UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

GABRIELA SILVA RIBEIRO

ORANGECAD WEB
UMA FERRAMENTA DE APOIO À SÍNTESE DE CIRCUITOS LÓGICOS SEQUENCIAIS

VITÓRIA
2011

Gabriela Silva Ribeiro

ORANGECAD WEB
UMA FERRAMENTA DE APOIO À SÍNTESE DE CIRCUITOS LÓGICOS SEQUENCIAIS

Parte manuscrita do Projeto de Graduação da aluna Gabriela Silva Ribeiro, apresentada ao Departamento de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para obtenção do Grau de Engenheira de Computação.

Orientador: Prof. Dr. rer. nat. Hans-Jorg Andreas Schneebeli

VITÓRIA
2011

Gabriela Silva Ribeiro

ORANGECAD WEB
UMA FERRAMENTA DE APOIO À SÍNTESE DE CIRCUITOS LÓGICOS SEQUENCIAIS

Projeto de Graduação submetido ao Departamento de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito para a obtenção do Grau de Engenheira de Computação.

Aprovado em 30 de Agosto de 2011.

COMISSÃO EXAMINADORA

Prof. Dr. rer. nat. Hans-Jorg Andreas Schneebeli
Universidade Federal do Espírito Santo
Orientador

Profa. Dra. Eliete Maria de Oliveira Caldeira
Universidade Federal do Espírito Santo

Prof. Dr. Evandro Ottoni Teatini Salles
Universidade Federal do Espírito Santo

Sumário

Lista de Figuras

Agradecimentos

A Deus que me permitiu esta vitória.

Aos meus pais Elisabeth e Silvestre e minha irmã Caroline, que me apoiaram e compreenderam minha ausência.

Ao meu namorado Daricque, que me acompanhou de perto e manteve minha mente calma.

Ao meu orientador Hans, pela paciência, sabedoria e objetividade ímpares.

Aos meus colegas de curso, pela torcida e bons conselhos.

Resumo

Este trabalho propõe uma metodologia sistemática de projeto de circuitos lógicos sequenciais, baseada em um conjunto de etapas bem definidas que partem do diagrama de estados de uma máquina de estados e levam à síntese em linguagens de descrição de hardware ou em equações booleanas.

São abordados a elaboração e o desenvolvimento da ferramenta OrangeCAD Web, que implementa essa metodologia de forma automatizada em uma interface gráfica intuitiva.

Palavras-chave: OrangeCAD, síntese, máquinas de estado, diagrama de estados, VHDL, ESPRESSO.

Abstract

This work proposes a systematic methodology for sequencial logic design, based on a set of well-defined steps that start from the state diagram of a state machine and leads to the synthesis with hardware description languages or Boolean equations.

It discusses the design and development of OrangeCAD Web, a tool that implements this methodology in an automated way with an intuitive graphical interface.

Keywords: OrangeCAD, synthesis, state machines, state diagram, statechart, VHDL, ESPRESSO.

Capítulo 1

Introdução

1.1 Motivação

Uma máquina de estados é uma abstração usada para determinar como um sistema deve reagir em determinada circunstância. Descrever o comportamento de sistemas sequenciais com máquinas de estado é um método de projeto muito usado (Cooper, 2008).

O uso de máquinas de estado provê um modo sistemático de projetar circuitos lógicos sequenciais (Cypress, 1995). A partir do diagrama de estados, é construído o mapa de transição, que após a codificação dos estados resulta na tabela de transição. Então, com a escolha do tipo de flip-flop, cria-se a tabela de excitação, a partir da qual se obtém a descrição do circuito.

Apesar de ser sistemático, esse processo é trabalhoso e propenso a erros se feito manualmente. Além disso, envolve escolhas, e mudanças de escolha demandam tempo de retrabalho.

Assim, é de grande utilidade ter em mãos uma ferramenta que possibilite criar diretamente o diagrama de estados e, a partir dele, gerar automaticamente o código na linguagem descritiva desejada, discriminando cada etapa de projeto. Essa ferramenta teria também um bom uso didático, por oferecer ao estudante uma visualização das etapas de síntese e reduzir a demanda de tempo, em uma interface amigável.

1.2 Descrição da Área

Existem vários programas de apoio a desenvolvimento de circuitos digitais, tais como o Ptolemy (Lee, 2009), o StateCAD (Xilinx,) e o OrangeCAD (Pegoretti, 2002) cada um com suas funcionalidades e limitações. Este trabalho toma como base o sistema OrangeCAD, por se aproximar mais dos objetivos deste projeto, e propõe adaptações de forma a resultar num sistema mais atraente e objetivo.

O OrangeCAD é uma ferramenta que oferece grande apoio na criação e análise de máquinas de estados, através da criação de diagramas de estado e geração automática de tabelas auxiliares e códigos. Nesse sistema, o usuário pode criar diagramas de estado de máquinas Moore, Mealy ou híbridas, adicionando em um painel os componentes gráficos do diagrama (estados e ramos) e editando propriedades da máquina, como o número de bits de entrada e saída.

De acordo com o diagrama desenhado pelo usuário, o sistema disponibiliza dinamicamente o código em VHDL e ESPRESSO, a tabela de transição de estados, a tabela de excitação e outras tabelas de apoio.

Após algum tempo de uso, já é possível utilizar o sistema OrangeCAD sem dificuldades, com o auxílio de legendas explicativas, que são apresentadas na parte inferior da tela, de acordo com as ações que o usuário pode tomar no momento.

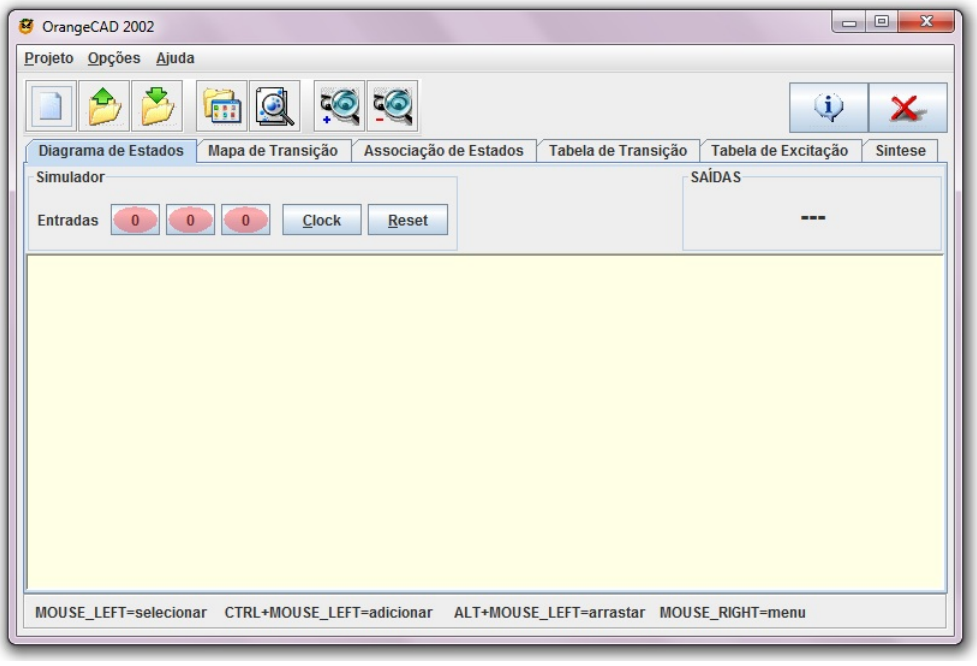


Figura 1.1: Legenda explicativa exibida durante a edição do diagrama de estados.

Contudo, os comandos não são intuitivos, tornando necessário o emprego dessas legendas sempre que um usuário destreinado inicia o sistema.

