ID: .....

Implementação de um contador numérico para um FPGA utilizando máquina de Mealy

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

Docente: Prof. Dr. Tiago de Oliveira

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência e Tecnologia – Campus São José dos Campos

São José dos Campos

Outubro de 2017

Resumo

O trabalho desenvolvido visou apresentar uma aplicação de máquina de estados finitos (utilizando máquina de Mealy), ao implementar um contador numérico para um FPGA (*Field Programmable Gate Array*), cuja sequência de contagem é a não usual (numeração ordenada) e possui quatro finalidades, sendo elas contar crescentemente, decrescentemente, manter o valor atual e apagar o display (*blank*). Para isso, foram implementados um temporizador de 2 segundos, pois a frequência da placa programável era de 50MHz e um decodificador para um display de 7 segmentos para visualização do resultado. Não houveram problemas referentes aos circuitos combinacionais e sequenciais implementados no *Quartus Prime* para compor os blocos do sistema digital em questão (Contador numérico), mas a abordagem da máquina de Mealy apresentou algumas características não desejadas nos resultados, como por exemplo a funcionalidade do mantêm não manter, de fato, o valor da saída e a escolha do estado inicial não gerar o início da contagem esperado, mostrando assim que seria necessário levar em consideração tais fatores para um projeto próximo do esperado.

Palavras-chave: FPGA, Máquina de Mealy, Contador numérico.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução

O que há em comum entre *laptops*, telefones, *smartphones*, semáforos, micro-ondas, console de jogos eletrônicos e até mesmo aparelhos karaokê? No ponto de vista funcional (o que é proporcionado ao usuário) não há (ou há pouca) características em comum, mas a caráter de implementação (o que faz ele fazer o que faz) existe, em outras palavras, todos esses aparelhos são sistemas digitais.

Segundo Tocci et. al. (2011), um sistema digital é uma combinação de dispositivos projetados para manipular informação lógica ou quantidades físicas representadas no formato digital, ou seja, as quantidades podem assumir apenas valores discretos. Esses sistemas são compostos pelo que chamamos de circuitos digitais (ou lógicos), nos quais são projetados para produzir tensões de saída que se encontram dentro das faixas de tensões determinadas para os níveis 0 e 1.

Esses aparelhos e muitos outros, que estão presentes no nosso cotidiano, possuem como objetivo principal facilitar as nossas atividades, tais como rotinas de escritório, comunicação, preparo de alimentos, dentre outros. Portando, o estudo e implementação de circuitos digitais contribui significativamente para compreensão do princípio de funcionamento desses aparelhos.

Dentre os tópicos principais no estudo de circuitos digitais tem-se a máquina de estados finitos (ou simplesmente máquina de estados), pois em muitas aplicações é necessário aguardar uma entrada, como ligar o micro-ondas, para que seja dado a respectiva resposta.

Nesse trabalho, visando apresentar uma aplicação de máquina de estados, será desenvolvido um projeto de circuitos digitais utilizando máquina de Mealy para implementação de um contador numérico cuja ordem de contagem seja a não usual 2-1-8-3-7-0-4-3-6 para utilização em um FPGA.

2 Objetivos

2.1 Geral

Desenvolver um contador numérico utilizando máquina de estados finitos de Mealy cuja sequência seja 2-1-8-3-7-0-4-3-6 com as funcionalidades de desligar, manter o valor e contar de forma crescente ou decrescente.

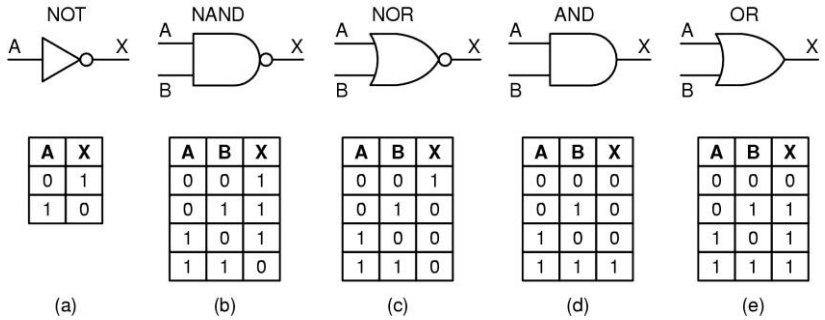
2.2 Específicos

- Implementar o circuito combinacional da saída para máquina de Mealy utilizando o software ***Quartus Prime***;
- Implementar o circuito combinacional do próximo estado para máquina de Mealy utilizando o software ***Quartus Prime***;
- Montar a máquina de Mealy para o contador numérico desejado;
- Implementar um temporizador de 2 segundos no ***Quartus Prime***;
- Implementar um circuito decodificador para um display de 7 segmentos no ***Quartus Prime***;
- Criar blocos para os circuitos do decodificador (do display de 7 segmentos), do temporizador de 2 segundos e da Máquina de Mealy do contador numérico e relaciona-los (integra-los) adequadamente;
- Avaliar *waveform* do decodificador para o display de 7 segmentos e para o contador que implementa o temporizador de 2 segundos;
- Avaliar *waveform* do projeto implementado no Quartus Prime:

