ID: 112269

Projeto 4: Desenvolvimento de uma Máquina de Estados de *Moore* para um Kit FPGA

São José dos Campos - Brasil

Outubro de 2018

ID: 112269

Projeto 4: Desenvolvimento de uma Máquina de Estados de *Moore* para um Kit FPGA

Relatório apresentado à Universidade Federal de São Paulo como parte dos requisitos para aprovação na disciplina de Laboratório de Sistemas Computacionais: Circuitos Digitais.

Aluno: Willian Dihanster Gomes de Oliveira

Docente: Prof. Dr. Lauro Paulo da Silva Neto

Universidade Federal de São Paulo - UNIFESP

Instituto de Ciência e Tecnologia - Campus São José dos Campos

São José dos Campos - Brasil

Outubro de 2018

Resumo

Este projeto teve como principal objetivo o desenvolvimento de uma Máquina de Estados (utilizando uma Máquina de *Moore*) para um contador númerico temporizado. A máquina desenvolvida possui 4 funções, que podem ser controladas pelas entradas *UP* e *DOWN*, sendo elas: contagem crescente, contagem decrescente, manter a contagem e a função *blank* (onde todos os LEDs são apagados). Além disso, o contador segue a sequência númerica 6-9-0-2-4-6-5-3-8, fazendo uma transição de estados a cada $1s$ e sua saída é mapeada para o display de 7 segmentos do Kit FPGA.

Palavras-chaves: Circuitos Digitais, Contador Automático, Máquina de *Moore*, Kit FPGA

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Circuitos eletrônicos são usualmente separados entre duas categorias: digitais e analógicos. Enquanto que na digital, trabalha-se com grandezas com valores discretos, na analógica há o uso de grandezas contínuas (1).

Sendo assim, circuitos e sistemas digitais trabalham com dois estados possíveis, que são representados por dois níveis de tensão diferentes: um ALTO e um BAIXO, que também podem ser representados por '0' e '1'. Dessa forma, circuitos digitais servem como base para diversas aplicações, como por exemplo na construção de computadores, na automação, robótica e etc.

Um exemplo mais simples, são as máquinas de estados finitos, que representam comportamentos de um sistema por um número finito de estados e de transições (2). Assim, pode-se representar contadores, como o desenvolvido neste projeto, que fazem a contagem com uma sequência númerica, fazendo uma transição de estados a cada $1s$, de acordo com a entrada definida pelo usuário.

O trabalho está organizado como segue: Capítulo 2 detalha os objetivos do trabalho, Capítulo 3, a fundamentação teórica, Capítulo 4, o desenvolvimento, Capítulo 5, os resultados obtidos e discussões e por fim, no Capítulo 6 as considerações finais.

2 Objetivos

2.1 Geral

Este projeto tem como principal objetivo o desenvolvimento de um contador númerico, implementado com uma Máquina de Estados de *Moore*, seguindo a sequência 6-9-0-2-4-6-5-3-8, com transição de estados a cada $1s$ e com 4 formas de contagem.

2.2 Específico

Para o pleno desenvolvimento do objetivo desejado, há uma sequência de passos, definidos a seguir:

- Confecção do diagrama de estados para a sequência definida.
- Implementação do circuito combinacional que define a Máquina de Estados de *Moore*, sendo esse circuito composto por 3 partes principais (próximo estado, registradores e saída).
- Implementação do circuito que simula um temporizador de $1s$.
- Decodificação da saída do estado da máquina para seu valor correspondente (em binário) da sequência numérica, para o display de 7 segmentos.
- Criação de *black-boxes* e organização dos circuitos criados em um único circuito principal.
- Avaliação com testes de *waveforms* no *software Quartus Prime* para cada bloco do projeto.
- Avaliação do circuito final (todas as partes integradas) com *waveforms* no *software Quartus Prime* e também no Kit FPGA.

3 Fundamentação Teórica

Neste capítulo, serão detalhados alguns dos principais conceitos utilizados para a realização deste projeto.

3.1 Princípios de Circuitos Digitais

Circuitos eletrônicos são geralmente separados entre duas categorias: digitais e analógicos. Na digital, há o uso de grandezas com valores discretos, já na analógica, de grandezas contínuas (1).

Sendo assim, circuitos e sistemas digitais trabalham com dois estados possíveis, que são representados por dois níveis de tensão diferentes: um ALTO e um BAIXO, que também podem ser representados por '0' e '1'. Em circuitos digitais há duas classes principais de circuitos: circuitos combinacionais e circuitos sequencias.

Em circuitos combinacionais há o uso de portas lógicas interconectadas para produzir uma saída especificada, usando a combinação de variáveis de entrada, sem o envolvimento de armazenamento de dados. Já os circuitos sequenciais há a existência de uma seção com lógica combinacional e de uma seção de memória (*flip-flops*).

Sendo assim, fazendo o uso de porta lógicas, tabela-verdade e álgebra booleana é possível desenvolver sistemas de circuitos digitais para as mais diversas aplicações.

3.2 *Flip-flops*

Um *flip-flop* ou multivibrador biestável, pode ser definido como um elemento de memória e, geralmente, é feito de uma configuração de portas lógicas. Comumente, possuem duas saídas: Q e $\bar{Q}$, a saída normal e saída invertida, respectivamente. No estado *ALTO*, temos $Q = 1$ e $\bar{Q} = 0$, e este estado pode ser chamado de *SET*. Já no estado *BAIXO*, temos $Q = 0$ e $\bar{Q} = 1$, e pode ser chamado de *CLEAR* ou *RESET*. (3) Há diversos tipos de *flip-flops*, como o tipo *D*, *J-K*, *T* e podem ser utilizados por exemplo, para a criação de contadores, registradores ou temporizadores.

3.3 Contadores

Contadores são circuitos que podem ser implementados a partir de N *flip-flops* conectados, sendo síncronos ou assíncronos. O valor de N e a forma na qual são conectados

determina o número de estados (ou módulo) e a sequência de estados do contador. Por exemplo, com N *flip-flops* é possível implementar um contador de 0 até $2^N - 1$. (1)

3.4 Temporizadores

Em certas aplicações, é utilizado a frequência de *clock* do dispositivo. Mas essa frequência pode ser muita alta para determinados casos. Sendo assim, um circuito temporizador pode ser usado. Uma implementação possível é a divisão de frequências, em que são usados *flip-flops* ou contadores para passar a informação, gerando um certo *delay*. (1)

3.5 Máquina de Estados Finitos

As máquinas de estados finitos representam comportamentos de um sistema por um número finito de estados e de transições (2). Há dois principais modelos amplamente utilizadas, sendo elas: Máquina de *Moore* e a Máquina de *Mealy*. As principais diferenças entre as duas é sua função de saída, onde na Máquina de *Moore*, depende apenas do estado atual, já na Máquina de *Mealy*, a saída depende do estado atual e das entradas.