Por essa razão, verificou-se a importância de modificar a interface gráfica com o usuário do OrangeCAD, propondo o emprego de representações de senso comum.

Além disso, o OrangeCAD é um aplicativo *desktop*, que tem que ser instalado no computador do usuário para ser utilizado. Isso intimida novos usuários a experimentarem o programa, e dificulta que usuários o utilizem em computadores variados.

Portanto, uma adaptação *online* do programa é desejável, incentivando e diversificando seu uso.

1.3 Definição do Problema

Desenvolver um aplicativo *web* que permita ao usuário descrever, sob forma de diagrama de estado, o funcionamento de um sistema sequencial síncrono. E então, o conduza através de uma sequência de etapas até a geração de um circuito que implemente o funcionamento desejado. As etapas para síntese são:

- Diagrama de estados
- Mapa de transição
- Associação de estados
- Tabela de transição
- Escolha do tipo de flip-flop
- Tabela de excitação
- Descrição do circuito combinacional

A interface do aplicativo deve moldar-se aos padrões usuais de GUI (*Graphical User Interface*).

A implementação do sistema deve apresentar arquitetura MVC (Modelo-Visualização-Controle ou *Model-View-Controller*) e prever suporte a Máquinas de Estado Hierárquicas (*Hierarquical State Machines*).

1.4 Metodologia

Nesse projeto adotou-se a seguinte metodologia:

- Estudo de tecnologias para a escolha da mais apropriada. Portabilidade, boa interface com o usuário, praticidade e segurança são fatores de grande importância neste projeto. Por isso, buscou-se uma linguagem de programação abrangente, em ambiente *web*, com flexibilidade de interface gráfica.
- Definição dos Casos de Uso do sistema. São elaborados os requisitos do sistema, listando quais serviços o sistema deve prover, e também os passos que o usuário deve tomar para efetuar cada ação.
- Implementação do sistema na linguagem e ambiente determinados, conforme os Casos de Uso da etapa anterior. Ao fim desta etapa, todas as funcionalidades do sistema estão implantadas.
- Ajuste da Interface Gráfica com o Usuário. São inseridos ícones, formas e cores de modo a tornar a interface gráfica mais atrativa e intuitiva, uma parte importante do objetivo deste trabalho.

1.5 Estrutura do Trabalho

Esse trabalho é composto de seis capítulos, sendo este primeiro o capítulo introdutório, que define o projeto numa visão geral.

O Capítulo 2 aborda os conceitos de Máquinas de Estado que foram aplicados no projeto.

O Capítulo 3 descreve o sistema e apresenta os Casos de Uso.

O Capítulo 4 aborda detalhes de implementação do projeto.

E finalmente, no Capítulo 5 são reunidos os resultados, cuja análise leva à conclusão do trabalho, no Capítulo 6.

Capítulo 2

Máquinas de Estados Finitos

O uso de máquinas de estado provê um modo sistemático de projetar circuitos lógicos sequenciais (Cypress, 1995).

Um circuito é dito sequencial quando as saídas dependem não somente das entradas correntes, mas também da sequência passada de entradas (Wakerly, 2005). Assim, além da lógica combinacional que determina o próximo estado e as saídas, se faz necessária a presença de um registrador para armazenar o estado das variáveis (Wikipédia, 2011).

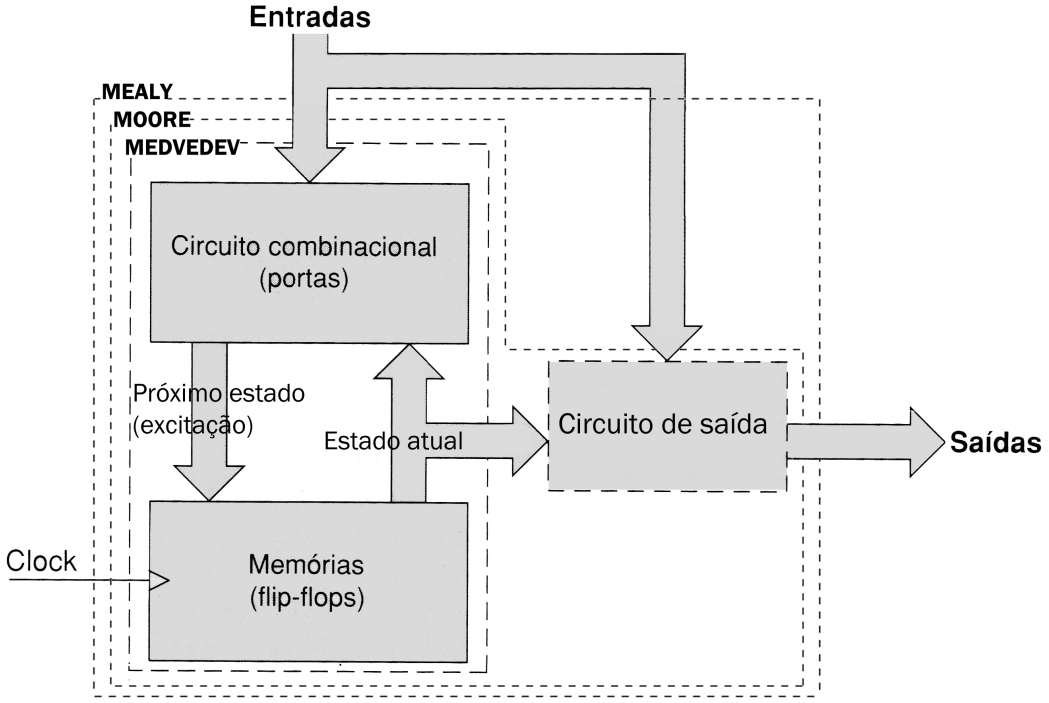


Figura 2.1: Diagrama em bloco da implementação de uma máquina de estados.

O diagrama em bloco mostrado na Figura 2.1 representa a implementação de uma máquina de estados Medvedev, Moore e Mealy.

A saída produzida por uma máquina de estado pode vir diretamente das saídas do flip-flop, ou alguns circuitos lógicos podem ser necessários, como indicado no diagrama em bloco.

O modelo mais simples é o Medvedev, onde a saída vem diretamente dos flip-flops, ou seja, a saída da máquina é igual ao estado atual dela (Glauert, 2011). No modelo Moore, a saída resulta de uma lógica que depende apenas do estado atual. E no modelo Mealy, os sinais de saída dependem também dos sinais de entrada (Moss, 2007).

O funcionamento de uma máquina de estados começa a partir do estado inicial e percorre os estados da máquina, de acordo com o vetor de entrada e a função de transição do estado corrente.

2.1 Visualização

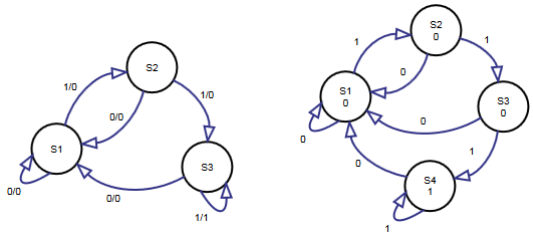
Uma máquina de estados pode ser representada graficamente de várias formas. Neste projeto, é adotada a representação por diagramas de estados, tabelas e linguagens de descrição de hardware (*Hardware Description Languages*).

Um diagrama de estados é composto por (Mano e Kime, 2007):

- Um nó de grafo para cada estado;
- Um arco direto do estado atual para o próximo estado, para cada transição de estados;
- Uma legenda em cada arco direto, com os valores de entrada que causam a transição de estados;
- E uma legenda:
  - Em cada círculo, com a saída Moore; ou
  - Em cada transição, com a saída Mealy.

A Figura 2.2 mostra o diagrama de estados de um detector da sequência ”111”, em máquinas Mealy e Moore.