3 Fundamentação Teórica

3.1 Princípios de Circuitos Digitais

Para que possamos projetar circuitos digitais é necessário utilizar o que denominamos de portas lógicas, sendo as primitivas *NOT, AND* e *OR*. Apenas com essas portas podemos implementar desde circuitos simples à sistemas complexos, como um computador, por exemplo. A figura 1 apresenta as tabelas-verdade de algumas portas e seus respectivos símbolos.



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Figura 1 Tabelas-verdade e seus respectivos símbolos das portas lógicas. Nas tabelas-verdade as letras A e B representam as entradas da porta lógica e a letra X a saída dela. Fonte: <http://www.dpi.inpe.br/~carlos/Academicos/Cursos/ArqComp/aula_5bn1.html>.

Em síntese, um projeto de circuitos digitais envolve as portas lógicas, a modelagem do problema (através da tabela-verdade) e álgebra booleana. Uma técnica com finalidade de simplificação do problema, bem como síntese do circuito é a que denominamos de Mapas de Karnaugh.

Um circuito lógico pode ser combinacional, quando a saída do circuito é dependente apenas e exclusivamente das variáveis de entrada, ou sequencial, quando a saída depende de um valor que está armazenado.

Segundo a PCS-EPUSP (2012), um componente essencial em circuitos sequenciais são os circuitos biestáveis, que são nada mais que circuitos que apresentam dois estados estáveis, ou seja, podem se manter até a mudança de determinada condição. Os mais comuns são os denominados *flip flop's* cuja característica é armazenar ou alterar bits na transição do *clock*, um exemplo é o *flip flop D* (figura 2).

3.2 Máquina de Estados Finitos de Mealy

Segundo Gomes et. al. (2014), um *flip flop* pode ser utilizado para criar dispositivos mais robustos com capacidades de guardar informações, dividir frequências, entre outras aplicações, tais dispositivos são denominados máquina de estados finitos ou máquina de estados, uma forma de implementar uma pode ser com a máquina de Mealy.

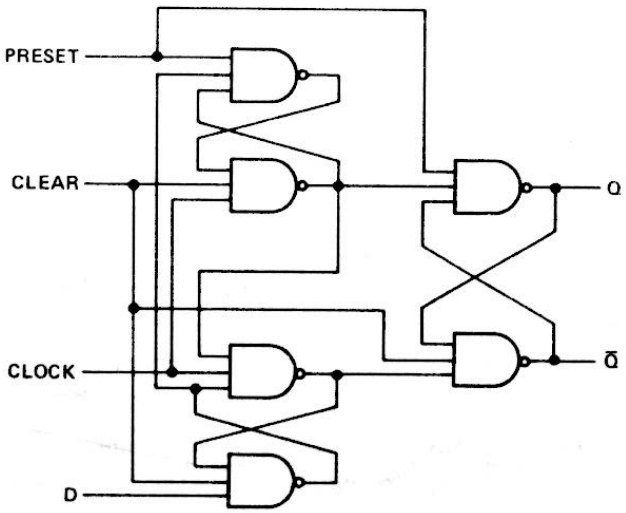


Figura 2 Circuito biestável do tipo flip flop D. A entrada Preset armazena 1, a Clear 0, a D o dado. Fonte: <http://lovqvist.net/DTL/7474.html>.

Em uma máquina de Mealy, a saída depende do valor da entrada e é expresso pelo seguinte diagrama (figura 3).