3.6 FPGA - *Field-Programmable Gate Array*

O FPGA - *field-programmable gate array* (matriz de portas programáveis) é um dispositivo lógico programável em que se é possível trabalhar e simular circuitos digitais. O Kit FPGA disponível no Laboratório de Circuitos Digitais é o Kit Altera DE2-115 Development and Education Board, com o FPGA Altera Cyclone IV E: EP4CE115F29C7. Esse kit contém, também, um display de 7 segmentos, que será detalhado a seguir. (4)

3.7 Display de 7 Segmentos

Um display de 7 segmentos é uma placa composta por segmentos com LEDs que podem ser ligados ou desligados individualmente, e podem, por exemplo, serem utilizados para a visualização de números decimais de 0 a 9.

3.8 *Intel Quartus Prime Software*

o *Intel Quartus Prime Software* é um software da *Intel* para o desenvolvimento de sistemas programáveis. Com ele é possível, por exemplo, criar circuitos atráves de esquemáticos ou com alguma linguagem de descrição de *hardware* como a *Verilog*. (5)

4 Desenvolvimento

Neste capítulo, serão detalhados os passos para a implementação da máquina de estados de Moore que represente a sequência ciclíca 6-9-0-2-4-6-5-3-8, com transição de estados a cada 1s e com 4 formas de contagem diferentes, como configurado na Tabela 1.

Tabela 1 – Exemplo de entradas e forma de contagem

Contagem	UP	$DOWN$
Mantém	0	0
Descrecente	0	1
Crescente	1	0
Blank	1	1

Fonte: O Autor

Portanto, considerando que estamos no estado inicial A, temos que:

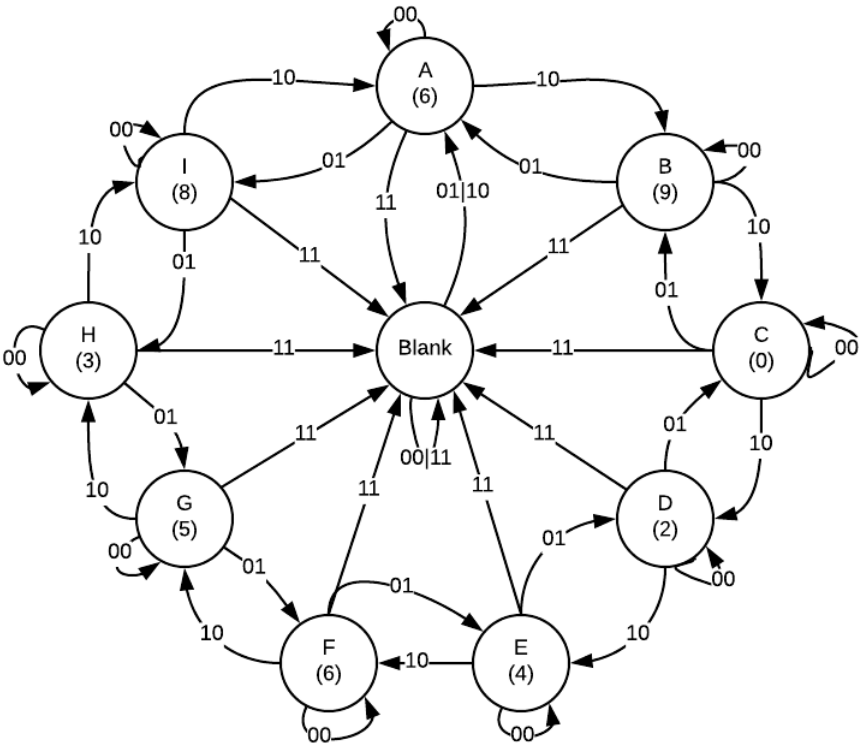
- se $UP = 0$ e $DOWN = 0$, o estado A será mantido.
- se $UP = 0$ e $DOWN = 1$, a máquina volta para o estado anterior, ou seja, a sequência apresentada será 8-3-5-6-4-2-0-9-6.
- se $UP = 1$ e $DOWN = 0$, temos a sequência crescente, ou seja, a sequência ciclíca será 6-9-0-2-4-6-5-3-8.
- se $UP = 1$ e $DOWN = 1$, a máquina mudará para o estado *Blank*, onde todos os LEDs são apagados.

Uma observação é que se o estado atual é *Blank*, e se $UP = 0$ e $DOWN = 1$ ou $UP = 1$ e $DOWN = 0$, ambos os casos levarão ao estado inicial A. Além disso, se a entrada *reset* for requisitada, a contagem também reiniciará do primeiro estado, A.

4.1 Diagrama de Estados

Inicialmente, para desenvolvimento deste projeto, foi realizada a confecção do diagrama de estados com as configurações possíveis para as entradas (UP e $DOWN$) e os estados da máquina, que pode ser conferido na Figura 1 a seguir. Este diagram foi feito com a ajuda do *software on-line, Lucid Chart* (6).

Figura 1 – Diagrama de Estados



Fonte: Autor

Sendo assim, foi usado uma esquema de codificação de 4 *bits* para a representação dos estados possíveis da máquina. Cada estado tem sua representação binária, conforme a Tabela 2. Posteriormente, cada estado será decodificado para a saída com seu valor decimal correspondente e mapeado para o display de 7 segmentos do kit FPGA.

Tabela 2 – Tabela de Codificação de Estados.

Número	Estado	$Q3$	$Q2$	$Q1$	$Q0$
6	A	0	0	0	0
9	B	0	0	0	1
0	C	0	0	1	0
2	D	0	0	1	1
4	E	0	1	0	0
6	F	0	1	0	1
5	G	0	1	1	0
3	H	0	1	1	1
8	I	1	0	0	0
Blank	*Blank*	1	0	0	1

Fonte: O Autor

4.2 Função do Próximo Estado

Um dos principais componentes de uma máquina de estados finitos é a função de próximo estado. Esta função tem como objetivo ditar qual o próximo estado em que a máquina deve estar. No caso da máquina de *Moore*, como a desejada nesse projeto, a função de próximo estado depende das entradas e do estado atual. Temos, então, na Tabela 3, as possibilidades de entradas, estado atual e próximo estado.

Tabela 3 – Tabela de próximo estado, dado uma entrada.