Há também casos em que são desejáveis máquinas híbridas, que apresentam saídas Mealy e Moore. A Figura 2.3 ilustra um exemplo de máquina de híbrida.



(a) (b)

Figura 2.2: Diagrama de estados de máquinas Mealy(a) e Moore(b).

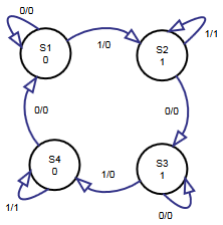


Figura 2.3: Diagrama de estados de uma máquina híbrida.

2.2 Geração de Circuito

Como citado na Seção 1.3, as etapas para a geração de circuito são: diagrama de estados, mapa de transição, associação de estados, tabela de transição, escolha do tipo de flip-flop, tabela de excitação e por fim a descrição do circuito.

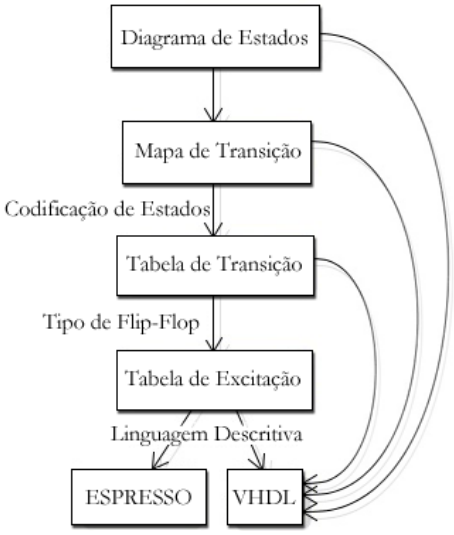


Figura 2.4: Fluxograma das Etapas de Síntese.

Para compreensão das etapas de projeto, será ilustrado o projeto de um circuito detector da sequência 1101 (Mano e Kime, 2008).

2.2.1 Diagrama de Estados

Trata-se da representação do comportamento desejado do circuito. Essa etapa é opcional, uma vez que o mapa de transição apresenta as mesmas informações do diagrama. Porém, é mais frequente e muito mais intuitivo construir o diagrama de estados do que o mapa de transição.

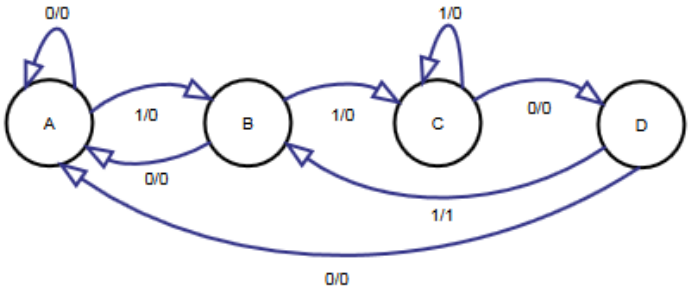


Figura 2.5: Diagrama de Estados de um Detector da Sequência “1101”.

2.2.2 Mapa de Transição

O mapa de transição apresenta em uma tabela os mesmos dados apresentados no diagrama de estados. O diagrama é mais intuitivo no primeiro momento, mas para partir para as etapas seguintes, é interessante passar para uma representação tabular.

O mapa de transição relaciona os estados atuais, os estados seguintes e a saída, de acordo com o vetor de entrada do sistema.

Estado Atual	Próximo Estado		Saída	
	X = 0	X = 1	X = 0	X = 1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Figura 2.6: Mapa de Transição para o Diagrama da Figura 2.5

2.2.3 Associação de Estados

Até esse ponto, os estados são referenciados por um nome da preferência do projetista. Para prosseguir, deve-se determinar uma codificação para cada estado.

Há vários tipos de codificação, como por exemplo, *One-hot*, *Gray* ou Binária. Codificações diferentes geram circuitos diferentes, às vezes de complexidades diferentes. Porém, não há um método automático de escolher a codificação que gerará o circuito mais simples, cabendo ao projetista essa avaliação.

Para esse exemplo, foi escolhida a codificação Gray de 2 bits. Ou seja, os estados A, B, C e D são apresentados como 00, 01, 11 e 10, respectivamente.

2.2.4 Tabela de Transição

Substituindo-se o nome dos estados por suas codificações no mapa de transição, obtém-se a tabela de transição.

Estado Atual	Próximo Estado		Saída	
AB	X = 0	X = 1	X = 0	X = 1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Figura 2.7: Tabela de Transição para o Diagrama da Figura 2.5

2.2.5 Escolha do Tipo de Flip-Flop

A escolha do tipo de flip-flop vai determinar de que forma o bloco de lógica combinacional representará o próximo estado para os flip-flops, de acordo com a Figura 2.8. Os flip-flops mais usados são do tipo D e JK.

Tipo de Flip-Flop	Estado atual → Próximo estado			
	0 →0	0 →1	1 →0	1 →1
D	0	1	0	1
T	0	1	1	0
JK	0X	1X	X1	X0
RS	X0	01	10	0X

Figura 2.8: Excitação necessária para causar cada transição, para flip-flops dos tipos D, T, JK e RS.

2.2.6 Tabela de excitação

Supondo que se escolha o flip-flop do tipo JK, obtém-se a seguinte tabela de excitação:

Estado Atual	Próximo Estado		Saída	
AB	X = 0	X = 1	X = 0	X = 1
00	0X0X	0X1X	0	0
01	0XX1	1XX0	0	0
11	X0X1	X0X0	0	0
10	X10X	X11X	0	1

Figura 2.9: Tabela de Excitação para o Diagrama da Figura 2.5

2.2.7 Descrição do circuito combinacional

Finalmente, a partir da tabela de excitação, tem-se a descrição do circuito.

Uma opção é representá-lo no formato ESPRESSO, que possibilita a minimização do circuito com o auxílio de um algoritmo desenvolvido por Robert Brayton, na Universidade de Califórnia, Berkeley (Brayton, 1982).

Outra opção é o uso de linguagens de descrição de hardware, como VHDL. Nesse caso, baseia-se apenas no diagrama de estados, sem levar em consideração as codificações de estados e flip-flops.

As sínteses do circuito em VHDL e ESPRESSO são apresentadas na Figuras 2.10 e 2.11.

```
library ieee;
use ieee.std_logic_1164.all;

entity MACHINE is
	port ( RST, CLOCK: in std_logic;
	       Inputs: in std_logic_vector (0 DOWNTO 0);
	       Mealy: out std_logic_vector (0 DOWNTO 0)
	      );
end;

architecture BEHAVIOR of MACHINE is
	type state_type is (A,B,C,D);
	signal CURRENT_STATE, NEXT_STATE: state_type;
begin
	combin: process (CURRENT_STATE, Inputs)
	begin
	case CURRENT_STATE is
		when A =>
			IF Inputs(0)='1' THEN
				Mealy <= "0";
				NEXT_STATE <= B;
			ELSIF Inputs(0)='0' THEN
				Mealy <= "0";
				NEXT_STATE <= A;
			END IF;

		when B =>
			IF Inputs(0)='1' THEN
				Mealy <= "0";
				NEXT_STATE <= C;
			ELSIF Inputs(0)='0' THEN
				Mealy <= "0";
				NEXT_STATE <= A;
			END IF;

		when C =>
			IF Inputs(0)='0' THEN
				Mealy <= "0";
				NEXT_STATE <= D;
			ELSIF Inputs(0)='1' THEN
				Mealy <= "0";
				NEXT_STATE <= C;
			END IF;

		when D =>
			IF Inputs(0)='1' THEN
				Mealy <= "1";
				NEXT_STATE <= B;
			ELSIF Inputs(0)='0' THEN
				Mealy <= "0";
				NEXT_STATE <= A;
			END IF;
		end case;
	end process;

	sync: process
	begin
		if RST='1' then
			CURRENT_STATE <= A;
		elsif rising_edge(CLOCK) then
			CURRENT_STATE <= NEXT_STATE;
		end if;
	end process;
end BEHAVIOR;
```

Figura 2.10: Descrção em VHDL para o Diagrama da Figura 2.5

```
.i 3
.o 3
001 0-1-0
000 0-0-0
010 0--10
011 1--00
111 -0-00
110 -0-10
100 -10-0
101 -11-1
.e
```

Figura 2.11: Descrição em ESPRESSO para o circuito de excitação representado na Figura 2.9

Capítulo 3

Descrição da Ferramenta

O OrangeCAD Web consiste em uma ferramenta de apoio à síntese de circuitos lógicos sequenciais. A partir de um diagrama de estados, desenhado através do próprio aplicativo, o usuário é guiado por etapas de projeto até a síntese do circuito, em código gerado automaticamente.

