hardware (*Hardware Description Language* - HDL) pode ser usada na descrição em vários níveis de abstração do circuito em desenvolvimento. Possibilitando a captura esquemática, partindo de uma descrição de alto nível, podendo ser usada para refinar e particionar esta descrição em outras de nível mais detalhado, bem como testá-la durante o processo de desenvolvimento. A descrição final deve conter componentes primitivos e blocos funcionais. Podendo ainda, trabalhar com descrições parametrizáveis, onde as dimensões das unidades internas são definidas no momento da síntese.

Ao utilizar HDLs é possível descrever a estrutura e o comportamento do hardware assim como as funções desejadas no circuito a ser projetado, o que possibilita, dentre outros recursos, representar diretamente operações complexas, como operações aritméticas por exemplo. Uma descrição em HDL em conjunto com uma biblioteca de componentes é usada por uma ferramenta de síntese para a geração automática de um circuito digital (KAESLIN, 2008).

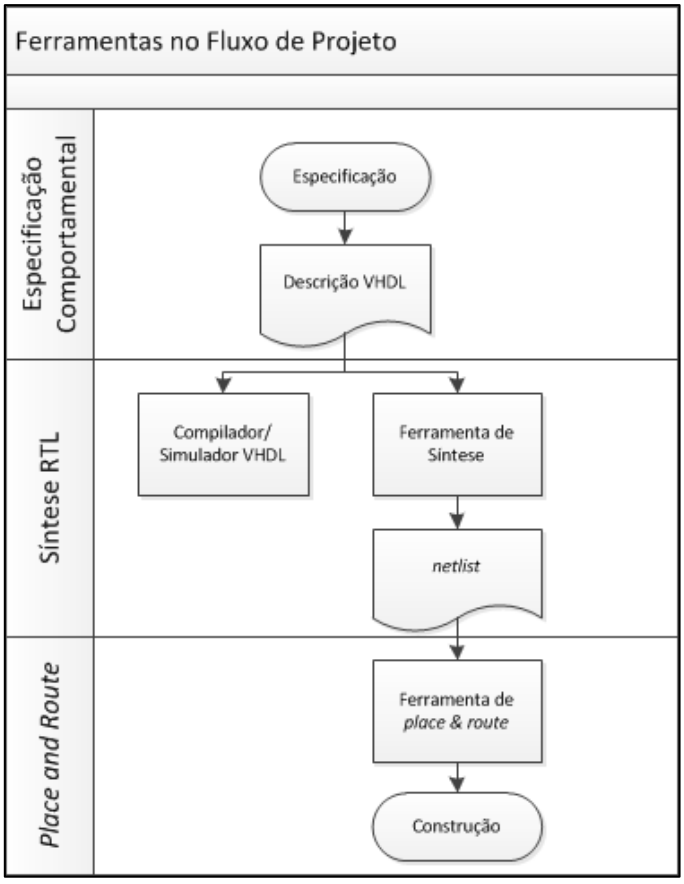
Atualmente, as HDLs mais utilizadas são o VHDL e o Verilog. Ambas as linguagens são hoje padrões aprovados e publicados pelo IEEE (*Institute of Electrical and Electronics Engineers*), tendo várias ferramentas comerciais compatíveis.

A linguagem VHDL é um acrônimo de *VHSIC Hardware Description Language*. Já o termo VHSIC é o acrônimo de *Very High Speed Integrated Circuit*. Assim podemos traduzir livremente o nome VHDL como "linguagem de descrição de hardware para circuitos integrados de velocidade muito alta". Todo componente descrito em VHDL requer ao menos duas estruturas: uma declaração de entidade (*entity*) e uma arquitetura (*architecture*). A declaração de entidade define os aspectos externos da função VHDL, isto é, os nomes das entradas e saídas e o nome da função. A arquitetura define os aspectos internos, isto é, como as entradas e saídas influenciam no funcionamento e como se relacionam com outros sinais internos.

3.2.5 Ferramentas

De acordo com as etapas do fluxo de um projeto, [d'Amore] apresenta a Figura 8 onde estas etapas são listadas incluindo as ferramentas inclusas em cada etapa.

Figura 8 Etapas de um Projeto



Fonte: Autor

A partir da especificação de um projeto, é gerada uma descrição VHDL, que é submetida a um simulador para a verificação da correspondência entre a especificação e o código [d'Amore]. A proposta deste trabalho é que esta etapa seja realizada no Ptolemy, a partir da utilização de atores e domínio específico.

A mesma descrição é interpretada por uma ferramenta de síntese que infere as estruturas necessárias para um circuito que corresponda à descrição. O resultado dessa etapa é um arquivo contendo uma rede de ligações – comumente denominada *netlist* – de elementos básicos disponíveis na tecnologia do dispositivo empregado. Nesta etapa, este trabalho propõe a ferramenta construída ao longo do desenvolvimento para a geração de código VHDL, bem como a utilização do Software Vasy – integrante do Alliance CAD System – como compilador/ Simulador VHDL para validação.

Esta *netlist* é a base de dados para a ferramenta que realiza o posicionamento e a interligação dos componentes, *Place and Route*. Ferramenta esta que gera um arquivo cujo disponibiliza as informações necessárias para a construção do dispositivo. O Alliance CAD System disponibiliza ferramentas como Nero, OCP, entre outros, para a realização e verificação nesta etapa.

Vale destacar que a Linguagem VHDL não foi originalmente concebida para a síntese de circuitos digitais; assim, nem todas as construções definidas são suportadas pelas ferramentas de síntese. Essas limitações não devem ser consideradas como um problema da ferramenta de síntese ou da linguagem, mas sim uma falha da própria descrição que está muito afastada de um circuito real.

3.3 Alliance CAD System

O Alliance CAD System é resultado de pesquisas realizadas no Laboratório MASI na Universidade de Pierre et Marie Curie University (UPMC), em Paris. O Objetivo inicial era proporcionar aos alunos de graduação e pós-graduação um framework CAD completo. A equipe MASI foca suas atividades em duas questões fundamentais: arquiteturas de computadores usando ASICs com alta complexidade, e ferramentas inovadoras de CAD para projeto VLSI (LIP6, 2011).

A ação principal CAD visa cumprir tanto as necessidades de designers experientes, fornecendo respostas práticas a state-of-the-art (problemas de síntese lógica, geração processual, verificação de layout, teste e interoperabilidade), e os designers iniciantes, através de um simples e conjunto consistente de ferramentas.

O Fluxo de projeto VLSI do Alliance é, portanto, com base tanto em ferramentas avançadas de CAD que não estão disponíveis nos sistemas comerciais de CAD, como ferramentas de abstração funcional ou análise estática de temporização, ferramentas padrão de projeto e validação.

O *Alliance* nos oferece uma biblioteca de células simbólicas, possibilitando efetuar o desenho de circuitos independente da tecnologia utilizada em sua etapa de fabricação. As bibliotecas de células incluem uma biblioteca de células *standard* e várias células específicas para o efeito memória e lógica de caminho de dados.

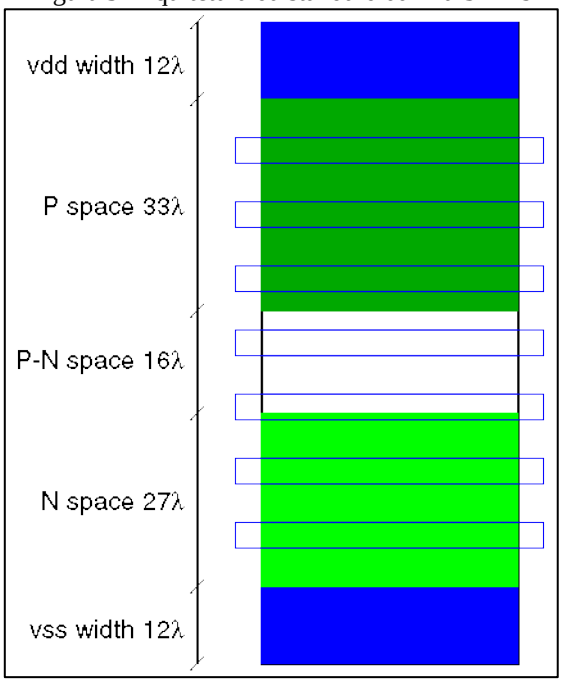
Nesta seção daremos uma noção geral sobre as ferramentas utilizadas, especificando sua função no processo, e demonstrando alguns dos comandos utilizados.

3.3.1 SXLib

O foi lançada em 1999, sendo fornecida em um formato proprietário. É uma biblioteca de standard *cells* que contém funções booleanas, buffers, multiplexadores, flip-flops, etc.

Todas as células possuem a mesma altura e N vezes sua largura, onde N é um número de *pitches* (distância mínima entre dois conectores), sendo disponíveis com 100λ de altura por 40λ para transistores P e 33λ para transistores N. A Figura 9 mostra como são divididos de 100 λ .

Figura 9 Arquitetura da standard *cell* na SXLib

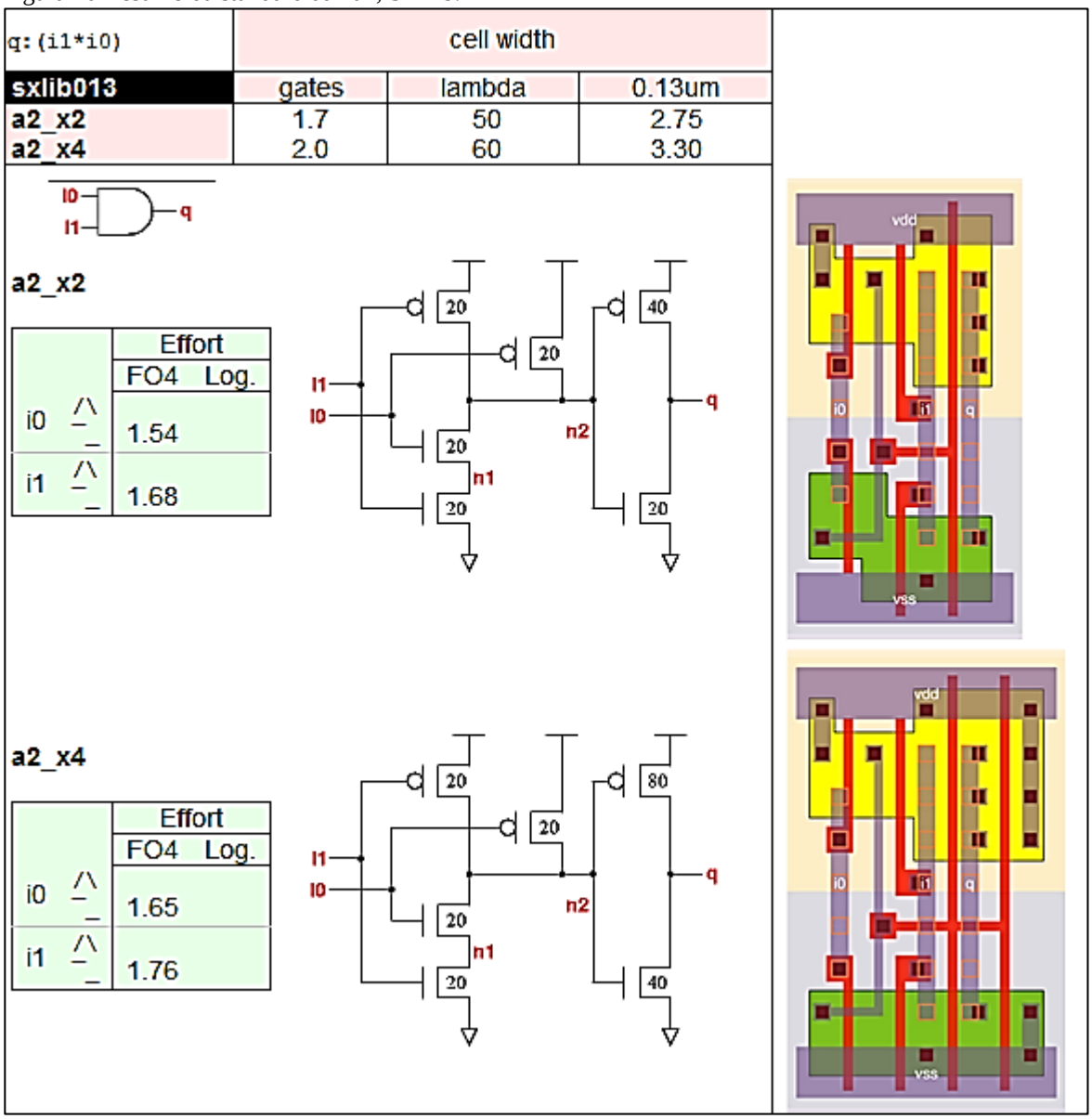


Fonte: VLSI Technology, 2012.

Ainda segundo VLSI Technology (2012) a eficiência da biblioteca é boa. Em alguns casos, porém as capacitâncias não são minimizadas, e alguns transistor o tamanho é aproximado para o desempenho elevado verdadeiro. Mas, em geral a biblioteca é larga e rápida.

Dentre as *standard cells* utilizadas neste trabalho, está a "a2" que corresponde a uma porta *and* com duas (2) entradas. Sendo disponibilizada em dois tamanhos como mostra a Figura 10.

Figura 10 Resumo da standard *cell* a2, SXLib.



Fonte: VLSI Technology

Por fim, é importante destacar que a utilização da SXLib neste projeto se deu apenas como estudo de caso. A ideia é que a ferramenta aqui proposta seja capaz de assimilar qualquer biblioteca de *standard cells*.

3.3.2 Vasy

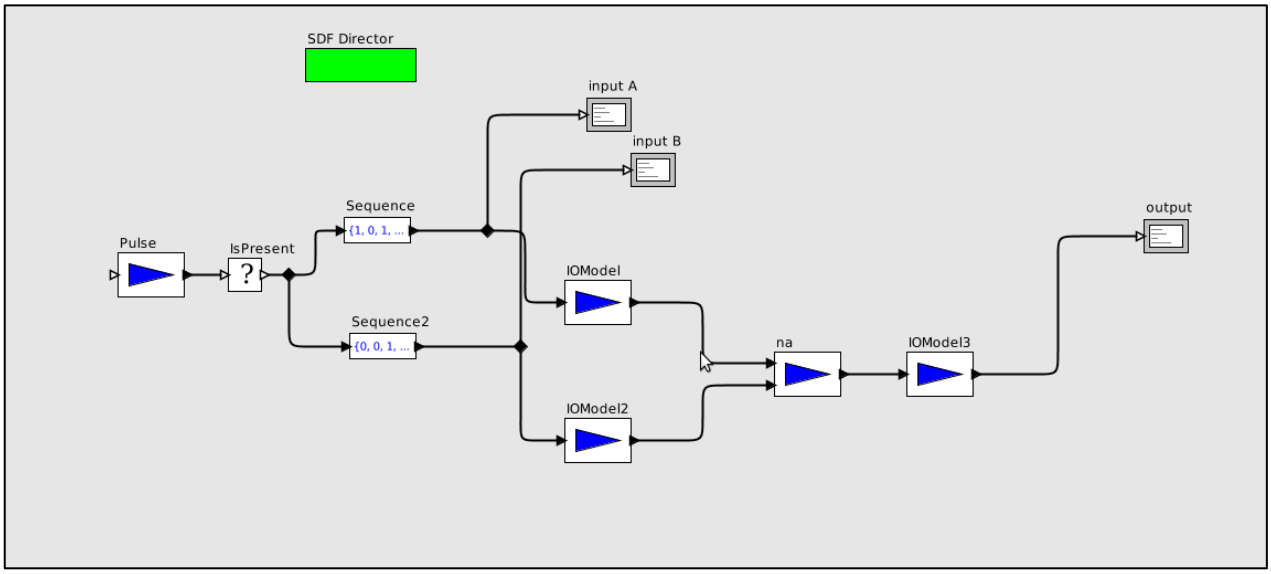
VASY é um analisador hierárquico de VHDL para Síntese. Realiza uma análise semântica de uma descrição VHDL, com um subconjunto VHDL mais extenso. Durante sua análise, VASY expande parâmetros genéricos, executa mapa genérico e gera declarações.

3.4 Equivalência Semântica entre Ptolemy e VHDL

Quando, no Ptolemy, um modelo é simulado e arquivado, gera um arquivo XML que permite acessar o modelo montado em momentos posteriores. Para a geração de código VHDL a ideia é que a ferramenta utilize um *parser* para mapear os atores do modelo do Ptolemy para as *standard cells* relacionadas e construir o relacionamento destas no arquivo VHDL correspondente ao modelo. As *standard cells* utilizadas serão as presentes na biblioteca SXLib.

Para efeito de exemplo é apresentado um modelo básico exibido na Figura 11, contendo uma das *standard cells* presentes na SXLib.

Figura 11 Modelo criado no Ptolemy



Fonte: Autor

O modelo anterior quando salvo é arquivado em formato XML. Este arquivo XML contem *tags* que delimitam características da entidade (modelo), portas, atores e suas relações.

Na Figura 12 é possível verificar o conteúdo do arquivo XML gerado. No final há a *tag* <*link* />, esta identifica a ligação entre os componentes do modelo. Na terceira ocorrência da *tag* link vê-se que ao componente “Input B” está atribuído a relação “relation4”, mesma relação atribuída – na quinta ocorrência da *tag* link – ao componente “na2_x. Input port B”.

Figura 12 Arquivo XML gerado pelo Ptolemy

```
    <port name="Input B" class="ptolemy.actor.TypedIOPort">
        <property name="input"/>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[345.0, 315.0]">
        </property></port>
    <entity name="na2_x" class="ptolemy.actor.VHDLGenerator.Na2_x1">
        <property name="_location" class="ptolemy.kernel.util.Location" value="[500.0, 270.0]">
        </property></entity>
    <relation name="relation2" class="ptolemy.actor.TypedIORelation">
        <property name="width" class="ptolemy.data.expr.Parameter" value="1">
        </property></relation>
    <relation name="relation" class="ptolemy.actor.TypedIORelation">
        <property name="width" class="ptolemy.data.expr.Parameter" value="1">
        </property></relation>
    <relation name="relation4" class="ptolemy.actor.TypedIORelation">
        <property name="width" class="ptolemy.data.expr.Parameter" value="1">
        </property> </relation>
    <link port="output" relation="relation2"/>
    <link port="Input A" relation="relation"/>
    <link port="Input B" relation="relation4"/>
    <link port="na2_x.Input port A" relation="relation"/>
    <link port="na2_x.Input port B" relation="relation4"/>
    <link port="na2_x.Result" relation="relation2"/>
</entity>
```

Fonte: Autor

Ao fazer esta análise e observar o componente da SXLib, na Figura 13, é perceptível que pode se construir um arquivo VHDL mapeando a *tag* "*entity*" do XML – que corresponde à declaração dos atores – para a *standard cell* no arquivo VHDL. As portas identificadas seriam declaradas na declaração da ENTITY no VHDL.