Estado Atual	Entradas		Estado Atual				Próximo Estado				Próximo Estado
	UP	$DOWN$	$Q3$	$Q2$	$Q1$	$Q0$	$D3$	$D2$	$D1$	$D0$	
A	0	0	0	0	0	0	0	0	0	0	A
B	0	0	0	0	0	1	0	0	0	1	B
C	0	0	0	0	1	0	0	0	1	0	C
D	0	0	0	0	1	1	0	0	1	1	D
E	0	0	0	1	0	0	0	1	0	0	E
F	0	0	0	1	0	1	0	1	0	1	F
G	0	0	0	1	1	0	0	1	1	0	G
H	0	0	0	1	1	1	0	1	1	1	H
I	0	0	1	0	0	0	1	0	0	0	I
Blank	0	0	1	0	0	1	1	0	0	1	*Blank*
A	0	1	0	0	0	0	0	0	0	1	I
B	0	1	0	0	0	1	0	0	1	0	A
C	0	1	0	0	1	0	0	0	1	1	B
D	0	1	0	0	1	1	0	1	0	0	C
E	0	1	0	1	0	0	0	1	0	1	D
F	0	1	0	1	0	1	0	1	1	0	E
G	0	1	0	1	1	0	0	1	1	1	F
H	0	1	0	1	1	1	1	0	0	0	G
I	0	1	1	0	0	0	0	0	0	0	H
Blank	0	1	1	0	0	1	0	0	0	0	A
A	1	0	0	0	0	0	1	0	0	0	B
B	1	0	0	0	0	1	0	0	0	0	C
C	1	0	0	0	1	0	0	0	0	1	D
D	1	0	0	0	1	1	0	0	1	0	E
E	1	0	0	1	0	0	0	0	1	1	F
F	1	0	0	1	0	1	0	1	0	0	G
G	1	0	0	1	1	0	0	1	0	1	H
H	1	0	0	1	1	1	0	1	1	0	I
I	1	0	1	0	0	0	0	1	1	1	A
Blank	1	0	1	0	0	1	0	0	0	0	A
A	1	1	0	0	0	0	1	0	0	1	*Blank*
B	1	1	0	0	0	1	1	0	0	1	*Blank*
C	1	1	0	0	1	0	1	0	0	1	*Blank*
D	1	1	0	0	1	1	1	0	0	1	*Blank*
E	1	1	0	1	0	0	1	0	0	1	*Blank*
F	1	1	0	1	0	1	1	0	0	1	*Blank*
G	1	1	0	1	1	0	1	0	0	1	*Blank*
H	1	1	0	1	1	1	1	0	0	1	*Blank*
I	1	1	1	0	0	0	1	0	0	1	*Blank*
Blank	1	1	1	0	0	1	1	0	0	1	*Blank*

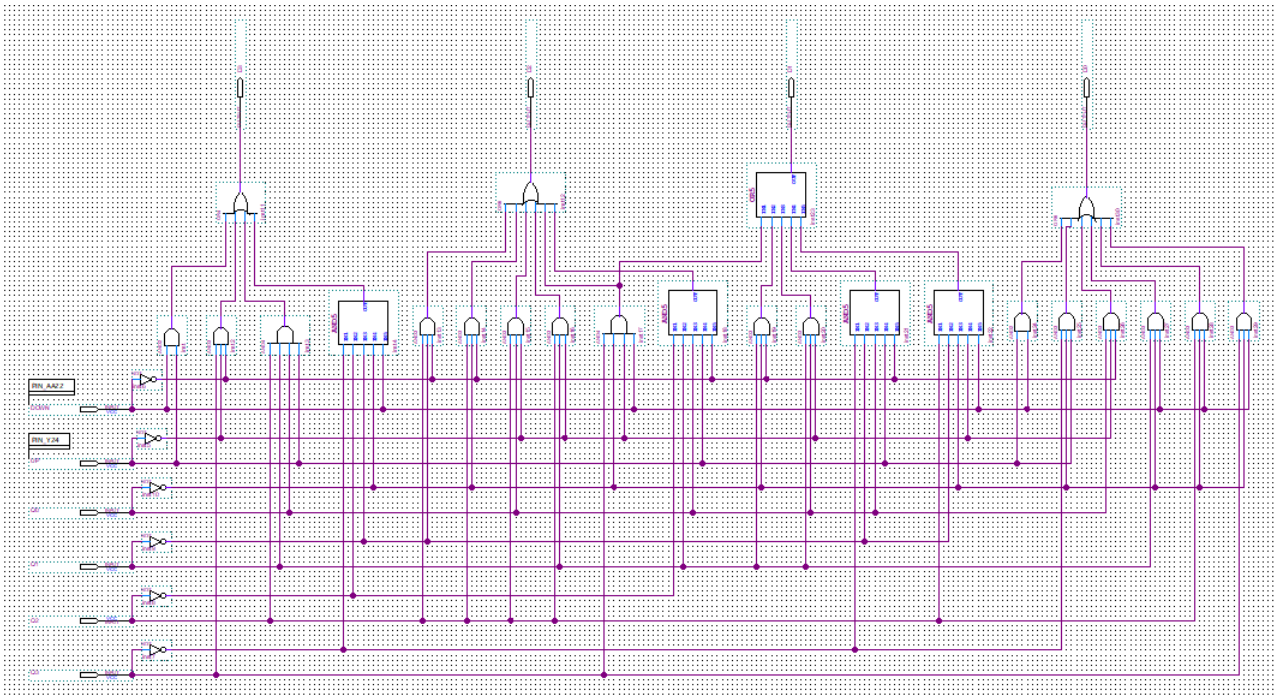
Fonte: O Autor

Nesta Tabela, as entradas são definidas por UP e $DOWN$, o estado atual é codificado pelas variáveis Q_3, Q_2, Q_1 e Q_0 e o próximo estado é codificado pelas variáveis D_3, D_2, D_1 e D_0. Dessa forma, agora é possível gerar o circuito correspondente à tabela. Para isso, extraiu-se os mintermos, e com ajuda do site $32x8$ (7) para simplificação com o Mapa de *Karnaugh*, foram retiradas as expressões que descrevem o circuito da função de próximo estado, que podem ser conferidas a seguir.

- $D_3 = UPDOWN + UP'DOWNQ_3 + UPQ2Q1Q0 + DOWNQ_3'Q_2'Q_1'Q_0'$
- $D_2 = DOWN'Q_2Q_1' + DOWN'Q_2q_0' + UP'Q_2q_0 + UP'Q_2Q_1 + UP'DOWNQ_3Q_0' + UPDOWN'Q_2'Q_1Q_0$
- $D_1 = DOWN'Q_1Q_0' + UP'Q_1Q_0 + UPDOWN'Q_3Q_0' + UP'DOWNQ_2Q_1'Q_0' + DOWNQ_3Q_0' + UPQ_3'Q_0'$
- $D_0 = UPDOWN + UP'DOWN'Q_0 + DOWNQ_1Q_0 + DOWNQ_2Q_0' + DOWNQ_3Q_0' + UPQ_3'Q_0'$

Na Figura 2, temos uma visão geral do circuito correspondente para as equações da função de próximo estado.