É um aplicativo de grande utilidade para qualquer projetista, mas o maior intuito é o uso didático. Por isso, é *online* e possui uma interface gráfica leve e intuitiva.

O OrangeCAD Web é desenvolvido em Javascript e HTML5, e roda inteiramente no cliente (*client-side*). Essa característica é muito vantajosa pois, uma vez carregada, a página não requer conexão ao servidor, tornando dispensável o acesso à Internet. É possível, também, que o usuário salve a página em seu computador e a execute diretamente da máquina. Nada disso seria possível se fosse utilizada uma linguagem voltada a servidor, como PHP e .NET, que requerem conexão com a internet para trocarem informações com o servidor.

3.1 Casos de Uso

O Diagrama de Casos de Uso apresenta uma visão externa do comportamento do sistema, possibilitando sua compreensão e ilustrando as funções disponíveis para o usuário (Guedes, 2004).

A Figura 3.1 exibe o Diagrama de Casos de Uso do OrangeCAD Web.

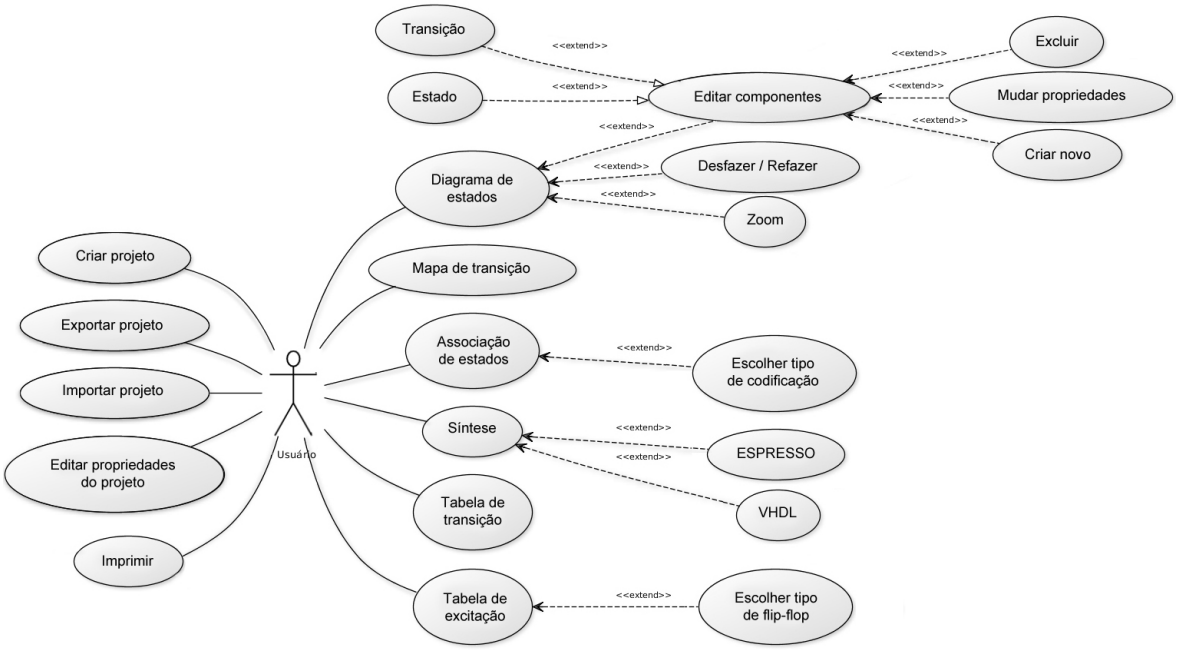


Figura 3.1: Diagrama de Casos de Uso do sistema.

3.1.1 Funções de uso geral

São opções e funcionalidades que se aplicarão ao projeto como um todo, e estarão disponíveis ao usuário a qualquer momento.

- **Criar Projeto:** Cria um novo projeto. As informações do projeto atual são perdidas se não forem salvas antes.
- **Editar Propriedades do Projeto:** Possibilita ao usuário determinar o número de bits de entrada, saída Mealy e saída Moore.
- **Exportar Projeto:** Permite salvar o projeto. Assim, o usuário pode fechar a página e continuar o projeto em outro momento.
- **Importar Projeto:** Carrega os dados de um projeto previamente exportado, permitindo sua manipulação.
- **Imprimir:** Gera página para impressão do diagrama, das tabelas ou da síntese.

3.1.2 Funções de uso específico

As etapas de projeto citadas na Seção 1.3 devem estar bem definidas no aplicativo. Assim, o usuário pode navegar entre elas, avançando cada etapa de projeto até obter a síntese.

- **Diagrama de Estados:** É a primeira etapa de projeto. Se trata de um menu de ferramentas e um painel, onde o usuário cria e modifica o diagrama de estados da máquina que ele deseja sintetizar. Nessa etapa, o usuário pode efetuar operações de:
  - **Manter Componente:** Criar e modificar propriedades de estados e transições entre estados.
  - **Zoom:** Modificar a escala de visualização do diagrama de estados.
  - **Simulação:** Visualizar o vetor de saídas e transições, resultantes de um vetor de entradas inserido pelo usuário.
  - **Desfazer / Refazer:** Desfazer e refazer as últimas modificações feitas no diagrama, facilitando sua confecção.
- **Mapa de Transição:** De acordo com as propriedades da máquina e do diagrama de estados, o sistema apresenta o mapa de transição do projeto atual. Dadas as possíveis entradas do sistema e os possíveis estados atuais, o mapa de transições exibe o próximo estado em cada caso.
- **Associação de Estados:** Nessa etapa, o usuário escolhe a codificação dos estados. Ele pode selecionar codificação **binária**, ***One-Hot*** ou ***Gray***. Para cada escolha, o sistema sugere a codificação numa tabela, mas permite ao usuário modificar a tabela manualmente, respeitando a condição de unicidade.
- **Tabela de Transição:** Nessa etapa, o usuário visualiza a tabela de transição que o sistema gera automaticamente, baseado no mapa de transição e na associação de estados.
- **Tabela de Excitação:** Dado o tipo de flip-flop escolhido pelo usuário, é gerada e exibida a tabela de excitação.
- **Síntese:** Disponibiliza ao usuário o código da máquina configurada. O usuário seleciona a linguagem e o sistema gera o código automaticamente.
  - **VHDL:** Escolhendo a linguagem VHDL para síntese, o código gerado pelo sistema pode ser usado em um dispositivo lógico programável, obtendo sua configuração em hardware.
  - **ESPRESSO:** Nesse formato, o código gerado pode servir como entrada para o software Espresso, que gera a minimização do circuito combinacional de excitação.

Capítulo 4

Implementação

4.1 Ambiente

Atendendo ao principal objetivo, que é a facilidade de uso, optou-se pelo ambiente *web*. Ultimamente, os navegadores têm se aprimorado muito, e com o uso da tecnologia HTML5, é possível desenvolver uma página de comportamento estável e compatível com a grande maioria dos navegadores atuais.