Figura 13 Definição do componente na2_x1.vhd na SXLib

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY na2_x1 IS
PORT(
  i0    : IN STD_LOGIC;
  i1    : IN STD_LOGIC;
  nq    : OUT STD_LOGIC
);
END na2_x1;

ARCHITECTURE RTL OF na2_x1 IS
BEGIN
  nq <= NOT((i0 AND i1));
END RTL;


Linha: 1 Col: 1 | INS | NORM | na2_x1.vhd
```

Fonte: Autor

Diante dos resultados destes estudos realizados, se deu inicio ao desenvolvimento da ferramenta. Sendo que os fatores preponderantes foram a capacidade do *framework* Ptolemy de simulação e adaptação às novas situações e a identificação da etapa do fluxo de projeto, etapa esta onde a ferramenta desenvolvida e sua utilização em conjunto com o Ptolemy seriam inseridas.

4 Uma Ferramenta para ensino de Microeletrônica utilizando o Ptolemy

Este capítulo apresenta a ferramenta proposta neste trabalho, trata da arquitetura da ferramenta, o ato de modelar com o Ptolemy e como se dá o processo desde o XML gerado até o resultado em linguagem VHDL. Mostrando, assim, o resultado do processo de desenvolvimento deste trabalho.

4.1 Arquitetura da Ferramenta

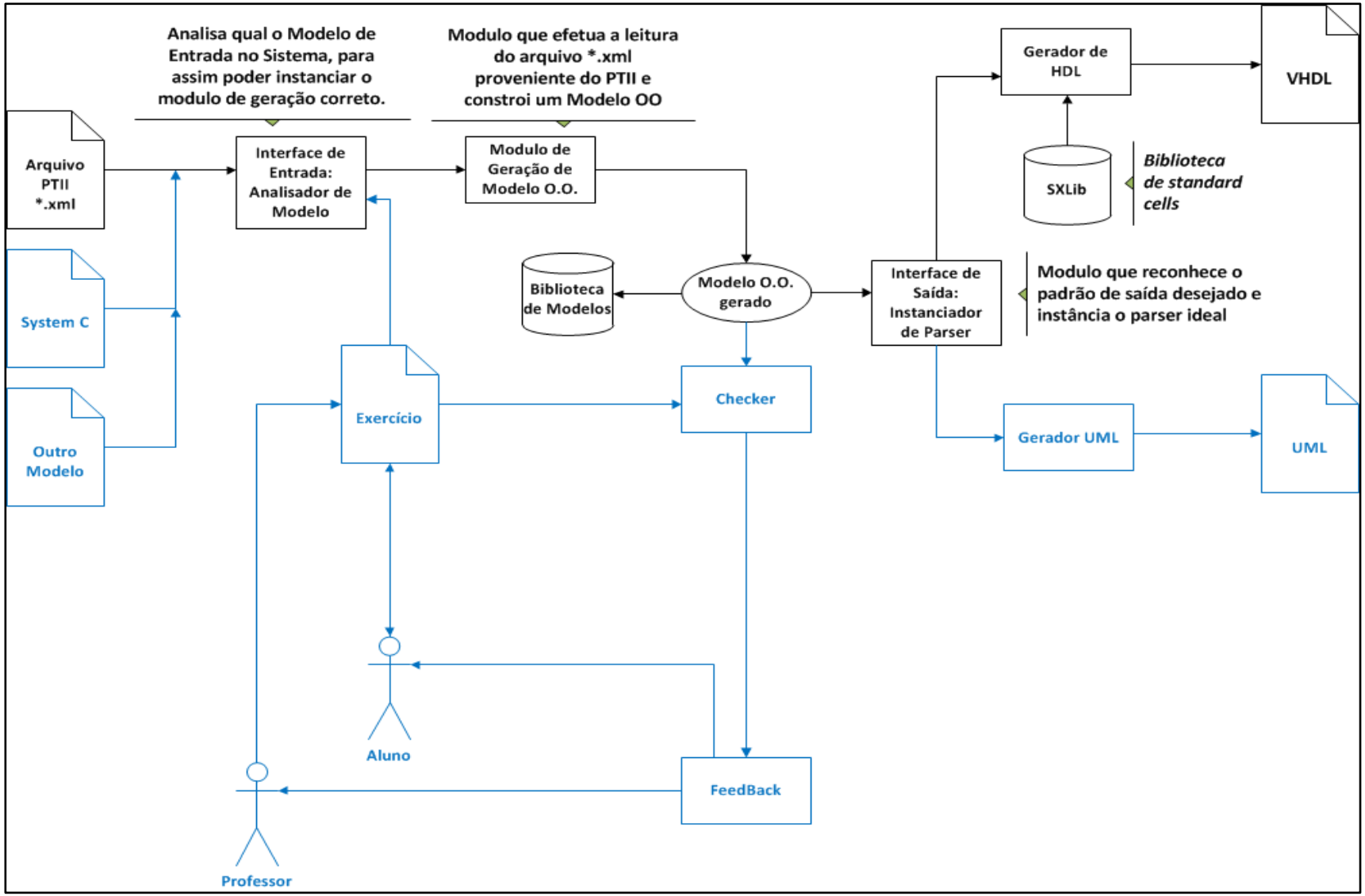
A arquitetura apresenta a ferramenta nos seus componentes, propriedades e relacionamentos entre os componentes.

Para Perry e Wolf (1992) a arquitetura permite diferentes vistas enfatizando aspectos particulares dos componentes.

Ainda sobre arquitetura, Bass *apud* Rocha, diz que arquitetura de software são as estruturas que incluem componentes, suas propriedades externas e os relacionamentos entre eles, constituindo uma abstração do sistema. Esta abstração suprime detalhes de componentes que não afetam a forma como eles são usados ou como eles usam outros componentes, auxiliando o gerenciamento da complexidade. Veja na Figura 14 a arquitetura proposta para a ferramenta.

A interação entre os componentes inicia a partir de um arquivo XML como o modelo construído no Ptolemy. A partir deste modelo, analisado pela interface inicial da ferramenta, há o reconhecimento da estrutura do arquivo a ser lido. Nesta etapa a possibilidade é de reconhecimento, apenas, de arquivos do Ptolemy, porém em trabalhos futuros podem ser assimiladas novas estruturas de arquivos provenientes de outros *frameworks* e outras linguagens. Sendo reconhecido que o arquivo é proveniente do Ptolemy, a ferramenta analisa e gera um modelo Orientado à Objetos correspondente ao modelo de atores salvo no arquivo XML analisado.

Figura 14 Arquitetura da Ferramenta

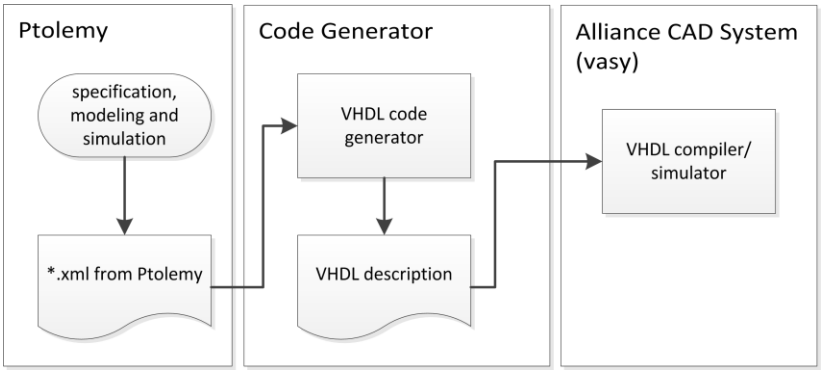


Fonte: Autor

De posse do modelo Orientado à Objetos (O.O.), há três possibilidades: Salvar para utilização futura, checar de acordo com um modelo prévio e gerar o código em linguagem VHDL. Para implementação futura, vislumbra-se a geração de código em outras linguagens como System Verilog ou System C, por exemplo.

Para exemplificar o processo de utilização do Ptolemy e a ferramenta aqui proposta, segue a Figura 15.

Figura 15 Fluxo de Uso/Experimento da Ferramenta



Fonte: Autor

Desta forma utiliza-se o Ptolemy e Gerador de Código para Modelagem, Simulação e Geração de Código; e o Alliance CAD System para Validação do código gerado.

4.2 Modelando com o Ptolemy

O ato de modelar sistemas no Ptolemy consiste na escolha, alocação e interligação de atores com a finalidade de simular o sistema em questão. Para que isso aconteça é necessário, dentre outros conhecimentos, do que foi descrito na seção 2 deste trabalho.

Para Lee (2005) a modelagem é o ato de representar um sistema ou subsistema formalmente. Podendo tal modelo ser construtivo, caso em que se define um procedimento computacional que imita um conjunto de propriedades do sistema.

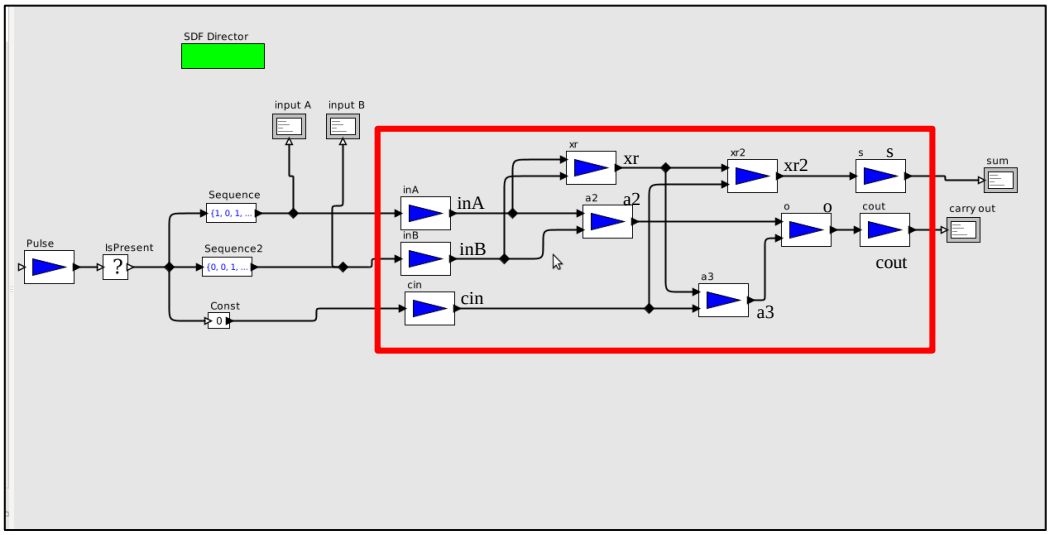
Para iniciar a modelagem se faz necessário levantamento dos atores, pois como dito na seção 2.3.1, no Ptolemy a modelagem e simulação é orientada a atores, “dirigida” por um modelo de computação. Caso não exista o ator em questão, há a possibilidade de cria-lo.

Para a modelagem de circuitos microeletrônicos é preciso o desenvolvimento de um novo conjunto atores com parâmetros específicos de *hardware* (FILIBA, LEUNG E NAGPA, 2008).

Assim, tem-se que considerar tipo dos dados, MoC a ser utilizado e, é claro, as características operacionais de cada novo ator.

Ao tratar do fluxo de projeto de sistemas digitais, o Ptolemy é alocado no nível mais alto deste fluxo. Neste acontece a concepção da ideia e a montagem da arquitetura desejada, é o que se denomina de captura comportamental. Neste momento, ao invés de utilizar linguagens de programação, utiliza-se os atores do Ptolemy. Veja na Figura 16, como se dá essa modelagem.

Figura 16 Modelagem no Ptolemy



Fonte: Autor

Na Figura, foram utilizados tanto atores nativos do Ptolemy quanto novos atores correspondentes aos componentes da SXLib. Os atores destacados correspondem ao modelo estudado – neste caso um somador completo –, já os demais atores acima são utilizados como *test bench*[2].

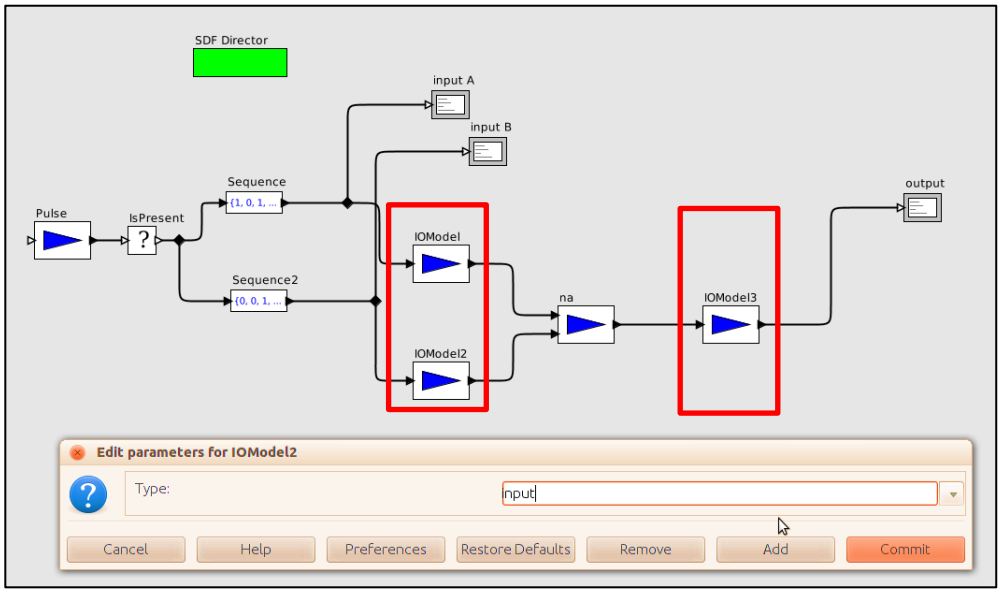
Quanto aos MoC, o ator composto usa o *SDF Director* que foi o escolhido para as simulações destes modelos.

Para que, posteriormente, a geração de código seja realizada corretamente é preciso que os atores "IOModel" sejam colocados como portas do modelo, de forma a separar o modelo do *testbench*.

[2] É um ambiente virtual utilizado para verificar a exatidão ou a solidez de um desenho ou modelo.

A Figura 17 mostra a utilização do ator IOModel. Mostra também a necessidade de configurar o tipo (*type*), que pode ser *input* ou *output*, que significa entrada ou saída, respectivamente.

Figura 17 Utilização do ator IOModel



Fonte: Autor

Note que os atores destacados na Figura 17 estão localizados antes e depois do ator “na” que, neste caso, representa o modelo a ser simulado. Os demais atores são responsáveis por gerar estímulos e exibir resultados.

4.3 Leitura de XML

Como já foi dito, quando, no Ptolemy, um modelo é simulado e arquivado, gera um arquivo XML que permite acessar o modelo montado em momentos posteriores.

XML, ou *eXtended Markup Language*, é um padrão para a formatação de dados, ou seja, uma maneira de organizar informações. Os documentos XML podem ser facilmente compreendidos por programadores facilitando o desenvolvimento de aplicativos compatíveis.

Para Bax (2001);

> Um arquivo XML é constituído de elementos. Como sempre, cada elemento possui uma marca inicial (como <*title*> ou <titulo>), uma marca final (como </*title*> ou </titulo>) e a informação propriamente dita entre as duas marcas. Porém, diferentemente de HTML, XML não propõe um número fixo de marcas. Um elemento XML pode ser marcado da forma que o autor do documento bem

> entender, ou seja, com o termo que melhor descreve a informação em sua opinião.

Seguindo essa premissa as marcas, também conhecidas como *tags*, no Ptolemy foram definidas de forma a representar as estruturas.

São exemplos de *tags* do arquivo do Ptolemy:

- <*entity*> e </*entity*>
- <property> e </property>
- <port> e </port>
- <relation> e </relation>
- <link />

Estes elementos, representados por *tags*, podem ter atributos. São eles:

- Name
- Class
- Value
- Port
- Relation

O Código Fonte 1 trax um trecho do conteúdo do Apêndice A, onde é possível verificar a organização do arquivo XML, a repetição da *tag* <*link* /> que representa a ligação entre os atores.

Código Fonte 1 Parte do Arquivo adder_1bit.xml

```
[…]
   <entity name="xr" class="ptolemy.actor.lib.vhdlsxlib.xr2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="xr2_x1">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[490.0, 210.0]">
      </property>
   </entity>
   <entity name="a2" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[530.0, 310.0]">
      </property>
   </entity>
   <entity name="a3" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
```

```
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[660.0, 385.0]">
        </property>
    </entity>
    <entity name="xr2" class="ptolemy.actor.lib.vhdlsxlib.xr2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="xr2_x1">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[670.0, 220.0]">
        </property>
    </entity>
    <entity name="o" class="ptolemy.actor.lib.vhdlsxlib.o2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="o2_x2">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[755.0, 320.0]">
        </property>
    </entity>
    <entity name="inA" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="{350.0, 265.0}">
        </property>
    </entity>
    <entity name="inB" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 320.0]">
        </property>
    </entity>
    <entity name="cin" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 395.0]">
        </property>
    </entity>
    <entity name="s" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[860.0, 220.0]">
        </property>
    </entity>
    <entity name="cout" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[865.0, 320.0]">
        </property>
    </entity>
[…]
    <link port="Sequence.enable" relation="relation5"/>
    <link port="Sequence.output" relation="relation6"/>
    <link port="Sequence2.enable" relation="relation5"/>
    <link port="Sequence2.output" relation="relation2"/>
    <link port="sum.input" relation="relation3"/>
    <link port="Pulse.output" relation="relation4"/>
    <link port="IsPresent.input" relation="relation4"/>
    <link port="IsPresent.output" relation="relation5"/>
    <link port="input A.input" relation="relation6"/>
    <link port="input B.input" relation="relation2"/>
    <link port="carry out.input" relation="relation14"/>
    <link port="Const.output" relation="relation"/>
    <link port="Const.trigger" relation="relation5"/>
    <link port="xr.i0" relation="relation10"/>
    <link port="xr.i1" relation="relation11"/>
```

```
    <link port="xr.q" relation="relation9"/>
    <link port="a2.i0" relation="relation10"/>
    <link port="a2.i1" relation="relation11"/>
    <link port="a2.q" relation="relation12"/>
    <link port="a3.i0" relation="relation9"/>
    <link port="a3.i1" relation="relation7"/>
    <link port="a3.q" relation="relation13"/>
    <link port="xr2.i0" relation="relation9"/>
    <link port="xr2.i1" relation="relation7"/>
    <link port="xr2.q" relation="relation8"/>
    <link port="o.i0" relation="relation12"/>
    <link port="o.i1" relation="relation13"/>
    <link port="o.q" relation="relation16"/>
    <link port="inA.in" relation="relation6"/>
    <link port="inA.out" relation="relation10"/>
    <link port="inB.in" relation="relation2"/>
    <link port="inB.out" relation="relation11"/>
    <link port="cin.in" relation="relation"/>
    <link port="cin.out" relation="relation7"/>
    <link port="s.in" relation="relation8"/>
    <link port="s.out" relation="relation3"/>
    <link port="cout.in" relation="relation16"/>
    <link port="cout.out" relation="relation14"/>
</entity>
```

Fonte: Autor

Outra marca importante no arquivo XML é a *tag <entity />*. No exemplo mostrado, as *tags <entity />* mostram a existência de atores e suas propriedades – *tag <property />*.

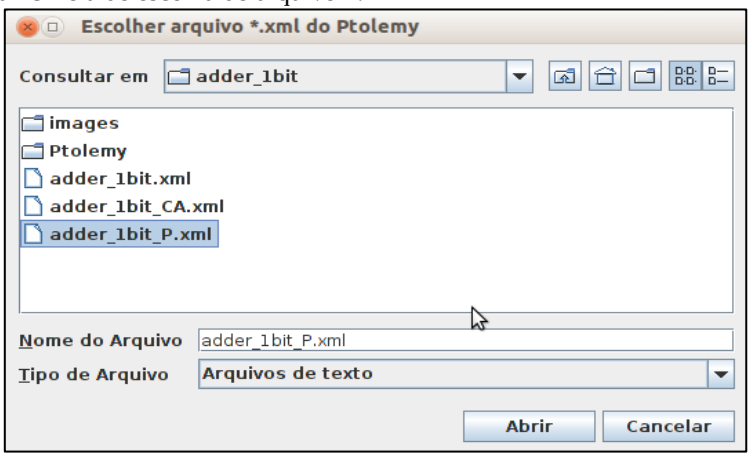
4.4 Execução da Ferramenta de Geração de Código.

De acordo com a arquitetura mostrada no início deste capítulo, há uma etapa que corresponde a geração de um modelo O.O. – orientado à objetos – após a identificação e interpretação do modelo inicial, o modelo de atores neste caso, para posterior geração de código.

O modelo O.O. é abstrato. A geração deste modelo O.O. é importante, pois a partir da instanciação de componente Java (objetos) estamos libertos daquele modelo inicial, ou seja, de posse do melo O.O. não importa qual o modelo inicial, podemos de posse das ferramentas necessárias, gerar qualquer modelo final.

A montagem deste modelo O.O. é baseada na leitura das *tags* no arquivo XML e a instanciação e objetos Java, considerando as propriedades. Após esta modelagem, acontece a geração de código VHDL. A Figura 18 demonstra a tela inicial de utilização da ferramenta.

Figura 18 Tela de escolha do arquivo *.xml



Fonte: Autor

Após a escolha do arquivo que faz referência a modelagem efetuada e simulada no Ptolemy é dado início o processo de geração de código. Este inicia pela identificação dos atores (*entity*) que possuem uma propriedade (*property)* de nome "*CellReference"*, como é possível constatar na Figura 15. Esta propriedade possui um valor (*value*) que corresponde ao nome da *standard cell* na biblioteca utilizada, neste caso a SXLib.

4.5 Arquivo VHDL Gerado (*.vst)

Localizando os *helpers*, ou seja, o arquivo VHDL correspondente a cada ator utilizado é chegado o momento de instanciá-los no arquivo a ser gerado. Para isso, a busca é pelo bloco, no arquivo VHDL, que declara a *entity*, ignorando assim, o conteúdo do bloco *architecture*. No Código Fonte 2 é possível verificar o conteúdo de um *helper* utilizado.

Código Fonte 2 Arquivo helper do ator a2 (and)