Figura 2 – Circuito para função de próximo estado



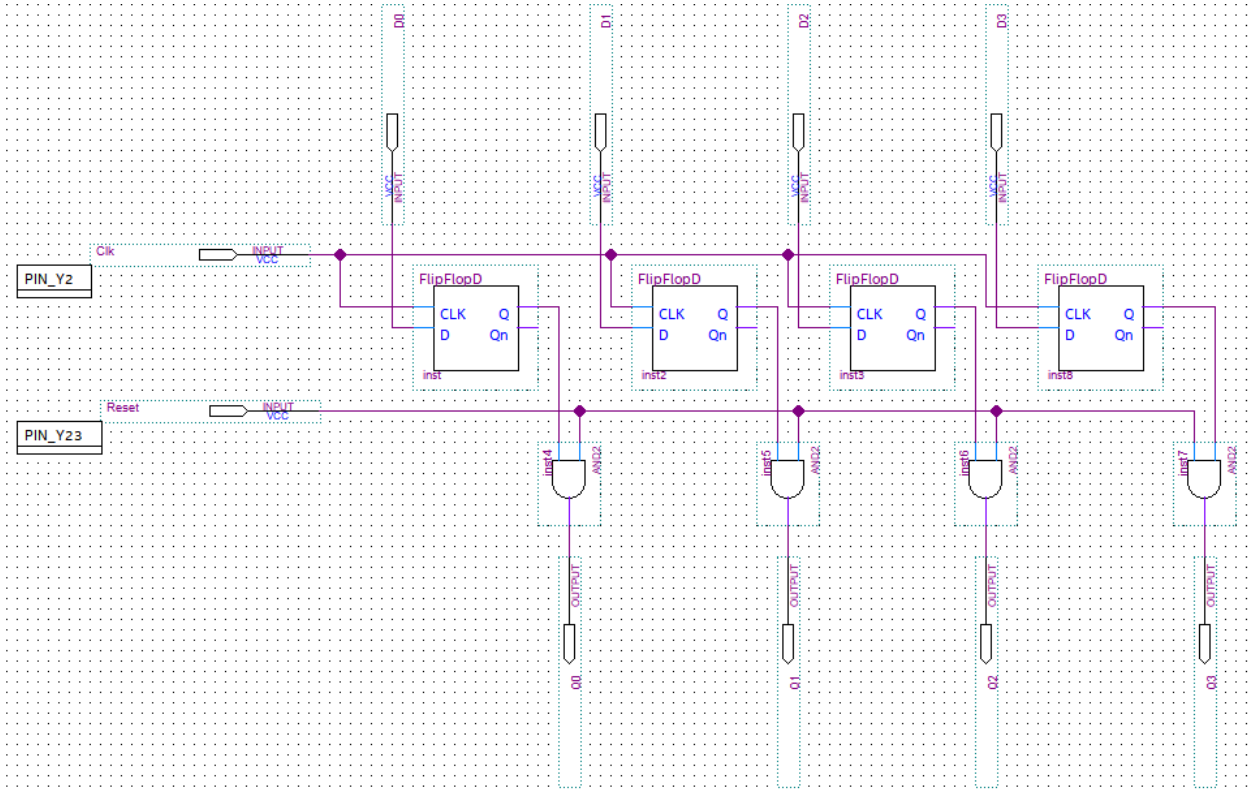
Fonte: Autor

4.3 Registradores

Uma outra parte importante de uma máquina de estados, são os registradores, que têm como principal função armazenar o estado atual, podendo esta função ser implementada atráves de *flip-flops*. Além disso, essa função pode conter um botão de *reset*, reinicializando a máquina. Para esta implementação, foram utilizados *flip-flops* do tipo *D*. Já para o *reset*, foi usado uma nova entrada, combinando-a com a saída *Q* do *flip-flop* em uma porta lógica *and*.

Uma visão geral do circuito desenvolvido pode ser conferida na Figura 3.

Figura 3 – Circuito para os Registradores.



Fonte: Autor

4.4 Função Saída

Em uma Máquina de *Moore*, as saídas dependem apenas do estado atual. Sendo assim, temos, na Tabela 4, o mapeamento dos estados e sua saída correspondente, que será enviada para o display de 7 segmentos do Kit FPGA.

Tabela 4 – Tabela da saída, dado um estado.

Estado Atual				Saída			
$Q3$	$Q2$	$Q1$	$Q0$	$Y3$	$Y2$	$Y1$	$Y0$
0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1
0	1	1	1	0	0	1	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	1	1

Fonte: O Autor

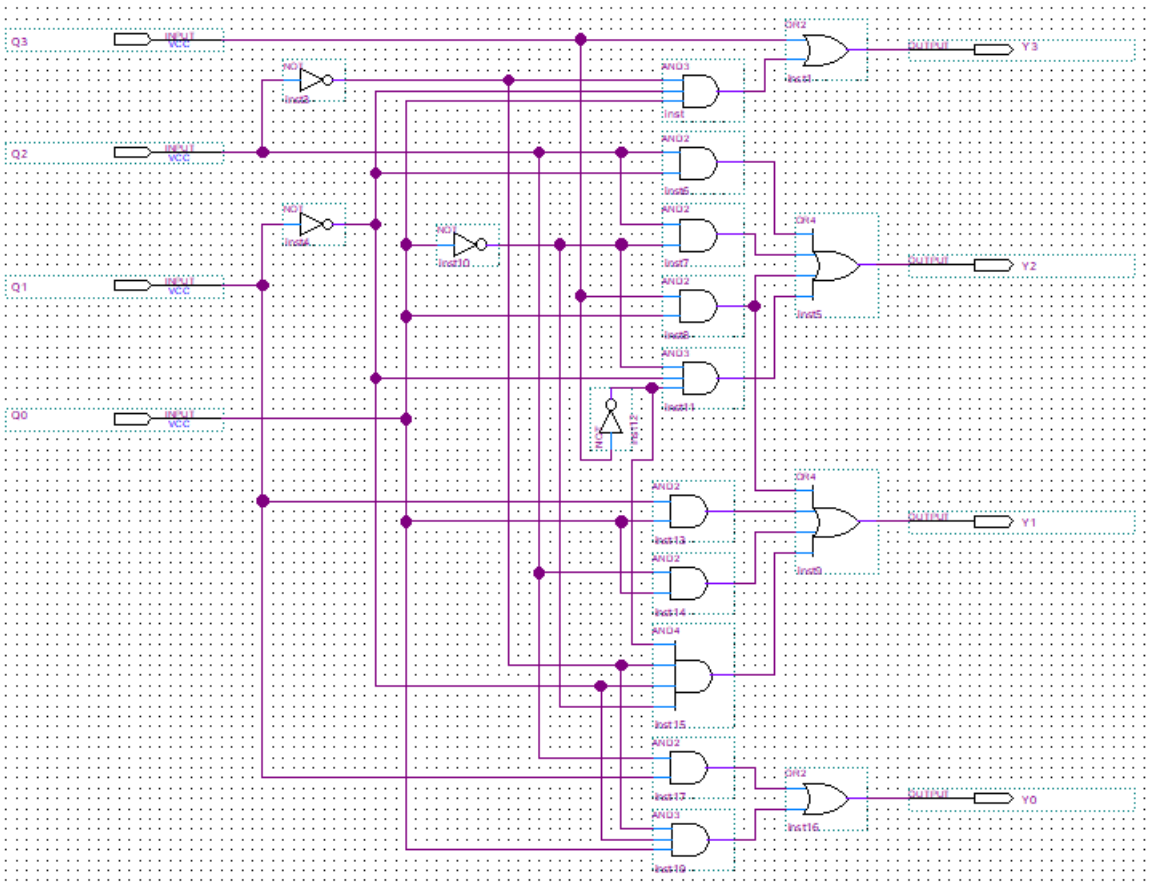
Cada estado é decodificado para o valor que ele de fato representa. As variáveis Q_3, Q_2, Q_1 e Q_0 correspondem à representação do estado atual da máquina, e as variáveis Y_3, Y_2, Y_1 e Y_0, ao número (em binário) representado por cada estado. Logo, Y_3, Y_2, Y_1 e Y_0 serão as entradas enviadas ao display de 7 segmentos do Kit FPGA.