4.2 Linguagem

Foi utilizada a linguagem Javascript, visando uma página dinâmica, ágil e que rodasse inteiramente no cliente.

Os componentes básicos da página, como menus e botões, foram controlados com o auxílio das bibliotecas JQuery e JQuery UI (JQuery, 2010; JQueryUI, 2010). Os componentes gráficos, presentes no editor do diagrama de estados, foram manipulados com o uso da biblioteca Raphaël (Baranovskiy, 2008). Para implementar a exportação e importação, os dados do projeto corrente são passados para o formato JSON (*JavaScript Object Notation*)(Crockford, 2006).

4.3 Arquitetura

Model-View-Controller (MVC) é um padrão de arquitetura onde os componentes de um *software* são divididos em três grupos:

- Modelo: Dados e funções elementares
- Visualização: Representação visual do Modelo na interface gráfica com o usuário
- Controlador: Permite ao usuário interagir com o Modelo e a Visualização.

O OrangeCAD Web foi desenvolvido baseado nessa arquitetura.

4.3.1 Modelo

Há três modelos no sistema: Diagrama, Estado e Caminho.

O modelo Diagrama guarda as propriedades gerais da máquina, como o número de bits de entrada e saídas e a lista de estados.

O modelo Estados guarda características elementares de cada estado, como sua posição, sua saída Moore e a lista de caminhos que chegam ou partem dele.

E o modelo Caminho armazena informações como seu rótulo, entradas, saídas Mealy e os estados origem e destino.

4.3.2 Visualização

Conforme mostrado no Capítulo 3, as etapas de projeto de síntese são bem definidas no aplicativo. Elas foram dispostas em abas: em cada aba, há uma forma diferente de

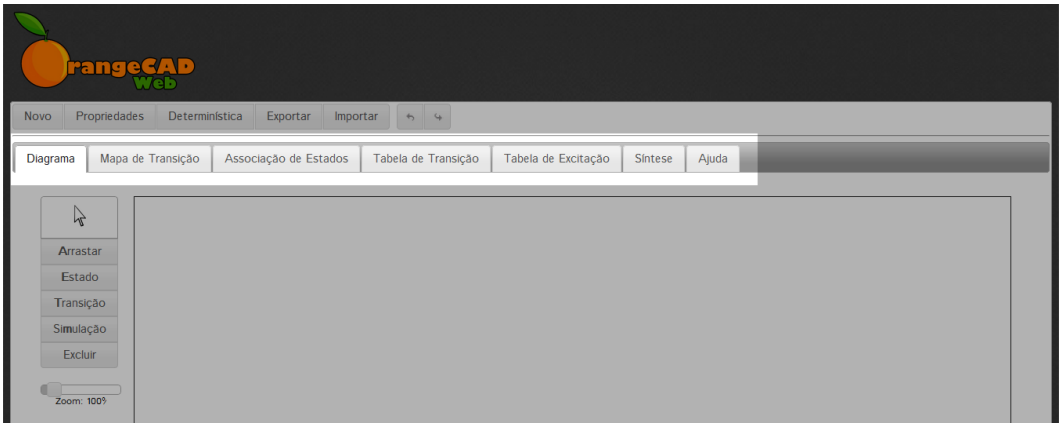
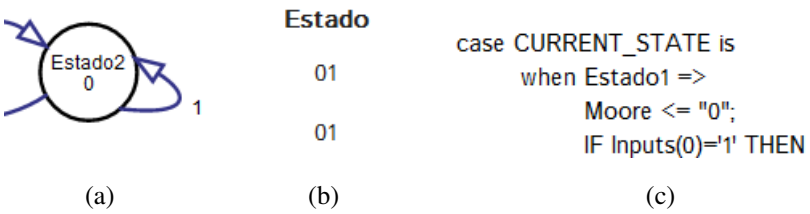


Figura 4.1: Página inicial. Em destaque, as abas representando cada etapa de projeto.

visualização dos mesmos dados. Um mesmo estado, por exemplo, pode ser representado em um diagrama de estados, em uma tabela ou em código, como mostrado na Figura 4.2. Ou seja, as abas determinam o modo de Visualização dos dados presentes no Modelo.



(a) (b) (c)

Figura 4.2: Vizualizações de um estado em um diagrama de estados (a), na tabela de excitação (b) e na síntese em VHDL (c).

4.3.3 Controladores

Para cada modelo do sistema há um controlador relacionado. Os controladores lêem e alteram informações na Visualização e no Modelo, criando uma interface do usuário com os dados internos do sistema. Cada clique, movimento de mouse e entrada de teclado é tratado por controladores e a ação devida é tomada.

Por exemplo, se o usuário clica na aba Diagrama de Estados, o Controlador modifica a Visualização de forma que sejam apresentados na tela, em forma de diagrama de estados, os dados atuais do Modelo. Cada informação que o usuário alterar no painel encadeará uma modificação na instância do respectivo Modelo. Se um estado é criado no diagrama, o Controlador altera o vetor de estados da instância do modelo Diagrama, o que acarreta a criação de mais uma instância de Estado.

4.4 Exportação e Importação

A idéia inicial era possibilitar ao usuário salvar um projeto em arquivo, para abri-lo posteriormente. Porém, por motivos de segurança, o Javascript não tem permissão para manipular o disco da máquina cliente. Como alternativa, optou-se pelo uso do JSON. JSON é um padrão nativo do Javascript e, através das funções *stringify* e *parse*, permite converter um objeto Javascript para texto e vice-versa, respectivamente. Assim, o usuário pode salvar o texto JSON gerado para exportar seu projeto e importá-lo depois, como era o objetivo.

Capítulo 5

Resultados

5.1 O OrangeCAD Web

A Figura 5.1 mostra a página inicial do OrangeCAD Web. Na parte superior da tela são dispostos botões de uso geral, que estarão disponíveis ao usuário a todo momento. Logo abaixo, são dispostas as abas. A aba inicial é o Diagrama de estados, que é a primeira etapa para síntese.

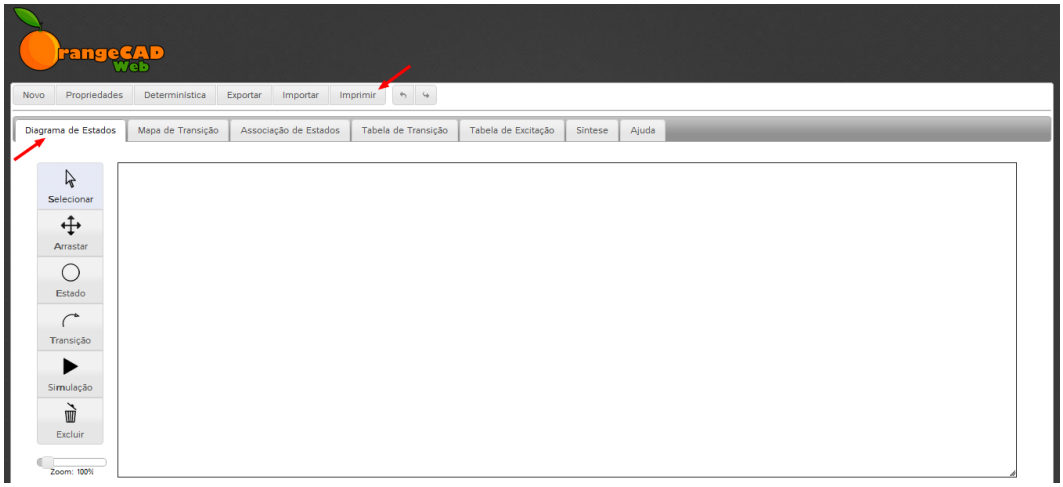


Figura 5.1: Página inicial. Em destaque, os botões de uso geral e as abas.