```
ENTITY a2_x2 IS
GENERIC (
  CONSTANT area   : NATURAL := 1250;
  CONSTANT cin_i0 : NATURAL := 9;
  CONSTANT cin_i1 : NATURAL := 11;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tphh_i1_q        : NATURAL := 203;
  CONSTANT tphh_i0_q        : NATURAL := 261;
  CONSTANT tpll_i0_q        : NATURAL := 388;
  CONSTANT tpll_i1_q        : NATURAL := 434;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0        : in  BIT;
  i1        : in  BIT;
  q         : out BIT;
  vdd       : in  BIT;
  vss       : in  BIT
);
END a2_x2;
```

```
ARCHITECTURE behaviour_data_flow OF a2_x2 IS

BEGIN
  ASSERT ((vdd and not (vss)) = '1')
  REPORT "power supply is missing on a2_x2"
  SEVERITY WARNING;
  q <= (i0 and i1) after 1000 ps;
END;
```

Fonte: SXLib

A ferramenta, após a identificação das instâncias, reconhece as ligações entre os componentes ao ler as *tags* <*relation* /> no arquivo XML e efetua as devidas instanciações dos *signals* no arquivo VHDL.

A seguir, no Código Fonte 3, é possível verificar o arquivo VHDL gerado a partir da modelagem demonstrada na Figura 16, utilizando a ferramenta proposta neste trabalho.

Código Fonte 3 Arquivo *.vst gerado

```
entity adder_1bit_P is
   port (
inA         :in         bit;
inB         :in         bit;
cin         :in         bit;
s           :out        bit;
cout        :out        bit;
vdd         :in         bit;
vss         :in         bit
 );
end adder_1bit_P;

architecture structural of adder_1bit_P is
COMPONENT xr2_x1
GENERIC (
  CONSTANT area   : NATURAL := 2250;
  CONSTANT cin_i0 : NATURAL := 21;
  CONSTANT cin_i1 : NATURAL := 22;
  CONSTANT rdown_i0_q       : NATURAL := 2850;
  CONSTANT rdown_i1_q       : NATURAL := 2850;
  CONSTANT rup_i0_q         : NATURAL := 3210;
  CONSTANT rup_i1_q         : NATURAL := 3210;
  CONSTANT tplh_i1_q        : NATURAL := 261;
  CONSTANT tphl_i0_q        : NATURAL := 292;
  CONSTANT tplh_i0_q        : NATURAL := 293;
  CONSTANT tphh_i0_q        : NATURAL := 366;
  CONSTANT tphl_i1_q        : NATURAL := 377;
  CONSTANT tpll_i1_q        : NATURAL := 388;
  CONSTANT tpll_i0_q        : NATURAL := 389;
  CONSTANT tphh_i1_q        : NATURAL := 405;
  CONSTANT transistors      : NATURAL := 12
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

COMPONENT a2_x2
GENERIC (
  CONSTANT area   : NATURAL := 1250;
  CONSTANT cin_i0 : NATURAL := 9;
  CONSTANT cin_i1 : NATURAL := 11;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tphh_i1_q        : NATURAL := 203;
  CONSTANT tphh_i0_q        : NATURAL := 261;
  CONSTANT tpll_i0_q        : NATURAL := 388;
  CONSTANT tpll_i1_q        : NATURAL := 434;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

COMPONENT o2_x2
GENERIC (
  CONSTANT area   : NATURAL := 1250;
```

```
  CONSTANT cin_i0 : NATURAL := 10;
  CONSTANT cin_i1 : NATURAL := 10;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tpll_i0_q        : NATURAL := 310;
  CONSTANT tphh_i1_q        : NATURAL := 335;
  CONSTANT tpll_i1_q        : NATURAL := 364;
  CONSTANT tphh_i0_q        : NATURAL := 406;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

signal relation : bit;
signal relation10 : bit;
signal relation11 : bit;
signal relation12 : bit;
signal relation13 : bit;
signal relation14 : bit;
signal relation2 : bit;
signal relation3 : bit;
signal relation16 : bit;
signal relation8 : bit;
signal relation9 : bit;
signal relation6 : bit;
signal relation7 : bit;
signal relation4 : bit;
signal relation5 : bit;

begin

xr : xr2_x1
Generic Map(
   area      => 2250,
   cin_i0    => 21,
   cin_i1    => 22,
   rdown_i0_q          => 2850,
   rdown_i1_q          => 2850,
   rup_i0_q            => 3210,
   rup_i1_q            => 3210,
   tplh_i1_q           => 261,
   tphl_i0_q           => 292,
   tplh_i0_q           => 293,
   tphh_i0_q           => 366,
   tphl_i1_q           => 377,
   tpll_i1_q           => 388,
   tpll_i0_q           => 389,
   tphh_i1_q           => 405,
   transistors         => 12
)
port map (
  i0 => inA ,
  i1 => inB ,
  vdd => vdd ,
  vss => vss ,
  q => relation9
 );

a2 : a2_x2
Generic Map(
   area      => 1250,
```

```
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => inA ,
  i1 => inB ,
  vdd => vdd ,
  vss => vss ,
  q => relation12
 );

a3 : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => relation9 ,
  i1 => cin ,
  vdd => vdd ,
  vss => vss ,
  q => relation13
 );

xr2 : xr2_x1
Generic Map(
   area       => 2250,
   cin_i0     => 21,
   cin_i1     => 22,
   rdown_i0_q          => 2850,
   rdown_i1_q          => 2850,
   rup_i0_q            => 3210,
   rup_i1_q            => 3210,
   tplh_i1_q           => 261,
   tphl_i0_q           => 292,
   tplh_i0_q           => 293,
   tphh_i0_q           => 366,
   tphl_i1_q           => 377,
   tpll_i1_q           => 388,
   tpll_i0_q           => 389,
   tphh_i1_q           => 405,
   transistors         => 12
)
port map (
  i0 => relation9 ,
  i1 => cin ,
  vdd => vdd ,
  vss => vss ,
  q => s
 );
```

```
o : o2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 10,
   cin_i1     => 10,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tpll_i0_q           => 310,
   tphh_i1_q           => 335,
   tpll_i1_q           => 364,
   tphh_i0_q           => 406,
   transistors         => 6
)
port map (
  i0 => relation12 ,
  i1 => relation13 ,
  vdd => vdd ,
  vss => vss ,
  q => cout
 );

end structural;
```

Fonte: Autor

Após a obtenção do arquivo *.vst a ferramenta proposta neste trabalho tem atingido seu objetivo, estando assim chegado o momento da verificação.

4.6 Gerando Arquivo *.vhd com Vasy

Como dito na seção 3.3.2 o Vasy é um software analisador VHDL. Assim podemos utilizá-lo para analisar o arquivo VHDL gerado pela ferramenta proposta neste trabalho.

Figura 19 Comando de utilização do Vasy

```
$ vasy –I vst adder_1bit
```

Fonte: Autor

Utilizando o comando presente na Figura 19, está se executando o Software Vasy passando como parâmetro um arquivo de entrada com a extensão *.vst e o nome adder_1bit.

Após a execução se obtem como resposta um arquivo com extensão *.vhd correspondente ao arquivo de entrada e outros arquivo, também *.vhd, correspondente as *standardcells* utilizadas no projeto.

5 Resultados

Este capítulo apresenta resultados da utilização da ferramenta proposta neste trabalho. Mostra exemplo de circuitos modelados com o Ptolemy e o resultado em linguagem VHDL.

Como cenário foram escolhido os seguintes circuito: somador, decodificador e Flip-flop D. As próximas seções apresentam conceitos gerais sobre cada circuito, a modelagem no Ptolemy, a geração de código e validação do código gerado.

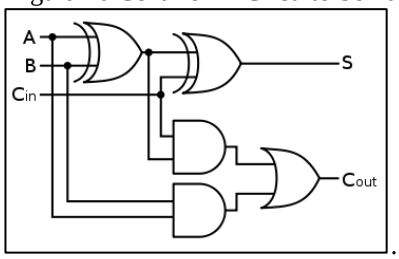
5.1 Cenário 1: Somador

O Somador é um circuito combinacional aritmético, este por sua vez implementa operações aritméticas como adição, subtração, multiplicação e divisão com números binários.

O somador completo (*full adder*) - um circuito que implementa a adição de três bits (dois bits significativos e um *carry*) – é um circuito aritmético básico a partir do qual todos os outros circuitos aritméticos são construídos.

A operação da adição de dois dígitos binários (bits), a qual pode ser vista como a adição de dois números binários de um bit cada, isso porque a operação aritmética mais simples é a adição de dois dígitos binários, que consiste de quatro possíveis operações elementares: $0+0=0$, $0+1=1$, $1+0=1$ e $1+1=10$. As três primeiras operações produzem um dígito de soma. Entretanto, quando ambos os operandos são iguais a um, são necessários dois dígitos para expressar seu resultado. Neste caso, o transporte (vai-um ou *carry*, em inglês) é somado ao próximo par mais significativo de bits. Assim, o somador completo, como mostrado na Figura 20, necessita de três entradas e duas saídas.

Figura 20 Cenário 1 - Circuito Somador Completo de 1 bit

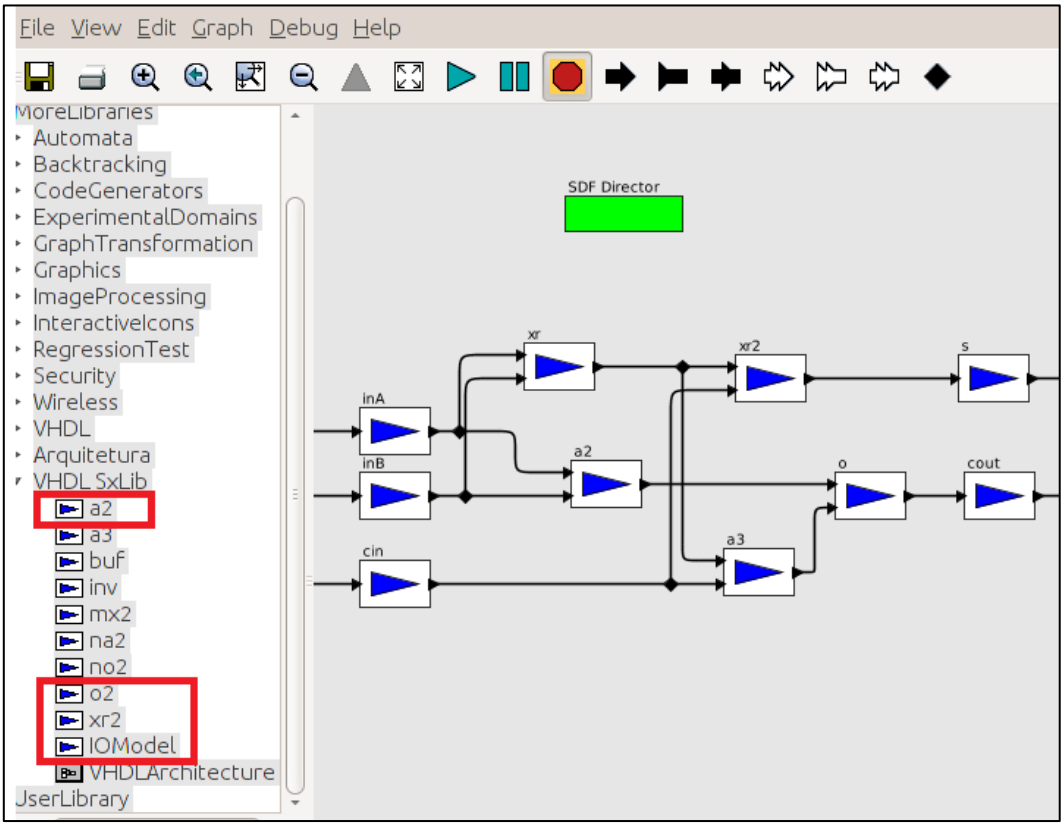


Fonte: Autor

O circuito exibido na Figura 20 mostra a utilização de cinco portas lógicas, sendo estas: duas portas XOR (ou exclusivo), duas portas AND (e) e uma porta OR (ou). Em decorrência da necessidade de utilização destes atores, foram criados atores correspondentes às portas citadas.

Como é possível verificar destacados na lateral esquerda da Figura 21 os atores criados para este cenário foram: a2, que é a porta AND com duas entradas; xr2, uma porta XOR com duas entradas; e o ator o2, que é uma porta OR de duas entradas. Os atores nomeados como “inA”, “inB”, “cin”, ”S” e “cout” são do tipo IOModel, sua utilização foi explicada no capítulo anterior.

Figura 21 Cenário 1 - Somador modelado no Ptolemy



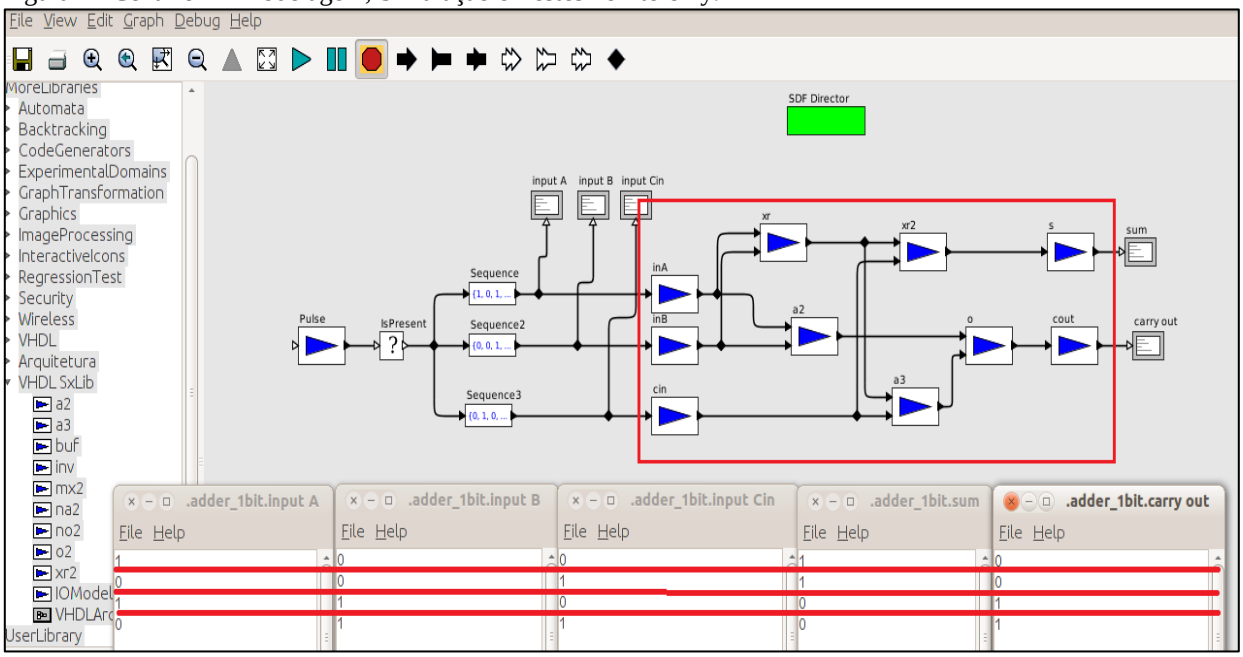
Fonte: Autor

Os atores nomeados como “inA”, “inB”, “cin”, ”S” e “cout” são do tipo IOModel, sua utilização foi explicada no capítulo anterior.

Quando modelado o circuito precisa ser simulado. Para isso tem-se que gerar estímulos para que o circuito possa responder, de acordo com seu processamento, para ser validado quanto a sua efetividade.

Após a escolha dos atores do circuito e aqueles que irão gerar os estímulos é o momento de iniciar a simulação. Após a simulação, a tela do Ptolemy é apresentada da seguinte forma integrante da Figura 22.

Figura 22 Cenário 1 - Modelagem, Simulação e Testes no Ptolemy.



Fonte: Autor

Na Figura 22 há duas partes destacadas. No destaque da parte inferior da figura é onde contas o resultado de cada monitor inserido no modelo. O segundo destaque diferencia os atores do modelo, que serão alvos, daqueles que representam o *testbench*, o gerador de estímulos e os monitores.

O Arquivo “adder_1bit.xml” está no Apêndice A. No Código Fonte 4 está a definição dos atores que são passíveis de geração de código e a ligação entre eles, ou seja, as *tags* *<entity />* e *<link />*.

Código Fonte 4 Cenário 1 – Parte do arquivo adder_1bit.xml

```
[...]
   <entity name="xr" class="ptolemy.actor.lib.vhdlsxlib.xr2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="xr2_x1">       </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[490.0, 210.0]">       </property>
   </entity>
   <entity name="a2" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">       </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[530.0, 310.0]">       </property>
   </entity>
   <entity name="a3" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">       </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[660.0, 385.0]">       </property>
   </entity>
   <entity name="xr2" class="ptolemy.actor.lib.vhdlsxlib.xr2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="xr2_x1">       </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[670.0, 220.0]">       </property>
   </entity>
   <entity name="o" class="ptolemy.actor.lib.vhdlsxlib.o2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="o2_x2">       </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[755.0, 320.0]">       </property>
   </entity>
   <entity name="inA" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
```