Dessa forma, é possível, então, extrair as expressões e montar o circuito correspondente. Asim sendo, fazendo a extração dos mintermos e simplificando com Mapa de *Karnaugh*, temos abaixo as seguintes expressões em soma de produtos:

- $Y_3 = Q_3 + Q_2{}'Q_1{}'Q_0$
- $Y_2 = Q_2Q_1{}' + Q_2Q_0{}' + Q_3Q_0 + Q_3{}'Q_1{}'Q_0{}'$
- $Y_1 = Q_1Q_0 + Q_2Q_0 + Q_3Q_0 + Q_3{}'Q_2{}'Q_1{}'Q_0{}'$
- $Y_0 = Q_2Q_1 + Q_2{}'Q_1{}'Q_0$

Na Figura 4, temos o circuito correspondente, desenvolvido para as equações obtidas da função de saída.

Figura 4 – Circuito para função de saída

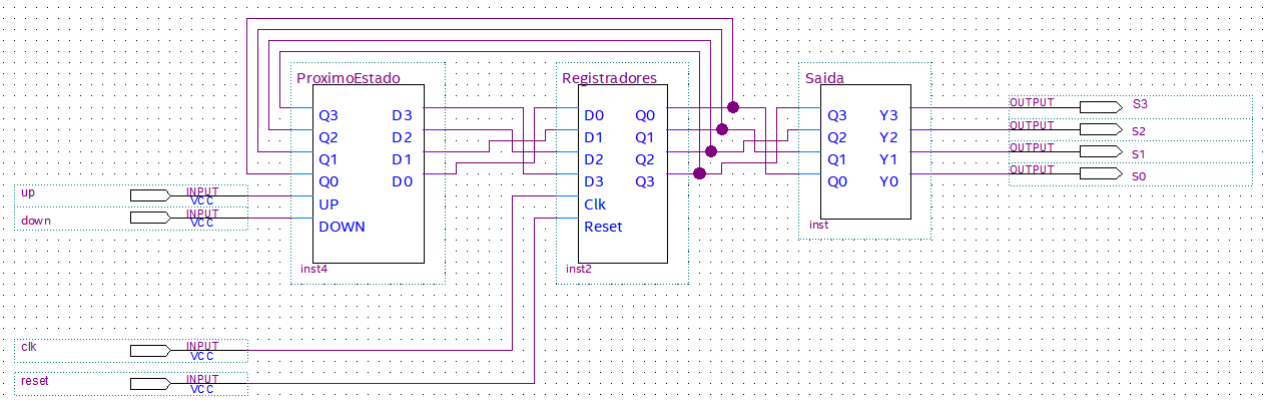


Fonte: Autor

4.5 Junção dos Componentes da Máquina de Estados

Feitas as 3 principais funções da máquina, as *black-boxes* desses circuitos foram geradas e agrupadas em um único circuito, formando a máquina de estados, como mostrado na Figura 5.

Figura 5 – Máquina de Estados



Fonte: Autor

4.6 Temporizador

Uma outra parte importante a ser desenvolvida é o circuito temporizador, pois o Kit FPGA trabalha com uma frequência de 50MHz, enquanto o objetivo deste projeto é obter transições de estados a cada $1s$. Uma estratégia possível é a divisão de frequências, que pode ser implementada atráves de *flip-flops*, ou mesmo por contadores, dividindo a sequência a cada uso. No caso dos *flip-flops*, a frequência é sempre dividida por 2.

Sabemos que $f = 1/T$ e $T = 1/f$, em que f é a frequência e T o período. Queremos $T = 1s$, e temos $f = 50\text{MHz}$. Logo:

$T = 2^N/50 * 10^6$, para $T = 1 \Rightarrow 1 = 2^N/50 * 10^6 \Rightarrow 2^N = 50 * 10^6$

Aplicando log_2 ao dois lados da equação, temos:

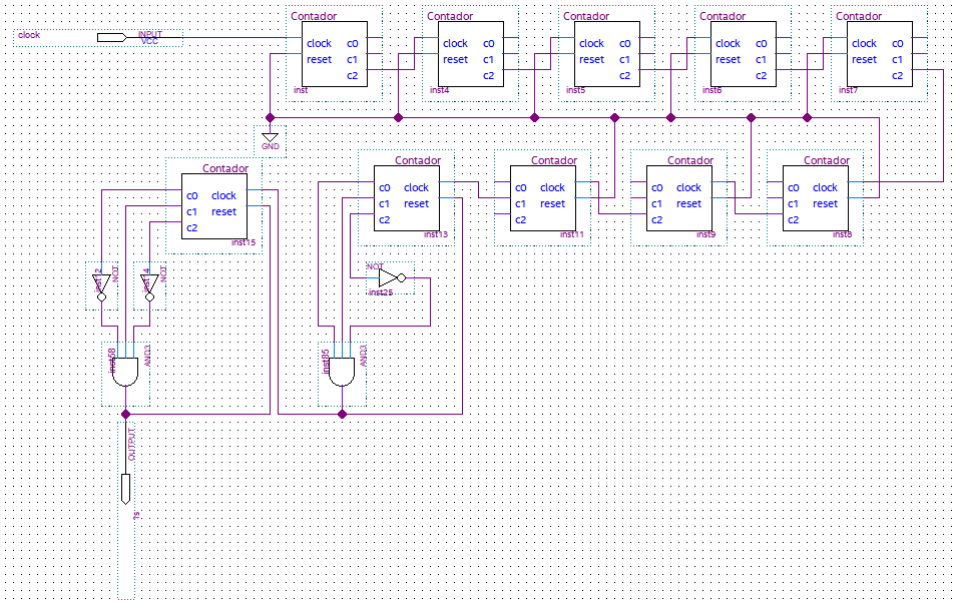
$log_2 2^N = log_2 50 * 10^6 \Rightarrow N = 25.57542$

Assim, para $T = 1s \Rightarrow N \approx 26$.