5.2 Diagrama de estados

Nesta primeira etapa, o OrangeCAD Web exibe um menu e um painel, que viabilizam a criação da máquina de estados em forma de diagrama.

5.2.1 Propriedades

O primeiro passo pra criação do diagrama é determinar as propriedades do projeto, clicando no botão **Propriedades**. É exibida uma tela como a da Figura 5.2, onde o usuário deve digitar o número de bits de entrada, de saída Moore e de saída Mealy.

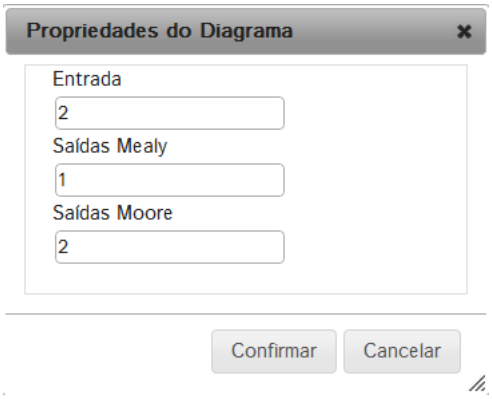


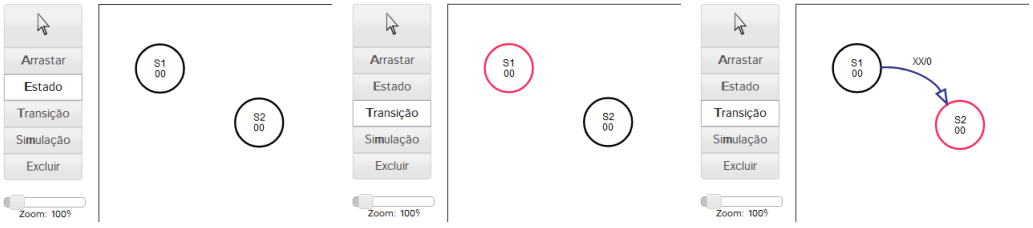
Figura 5.2: Tela de configuração das propriedades do projeto.

5.2.2 Criando componentes do diagrama

Após configuradas as propriedades, pode-se iniciar a edição do diagrama, com o uso do menu que aparece do lado esquerdo da tela.

Para criar um estado, deve-se selecionar o botão **Estado** ou digitar o atalho **E**. Assim, entra-se no modo de criação de estados, e cada clique no painel criará um estado novo.

Para criar uma transição entre dois estados, seleciona-se o botão **Transição** ou pode-se usar o atalho **T**. Para cada transição a ser criada, são necessários dois cliques: o primeiro seleciona o estado de origem da transição a ser criada, e o segundo seleciona o estado destino.

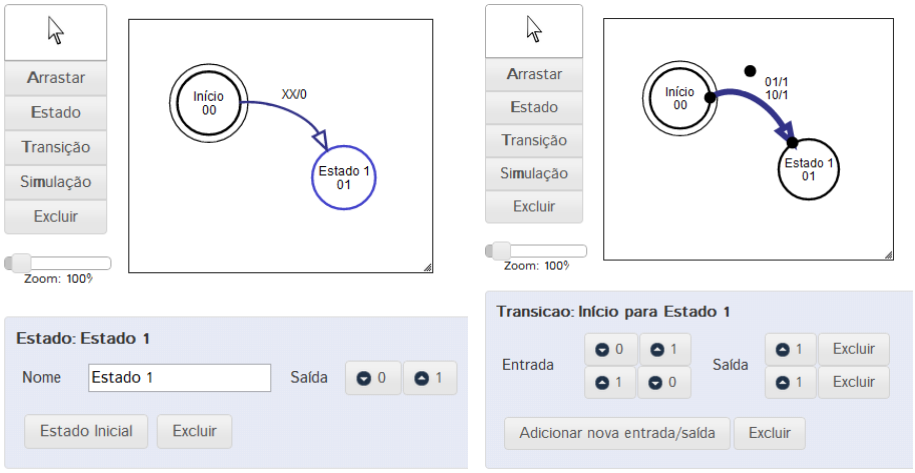


(a) (b) (c)

Figura 5.3: Criação de estados (a) e de uma transição (b) e (c).

5.2.3 Manipulando componentes do diagrama

Para obter um diagrama de estados válido, após criar estados e transições, é necessário editar as propriedades desses componentes. Para isso, o projetista deve clicar no ícone , ativando o modo **Selecionar** (atalho: **S**). Nesse modo, ao selecionar um componente (estado ou transição), o projetista percebe a exibição de um quadro de propriedades abaixo do painel do diagrama, conforme a Figura 5.4.



(a) (b)

Figura 5.4: Propriedades de um estado (a) e de uma transição (b).

No caso de um estado, as propriedades disponíveis são: nome, saída Moore e a opção de ser o estado inicial. A manipulação do nome é importante para a legibilidade do projeto. É o nome que aparece no círculo do diagrama, no mapa de transição e no código VHDL da síntese. A saída Moore é a saída que depende somente do estado atual da máquina, e é exibida no círculo do estado, abaixo do nome. No OrangeCAD Web, é obrigatória a existência de um estado inicial na máquina. Assim, por padrão, o primeiro estado criado é considerado o estado inicial da máquina, sendo permitido ao usuário selecionar outro estado para ser o inicial. O estado inicial é exibido de forma diferenciada no diagrama, como um círculo duplo. É possível, ainda, mudar a posição do estado, simplesmente clicando sobre ele e o arrastando para a posição desejada.

A principal propriedade de um transição é seu rótulo, um texto que inclui a entrada e a saída Mealy da transição, separados por uma barra (“/”). Uma transição, ou ramo, pode ter vários rótulos. Ou seja, várias entradas diferentes podem causar a mesma transição. A Figura 5.4(b) mostra os botões da janela de propriedades de transição, que possibilitam criar e editar rótulos. Cada bit é controlado através de um botão que, ao ser clicado, alterna seu valor entre as opções **0**, **1** e **X** (*don’t-care*).

Para modificar o formato dos arcos de transição, são exibidos três pontos de controle sobre o ramo selecionado: um em cada extremo do ramo, e um ponto próximo ao meio do arco. O usuário pode mover os pontos dos extremos livremente, apenas respeitando os limites do círculo onde o arco está fixo. Movendo o terceiro ponto, o usuário pode controlar a inclinação do arco conforme desejar.

Há outras opções de manipulação do diagrama. São elas:

- **Arrastar:** Permite que o usuário arraste todo o diagrama, ajustando sua posição no painel. Tecla de atalho: **A**.
- **Zoom:** Ajusta a escala do diagrama, entre 30 e 500 por cento.
- **Excluir:** Clicando nesse botão, o sistema entra no modo excluir e exclui todo componente selecionado. Também é possível excluir um componente através da tecla **Del**.
- **Completar:** Essa opção completa a máquina, criando transições que explicitem qual o próximo estado em todos os casos possíveis de entradas. O sistema toma como padrão que entradas não cobertas anteriormente geram transições para o estado corrente.

5.2.4 Simulação

O modo Simulação pode ser ativado através do botão **Simulação** ou do atalho **M**. Nesse modo, pode-se testar o comportamento da máquina, dado um vetor de entradas. Abaixo do painel do diagrama, é exibido um quadro (Figura 5.5), onde o usuário insere o vetor de entradas a serem testadas.