```
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="{350.0, 265.0}">      </property>
   </entity>
   <entity name="inB" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 320.0]">      </property>
   </entity>
   <entity name="cin" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 395.0]">      </property>
   </entity>
   <entity name="s" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[860.0, 220.0]">      </property>
   </entity>
   <entity name="cout" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[865.0, 320.0]">      </property>
   </entity>
[…]
   <link port="input A.input" relation="relation6"/>
   <link port="input B.input" relation="relation2"/>
   <link port="carry out.input" relation="relation14"/>
   <link port="xr.i0" relation="relation10"/>
   <link port="xr.i1" relation="relation11"/>
   <link port="xr.q" relation="relation9"/>
   <link port="a2.i0" relation="relation10"/>
   <link port="a2.i1" relation="relation11"/>
   <link port="a2.q" relation="relation12"/>
   <link port="a3.i0" relation="relation9"/>
   <link port="a3.i1" relation="relation7"/>
   <link port="a3.q" relation="relation13"/>
   <link port="xr2.i0" relation="relation9"/>
   <link port="xr2.i1" relation="relation7"/>
   <link port="xr2.q" relation="relation8"/>
   <link port="o.i0" relation="relation12"/>
   <link port="o.i1" relation="relation13"/>
   <link port="o.q" relation="relation16"/>
   <link port="inA.in" relation="relation6"/>
   <link port="inA.out" relation="relation10"/>
   <link port="inB.in" relation="relation2"/>
   <link port="inB.out" relation="relation11"/>
   <link port="cin.in" relation="relation"/>
   <link port="cin.out" relation="relation7"/>
   <link port="s.in" relation="relation8"/>
   <link port="s.out" relation="relation3"/>
   <link port="cout.in" relation="relation16"/>
   <link port="cout.out" relation="relation14"/>
   <link port="Sequence3.enable" relation="relation5"/>
   <link port="Sequence3.output" relation="relation"/>
   <link port="input Cin.input" relation="relation"/>
</entity>
```

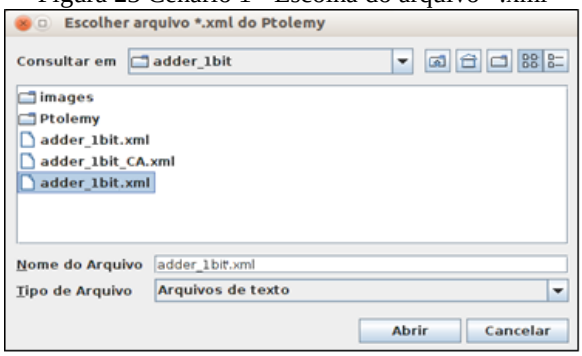
É importante destacar que para a correta geração do código VHDL é necessário, ainda durante a modelagem no Ptolemy, configurar o parâmetro "*CellReference*" para os atores do circuito estudado.

5.1.1 Resultado

Ao escolher o arquivo correspondente ao circuito que se deseja gerar o arquivo VHDL correspondente, a ferramenta mostra o resultado e o arquivo VHDL é salvo.

Depois de realizar a modelagem e simulação do circuito desejado no Ptolemy, é chegado o momento de utilizar a ferramenta desenvolvida neste trabalho. Ao executar a ferramenta, é possível escolher um arquivo com extensão *.xml, como mostra a Figura 23.

Figura 23 Cenário 1 - Escolha do arquivo *.xml

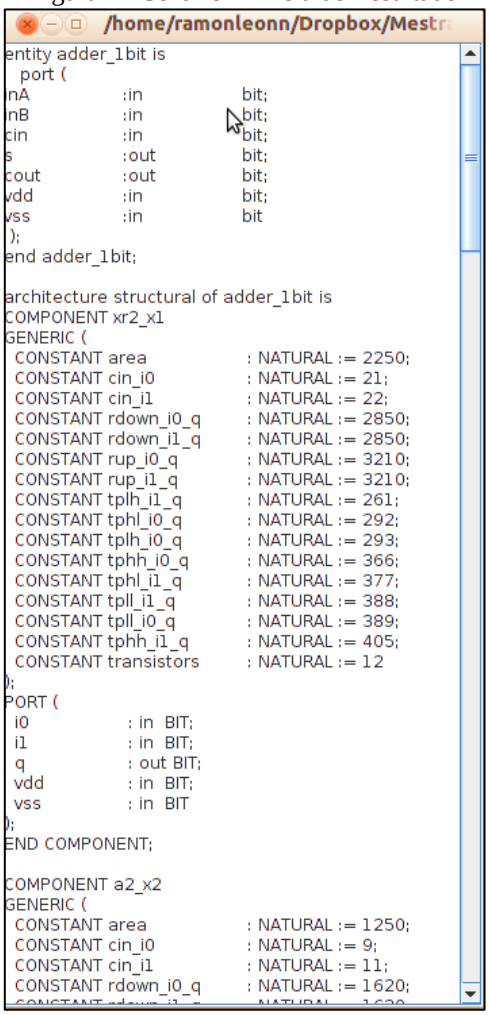


Fonte: Autor

Ao mostrar a janela para escolha do arquivo, a ferramenta filtra o tipo de arquivo para *.xml. Durante a geração do código VHDL pode aparecer algumas telas com alertas acerca de possíveis erros como incompatibilidade do arquivo *.xml ou a inexistência/ não configuração de atores para geração de código.

Ao final da geração de código, a ferramenta exibe uma tela com o resultado, o arquivo VHDL gerado e salvo como mostra a Figura 24.

Figura 24 Cenário 1 - Tela de Resultado



Fonte: Autor

O conteúdo do arquivo “adder_1bit.vst” – resultado da geração de código – é exibido na integra no Apêndice B.

De posse do arquivo a próxima etapa é verificação, que neste estudo foi feita utilizando ferramenta do Alliance CAD System.

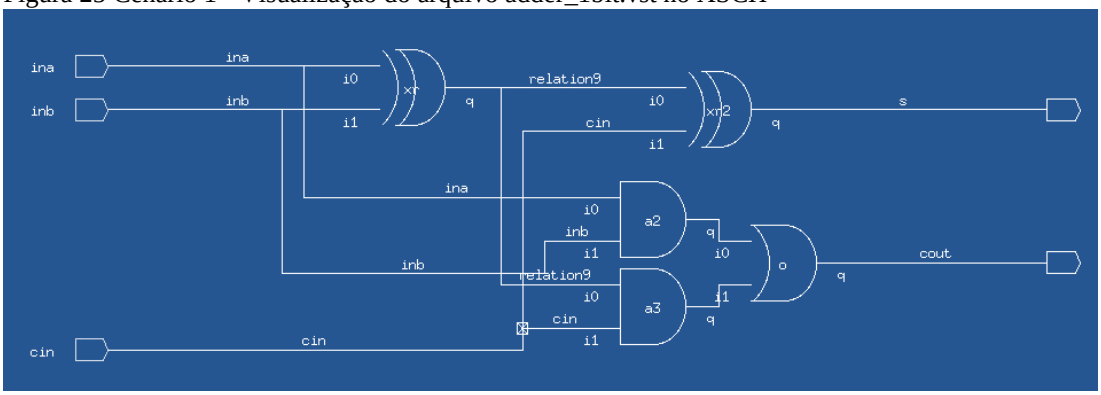
5.1.2 Verificação

Esta seção mostra a verificação feita no circuito gerado pelo Ptolemy. Para o circuito do somador além da verificação básica – visualização do circuito com XSCH[3] e compilação com Vasy – foi realizada uma verificação funcional com ASIMUT a partir de vetor de teste gerado com GENPAT.

[3] Software Visualizador Gráfico de Esquemáticos presente no Alliance CAD System.

Após a execução da ferramenta proposta neste trabalho e, como já se sabe, a geração de um arquivo em linguagem VHDL, um arquivo estrutural que podemos visualizá-lo utilizando o software XSCH.

Figura 25 Cenário 1 - Visualização do arquivo adder_1bit.vst no XSCH



Fonte: Autor

A Figura 25 mostra o circuito do somador contendo portas XOR, AND, OR, entradas/saídas e coneções. Esta visualização já pode ser considera, apesar de básica, uma verificação. Isso porque, visualmente é possível constatar que existem os componentes e as ligações entre eles.

Após a visualização, se utilizou o arquivo "adder_1bit.vst" no software Vasy com a finalidade de compilar e validar o VHDL gerado. Ao executar o Vasy, este cria arquivos com extensão *.vhd, são eles: "adder_1bit.vhd" e um arquivo *.vhd para cada componente utilizado da biblioteca de *standard cells*. A criação destes arquivos, possibilida a utilização do circuito em projeto utilizando ferramentas proprietárias, por exemplo.

Utilizando os softwares do Alliance possível realizar, também, verificação funcional dos circuitos. Para isso, inicialmente, é preciso desenvolver um programa em linguagem C utilizando as funções reconhecidar pelo GENPAT, software que gera vetores de teste, arquivos com extenção *.pat.

Código Fonte 5 Cenário 1 - Arquivo adder_1bit_genpat.c

```
#include <stdio.h>
#include "genpat.h"

/*Constante de Delay para o tempo*/
#define UNIT_TIME 15000

char *inttostr(entier)
int entier;
{
        char *str;
        str = (char *) mbkalloc (32 * sizeof (char));
```

```
		sprintf (str, "%d",entier);
		return(str);
}
/*
*Definindo Funções de teste
*/
int S (int a, int b, int cin) {
		int soma = a+b+cin;
		if (soma < 2) {
				return soma;
		}else{
				if (soma == 2) return 0;
				else return 1;
		}
}

int cout (int a, int b, int cin) {
		if ((a+b+cin)> 1) {	return 1;
		}else {		return 0; }
}

main ()
{
	int inA[16] ;
		int inB[16] ;
		int cin[16] ;
		int vect_date = 0;
		int x = 0;
		int i, j, k;
/*Definindo nome do arquito *.pat de saída*/
		DEF_GENPAT("adder_1bit_delay");

/*Definindo interface */
		DECLAR ("inA", ":2", "B", IN, "", "" );
		DECLAR ("inB", ":2", "B", IN, "", "" );
		DECLAR ("cin", ":2", "B", IN, "", "" );
		DECLAR ("s", ":2", "B", OUT, "", "" );
		DECLAR ("cout", ":2", "B", OUT, "", "" );
		DECLAR ("vdd", ":2", "B", IN, "", "" );
		DECLAR ("vss", ":2", "B", IN, "", "" );

/*Definindo LABEL e AFFECT para vdd e vss*/
		LABEL ("adder");
	AFFECT ("0", "vdd", "0b1");
	AFFECT ("0", "vss", "0b0");

/* Preenchendo os vetores 'a', 'b' e 'cin'*/
		for (i = 0; i <2; i++)
		{
				for (j = 0; j < 2; j++)
				{
						for(k = 0; k < 2; k++)
						{
								inA[2*x]=i;
								inA[(2*x)+1]=i;

								inB[2*x]=j;
								inB[(2*x)+1]=j;
								cin[2*x]=k;
								cin[(2*x)+1]=k;
								x++;
						}
				}
		}
/* AFFECT */
		x = 0;
		AFFECT (inttostr(vect_date), "inA", inttostr(inA[x]) );
		AFFECT (inttostr(vect_date), "inB", inttostr(inB[x]) );
		AFFECT (inttostr(vect_date), "cin", inttostr(cin[x]) );
```

```
        AFFECT (inttostr(vect_date), "S", "?0B*");
        AFFECT (inttostr(vect_date), "cout", "?0B*");
        vect_date += UNIT_TIME;
        AFFECT (inttostr(vect_date), "inA", inttostr(inA[x+1]) );
        AFFECT (inttostr(vect_date), "inB", inttostr(inB[x+1]) );
        AFFECT (inttostr(vect_date), "cin", inttostr(cin[x+1]) );
        AFFECT (inttostr(vect_date), "S", inttostr(S(inA[x+1],inB[x+1],cin[x+1])) );
        AFFECT (inttostr(vect_date), "cout", inttostr(cout(inA[x+1], inB[x+1], cin[x+1])) );

        for (x = 1; x < 8; x++)
        {
                vect_date += UNIT_TIME;
                AFFECT (inttostr(vect_date), "inA", inttostr(inA[2*x]) );
                AFFECT (inttostr(vect_date), "inB", inttostr(inB[2*x]) );
                AFFECT (inttostr(vect_date), "cin", inttostr(cin[2*x]) );
                AFFECT (inttostr(vect_date), "S",inttostr(S(inA[(2*x)-1],inB[(2*x)-1],cin[(2*x)-1])) );
                AFFECT (inttostr(vect_date), "cout", inttostr(cout(inA[(2*x)-1],inB[(2*x)-1],cin[(2*x)-1])));
                vect_date += UNIT_TIME;
                AFFECT (inttostr(vect_date), "inA", inttostr(inA[(2*x)+1]) );
                AFFECT (inttostr(vect_date), "inB", inttostr(inB[(2*x)+1]) );
                AFFECT (inttostr(vect_date), "cin", inttostr(cin[(2*x)+1]) );
                AFFECT (inttostr(vect_date), "S", inttostr(S(inA[(2*x)+1],inB[(2*x)+1],cin[(2*x)+1])) );
                AFFECT (inttostr(vect_date), "cout", inttostr(cout(inA[(2*x)+1], inB[(2*x)+1], cin[(2*x)+1])) );

        }
        SAV_GENPAT ();
}
```

Fonte: Autor

O Código Fonte 5 traz o conteúdo do arquivo em linguagem C utilizado para gerar os vetores de teste. A geração destes vetores só é possível a partir da utilização da biblioteca "genpat.h" como consta na segunda linha do código. É perceptível que as funções utilizadas não são convencionais na linguagem C.

De posse do Código Fonte 5 foi gerado o arquivo "adder_1bit_delay.pat" executando o seguinte comando:

$ genpat adder_1bit_genpat

O arquvo "adder_1bit_delay.pat" consta no Código Fonte 6.

Código Fonte 6 Cenário 1 - Arquivo adder_1bit_delay.pat

```
-- description generated by Pat driver
--          date     : Fri Aug  3 16:55:51 2012
--          revision : v109
--          sequence : adder_1bit_delay
-- input / output list :
in       ina B;;;
in       inb B;;;
in       cin B;;;
out      s B;;;
out      cout B;;;
in       vdd B;;;
in       vss B;;;