Sendo assim, serão necessários aproximadamente 26 *flip-flops* para que a frequência de 50MHz seja dividida, até alcançarmos $1s$. No caso, foram usados 10 contadores de 3 bits, pois cada contador utiliza 3 *flip-flops*, e no caso do 8º *flip-flop* foi obtida a saída do *bit* 23.

Assim, temos $\approx 0.16s$, e portanto, as 3 saídas entram no 9º contador, para que essa frequência seja multiplicada por 3, conseguindo $\approx$ 0.5s. Agora, essa nova saída e ligado em um novo contador, permitindo que essa frequência seja multiplicada por 2, chegando mais próximo de $1s$. O circuito obtido pode ser visualizado na Figura 6.

Figura 6 – Temporizador



Fonte: Autor

4.7 Display de 7 Segmentos

Dado o circuito de saída, temos agora a entrada que pode ser mapeada para o display de 7 segmentos do Kit FPGA. Sendo assim, essa entrada deve ser convertida para os segmentos do display, que devem estar ligados ou desligados, conforme a Tabela 5.

Tabela 5 – Tabela de Saída, dado um estado.

Entradas					Saídas						
W	X	Y	Z	Decimal	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1
1	1	1	1	*Blank*	0	0	0	0	0	0	0

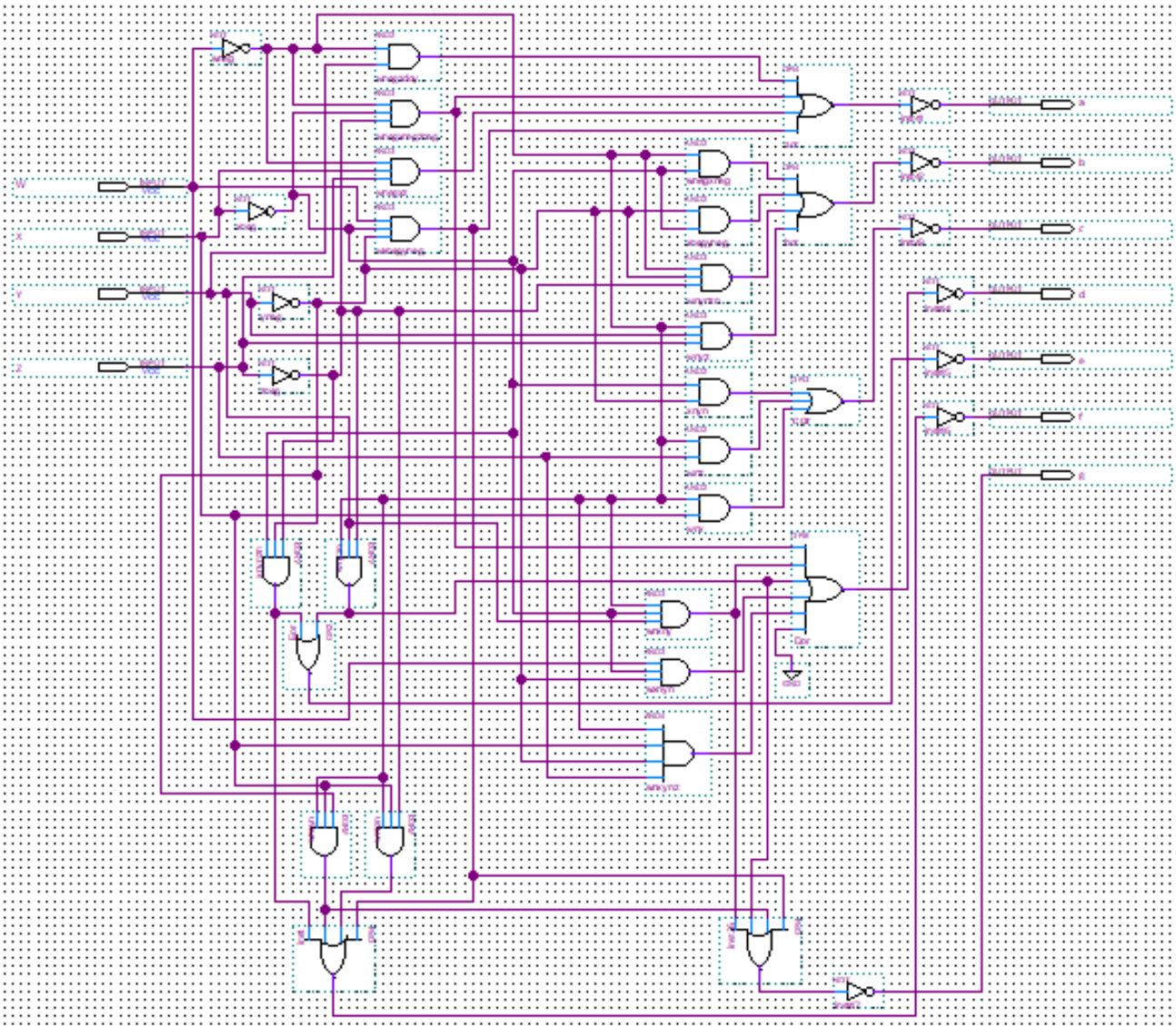
Fonte: Autor

Nesse caso, as entradas W, X, Y e Z representam as variáveis de entradas (isto é, o número em binário que deve ser mostrado no display) e as variáveis a, b, c, d, e, f, g representam cada um dos 7 segmentos do display. Extraindo as expressões da Tabela 5, temos abaixo, as expressões para geração do circuito:

- $a = W'Y + W'X'Z' + W'XZ + WX'Y'$
- $b = A'B' + X'Y' + W'Y'Z' + W'YZ$
- $c = X'Y' + W'Z + W'X$
- $d = W'X'Z' + W'X'Y + W'YZ' + WX'Y' + W'XY'Z$
- $e = X'Y'Z' + W'YZ'$
- $f = W'Y'Z' + W'XY' + W'XZ' + WX'Y'$
- $g = W'X'Y + W'YZ' + W'XY' + WX'Y'$

Então, montando os circuitos correspondentes para as equações anteriores, temos o circuito desenvolvido na Figura 7. Vale ressaltar que o display contido no Kit FPGA é do tipo ânodo comum (ou seja, é ativo em '0'). Portanto, há inversores antes da saída de cada segmento.