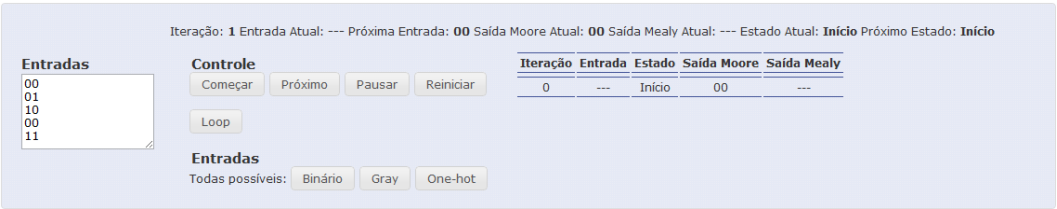
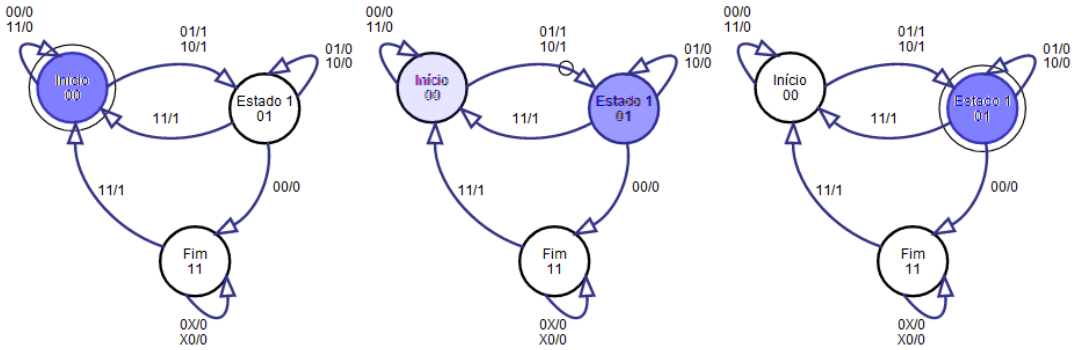


Figura 5.5: Quadro para controle de simulação.

A Figura 5.6 mostra que reação da máquina a cada entrada pode ser vista em uma tabela no quadro (a) e também em uma animação no diagrama (b).

Iteração	Entrada	Estado	Saída Moore	Saída Mealy
0	---	Início	00	---
1	00	Início	00	0
2	01	Estado 1	01	1
3	10	Estado 1	01	0
4	00	Fim	11	0
5	11	Início	00	1

(a)



(b)

Figura 5.6: Resultados da simulação exibidos em forma de tabela (a) e animação (b).

	00	01	10	11
Início	Início	Estado 1	Estado 1	Início
Estado 1	Fim	Estado 1	Estado 1	Início
Fim	Fim	Fim	Fim	Início

1

Figura 5.7: Mapa de transição.

5.3 Mapa de transição

Nesta aba, o usuário visualiza o mapa de transição do diagrama criado.

Para cada estado da máquina, o mapa de estados exibe o próximo estado em todos os casos possíveis de entradas. Note que o número de entradas possíveis pode ser elevado, o que tornaria difícil a visualização da tabela. Assim, a tabela é paginada, com 10 colunas por página.

No mapa de transição, os estados são representados pelo nome, o que facilita sua análise. É também útil para detecção de duas possíveis características da máquina em construção: inconsistência e não-determinismo. O primeiro caso ocorre quando, para determinado estado, uma mesma entrada ativa mais de uma transição, gerando uma indeterminação do próximo estado. Nesse caso, o mapa exibe o próximo estado como **Indefinido** e o usuário pode, então, voltar ao diagrama de estados para corrigir o problema. O segundo caso é quando as

transições que saem de determinado estado não cobrem todas as entradas possíveis. Nesse caso, o mapa de transição exibe o símbolo "**-**". O usuário tem a opção de voltar ao diagrama e fazer as correções manualmente, ou deixar que a correções sejam feitas automaticamente, através do botão **Determinística**. É possível também prosseguir para a próxima etapa, sem que sejam feitas correções.

	00	01	10	11
Início	Início	Estado 1	-	Início
Estado 1	Fim	Indefinido	-	Início
Fim	Fim	Fim	Fim	Início

Figura 5.8: Mapa de transição de uma máquina inconsistente e não-determinística.

5.4 Associação de estados

Associa um código único para cada estado. Esse código irá representar o estado nas etapas seguintes, se aproximando da síntese.

Podem ser escolhidas três codificações pré-definidas no aplicativo: binária, Gray e One-Hot. O sistema gera, então uma tabela que relaciona o nome do estado com sua codificação. Além das três opções de codificação, o usuário pode editar a tabela gerada e fazer a codificação manualmente.

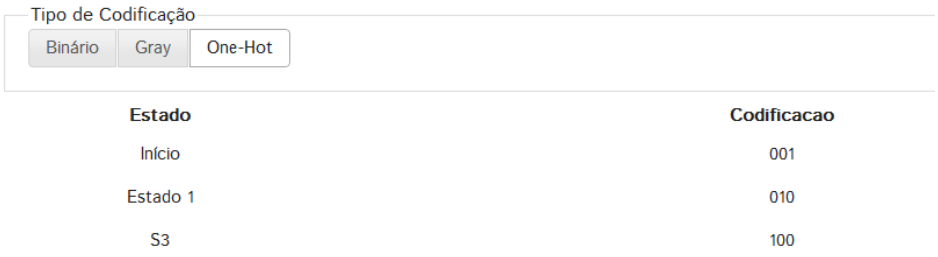


Figura 5.9: Tabela para a associação de estados.

5.5 Tabela de transição

Esta tabela tem a mesma estrutura do mapa de transição, porém os estados não são representados pelo nome, mas pela codificação definida na etapa anterior.

	00	01	10	11
001	001	010	010	001
010	100	010	010	001
100	100	100	100	001

Figura 5.10: Exemplo de tabela de transição.

5.6 Tabela de excitação

Nesta etapa, o sistema exibe as opções de tipo de flip-flop e, de acordo com a escolha do usuário, gera a tabela de excitação correspondente. Após essa última escolha de projeto, tem-se todos os dados necessários para a síntese.

Tipo de FlipFlop

D T JK RS

Estado	Entrada	Próximo	Saída
001	01	0X1XX1	Moore: 01 Mealy: 1
001	10	0X1XX1	Moore: 01 Mealy: 1
001	00	0X0XX0	Moore: 00 Mealy: 0
001	11	0X0XX0	Moore: 00 Mealy: 0
010	01	0XX00X	Moore: 01 Mealy: 0
010	10	0XX00X	Moore: 01 Mealy: 0
010	11	0XX11X	Moore: 00 Mealy: 1
010	00	1XX10X	Moore: 11 Mealy: 0
100	11	X10X1X	Moore: 00 Mealy: 1
100	0X	X00X0X	Moore: 11 Mealy: 0
100	X0	X00X0X	Moore: 11 Mealy: 0

Figura 5.11: Exemplo de tabela de excitação.

5.7 Síntese

Dados o diagrama de estados e as escolhas de projeto tomadas nas etapas anteriores, o OrangeCAD Web gera automaticamente a síntese da máquina de estados projetada. Para obter a síntese, basta um clique na linguagem de preferência do projetista: ESPRESSO ou VHDL. A seguir, a síntese do diagrama da Figura 5.6 nos dois formatos.