begin
-- Pattern description :
--                        i  i  c   s   c  v  v
--                        n  n  i       o  d  s
--                        a  b  n       u  d  s
--                                      t
-- Beware : unprocessed patterns
<          0 ps>adder_0 : 0  0  0  ?*  ?*  1  0  ;
<      15000 ps>        : 0  0  0  ?0  ?0  1  0  ;
<      30000 ps>        : 0  0  1  ?0  ?0  1  0  ;
<      45000 ps>        : 0  0  1  ?1  ?0  1  0  ;
<      60000 ps>        : 0  1  0  ?1  ?0  1  0  ;
<      75000 ps>        : 0  1  0  ?1  ?0  1  0  ;
<      90000 ps>        : 0  1  1  ?1  ?0  1  0  ;
<     105000 ps>        : 0  1  1  ?0  ?1  1  0  ;
<     120000 ps>        : 1  0  0  ?0  ?1  1  0  ;
<     135000 ps>        : 1  0  0  ?1  ?0  1  0  ;
<     150000 ps>        : 1  0  1  ?1  ?0  1  0  ;
<     165000 ps>        : 1  0  1  ?0  ?1  1  0  ;
<     180000 ps>        : 1  1  0  ?0  ?1  1  0  ;
<     195000 ps>        : 1  1  0  ?0  ?1  1  0  ;
<     210000 ps>        : 1  1  1  ?0  ?1  1  0  ;
<     225000 ps>        : 1  1  1  ?1  ?1  1  0  ;
end;
```

Fonte: Autor

Para dar continuidade a verificação funcional, há a necessidade de utilizar a ferramenta ASIMUT[4]. Esta ferramenta utiliza o arquivo “adder_1bit.vst” e o “adder_1bit_delay.pat”, sendo que este contém vetor de teste para aquele.

$ asimut adder_1bit adder_1bit_delay adder_1bit_res

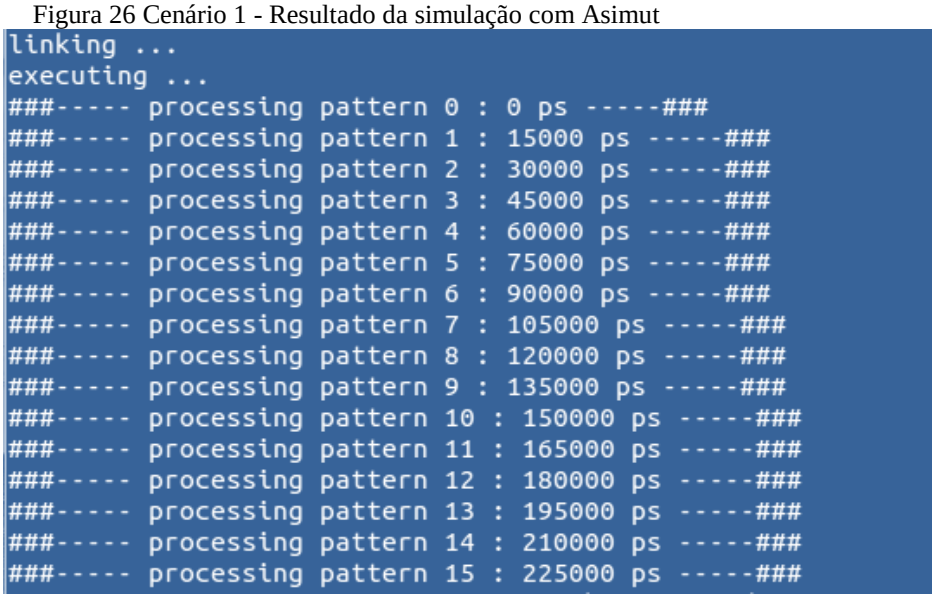
O comando acima, necessário para execução do ASIMUT, contem três parâmetros, são eles: o primeiro é o nome do arquivo com a descrição estrutural do circuito; o segundo é o

[4] Ferramenta de simulação para descrição de hardware presente no Alliance CAD System.

arquivo *.pat que contem o vetor de teste; e, por último, o nome do arquivo que guardará o resultado da simulação.

Além do arquivo de resposta, neste caso denominado “adder_1bit_res.pat”, o ASIMUT exibe a mensagem presente na Figura 26 quando a simulação é efetuada com sucesso.

Figura 26 Cenário 1 - Resultado da simulação com Asimut

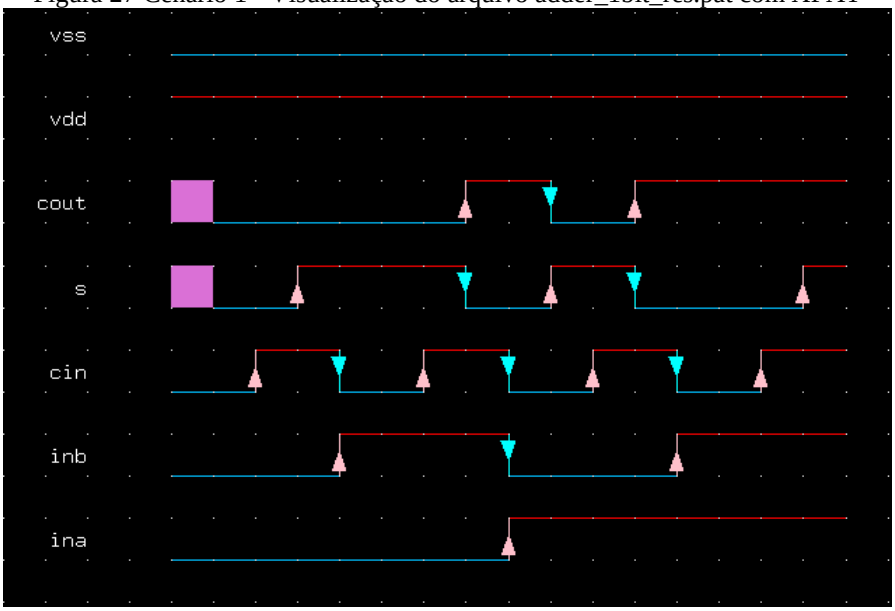


Fonte: Autor

O ASIMUT realiza uma simulação levando em conta os tempos de propagação dos componentes do circuito. Em caso de uma mensagem diferente da exibida na Figura 26 pode ser necessário uma modificação no arquivo que contem o vetor de teste, uma alteração no tempo de delay do circuito. Além desta mensagem, é possível, utilizando a ferramenta XPAT[5], visualizar de forma gráfica o arquivo “adder_1bit_res.pat”, como mostra a Figrura 27.

[5] Ferramenta de visualização gráfica de vetores de teste (patterns) do Alliance CAD System

Figura 27 Cenário 1 - Visualização do arquivo adder_1bit_res.pat com XPAT



Fonte: Autor

Na Figura 27 existe a variação de estado em função do tempo. Estes estados são as portas de entrada e saída do circuito estudado, tornando possível analisar visualmente o comportamento das saídas em função das entradas em cada tempo.

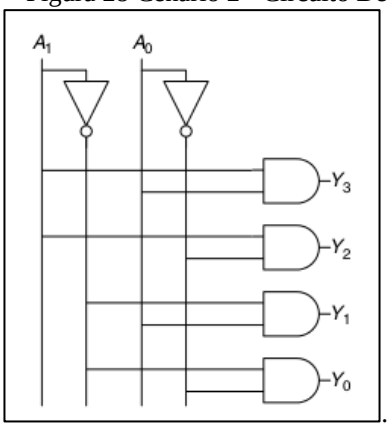
Assim, tanto com esta última quanto com as demais verificações utilizadas, pode-se validar a descrição criada em VHDL para o circuito modelado no Ptolemy.

5.2 Cenário 2: Decodificador

Nos circuitos combinacionais existem os codificadores/decodificadores. Enquanto os codificadores são circuitos que transformam uma representação de um número para binário, os decodificadores fazem o papel inverso. Estes pegam o número representado em binário e transformam em representação de chaveamento, ou seja, dado um número numa representação binária o decodificador vai resolver várias saídas em que cada saída representa um número.

Para este cenário foi escolhido um decodificador 2 para 4, como mostrado na Figura 28. Este necessita de 2 entradas e 4 saídas.

Figura 28 Cenário 2 - Circuito Decodificador 2x4

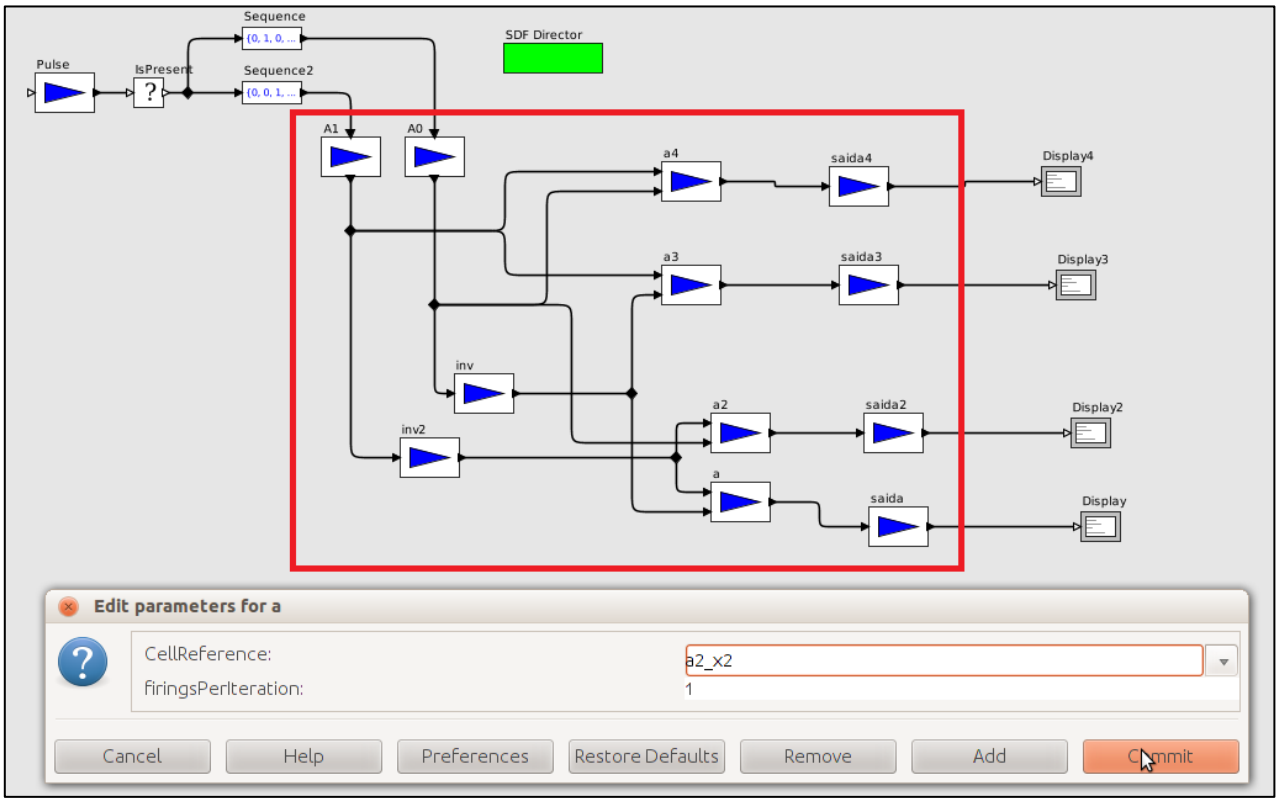


Fonte: Harris e Harris, 2007.

O circuito exibido na Figura 28 mostra a implementação do decodificador com quatro portas lógicas AND. Em decorrência da necessidade de utilização destes atores, foram criados atores correspondentes às portas citadas.

Como é possível verificar destacados na lateral esquerda da Figura 29 os atores criados para este cenário foram: a3, que é a porta AND com três entradas; e o ator inv, que é uma porta inversora. Os atores nomeados como “A0”, “A1”, ”saida”, ”saida2”, ”saida3” e ”saida4” são do tipo IOModel, que serve como delimitação do modelo e identificação das portas de entrada/saída.

Figura 29 Cenário 2 - Decodificador modelado no Ptolemy



Fonte: Autor

Quando modelado o circuito precisa ser simulado. Para isso tem-se que gerar estímulos para que o circuito possa responder, de acordo com seu processamento, para ser validado quanto a sua efetividade.

Ainda na Figura 29, na parte inferior, é exibida a tela de configuração do parâmetro "CellReference". Parâmetro que foi criado para identificar a *standard cell* (ou o *helper*) correspondente ao ator quando este faz parte do circuito o qual vai ser gerado o código em linguagem de descrição de hardware.

O Arquivo "dec2x4.xml" está no Apêndice C.

É importante destacar que para a correta geração do código VHDL é necessário, ainda durante a modelagem no Ptolemy, configurar o parâmetro "*CellReference*" para os atores do circuito estudado.

5.2.1 Resultado

Ao escolher o arquivo correspondente ao circuito que se deseja gerar o arquivo VHDL correspondente, a ferramenta mostra o resultado e o arquivo VHDL é salvo.

Ao mostrar a janela para escolha do arquivo, a ferramenta filtra o tipo de arquivo para *.xml.

O conteúdo do arquivo “dec2x4.vst” – resultado da geração de código – é exibido na integra no Código Fonte 9.

Código Fonte 7 Cenário 2 - Arquivo dec2x4.vst

```
entity dec2x4 is
   port (
A0          :in          bit;
A1          :in          bit;
saida       :out         bit;
saida2      :out         bit;
saida3      :out         bit;
saida4      :out         bit;
vdd         :in          bit;
vss         :in          bit
 );
end dec2x4;

architecture structural of dec2x4 is
COMPONENT inv_x1
GENERIC (
  CONSTANT area   : NATURAL := 750;
  CONSTANT cin_i   : NATURAL := 8;
  CONSTANT rdown_i_nq       : NATURAL := 3640;
  CONSTANT rup_i_nq         : NATURAL := 3720;
  CONSTANT tphl_i_nq        : NATURAL := 101;
  CONSTANT tplh_i_nq        : NATURAL := 139;
  CONSTANT transistors      : NATURAL := 2
);
PORT (
  i           : in  BIT;
  nq          : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

COMPONENT a2_x2
GENERIC (
  CONSTANT area   : NATURAL := 1250;
  CONSTANT cin_i0 : NATURAL := 9;
  CONSTANT cin_i1 : NATURAL := 11;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tphh_i1_q        : NATURAL := 203;
  CONSTANT tphh_i0_q        : NATURAL := 261;
  CONSTANT tpll_i0_q        : NATURAL := 388;
  CONSTANT tpll_i1_q        : NATURAL := 434;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

signal relation : bit;
signal relation10 : bit;
```

```
signal relation11 : bit;
signal relation12 : bit;
signal relation13 : bit;
signal relation14 : bit;
signal relation2 : bit;
signal relation18 : bit;
signal relation3 : bit;
signal relation16 : bit;
signal relation8 : bit;
signal relation9 : bit;
signal relation6 : bit;
signal relation7 : bit;
signal relation4 : bit;
signal relation5 : bit;

begin

inv : inv_x1
Generic Map(
   area       => 750,
   cin_i      => 8,
   rdown_i_nq          => 3640,
   rup_i_nq            => 3720,
   tphl_i_nq           => 101,
   tplh_i_nq           => 139,
   transistors         => 2
)
port map (
  i => A0 ,
  vdd => vdd ,
  vss => vss ,
  nq => relation7
 );

inv2 : inv_x1
Generic Map(
   area       => 750,
   cin_i      => 8,
   rdown_i_nq          => 3640,
   rup_i_nq            => 3720,
   tphl_i_nq           => 101,
   tplh_i_nq           => 139,
   transistors         => 2
)
port map (
  i => A1 ,
  vdd => vdd ,
  vss => vss ,
  nq => relation
 );

a : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => relation ,
  i1 => relation7 ,
```

```
  vdd => vdd ,
  vss => vss ,
  q => saida
 );

a2 : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => relation ,
  i1 => A0 ,
  vdd => vdd ,
  vss => vss ,
  q => saida2
 );

a3 : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => A1 ,
  i1 => relation7 ,
  vdd => vdd ,
  vss => vss ,
  q => saida3
 );

a4 : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => A1 ,
  i1 => A0 ,
```

```
  vdd => vdd ,
  vss => vss ,
  q => saida4
 );

end structural;
```

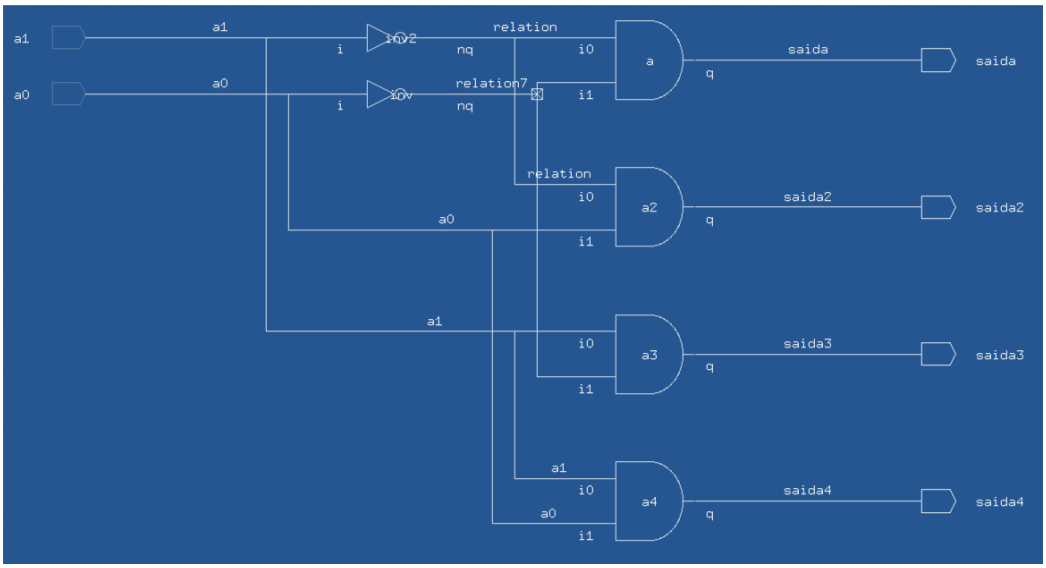
Fonte: autor

De posse do arquivo a próxima etapa é verificação, que neste estudo foi feita utilizando ferramenta do Alliance CAD System.

5.2.2 Verificação

Esta seção mostra a verificação feita no circuito gerado pelo Ptolemy. Para o circuito do decodificador foi, inicialmente, realizada uma visualização do circuito com o XSCH.

Figura 30 Cenário 2 - Visualização do arquivo dec2x4.vst no XSCH



Fonte: Autor

A Figura 30 mostra o circuito do somador contendo portas lógicas NOT, AND, entradas/saídas e coneções. Esta visualização já pode ser considera, apesar de básica, uma verificação. Isso porque, visualmente é possível constatar que existem os componentes e as ligações entre eles.

Após a visualização, se utilizou o arquivo "dec2x4.vst" no software Vasy com a finalidade de compilar e validar o VHDL gerado.

Código Fonte 8 Cenário 2 - Arquivo dec2x4.vhd

```
--
-- Generated by VASY
--
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;
```

```
ENTITY dec2x4 IS
PORT(
  a0        : IN BIT;
  a1        : IN BIT;
  saida     : OUT BIT;
  saida2    : OUT BIT;
  saida3    : OUT BIT;
  saida4    : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
);
END dec2x4;

ARCHITECTURE RTL OF dec2x4 IS
  SIGNAL relation    : BIT;
  SIGNAL relation7   : BIT;
  COMPONENT inv_x1_inv
  PORT(
  i         : IN BIT;
  nq        : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

  COMPONENT inv_x1_inv2
  PORT(
  i         : IN BIT;
  nq        : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

  COMPONENT a2_x2_a
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

  COMPONENT a2_x2_a2
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

  COMPONENT a2_x2_a3
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

  COMPONENT a2_x2_a4
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
```

```
  vdd       : IN BIT;
  vss       : IN BIT
   );
  END COMPONENT;

BEGIN
  a4 : a2_x2_a4
  PORT MAP (
    i0 => a1,
    i1 => a0,
    vdd => vdd,
    vss => vss,
    q => saida4
  );
  a3 : a2_x2_a3
  PORT MAP (
    i0 => a1,
    i1 => relation7,
    vdd => vdd,
    vss => vss,
    q => saida3
  );
  a2 : a2_x2_a2
  PORT MAP (
    i0 => relation,
    i1 => a0,
    vdd => vdd,
    vss => vss,
    q => saida2
  );
  a : a2_x2_a
  PORT MAP (
    i0 => relation,
    i1 => relation7,
    vdd => vdd,
    vss => vss,
    q => saida
  );
  inv2 : inv_x1_inv2
  PORT MAP (
    i => a1,
    vdd => vdd,
    vss => vss,
    nq => relation
  );
  inv : inv_x1_inv
  PORT MAP (
    i => a0,
    vdd => vdd,
    vss => vss,
    nq => relation7
  );
END RTL;
```

Fonte: Autor

Ao executar o Vasy, este cria arquivos com extensão *.vhd, são eles: “dec2x4.vhd” e um arquivo *.vhd para cada componente utilizado da biblioteca de *standard cells*. A criação destes arquivos, possibilida a utilização do circuito em projeto utilizando ferramentas proprietárias, por exemplo.

5.3 Cenário 3: Latch D

Latch é um circuito digital sequencial. Na lógica sequencial, as saídas não dependem apenas dos valores atuais mais também de valores anteriores (HARRIS E HARRIS, 2007).

É um circuito constituído por portas lógicas, capaz de armazenar um bit de informação, onde as saídas de certo instante dependem dos valores de entrada do instante mais os valores anteriores de saída, isto é, do seu estado atual, e onde as saídas mudam a qualquer instante de tempo, podendo ter ou não variáveis de controle.

O mais básico é o Latch NAND SR que é um dos tipos existentes de Latches SR, este é composto por duas portas NAND que ficam interconectas, contando com os bits de entrada S e R, sendo o R a entrada de RESET e a S sendo a entrada de SET, e um bit que armazena a saída do Latch , geralmente representada por ‘Q’, e uma outra ‘/Q’ que é o seu complemento.

Entretanto, foi escolhido para o experimento o Latch D que possui duas entradas (D e CLK) e duas saídas (‘Q’ e ‘/Q’). Sua característica principal de funcionamento é transferir para a saída Q o valor da entrada de dados D sempre que CLK for 1, e manter o mesmo estado na saída se CLK for 0.

A utilização deste latch iniciou a partir da necessidade de evitar, no latch RS, a ocorrência do estado proibido (TANENBAUM, 2007, p.93). É construído a partir deste ao se colocar um inversor entre as entradas R e S, evitando assim que R=1 e S=1 simultaneamente, o que permitia a ocorrência do estado proibido. Desta maneira, R e S passam a ser denominados D (onde D=S).

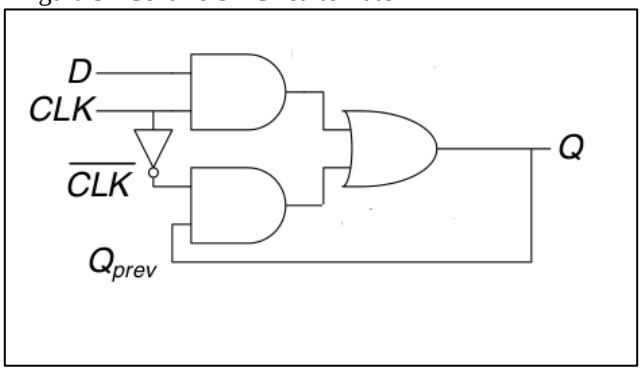
Tabela 1 Tabela-Verdade do Latch D

Enable	**D**	**$Q_{próximo}$**
0	X	Q
1	0	0
1	1	1

Fonte:

Como mostra sua tabela verdade, a saída ‘Q’ segue a entrada D, e por isto o latch D é às vezes chamado latch transparente.

Figura 31 Cenário 3 - Circuito Latch D

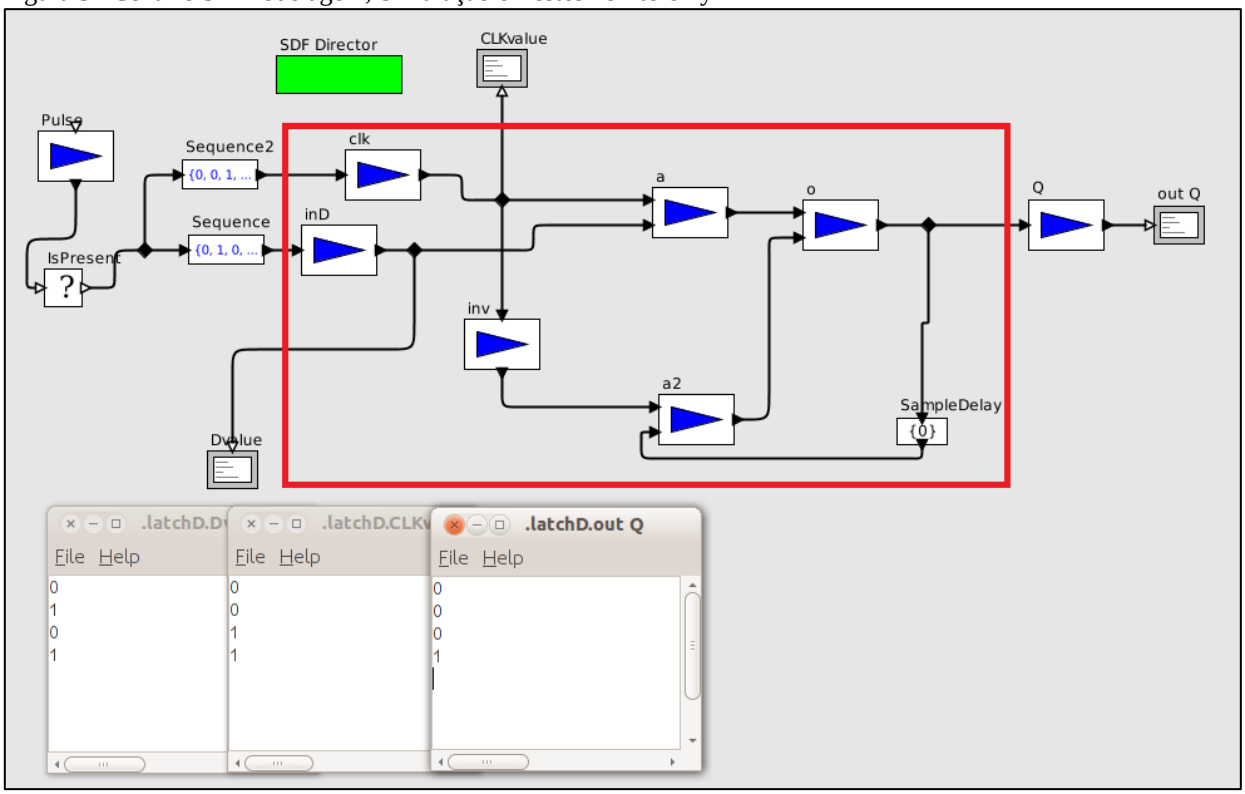


Fonte: Harris e Harris (p.114, 2007)

Harris e Harris (p.114, 2007) propõe o latch D como descrito na Figura 31. O Latch é desenvolvido com apenas duas portas AND, uma porta OR e uma NOT. As portas AND recebem os sinais de D e CLK, CLK invertido e Q_{prev} (estado anterior da saída Q). As saídas das duas portas AND são destinadas à entrada da porta OR, que por suas vez gera a saída Q.

Já no Ptolemy, o circuito foi modelado como mostra a Figura 32. Nele há a presença de um ator denominado "SampleDaley" que ainda não tinha sido utilizado nos outros cenários. Este ator serve para, no Ptolemy, evitar loop e/ou "*deadlock*".

Figura 32 Cenário 3 - Modelagem, Simulação e Testes no Ptolemy



Fonte: Autor

O circuito correspondente ao que consta Figura 31 está destacado, na Figura 32, pelo retângulo vermelho. Já no canto inferior da Figura 32 é possível verificar as telas de saída do circuito, onde os valores correspondem ao valor de D, clk e Q, se visto da esquerda para a direita. Ainda sobre a modelagem, é importante destacar que utilização do ator *SampleDalay* é um forma de superar limitações ainda existentes no *framework* Ptolemy.

O Arquivo "latch_D.xml" está no Apêndice D.

No caso deste circuito, por ser sequencias e, assim, necessitar da utilização do ator *SampleDaley*, este não será considerado no momento da geração de código.

5.3.1 Resultado

Ao iniciar a ferramenta e visualizar a janela para escolha do arquivo, acontece a geração do código VHDL correspondente ao circuito modelado.

O conteúdo do arquivo "latchD.vst" – resultado da geração de código – é exibido na integra no Código Fonte 12.

Código Fonte 9 Cenário 3 - Arquivo latchD.vst