Figura 7 – Circuito que mapeia para o display de 7 segmentos

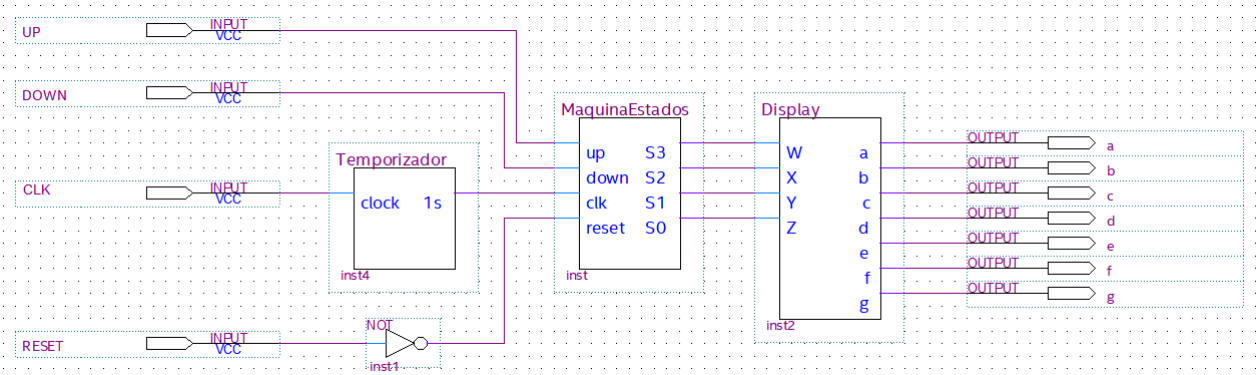


Fonte: Autor

5 Resultados Obtidos e Discussões

Finalizada a implementação da máquina de estados, o circuito temporizador, o decodificador para o display de 7 segmentos e feita a junção em um único circuito, temos então o circuito final, que pode ser visualizado na Figura 8.

Figura 8 – Circuito final desenvolvido.



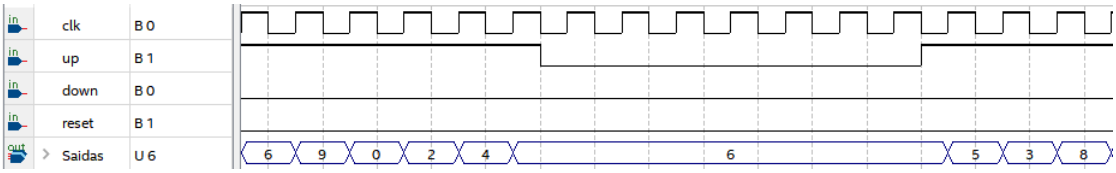
Fonte: Autor

Para validar o circuito desenvolvido, foram feitas simulações atráves de *waveforms* no *software Quartus*. Além disso, também foi feita uma simulação e apresentação do pleno funcionamento do projeto no Kit FPGA. Estas simulações podem ser conferidas a seguir.

5.1 Mantém - $UP = 0$ e $DOWN = 0$

Com $UP = 0$ e $DOWN = 0$, é esperado que a sequência pare e mantenha seu resultado, como mostra a simulação da Figura 9, onde a sequência seguia de forma crescente, até que mudou para o estado mantém, onde ficou parada no estado F = 6, até que a entrada mudasse.

Figura 9 – Simulação contagem mantém.

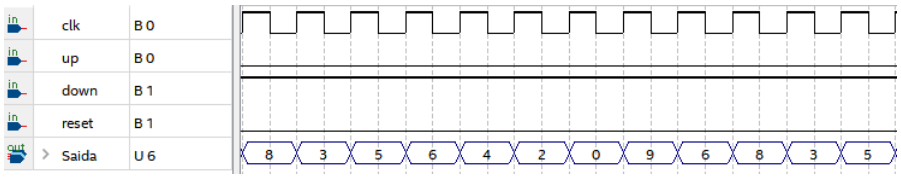


Fonte: Autor

5.2 Decrescente - $UP = 0$ e $DOWN = 1$

Para $UP = 0$ e $DOWN = 1$, a sequência apresentada será da forma decrescente. Sendo assim, a sequência 8-3-5-6-4-2-0-9-6 deve ser apresentada, como pode ser comprovado pela simulação da Figura 10.

Figura 10 – Simulação contagem decrescente.

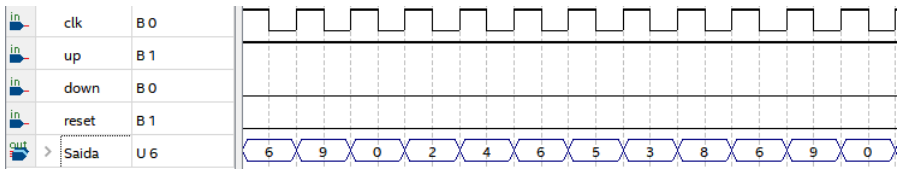


Fonte: Autor

5.3 Crescente - $UP = 1$ e $DOWN = 0$

Para $UP = 1$ e $DOWN = 0$, a contagem será da forma crescente. Sendo assim, a sequência 6-9-0-2-4-6-5-3-8 deve ser apresentada, como pode ser visualizado na simulação da Figura 11.

Figura 11 – Simulação contagem crescente.

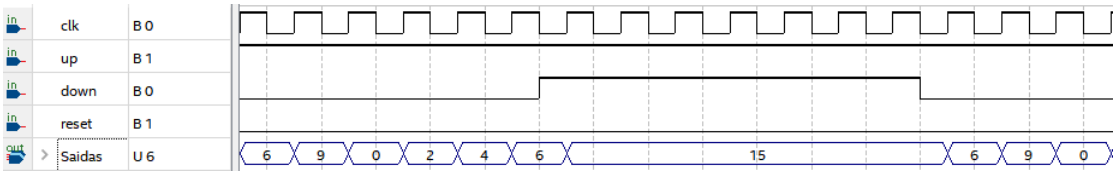


Fonte: Autor

5.4 *Blank* - $UP = 1$ e $DOWN = 1$

Já para $UP = 1$ e $DOWN = 1$, é esperado que a máquina avance para um estado de *blank*, onde todos os LEDs estarão apagados. Isso pode ser implementado atráves de um estado inválido para o display, como o número 15, representado na simulação da Figura 12. Nesse exemplo, a sequência seguia na forma crescente, até que a entrada mudou para *blank* e a sequência ficou nesse estado até que a entrada mudasse, e então a máquina volta para o primeiro estado e a contagem segue de forma crescente.

Figura 12 – Simulação contagem *blank*.