5.7.1 Síntese em ESPRESSO

```
# ================================================
#   UFES - Universidade Federal do Espirito Santo
#   Descricao do circuito no formato ESPRESSO
#   Arquivo gerado automaticamente por OrangeCAD Web em
#   Wed Aug 17 2011 02:30:02 GMT-0300 (E. South America Standard Time)
# ================================================
.i 5
.o 6
00101 0-1--1011
00110 0-1--1011
00100 0-0--0000
00111 0-0--0000
01001 0--00-010
01010 0--00-010
01011 0--11-001
01000 1--10-110
10011 -10-1-001
1000- -00-0-110
100-0 -00-0-110
10010 -00-0-110
.e
```

5.7.2 Síntese em VHDL

```
# ================================================
#   UFES - Universidade Federal do Espirito Santo
#   Descricao do circuito no formato VHDL
#   Arquivo gerado automaticamente por OrangeCAD Web em
#   Fri Aug 05 2011 06:17:17 GMT-0300 (E. South America Standard Time)
# ================================================
library ieee;
use ieee.std_logic_1164.all;

entity MACHINE is
port ( RST, CLOCK: in std_logic;
Inputs: in std_logic_vector (1 DOWNTO 0);
```

```
Moore: in std_logic_vector (1 DOWNTO 0);
Mealy: in std_logic_vector (0 DOWNTO 0)
);
end;
architecture BEHAVIOR of MACHINE is
type state_type is (Início,Estado 1,Fim);
signal CURRENT_STATE, NEXT_STATE: state_type;
begin
combin: process (CURRENT_STATE, Inputs)
begin
case CURRENT_STATE is
when Início =>
Moore <= "00";
IF Inputs(0)='0' and Inputs(1)='1' THEN
Mealy <= "1";
NEXT_STATE <= Estado 1;
END IF;
IF Inputs(0)='1' and Inputs(1)='0' THEN
Mealy <= "1";
NEXT_STATE <= Estado 1;
END IF;
IF Inputs(0)='0' and Inputs(1)='0' THEN
Mealy <= "0";
NEXT_STATE <= Início;
END IF;
IF Inputs(0)='1' and Inputs(1)='1' THEN
Mealy <= "0";
NEXT_STATE <= Início;
END IF;
when Estado 1 =>
Moore <= "01";
IF Inputs(0)='0' and Inputs(1)='1' THEN
Mealy <= "0";
NEXT_STATE <= Estado 1;
END IF;
IF Inputs(0)='1' and Inputs(1)='0' THEN
Mealy <= "0";
NEXT_STATE <= Estado 1;
```

```
END IF;
IF Inputs(0)='1' and Inputs(1)='1' THEN
Mealy <= "1";
NEXT_STATE <= Início;
END IF;
IF Inputs(0)='0' and Inputs(1)='0' THEN
Mealy <= "0";
NEXT_STATE <= Fim;
END IF;
when Fim =>
Moore <= "11";
IF Inputs(0)='1' and Inputs(1)='1' THEN
Mealy <= "1";
NEXT_STATE <= Início;
END IF;
IF Inputs(0)='0' THEN
Mealy <= "0";
NEXT_STATE <= Fim;
END IF;
IF Inputs(1)='0' THEN
Mealy <= "0";
NEXT_STATE <= Fim;
END IF;
end process;

sync: process
begin
if RST='1' then
CURRENT_STATE <= Início;
elsif rising_edge(CLOCK) then
CURRENT_STATE <= NEXT_STATE;
end if;
end process;
end behavior;
```

Capítulo 6

Conclusão

6.1 Considerações Finais

O aplicativo OrangeCAD Web foi desenvolvido de acordo com a metodologia citada no Capítulo 1 e os resultados obtidos foram satisfatórios.

Os resultados mais significativos deste trabalho são as características favoráveis apresentadas pelo OrangeCAD Web. Entre elas, cabe destacar:

- Apresenta etapas de projeto bem definidas, auxiliando na tomada de decisão e facilitando ajustes.
- Tem grande valor didático: ao prover uma visualização da sistemática utilizada para síntese, viabiliza a compreensão do processo e a fixação dos conceitos envolvidos.
- É disponibilizado em ambiente *web*, tornando seu uso mais convidativo e prático, se comparado a um software que precisa ser instalado antes do uso.
- Possui interface com o usuário bastante leve e intuitiva, provocando o interesse do usuário desde seu primeiro contato com o aplicativo.
- Permite que o usuário crie um vetor de entradas e simule o comportamento da máquina projetada, podendo, assim, detectar erros ou situações imprevistas e indesejadas.
- Detecta a criação de máquinas incompletas ou inconsistentes.

Tendo em vista os objetivos propostos e os resultados obtidos, considera-se que este trabalho teve seu propósito alcançado, ao oferecer apoio efetivo à síntese de circuitos lógicos

sequenciais. A partir da descrição do circuito desejado em um diagrama de estados criado no próprio aplicativo, segue-se um conjunto de etapas pré-definidas e atinge-se a síntese do circuito, de forma intuitiva e rápida.

6.2 Trabalhos Futuros

Este trabalho pode servir como ponto de partida para novos projetos. Sugere-se, como continuidade, a implementação de algumas melhorias e extensões.

Uma possível adaptação desse aplicativo prevê a criação de máquinas de estados hierárquicos, possibilitando um uso mais abrangente do aplicativo.

O OrangeCAD Web oferece a opção de síntese no formato ESPRESSO, visando sua minimização. Para efetuar essa minimização, contudo, é necessário o uso de uma ferramenta externa. Portanto, outra melhoria sugerida é inserir, no próprio aplicativo, uma etapa de minimização de circuito. Assim, o processo de minimização se tornaria tão prático quanto as demais etapas automatizadas do aplicativo.

Por fim, seria interessante permitir ao usuário entrar com a descrição da máquina através de outras etapas do processo, como a Tabela de Transição ou a própria síntese. Nesse caso, o sistema faria o caminho inverso e geraria o Diagrama de Estados correspondente. Essa alternativa teria utilidade no caso de o projetista já possuir de antemão um circuito pronto, e desejar voltar às etapas iniciais afim de efetuar pequenas mudanças no circuito, ou mesmo para formalizar seu projeto e estudá-lo.

Bibliografia

Baranovskiy, D. (2008). Raphaël. http://raphaeljs.com/.

Brayton, R. K. (1982). Espresso. http://lnk.nu/embedded.eecs.berkeley.edu/1rdz.htm.

Cooper, D. (2008). *Finite State Machine in PSoC Designer*. Cypress Semiconductor Corporatio. http://www.cypress.com/?docID=28701.

Crockford, D. (2006). Json.org. http://www.ietf.org/rfc/rfc4627.txt?number=4627.

Cypress (1995). State machine design considerations and methodologies. http://lnk.nu/embeddeddesignindia.co.in/1rdy.PDF.

Glauert, W. H. (2011). Fsm tutorial. Institute for Computer Aided Circuit Design, University of Erlangen-Nuremberg. http://lnk.nu/vhdl-online.de/1rdx.htm.

Guedes, G. T. A. (2004). *UML Uma Abordagem Prática*. Novatec Editora Ltda.

JQuery (2010). http://jquery.com/.

JQueryUI (2010). http://jqueryui.com/.

Lee, E. A. (2009). Finite state machines and modal models in ptolemy ii. Relatório Técnico UCB/EECS-2009-151, EECS Department, University of California, Berkeley. http://lnk.nu/eecs.berkeley.edu/1re1.html.

Mano, M. M. e Kime, C. (2007). *Logic and Computer Design Fundamentals (4th Edition)*. Prentice Hall.

Mano, M. M. e Kime, C. (2008). Logic and computer design fundamentals.

Moss, R. J. T. N. S. W. G. L. (2007). *Sistemas Digitais: princípios e aplicações*. Pearson Prentice Hall.

Pegoretti, G. (2002). Orangecad. https://sites.google.com/site/gpegoretti/orange.

Wakerly, J. F. (2005). *Digital Design: Principles and Practices (4th Edition, Book only)*. Prentice Hall.

Wikipédia (2011). Máquina de estados finitos. [Online; accessed 20-junho-2011] http://lnk.nu/pt.wikipedia.org/1rdw.php.

Xilinx, I. *StateCAD Users Guide*. http://lnk.nu/xilinx.com/1re0.pdf.