```
--
-- Generated by VASY
--
ENTITY latchd IS
PORT(
  clk        : IN BIT;
  ind        : IN BIT;
  saida      : OUT BIT;
  vdd        : IN BIT;
  vss        : IN BIT
);
END latchd;

ARCHITECTURE VST OF latchd IS

  SIGNAL rtl_internal_saida     : BIT;
  SIGNAL relation15: BIT;
  SIGNAL relation7  : BIT;
  SIGNAL relation12: BIT;

  COMPONENT latchd_model
  PORT(
  saida     : OUT BIT;
  rtl_internal_saida   : IN BIT
  );
  END COMPONENT;

  COMPONENT o2_x2_o
  PORT(
  i0         : IN BIT;
  i1         : IN BIT;
  q          : OUT BIT;
  vdd        : IN BIT;
  vss        : IN BIT
  );
```

```
  END COMPONENT;

  COMPONENT inv_x1_inv
  PORT(
  i         : IN BIT;
  nq        : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
  );
  END COMPONENT;

  COMPONENT a2_x2_a2
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
  );
  END COMPONENT;

  COMPONENT a2_x2_a
  PORT(
  i0        : IN BIT;
  i1        : IN BIT;
  q         : OUT BIT;
  vdd       : IN BIT;
  vss       : IN BIT
  );
  END COMPONENT;

BEGIN
  latchd_inst : latchd_model
  PORT MAP (
    saida => saida,
    rtl_internal_saida => rtl_internal_saida
  );
  a : a2_x2_a
  PORT MAP (
    q => relation12,
    vss => vss,
    vdd => vdd,
    i1 => ind,
    i0 => clk
  );
  a2 : a2_x2_a2
  PORT MAP (
    q => relation15,
    vss => vss,
    vdd => vdd,
    i1 => rtl_internal_saida,
    i0 => relation7
  );
  inv : inv_x1_inv
  PORT MAP (
    nq => relation7,
    vss => vss,
    vdd => vdd,
    i => clk
  );
  o : o2_x2_o
  PORT MAP (
    q => rtl_internal_saida,
    vss => vss,
    vdd => vdd,
    i1 => relation15,
    i0 => relation12
  );
END VST;
```

Fonte: autor

De posse do arquivo a próxima etapa é verificação, que neste estudo foi feita utilizando ferramenta do Alliance CAD System.

5.3.2 Verificação

Utilizando os softwares do Alliance possível realizar, também, verificação funcional do circuito. Para isso, inicialmente, é preciso desenvolver um programa em linguagem C utilizando as funções reconhecidas pelo GENPAT, software que gera vetores de teste, arquivos com extenção *.pat.

Código Fonte 10 Cenário 3 - Arquivo latchD_genpat.c

```
#include <stdio.h>
#include "genpat.h"

/*Constante de Delay para o tempo*/
#define UNIT_TIME 15000

char *inttostr(entier)
int entier;
{
        char *str;
        str = (char *) mbkalloc (32 * sizeof (char));
        sprintf (str, "%d",entier);
        return(str);
}
/*
*Definindo Funções de teste
*/

int output (int ind, int clk, int saida) {
        if (clk == 1) {
                return ind;
        }else{
                return saida;
        }
}

main ()
{
    int clk[8] ;
        int ind[8] ;
        int saida = 0;
        int vect_date = 0;
        int x = 0;
        int i, j;
/*Definindo nome do arquito *.pat de saída*/
        DEF_GENPAT("latchd_delay");

/*Definindo interface */
        DECLAR ("ind", ":2", "B", IN, "", "" );
        DECLAR ("clk", ":2", "B", IN, "", "" );
        DECLAR ("saida", ":2", "B", OUT, "", "" );
        DECLAR ("vdd", ":2", "B", IN, "", "" );
        DECLAR ("vss", ":2", "B", IN, "", "" );

/*Definindo LABEL e AFFECT para vdd e vss*/
        LABEL ("adder");
    AFFECT ("0", "vdd", "0b1");
    AFFECT ("0", "vss", "0b0");

/* Preenchendo os vetores 'ind' e 'clk'*/
```

```
	for (i = 0; i <2; i++)
	{
		for (j = 0; j < 2; j++)
		{
			if (i ==0){
				ind[2*x]=i+j;
				ind[(2*x)+1]=i+j;
			}else{
				ind[2*x]=j;
				ind[(2*x)+1]=j;
			}
			clk[2*x]=0;
			clk[(2*x)+1]=1;
			x++;
		}
	}

/* AFFECT */
	x = 0;
	AFFECT (inttostr(vect_date), "ind", inttostr(ind[x]));
	AFFECT (inttostr(vect_date), "clk", inttostr(clk[x]));
	AFFECT (inttostr(vect_date), "saida", "?0B*");

	vect_date += UNIT_TIME;
	AFFECT (inttostr(vect_date), "ind", inttostr(ind[x+1]) );
	AFFECT (inttostr(vect_date), "clk", inttostr(clk[x+1]) );
	AFFECT (inttostr(vect_date), "saida", inttostr(output(ind[x+1],clk[x+1],saida) ));

	for (x = 1; x < 4; x++)
	{
		vect_date += UNIT_TIME;
		AFFECT (inttostr(vect_date), "ind", inttostr(ind[2*x]) );
		AFFECT (inttostr(vect_date), "clk", inttostr(clk[2*x]) );
		AFFECT (inttostr(vect_date), "saida", inttostr( output(ind[(2*x)-1],clk[(2*x)-1],saida) ));

		vect_date += UNIT_TIME;
		AFFECT (inttostr(vect_date), "ind", inttostr(ind[(2*x)+1]) );
		AFFECT (inttostr(vect_date), "clk", inttostr(clk[(2*x)+1]) );
		AFFECT (inttostr(vect_date), "saida", inttostr( output(ind[(2*x)+1], clk[(2*x)+1],saida) ));
	}
	SAV_GENPAT ();
}
```

Fonte: Autor

O Código Fonte 13 traz o conteúdo do arquivo em linguagem C utilizado para gerar os vetores de teste. A geração destes vetores só é possível a partir da utilização da biblioteca “genpat.h” como consta na segunda linha do código. É perceptível que as funções utilizadas não são convencionais na linguagem C.

De posse do Código Fonte 13 foi gerado o arquivo “adder_1bit_delay.pat” executando o seguinte comando:

$ genpat adder_1bit_genpat

O arquvo “adder_1bit_delay.pat” consta no Código Fonte 14.

Código Fonte 11 Cenário 3 - latchd_delay.pat

```
-- description generated by Pat driver
--          date     : Tue Aug  7 01:47:32 2012
--          revision : v109
--          sequence : latchd_delay

-- input / output list :
in       ind B;;;
in       clk B;;;
out      saida B;;;
in       vdd B;;;
in       vss B;;;