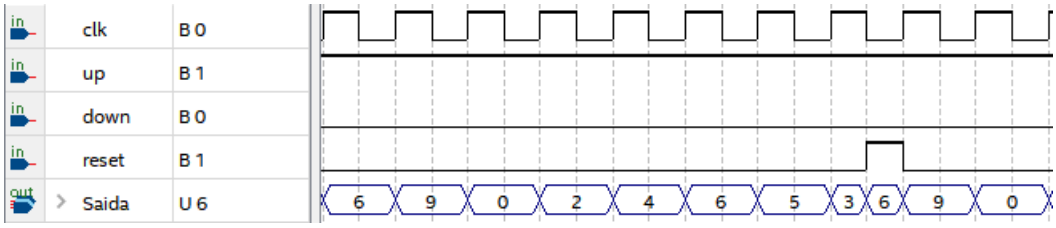


Fonte: Autor

5.5 Uso do *reset*

Ainda, há uma opção de *reset*, onde não importando qual seja o estado atual da máquina, ela deve voltar ao estado inicial (A = 6), e então, após desligado o *reset*, a contagem deve prosseguir. Essa simulação pode ser visualizada na Figura 13, onde a contagem seguia a ordem crescente e um *reset* foi acionado, forçando a máquina a voltar ao estado inicial e prosseguindo a contagem crescente, como o esperado.

Figura 13 – Simulação contagem *reset*.



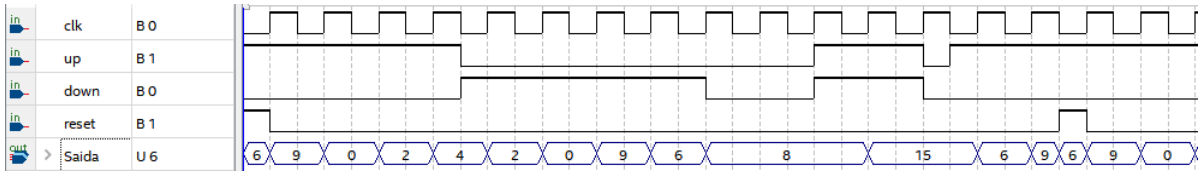
Fonte: Autor

5.6 Contagem Genérica

Foi simulado, também, um caso mais genérico, com uma mistura de combinações das entradas em uma única simulação, como mostrado na Figura 14.

Pode-se notar que a contagem se mantém crescente, com $UP = 1$ e $DOWN = 0$, até que há uma troca, para $UP = 0$ e $DOWN = 1$, e a contagem segue de modo decrescente. Após isso, a entrada $DOWN$ é trocada para $DOWN = 0$, e a contagem se mantém no estado atual. Então, há uma nova troca, e agora $UP = 1$ e $DOWN = 1$, e agora, a máquina vai para o estado *blank*. Por último, temos $UP = 1$ e $DOWN = 0$, onde a contagem segue crescente, até que o *reset* é acionado, e então, a contagem é reiniciada do primeiro estado, conforme esperado.

Figura 14 – Simulação contagem genérica

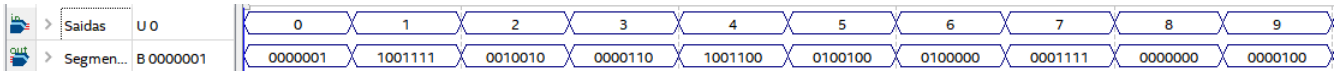


Fonte: Autor

5.7 Display de 7 Segmentos

Para este bloco, foram testadas os números possíveis (0-9), como pode ser visto pela *waveforms* da Figura 15. Como o esperado, o circuito opera corretamente. Por exemplo, o número 1 tem todos os segmentos ligados (em 0) exceto pelo último segmento. Já o número 8 tem todos segmentos ligados.

Figura 15 – Simulação display de 7 segmentos

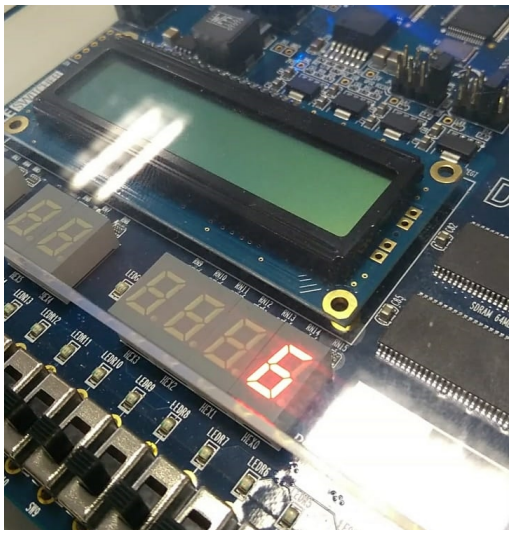


Fonte: Autor

5.8 Simulação no Kit FPGA

As simulações também foram feitas no Kit FPGA disponível no laboratório da disciplina e apresentado ao professor. Na Figura 16, temos uma imagem da simulação realizada, mostrando apenas o estado inicial da máquina, com valor 6.

Figura 16 – Simulação no Kit FPGA



Fonte: Autor

6 Considerações Finais

Com o desenvolvimento deste projeto, foi possível unir todo o conhecimento adquirido na disciplina de Circuitos Digitais e os do Laboratório de Sistemas Computacionais: Circuitos Digitais.

Então, após a modelagem da máquina, as simulações e os resultados obtidos, pode-se concluir o sucesso em sua implementação, pois a máquina possui todas as suas funcionalidades sendo executadas de acordo com o esperado.

Sendo assim, pode-se concluir a eficiência de uma máquina de estados para o problma apresentado e também suas diversas aplicações. Assim como, a de de um Kit FPGA para essas simulações e ver sua aplicabilidade em outros sistemas.

No entanto, o desenvolvimento de uma máquina de estados é uma tarefa longa e requer muita atenção, pois há a existência de várias partes e geram circuitos complexos e longos.

Além disso, com o desenvolvimento deste projeto, foi possível ver mais na prática uma aplicação de circuitos digitais, ter novas noções de *hardware* e a integração *software* e *hardware*.

Referências

1 FLOYD, T. *Sistemas digitais: fundamentos e aplicações*. [S.l.]: Bookman Editora, 2009. Citado 3 vezes nas páginas 7, 11 e 12.

2 GILL, A. et al. Introduction to the theory of finite-state machines. McGraw-Hill, 1962. Citado 2 vezes nas páginas 7 e 12.

3 TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. *Sistemas digitais: princípios e aplicações*. [S.l.]: Prentice Hall, 2003. v. 8. Citado na página 11.

4 FPGA.Altera DE2-115 Development and Education Board. <https://www.ee.ryerson.ca/~courses/coe608/labs/DE2_115_User_Manual.pdf>. Acessado em 22/10/2018. Citado na página 12.

5 QUARTUS PRIME. *Intel Quartus Prime Software Suite*. <https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html>. Acessado em 23/10/2018. Citado na página 12.

6 LUCID CHART. *Software para a criação de de fluxogramas e diagramas on-line*. <https://www.lucidchart.com>. Acessado em 20/10/2018. Citado na página 13.

7 32X8. *Logic circuit simplification (SOP and POS)*. <http://www.32x8.com/>. Acessado em 15/10/2018. Citado na página 16.