begin
-- Pattern description :
--                         i  c   s  v  v
--                         n  l   a  d  s
--                         d  k   i  d  s
--                               d
--                               a
-- Beware : unprocessed patterns
<          0 ps>adder_0 : 0  0  ?*  1  0  ;
<      15000 ps>        : 0  1  ?0  1  0  ;
<      30000 ps>        : 1  0  ?0  1  0  ;
<      45000 ps>        : 1  1  ?1  1  0  ;
<      60000 ps>        : 0  0  ?1  1  0  ;
<      75000 ps>        : 0  1  ?0  1  0  ;
<      90000 ps>        : 1  0  ?0  1  0  ;
<     105000 ps>        : 1  1  ?1  1  0  ;
end;
```

Fonte: Autor

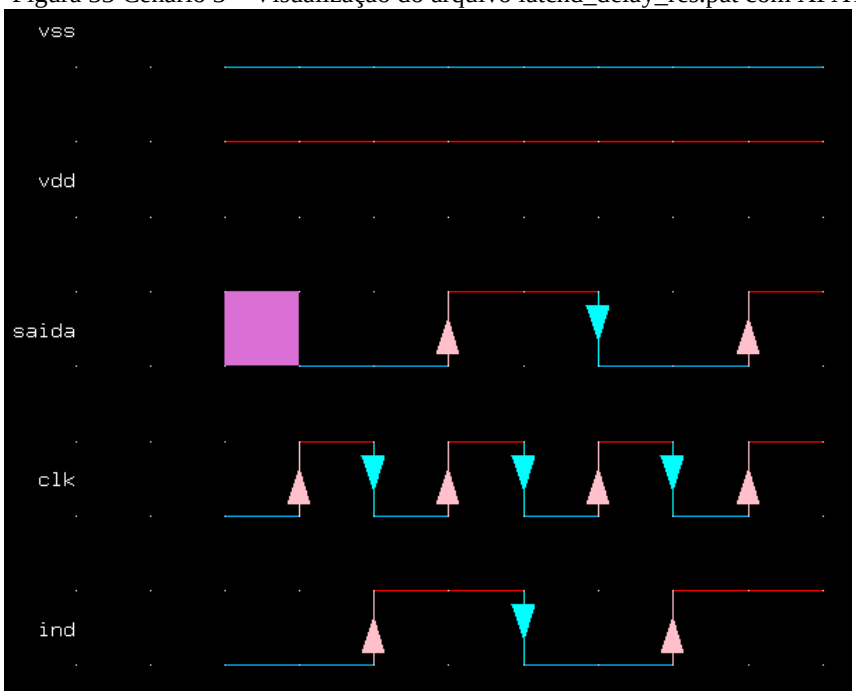
Para dar continuidade a verificação funcional é necessário utilizar a ferramenta ASIMUT. Esta ferramenta utiliza o arquivo “latchD.vst” e o “latchd_delay.pat”, sendo que este contém vetor de teste para aquele.

$ asimut latchD latchd_delay latchd_delay_res

O comando acima, necessário para execução do ASIMUT, contem três parâmetros, são eles: o primeiro é o nome do arquivo com a descrição estrutural do circuito; o segundo é o arquivo *.pat que contem o vetor de teste; e, por último, o nome do arquivo que guardará o resultado da simulação.

O ASIMUT realiza uma simulação levando em conta os tempos de propagação dos componentes do circuito. É possível, utilizando a ferramenta XPAT, visualizar de forma gráfica o arquivo de resposta, neste caso denominado “latchd_delay_res.pat”, como mostra a Figrura 33.

Figura 33 Cenário 3 – Visualização do arquivo latchd_delay_res.pat com XPAT



Fonte: Autor

Na Figura 33 existe a variação do sinal ‘saída’ em função do tempo representado pelo sinal ‘clk’ e da entrada ‘ind’. Analisando ‘saida’ e ‘clk’ é possível perceber que o sinal ‘saida’ só muda quando o ‘clk’ está mudando de baixo para alto, ouseja, de 0 para 1. Momento no qual a ‘saida’ passa a ter um valor igual a ‘ind’.

Assim, tanto com esta última quanto com as demais verificações utilizadas, pode-se validar a descrição criada em VHDL para o circuito modelado no Ptolemy.

6 Considerações Finais

Neste trabalho foi estudada a viabilidade do Modelo de Atores implementado a partir do Ptolemy, servir como ferramenta de modelagem/simulação de circuitos e geração de código em linguagem de descrição de hardware, neste caso VHDL.

A ferramenta foi implementada para ser incorporada e, assim, utilizada em conjunto com o Ptolemy, que utiliza linguagem Java. Linguagem esta, também, utilizada no desenvolvimento da ferramenta proposta, principalmente devido à sua independência de plataforma.

Para o desenvolvimento da ferramenta foram estudados conceitos de modelagem orientada a objetos, linguagem de marcação XML e fluxo de projeto de hardware, sem esquecer, dos estudos sobre o Ptolemy e linguagens de descrição de hardware. Tudo isso, com objetivo de desenvolver uma ferramenta eficiente.

Ao final do trabalho obtemos um protótipo que gera código HDL. Apesar da simplicidade dos circuitos validados, estes representam que a relação do Ptolemy (atores e modelos) com o VHDL (*standard cells* e arquivo final) foi construída corretamente e validada por ferramentas presentes no Alliance CAD System.

6.1 Contribuições

Este trabalho buscou desenvolver uma ferramenta para a geração de código em linguagem de descrição de hardware (HDL) que possibilitasse uma visualização gráfica e simulação a partir da utilização de modelo de atores, se apresentando como possibilidade acessível a pesquisadores, instituições e alunos como objeto facilitador do processo de desenvolvimento e/ou ensino aprendizagem de microeletrônica.

Durante o desenvolvimento deste trabalho foi possível estudar e desenvolver novos atores para o Ptolemy. Não apenas os atores básicos utilizados nos cenários demonstrados anteriormente, mas também foi possível modelar um processador básico e simular no Ptolemy.

Os atores criados durante o trabalho vão desde modelos avulsos até itens de bibliotecas de *standard cell*, como a utilizada para demostrar o desenvolvimento dos cenários no capítulo 5.

A partir dos estudos executados foi possível obter subsídio para o desenvolvimento do objetivo principal deste trabalho. Desenvolver uma ferramenta que fosse capaz de ler e identificar um arquivo com extensão *.xml do Ptolemy, identificar os atores presentes no modelo que

correspondessem a itens da biblioteca de *standard cell* e os mapear para um arquivo em linguagem VHDL.

Apesar da interface simplista que a ferramenta apresenta e as funcionalidades básica, ela cumpre seu papel: gerar código VHDL a partir de um modelo gerado no Ptolemy.

O aumento da complexidade dos modelos resultará, inevitavelmente, em modificações na ferramenta desenvolvida. Contudo, este trabalho eleva e destaca a capacidade do Ptolemy de servir como base para implementação de ferramentas com o objetivo desenvolvido aqui. Lembrando que o grande destaque é a possibilidade de ter uma ferramenta com as propriedades que o Ptolemy detém – simulação concorrente e heterogênea, *open-source*, entre outras – e a capacidade de gerar código HDL.

Mostramos também que o código HDL gerado é totalmente verificável com as ferramentas presentes no Alliance CAD System como: Vasy, Asimut, XSCH e Xpat. Não só isso, há algumas validações em andamento que indicam a possibilidade de verificar o HDL gerado por nossa ferramenta utilizando software proprietário.

Conhecendo tudo que foi explicitado, passamos a imaginar o que pode e vai ser desenvolvido tanto na ferramenta aqui desenvolvida quanto no Ptolemy, e na relação entre estas. A seguir falamos de algumas das possibilidades que podem ser adotas em trabalhos futuros.

6.2 Trabalhos Futuros

No capítulo 4, apresenta-se uma descrição da ferramenta desenvolvida neste trabalho. É importante ressaltar que nem todas as funções necessárias para utilização plena da capacidade do Ptolemy foram implementadas no protótipo da ferramenta e, portanto, serão implementadas futuramente. Tanto antes quanto durante a implementação, observaram-se novas funcionalidades que poderiam ser incorporadas à ferramenta, algumas destas serão descritas nas próximas seções.

6.2.1 Salvar modelos criados como novos atores

A ideia é possibilitar que os modelos criados, simulados e que tenham seus códigos HDL gerados, possam ser salvos para utilização posterior. De forma que estes circuitos sejam exibidos como novos atores da biblioteca, evitando assim, que diante da necessidade não seja preciso modelá-los para inseri-los em outros modelos.

6.2.2 Possibilitar a criação de atores e código HDL no momento da modelagem

Se houver a necessidade de um novo ator para a modelagem de um circuito que não exista um arquivo auxiliar, que a ferramenta possibilite a criação do ator e do seu código HDL auxiliar, os relacionando e, permitindo assim, a geração de código HDL.

6.2.3 Gerar código em outras linguagens de descrição de hardware

Neste primeiro protótipo, o a ferramenta gera descrições do circuito usando a linguagem VHDL. Porém, é desejado que a ferramenta possibilitasse a geração de descrições em mais de um formato padrão. Linguagens como *System Verilog*, *System* C, entre outras, devem ser estudadas e a geração de descrições nestes formatos deve ser inserida na ferramenta.

6.2.4 Utilizar como ferramenta educacional

Utilizar o potencial de modelagem e simulação do Ptolemy em conjunto com a geração de código da ferramenta proposta aqui e, ainda, as características educacionais do Alliance CAD System para subsidiar o ensino de microeletrônica.

6.2.5 Disponibilizar como ferramenta de ensino à distância

Pelo fato do Ptolemy e da ferramenta proposta neste trabalho serem desenvolvidos em linguagem Java, é possível adapta-los para utilização na modalidade de ensino à distância. Modalidade que requer, entre outras características, sistemas funcionais independentes de plataforma.

6.2.6 Teste com FPGA

Modelar, simular e gerar código HDL faz mais sentido se for para ver o circuito em funcionamento. Por isso, entendemos que facilitar a programação de FPGA a partir da ferramenta facilitaria esta utilização.

Referências

BERTASI, D. Implementação de um sistema de Síntese de Alto Nível baseado em modelos Java. Porto Alegre: PPGC da UFRGS. 2002.

BRITO, A. V. Simulação Baseada em Atores para o Ensino de Arquitetura de Computadores. In: WORKSHOP SOBRE EDUCAÇÃO EM ARQUITETURA DE COMPUTADORES (WEAC), 60, 2009, São Paulo. **Anais...** São Paulo: Simpósio Brasileiro de Arquitetura de Computadores (SBAC), 2009.

COYLE, F. P. e THORNTON, M. A. From UML to HDL: a Model Driven Architectural Approach to Hardware-Software Co-Design. Computer Science and Engineering Dept. Southern Methodist University. 2005.

DAVIS, J. Order and Containment in Concurrent System Design," Ph.D. thesis, Memorandum UCB/ERL M00/47, Electronics Research Laboratory, University of California, Berkeley, 2000.

D'AMORE, R. VHDL: descrição e síntese de circuitos digitais. LTC

FILIBA, T.E.; LEUNG, M. e NAGPAL, V. VHDL Code Generation in the Ptolemy II Environment. Electrical Engineering and Computer Sciences, University of California at Berkeley. Technical Report No. UCB/EECS-2008-140. 2006

GROUT, I. Digital systems design with FPGAs and CPLDs. USA: Elsevier, 2008.

HARRIS, D. M.; HARRIS S. L. **Digital Design and Computer Architecture**. 1ed. Editora Elsevier, 2007.

HEXSEL, R. A, CARMO, R. Ensino de Arquitetura de Computadores com enfoque na interface Hardware/Software. Apresentado no Workshop sobre Educação em Arquitetura de Computadores (WEAC) – Simpósio Brasileiro de Arquitetura de Computadores. SBAC, Ouro Preto - MG, 2006.

KAESLIN, H. Digital Integrated Circuit Design from VLSI Architecture to CMOS Fabrication. Cambridge University Press: Cambridge. 2008

KULLER, R. L. Simulador de redes orientado a eventos para o sistema operacional Linux. Dissertação para obtenção do título de mestre em Engenharia Elétrica. Universidade Estadual de Campinas – Unicamp. Campinas – SP. 2003.

LEE, E. A. Overview of the Ptolemy Project, Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, EUA. 2003.

LEE, E. A. AND NEUENDORFFER, S. Concurrent Models of Computation for Embedded Software, *Technical Memorandum UCB/ERL M04/26*, University of California, Berkeley, CA 94720, July 22, 2004.

LEE, A. et al. Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II). Technical Report No. UCB/EECS-2005, 2005.

LEE, A. et al. Tutorial: Building Ptolemy II Models Graphically. Technical Report No. UCB/EECS-2007-129, 2007.

LEE, A. et al. Heterogeneous Concurrent Modeling and Design in Java (Volume 2: Ptolemy II Software Architecture). Technical Report No. UCB/EECS-2008-29, 2008.

LEE, A. et al. Heterogeneous Concurrent Modeling and Design in Java (Volume 3: Ptolemy II Domains). Technical Report No. UCB/EECS-2008-37, 2008.

LIP6, Laboratoire d'Informatique de Paris 6. Alliance: A Complete CAD System for **VLSI** Design. Équipe Achitecture des Systèmes et Micro-Électronique. Université Pierre et Marie Curie. Disponível em: <> Acesso em: 15 Maio 2011

LIU, J. Continuous Time and Mixed-Signal Simulation in Ptolemy II, MS Report, UCB/ERL Memorandum M98/74, Dept. of EECS, University of California, Berkeley, CA 94720, 1998.

LIU, J., XIONG, Y. AND LEE, E. A. The Ptolemy II Framework for Visual Languages. Proceedings of the IEEE Symposia on Human-Centric Computing Languages and Environments (HCC'01), 2001.

MARTINS, C. A. P. S; POUSA, C. V.; CARVALHO, M. B.; PENHA, D. O. Computer Architecture Education: Application of a New Learning Method Based on Design and Simulator Development. Proceedings of the Frontiers in Education Conference (FIE) - 2003

MAINART, D. DE A1. SANTOS, C. M. A Importância da Tecnologia Processo Ensino-Aprendizagem. Faculdade Presidente Antonio Carlos1, Universidade Federal dos Vales do Jequitinhonha e Mucuri – UFVJM.

MEC. Diretrizes Curriculares de Cursos de Computação. Versão final disponível em www.mec.gov.br/Ftp/sesu/diretriz/**Computa**.doc, 1998.

MORAN, J.M. Ensino e aprendizagem inovadores com tecnologias. Rev. ***Informática na Educação: Teoria & Prática***. Porto Alegre, vol. 3, n.1 (set. 2000) UFRGS. Programa de Pós-Graduação em Informática na Educação, pág. 137-144.

MOREIRA, M. P; FAVERO, E. L. Um Ambiente para Ensino de Programação com Feedback Automático de Exercícios. XVII Workshop sobre Educação em Computação (WEI09) – XXIX Congresso da Sociedade Brasileira de Computação (CSBC09), Bento Gonçalves – RS, 2009.

MULIADI, L. Discrete Event Modeling in Ptolemy II," MS Report, Dept. of EECS, University of California, Berkeley, CA 94720, 1999.

MURARI M.L. Desenvolvimento de uma ferramenta computacional de apoio ao ensino de sistemas eletrônicos digitais. Dissertação de Mestrado – Engenharia Elétrica – Universidade Estadual de São Paulo (UNESP – Ilha Solteira). 2008

OYAMADA, M. S. Estudo de ambientes para Co-Simulação. Dissertação de Mestrado – Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, 1999.

OH, M. HÁ, S. Synthesizable VHDL Code Generation from Data Flow Graph. In: Proceedings 5th Asia Pacific Conference on Hardware Description Languages. Seoul, Korea, July 1998.

PATTERSON, D. A; HENNESSY, J. L. Organização e Projeto de Computadores – A Interface Hardware/Software. 3 edição. Rio de Janeiro: Elsevier Editora, 2005.

PERRY, D. E. WOLF, A. L. Foundations for the Study of Software Architecture. Software Engineering Notes. Vol. 17. N.4. 1992

PRITSKER, A. A. B. Principies of simulation modeling. In Banks, J (ed.) Handbook of simulation – principles, metholdoly, advances, applications and practice. New York, 1998.

ROCHA. A. M. M. Introdução à Arquitetura de Software. Disponível em: <http://www.fabsoft.cesupa.br/site/images/introducao_arquitetura_software.pdf > Acesso em: 22 fev. 2012.

SANTOS. L.C.V. et al. An Open Source Binary Utility Generator. ACM Transactions on Design Automation of Electronic Systems, v.13, n.2, 2008.

SANTOS, L. C. V. . A Síntese de Alto Nível na Automação do Projeto de Sistemas Computacionais. In: Sociedade Brasileira de Computação. (Org.). VIII Escola de Informática da SBC Sul. VIII Escola de Informática da SBC Sul. Porto Alegre: Editora da UFRGS, 2000, v. , p. 211-232.

SCHOEBERL, M. BROOKS, C. LEE, E.A. Code Generation Embedded java whit Ptolemy. In: Proceedings of the 8th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2010), Oct, 2010.

SCOTT, M. WinMips64, version 1.5, School of Computing, Dublin City University, Ireland, 2006.

SENTURIA, S.D. CAD Challenges for Microsensors, Microactuators, and Microsystems. In: Proceedings of the IEEE, V. 86, N. 8, Aug, 1998.

SMYTH, N. Communicating Sequential Processes Domain in Ptolemy II, MS Report, UCB/ERL Memorandum M98/70, Dept. of EECS, University of California, Berkeley, CA 94720, 1998.

SOBREIRA, P. L, et al. Competição como uma Técnica Motivacional no Ensino de Arquitetura de Computadores. Apresentado no Workshop sobre Educação em Arquitetura de Computadores (WEAC) – Simpósio Brasileiro de Arquitetura de Computadores. SBAC, Gramado-RS, 2007.

SOUTO, A. V. M.; DUDUCHI, M. Um processo de avaliação baseado em ferramenta computadorizada para o apoio ao ensino de programação de computadores. Workshop sobre Educação em Informática (WEI09), Bento Gonçalves – RS, 2009.

SPOLON, R. Um editor gráfico para um ambiente de simulação automático. Dissertação de Mestrado. ICMSC – USP. São Paulo. 1994.

SUNG, W. et al. Demonstration of Codesign Workflow in PeaCe. In: Proc. of International Conference of VLSI Circuit, Seoul, Koera, 1997.

TANENBAUM, A. S. Organização Estruturada de Computadores. 5ed. São Paulo: Editora Pearson, 2007.

VLSI Technology. Sxlib Standard Cell Library Description. Disponível em: < http://www.vlsitechnology.org/html/sx_description.html > Acesso em: 03 Abr. 2012.

WEST, N.H.E.; HARRIS, D. **CMOS VLSI Design: A circuit and system perspective**. 3ed. Editora Pearson, 2005.

WILLIAMSON, M. C. Synthesis of parallel hardware implementations from synchronous dataflow graph specifications. Technical Report UCB/ERL M98/45, University of California, Berkeley, June 1998.

ZHOU, G. Partial Evaluation for Optimized Compilation of Actor-Oriented Models. In: Technical Report No. UCB/EECS-2008-53, 2008.

ZHOU, G. LEUNG, M. LEE, E. A. A Code Generation Framework for Actor Oriented Models with Partial Evaluation. In: Proceedings of International Conference on Embedded Software and Systems 2007, LNCS 4523, pp. 786-799, Daegu, South Korea, May, 2007

APÊNDICE A – Arquivo adder_1bit.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE entity PUBLIC "-//UC Berkeley//DTD MoML 1//EN"
   "http://ptolemy.eecs.berkeley.edu/xml/dtd/MoML_1.dtd">
<entity name="adder_1bit" class="ptolemy.actor.TypedCompositeActor">
   <property name="_createdBy" class="ptolemy.kernel.attributes.VersionAttribute"
value="8.0.1_20101021"> </property>
   <property name="SDF Director" class="ptolemy.domains.sdf.kernel.SDFDirector">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[545.0, 80.0]">
</property>
   </property>
   <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={49, 24, 1317, 744}, maximized=true}"> </property>
   <property name="_vergilSize" class="ptolemy.actor.gui.SizeAttribute" value="[1111, 649]">
</property>
   <property name="_vergilZoomFactor" class="ptolemy.data.expr.ExpertParameter" value="0.8">
</property>
   <property name="_vergilCenter" class="ptolemy.data.expr.ExpertParameter"
value="{437.9258996212121, 444.1680871212121}"> </property>
   <entity name="Sequence" class="ptolemy.actor.lib.Sequence">
      <property name="values" class="ptolemy.data.expr.Parameter" value="{1,0,1,0}"> </property>
      <property name="repeat" class="ptolemy.data.expr.Parameter" value="false"> </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[115.0, 265.0]">
</property>
   </entity>
   <entity name="Sequence2" class="ptolemy.actor.lib.Sequence">
      <property name="values" class="ptolemy.data.expr.Parameter" value="{0,0,1,1}"> </property>
      <property name="repeat" class="ptolemy.data.expr.Parameter" value="false"> </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[115.0, 320.0]">
</property>
   </entity>
   <entity name="sum" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={867, 461, 200, 209}, maximized=false}"> </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[200, 182]">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[950.0, 220.0]">
</property>
   </entity>
   <entity name="Pulse" class="ptolemy.actor.lib.Pulse">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[-105.0, 320.0]">
</property>
   </entity>
   <entity name="IsPresent" class="ptolemy.actor.lib.logic.IsPresent">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[-15.0, 320.0]">
</property>
   </entity>
   <entity name="input A" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={162, 461, 226, 209}, maximized=false}"> </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[226, 182]">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[115.0, 170.0]">
</property>
```

```
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">       </property>
   </entity>
   <entity name="input B" class="ptolemy.actor.lib.gui.Display">
      <property       name="_windowProperties"       class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={391, 460, 229, 209}, maximized=false}">       </property>
      <property     name="_paneSize"     class="ptolemy.actor.gui.SizeAttribute"     value="[229,     182]">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="{180.0,     170.0}">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">       </property>
   </entity>
   <entity name="carry out" class="ptolemy.actor.lib.gui.Display">
      <property       name="_windowProperties"       class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={1069, 461, 238, 209}, maximized=false}">       </property>
      <property     name="_paneSize"     class="ptolemy.actor.gui.SizeAttribute"     value="[238,     182]">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[960.0,     320.0]">
</property>
   </entity>
   <entity name="xr" class="ptolemy.actor.lib.vhdlsxlib.xr2">
      <property    name="CellReference"    class="ptolemy.data.expr.StringParameter"    value="xr2_x1">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[490.0,     210.0]">
</property>
   </entity>
   <entity name="a2" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property     name="CellReference"     class="ptolemy.data.expr.StringParameter"     value="a2_x2">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[530.0,     310.0]">
</property>
   </entity>
   <entity name="a3" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property     name="CellReference"     class="ptolemy.data.expr.StringParameter"     value="a2_x2">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[660.0,     385.0]">
</property>
   </entity>
   <entity name="xr2" class="ptolemy.actor.lib.vhdlsxlib.xr2">
      <property    name="CellReference"    class="ptolemy.data.expr.StringParameter"    value="xr2_x1">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[670.0,     220.0]">
</property>
   </entity>
   <entity name="o" class="ptolemy.actor.lib.vhdlsxlib.o2">
      <property     name="CellReference"     class="ptolemy.data.expr.StringParameter"     value="o2_x2">
</property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="[755.0,     320.0]">
</property>
   </entity>
   <entity name="inA" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">       </property>
      <property     name="_location"     class="ptolemy.kernel.util.Location"     value="{350.0,     265.0}">
</property>
   </entity>
   <entity name="inB" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">       </property>
```

```
      <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 320.0]">
</property>
   </entity>
   <entity name="cin" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[350.0, 395.0]">
</property>
   </entity>
   <entity name="s" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[860.0, 220.0]">
</property>
   </entity>
   <entity name="cout" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[865.0, 320.0]">
</property>
   </entity>
   <entity name="Sequence3" class="ptolemy.actor.lib.Sequence">
      <property name="values" class="ptolemy.data.expr.Parameter" value="{0,1,0,1}">      </property>
      <property name="repeat" class="ptolemy.data.expr.Parameter" value="false">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[110.0, 395.0]">
</property>
   </entity>
   <entity name="input Cin" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={619, 432, 250, 238}, maximized=false}">      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[248, 182]">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[240.0, 170.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">      </property>
   </entity>
   <relation name="relation5" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[40.0, 320.0]">      </vertex>
   </relation>
   <relation name="relation4" class="ptolemy.actor.TypedIORelation">   </relation>
   <relation name="relation6" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[175.0, 265.0]">      </vertex>
   </relation>
   <relation name="relation2" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[225.0, 320.0]">      </vertex>
   </relation>
   <relation name="relation7" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[585.0, 395.0]">      </vertex>
   </relation>
   <relation name="relation9" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[595.0, 210.0]">      </vertex>
   </relation>
   <relation name="relation10" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[405.0, 265.0]">      </vertex>
   </relation>
   <relation name="relation11" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[410.0, 320.0]">      </vertex>
   </relation>
   <relation name="relation12" class="ptolemy.actor.TypedIORelation">   </relation>
```

```
    <relation name="relation13" class="ptolemy.actor.TypedIORelation">    </relation>
    <relation name="relation8" class="ptolemy.actor.TypedIORelation">    </relation>
    <relation name="relation14" class="ptolemy.actor.TypedIORelation">    </relation>
    <relation name="relation16" class="ptolemy.actor.TypedIORelation">    </relation>
    <relation name="relation3" class="ptolemy.actor.TypedIORelation">    </relation>
    <relation name="relation" class="ptolemy.actor.TypedIORelation">
        <vertex name="vertex1" value="[265.0, 395.0]">        </vertex>
    </relation>
    <link port="Sequence.enable" relation="relation5"/>
    <link port="Sequence.output" relation="relation6"/>
    <link port="Sequence2.enable" relation="relation5"/>
    <link port="Sequence2.output" relation="relation2"/>
    <link port="sum.input" relation="relation3"/>
    <link port="Pulse.output" relation="relation4"/>
    <link port="IsPresent.input" relation="relation4"/>
    <link port="IsPresent.output" relation="relation5"/>
    <link port="input A.input" relation="relation6"/>
    <link port="input B.input" relation="relation2"/>
    <link port="carry out.input" relation="relation14"/>
    <link port="xr.i0" relation="relation10"/>
    <link port="xr.i1" relation="relation11"/>
    <link port="xr.q" relation="relation9"/>
    <link port="a2.i0" relation="relation10"/>
    <link port="a2.i1" relation="relation11"/>
    <link port="a2.q" relation="relation12"/>
    <link port="a3.i0" relation="relation9"/>
    <link port="a3.i1" relation="relation7"/>
    <link port="a3.q" relation="relation13"/>
    <link port="xr2.i0" relation="relation9"/>
    <link port="xr2.i1" relation="relation7"/>
    <link port="xr2.q" relation="relation8"/>
    <link port="o.i0" relation="relation12"/>
    <link port="o.i1" relation="relation13"/>
    <link port="o.q" relation="relation16"/>
    <link port="inA.in" relation="relation6"/>
    <link port="inA.out" relation="relation10"/>
    <link port="inB.in" relation="relation2"/>
    <link port="inB.out" relation="relation11"/>
    <link port="cin.in" relation="relation"/>
    <link port="cin.out" relation="relation7"/>
    <link port="s.in" relation="relation8"/>
    <link port="s.out" relation="relation3"/>
    <link port="cout.in" relation="relation16"/>
    <link port="cout.out" relation="relation14"/>
    <link port="Sequence3.enable" relation="relation5"/>
    <link port="Sequence3.output" relation="relation"/>
    <link port="input Cin.input" relation="relation"/>
</entity>
```

APÊNDICE B – Arquivo adder_1bit.vst

```
entity adder_1bit is
   port (
inA         :in         bit;
inB         :in         bit;
cin         :in         bit;
s           :out        bit;
cout        :out        bit;
vdd         :in         bit;
vss         :in         bit
 );
end adder_1bit;

architecture structural of adder_1bit is
COMPONENT xr2_x1
GENERIC (
  CONSTANT area    : NATURAL := 2250;
  CONSTANT cin_i0 : NATURAL := 21;
  CONSTANT cin_i1 : NATURAL := 22;
  CONSTANT rdown_i0_q       : NATURAL := 2850;
  CONSTANT rdown_i1_q       : NATURAL := 2850;
  CONSTANT rup_i0_q         : NATURAL := 3210;
  CONSTANT rup_i1_q         : NATURAL := 3210;
  CONSTANT tplh_i1_q        : NATURAL := 261;
  CONSTANT tphl_i0_q        : NATURAL := 292;
  CONSTANT tplh_i0_q        : NATURAL := 293;
  CONSTANT tphh_i0_q        : NATURAL := 366;
  CONSTANT tphl_i1_q        : NATURAL := 377;
  CONSTANT tpll_i1_q        : NATURAL := 388;
  CONSTANT tpll_i0_q        : NATURAL := 389;
  CONSTANT tphh_i1_q        : NATURAL := 405;
  CONSTANT transistors      : NATURAL := 12
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

COMPONENT a2_x2
GENERIC (
  CONSTANT area    : NATURAL := 1250;
  CONSTANT cin_i0 : NATURAL := 9;
  CONSTANT cin_i1 : NATURAL := 11;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tphh_i1_q        : NATURAL := 203;
  CONSTANT tphh_i0_q        : NATURAL := 261;
  CONSTANT tpll_i0_q        : NATURAL := 388;
  CONSTANT tpll_i1_q        : NATURAL := 434;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

COMPONENT o2_x2
GENERIC (
```

```
  CONSTANT area    : NATURAL := 1250;
  CONSTANT cin_i0 : NATURAL := 10;
  CONSTANT cin_i1 : NATURAL := 10;
  CONSTANT rdown_i0_q       : NATURAL := 1620;
  CONSTANT rdown_i1_q       : NATURAL := 1620;
  CONSTANT rup_i0_q         : NATURAL := 1790;
  CONSTANT rup_i1_q         : NATURAL := 1790;
  CONSTANT tpll_i0_q        : NATURAL := 310;
  CONSTANT tphh_i1_q        : NATURAL := 335;
  CONSTANT tpll_i1_q        : NATURAL := 364;
  CONSTANT tphh_i0_q        : NATURAL := 406;
  CONSTANT transistors      : NATURAL := 6
);
PORT (
  i0          : in  BIT;
  i1          : in  BIT;
  q           : out BIT;
  vdd         : in  BIT;
  vss         : in  BIT
);
END COMPONENT;

signal relation : bit;
signal relation10 : bit;
signal relation11 : bit;
signal relation12 : bit;
signal relation13 : bit;
signal relation14 : bit;
signal relation2 : bit;
signal relation3 : bit;
signal relation16 : bit;
signal relation8 : bit;
signal relation9 : bit;
signal relation6 : bit;
signal relation7 : bit;
signal relation4 : bit;
signal relation5 : bit;

begin

xr : xr2_x1
Generic Map(
   area      => 2250,
   cin_i0    => 21,
   cin_i1    => 22,
   rdown_i0_q          => 2850,
   rdown_i1_q          => 2850,
   rup_i0_q            => 3210,
   rup_i1_q            => 3210,
   tplh_i1_q           => 261,
   tphl_i0_q           => 292,
   tplh_i0_q           => 293,
   tphh_i0_q           => 366,
   tphl_i1_q           => 377,
   tpll_i1_q           => 388,
   tpll_i0_q           => 389,
   tphh_i1_q           => 405,
   transistors         => 12
)
port map (
  i0 => inA ,
  i1 => inB ,
  vdd => vdd ,
  vss => vss ,
  q => relation9
 );

a2 : a2_x2
Generic Map(
   area      => 1250,
```

```
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => inA ,
  i1 => inB ,
  vdd => vdd ,
  vss => vss ,
  q => relation12
 );

a3 : a2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 9,
   cin_i1     => 11,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tphh_i1_q           => 203,
   tphh_i0_q           => 261,
   tpll_i0_q           => 388,
   tpll_i1_q           => 434,
   transistors         => 6
)
port map (
  i0 => relation9 ,
  i1 => cin ,
  vdd => vdd ,
  vss => vss ,
  q => relation13
 );

xr2 : xr2_x1
Generic Map(
   area       => 2250,
   cin_i0     => 21,
   cin_i1     => 22,
   rdown_i0_q          => 2850,
   rdown_i1_q          => 2850,
   rup_i0_q            => 3210,
   rup_i1_q            => 3210,
   tplh_i1_q           => 261,
   tphl_i0_q           => 292,
   tplh_i0_q           => 293,
   tphh_i0_q           => 366,
   tphl_i1_q           => 377,
   tpll_i1_q           => 388,
   tpll_i0_q           => 389,
   tphh_i1_q           => 405,
   transistors         => 12
)
port map (
  i0 => relation9 ,
  i1 => cin ,
  vdd => vdd ,
  vss => vss ,
  q => s
 );
```

```
o : o2_x2
Generic Map(
   area       => 1250,
   cin_i0     => 10,
   cin_i1     => 10,
   rdown_i0_q          => 1620,
   rdown_i1_q          => 1620,
   rup_i0_q            => 1790,
   rup_i1_q            => 1790,
   tpll_i0_q           => 310,
   tphh_i1_q           => 335,
   tpll_i1_q           => 364,
   tphh_i0_q           => 406,
   transistors         => 6
)
port map (
  i0 => relation12 ,
  i1 => relation13 ,
  vdd => vdd ,
  vss => vss ,
  q => cout
 );

end structural;
```

APÊNDICE C – Arquivo dec2x4.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE entity PUBLIC "-//UC Berkeley//DTD MoML 1//EN"
    "http://ptolemy.eecs.berkeley.edu/xml/dtd/MoML_1.dtd">
<entity name="dec2x4" class="ptolemy.actor.TypedCompositeActor">
    <property name="_createdBy" class="ptolemy.kernel.attributes.VersionAttribute"
value="8.0.1_20101021">
    </property>
    <property name="SDF Director" class="ptolemy.domains.sdf.kernel.SDFDirector">
        <property name="_location" class="ptolemy.kernel.util.Location" value="[655.0, 220.0]">
        </property>
    </property>
    <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={49, 24, 1317, 744}, maximized=true}">
    </property>
    <property name="_vergilSize" class="ptolemy.actor.gui.SizeAttribute" value="[1111, 649]">
    </property>
    <property name="_vergilZoomFactor" class="ptolemy.data.expr.ExpertParameter" value="0.8">
    </property>
    <property name="_vergilCenter" class="ptolemy.data.expr.ExpertParameter"
value="{785.4354482323233, 567.8435921717172}">
    </property>
    <entity name="Sequence" class="ptolemy.actor.lib.Sequence">
        <property name="values" class="ptolemy.data.expr.Parameter" value="{0,1,0,1,0,1,0,1,0,1,0,1}">
        </property>
        <property name="repeat" class="ptolemy.data.expr.Parameter" value="false">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[370.0, 200.0]">
        </property>
    </entity>
    <entity name="Sequence2" class="ptolemy.actor.lib.Sequence">
        <property name="values" class="ptolemy.data.expr.Parameter" value="{0,0,1,1,0,0,1,1,0,0,1,1}">
        </property>
        <property name="repeat" class="ptolemy.data.expr.Parameter" value="false">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[370.0, 255.0]">
        </property>
    </entity>
    <entity name="Pulse" class="ptolemy.actor.lib.Pulse">
        <property name="_location" class="ptolemy.kernel.util.Location" value="[160.0, 255.0]">
        </property>
    </entity>
    <entity name="IsPresent" class="ptolemy.actor.lib.logic.IsPresent">
        <property name="_location" class="ptolemy.kernel.util.Location" value="[245.0, 255.0]">
        </property>
    </entity>
    <entity name="A0" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
        <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[535.0, 320.0]">
        </property>
        <property name="_flipPortsVertical" class="ptolemy.data.expr.Parameter" value="true">
        </property>
        <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">
```

```
      </property>
   </entity>
   <entity name="A1" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[450.0, 320.0]">
      </property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="90">
         <property name="_editorFactory" class="ptolemy.vergil.toolbox.VisibleParameterEditorFactory">
         </property>
      </property>
   </entity>
   <entity name="saida" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[1005.0, 695.0]">
      </property>
   </entity>
   <entity name="saida2" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[1000.0, 600.0]">
      </property>
   </entity>
   <entity name="saida3" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[975.0, 450.0]">
      </property>
   </entity>
   <entity name="saida4" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[965.0, 350.0]">
      </property>
   </entity>
   <entity name="Display" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={275, 211, 522, 209}, maximized=false}">
      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[522, 182]">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[1210.0, 695.0]">
      </property>
   </entity>
   <entity name="Display2" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={468, 184, 524, 238}, maximized=false}">
      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[522, 182]">
      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[1200.0, 600.0]">
      </property>
   </entity>
   <entity name="Display3" class="ptolemy.actor.lib.gui.Display">
```

```
        <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={727, 333, 538, 514}, maximized=false}">
        </property>
        <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[538, 487]">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[1185.0, 450.0]">
        </property>
    </entity>
    <entity name="Display4" class="ptolemy.actor.lib.gui.Display">
        <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={592, 133, 524, 238}, maximized=false}">
        </property>
        <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[522, 182]">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[1170.0, 345.0]">
        </property>
    </entity>
    <entity name="inv" class="ptolemy.actor.lib.vhdlsxlib.inv">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="inv_x1">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[585.0, 560.0]">
        </property>
        <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="0">
            <property name="_editorFactory" class="ptolemy.vergil.toolbox.VisibleParameterEditorFactory">
            </property>
        </property>
    </entity>
    <entity name="inv2" class="ptolemy.actor.lib.vhdlsxlib.inv">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="inv_x1">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[530.0, 625.0]">
        </property>
        <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="0">
        </property>
    </entity>
    <entity name="a" class="ptolemy.actor.lib.vhdlsxlib.a2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[845.0, 670.0]">
        </property>
    </entity>
    <entity name="a2" class="ptolemy.actor.lib.vhdlsxlib.a2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[845.0, 600.0]">
        </property>
    </entity>
    <entity name="a3" class="ptolemy.actor.lib.vhdlsxlib.a2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
        </property>
        <property name="_location" class="ptolemy.kernel.util.Location" value="[795.0, 450.0]">
        </property>
    </entity>
    <entity name="a4" class="ptolemy.actor.lib.vhdlsxlib.a2">
        <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
        </property>
```

```
    <property name="_location" class="ptolemy.kernel.util.Location" value="[795.0, 345.0]">
    </property>
</entity>
<relation name="relation5" class="ptolemy.actor.TypedIORelation">
    <vertex name="vertex1" value="[285.0, 255.0]">
    </vertex>
</relation>
<relation name="relation4" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation6" class="ptolemy.actor.TypedIORelation">
    <vertex name="vertex1" value="[535.0, 470.0]">
    </vertex>
</relation>
<relation name="relation2" class="ptolemy.actor.TypedIORelation">
    <vertex name="vertex1" value="[450.0, 395.0]">
    </vertex>
</relation>
<relation name="relation9" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation11" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation3" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation14" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation16" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation18" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation" class="ptolemy.actor.TypedIORelation">
    <vertex name="vertex1" value="[780.0, 625.0]">
    </vertex>
</relation>
<relation name="relation7" class="ptolemy.actor.TypedIORelation">
    <vertex name="vertex1" value="[735.0, 560.0]">
    </vertex>
</relation>
<relation name="relation10" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation12" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation8" class="ptolemy.actor.TypedIORelation">
</relation>
<relation name="relation13" class="ptolemy.actor.TypedIORelation">
</relation>
<link port="Sequence.enable" relation="relation5"/>
<link port="Sequence.output" relation="relation9"/>
<link port="Sequence2.enable" relation="relation5"/>
<link port="Sequence2.output" relation="relation11"/>
<link port="Pulse.output" relation="relation4"/>
<link port="IsPresent.input" relation="relation4"/>
<link port="IsPresent.output" relation="relation5"/>
<link port="A0.in" relation="relation9"/>
<link port="A0.out" relation="relation6"/>
<link port="A1.in" relation="relation11"/>
<link port="A1.out" relation="relation2"/>
```

```
    <link port="saida.in" relation="relation13"/>
    <link port="saida.out" relation="relation3"/>
    <link port="saida2.in" relation="relation8"/>
    <link port="saida2.out" relation="relation18"/>
    <link port="saida3.in" relation="relation10"/>
    <link port="saida3.out" relation="relation16"/>
    <link port="saida4.in" relation="relation12"/>
    <link port="saida4.out" relation="relation14"/>
    <link port="Display.input" relation="relation3"/>
    <link port="Display2.input" relation="relation18"/>
    <link port="Display3.input" relation="relation16"/>
    <link port="Display4.input" relation="relation14"/>
    <link port="inv.i" relation="relation6"/>
    <link port="inv.nq" relation="relation7"/>
    <link port="inv2.i" relation="relation2"/>
    <link port="inv2.nq" relation="relation"/>
    <link port="a.i0" relation="relation"/>
    <link port="a.i1" relation="relation7"/>
    <link port="a.q" relation="relation13"/>
    <link port="a2.i0" relation="relation"/>
    <link port="a2.i1" relation="relation6"/>
    <link port="a2.q" relation="relation8"/>
    <link port="a3.i0" relation="relation2"/>
    <link port="a3.i1" relation="relation7"/>
    <link port="a3.q" relation="relation10"/>
    <link port="a4.i0" relation="relation2"/>
    <link port="a4.i1" relation="relation6"/>
    <link port="a4.q" relation="relation12"/>
</entity>
```

APÊNDICE D – Arquivo latch_D.xml

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE entity PUBLIC "-//UC Berkeley//DTD MoML 1//EN"
   "http://ptolemy.eecs.berkeley.edu/xml/dtd/MoML_1.dtd">
<entity name="latch_D" class="ptolemy.actor.TypedCompositeActor">
   <property name="_createdBy" class="ptolemy.kernel.attributes.VersionAttribute"
value="8.0.1_20101021">  </property>
   <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={48, -4, 1319, 773}, maximized=true}">  </property>
   <property name="_vergilSize" class="ptolemy.actor.gui.SizeAttribute" value="[1111, 649]">
</property>
   <property name="_vergilZoomFactor" class="ptolemy.data.expr.ExpertParameter" value="1.0">
</property>
   <property name="_vergilCenter" class="ptolemy.data.expr.ExpertParameter"
value="{601.5075757575758, 378.4375}">
   </property>
   <property name="SDF Director" class="ptolemy.domains.sdf.kernel.SDFDirector">
      <property name="iterations" class="ptolemy.data.expr.Parameter" value="0">      </property>
      <property name="allowRateChanges" class="ptolemy.data.expr.Parameter" value="false">
</property>
      <property name="period" class="ptolemy.data.expr.Parameter" value="0.0">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[325.0, 130.0]">
</property>
   </property>
   <entity name="Sequence" class="ptolemy.actor.lib.Sequence">
      <property name="values" class="ptolemy.data.expr.Parameter" value="{0,1,0,1}">      </property>
      <property name="repeat" class="ptolemy.data.expr.Parameter" value="false">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[235.0, 270.0]">
</property>
   </entity>
   <entity name="Pulse" class="ptolemy.actor.lib.Pulse">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[115.0, 195.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="90">      </property>
   </entity>
   <entity name="IsPresent" class="ptolemy.actor.lib.logic.IsPresent">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[105.0, 300.0]">
</property>
   </entity>
   <entity name="Dvalue" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={301, 537, 215, 185}, maximized=false}">      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[215, 158]">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[240.0, 445.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="90">      </property>
   </entity>
   <entity name="CLKvalue" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={444, 509, 202, 213}, maximized=false}">      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[200, 157]">
</property>
```

```
      <property name="_location" class="ptolemy.kernel.util.Location" value="[455.0, 125.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">
        <property name="_editorFactory" class="ptolemy.vergil.toolbox.VisibleParameterEditorFactory">
</property>
      </property>
   </entity>
   <entity name="clk" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[360.0, 210.0]">
</property>
   </entity>
   <entity name="Sequence2" class="ptolemy.actor.lib.Sequence">
      <property name="values" class="ptolemy.data.expr.Parameter" value="{0,0,1,1}">      </property>
      <property name="repeat" class="ptolemy.data.expr.Parameter" value="false">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[230.0, 210.0]">
</property>
   </entity>
   <entity name="inD" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="input">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[325.0, 270.0]">
</property>
   </entity>
   <entity name="out Q" class="ptolemy.actor.lib.gui.Display">
      <property name="_windowProperties" class="ptolemy.actor.gui.WindowPropertiesAttribute"
value="{bounds={606, 510, 217, 211}, maximized=false}">
      </property>
      <property name="_paneSize" class="ptolemy.actor.gui.SizeAttribute" value="[215, 155]">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[995.0, 250.0]">
</property>
   </entity>
   <entity name="Saida" class="ptolemy.actor.lib.vhdlsxlib.IOModel">
      <property name="Type" class="ptolemy.data.expr.StringParameter" value="output">      </property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[905.0, 250.0]">
</property>
   </entity>
   <entity name="a" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[605.0, 240.0]">
</property>
   </entity>
   <entity name="a2" class="ptolemy.actor.lib.vhdlsxlib.a2">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="a2_x2">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[655.0, 405.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="-90">      </property>
   </entity>
   <entity name="inv" class="ptolemy.actor.lib.vhdlsxlib.inv">
      <property name="CellReference" class="ptolemy.data.expr.StringParameter" value="inv_x1">
</property>
      <property name="_location" class="ptolemy.kernel.util.Location" value="[455.0, 345.0]">
</property>
      <property name="_rotatePorts" class="ptolemy.data.expr.Parameter" value="270">      </property>
```

```
      <property name="_flipPortsHorizontal" class="ptolemy.data.expr.Parameter" value="true">
</property>
      <property name="_flipPortsVertical" class="ptolemy.data.expr.Parameter" value="true">
</property>
   </entity>
   <entity name="o" class="ptolemy.actor.lib.vhdlsxlib.o2">
      <property name="_location" class="ptolemy.kernel.util.Location" value="[725.0, 250.0]">
</property>
   </entity>
   <relation name="relation4" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation6" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[455.0, 230.0]">
      </vertex>
   </relation>
   <relation name="relation3" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex" value="{0.0, 0.0}">
      </vertex>
   </relation>
   <relation name="relation9" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[795.0, 250.0]">
      </vertex>
   </relation>
   <relation name="relation2" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[385.0, 270.0]">
      </vertex>
   </relation>
   <relation name="relation17" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex1" value="[170.0, 270.0]">
      </vertex>
   </relation>
   <relation name="relation5" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex" value="{0.0, 0.0}">
      </vertex>
   </relation>
   <relation name="relation16" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation" class="ptolemy.actor.TypedIORelation">
      <vertex name="vertex" value="{0.0, 0.0}">
      </vertex>
   </relation>
   <relation name="relation13" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation8" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation7" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation12" class="ptolemy.actor.TypedIORelation">
   </relation>
   <relation name="relation15" class="ptolemy.actor.TypedIORelation">
   </relation>
   <link port="Sequence.enable" relation="relation17"/>
   <link port="Sequence.output" relation="relation8"/>
   <link port="Pulse.output" relation="relation4"/>
   <link port="IsPresent.input" relation="relation4"/>
   <link port="IsPresent.output" relation="relation17"/>
```

```
    <link port="Dvalue.input" relation="relation2"/>
    <link port="CLKvalue.input" relation="relation6"/>
    <link port="clk.in" relation="relation13"/>
    <link port="clk.out" relation="relation6"/>
    <link port="Sequence2.enable" relation="relation17"/>
    <link port="Sequence2.output" relation="relation13"/>
    <link port="inD.in" relation="relation8"/>
    <link port="inD.out" relation="relation2"/>
    <link port="out Q.input" relation="relation16"/>
    <link port="Saida.in" relation="relation9"/>
    <link port="Saida.out" relation="relation16"/>
    <link port="a.i0" relation="relation6"/>
    <link port="a.i1" relation="relation2"/>
    <link port="a.q" relation="relation12"/>
    <link port="a2.i0" relation="relation7"/>
    <link port="a2.i1" relation="relation9"/>
    <link port="a2.q" relation="relation15"/>
    <link port="inv.i" relation="relation6"/>
    <link port="inv.nq" relation="relation7"/>
    <link port="o.i0" relation="relation12"/>
    <link port="o.i1" relation="relation15"/>
    <link port="o.q" relation="relation9"/>
</entity>
```