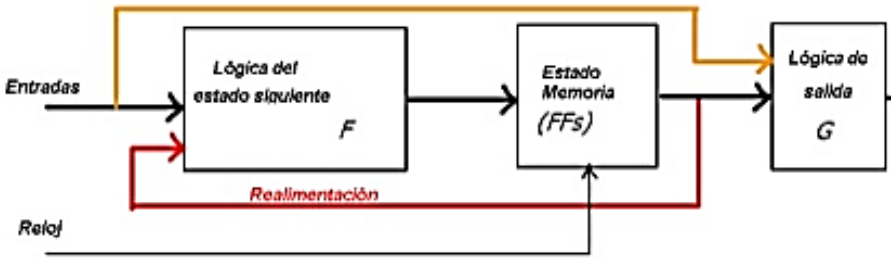


Figura 3 Esquema de uma máquina de estados de Mealy. Tanto o circuito combinacional do próximo estado (Lógica del estado siguiente) quanto da saída (Lógica de salide) utilizam a entrada para apresentação do resultado. O estado atual é então armazenado no banco de registradores (Estado Memória) na transição do clock (Reloj). Fonte: <https://es.slideshare.net/faurbano/mquinas-de-estado>.

Nesse caso, o banco de registradores pode ser implementado utilizando um conjunto de **flip flop's D** para armazenar o estado atual.

3.3 Display de 7 Segmentos

Um display de 7 segmentos é um dispositivo para representação visual dos números binário (o que o computador trabalha) em decimais (que é a usual e facilmente compreendida pelos humanos). É importante ressaltar que os segmentos em questão são nada mais que *LED's*.

Para conversão de números binários em um conjunto de saídas que consiga representar os decimais, utiliza-se o que denominamos de circuito decodificador e ele é implementado utilizando como base a seguinte representação (figura 4).

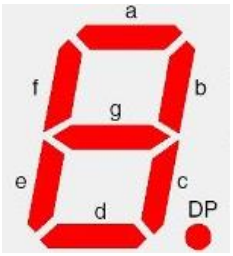


Figura 4 Ilustração de um display de 7 segmentos (nomeados com as letras de a-g). Fonte: <https://www.electronica-pt.com/eletronica-digital/display-7-segmentos>.

O princípio do decodificador em questão (figura 5), é nada mais que indicar os segmentos que devem estar ligados para representação visual dos números binários.

	a	b	c	d	e	f	g
0	0	1	1	0	0	0	0
1	1	1	0	1	1	0	1
2	1	1	1	1	0	0	1
3	0	1	1	0	0	1	1
4	1	0	1	1	0	1	1
5	1	0	1	1	1	1	1
6	1	1	1	0	0	0	0
7	1	1	1	1	1	1	1
8	1	1	1	1	0	1	1
9	1	1	1	1	1	1	0

Figura 5 O princípio para implementação (e para fazer a tabela verdade) do decodificador para o display de 7 segmentos. Fonte: <http://tecnomelque.blogspot.com.br/2011/01/arduino-dimmer-com-display-de-sete.html>.

A tabela verdade desse decodificador, seguindo a ideia da figura 5 é (tabela 1):

Tabela 1 Tabela verdade do decodificador para o display de 7 segmentos. Null indica display apagado.

	Entrada				**Saída (para o segmento)**						
Nº	W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
Null	1	0	1	0	0	0	0	0	0	0	0
Null	1	0	1	1	0	0	0	0	0	0	0
Null	1	1	0	0	0	0	0	0	0	0	0
Null	1	1	0	1	0	0	0	0	0	0	0
Null	1	1	1	0	0	0	0	0	0	0	0
Null	1	1	1	1	0	0	0	0	0	0	0

3.4 Temporizador

Em algumas aplicações, é necessário dividir a frequência do *clock*, uma vez que sua frequência (f) pode estar alta demais, como por exemplo na escala de 10^6. Nesse contexto, se insere o dispositivo que denominamos temporizador, que é exatamente para essa finalidade, pois ele consegue reduzir essa frequência.

Observando os números em binário na tabela 1 nota-se que o bit mais significativo (A) irá mudar apenas no 8, ou seja, em uma máquina de estados que implemente um contador numérico sequencialmente, esse bit irá necessitar de exatamente 8 ciclos de clock para alterar o bit A e é essa a ideia de um temporizador. Partindo disso, concluímos que, se considerarmos como a saída do *clock* o bit A, a frequência então será dividida por 2^4. A relação entre o tempo e a frequência será dada por

Como

$$t(s) = \frac{1}{f(Hertz)} \rightarrow t'(s) = \frac{2^{qb}}{f(Hertz)} \qquad (1)$$

Onde t(s) é o tempo em segundos, f em Hertz é a frequência do clock, qb a quantidade de bits e t'(s) é o tempo que o clock terá após a frequência ser dividida.

A figura 6 apresenta um diagrama de um contador numérico de 3 bits implementado com Máquina de Mealy.

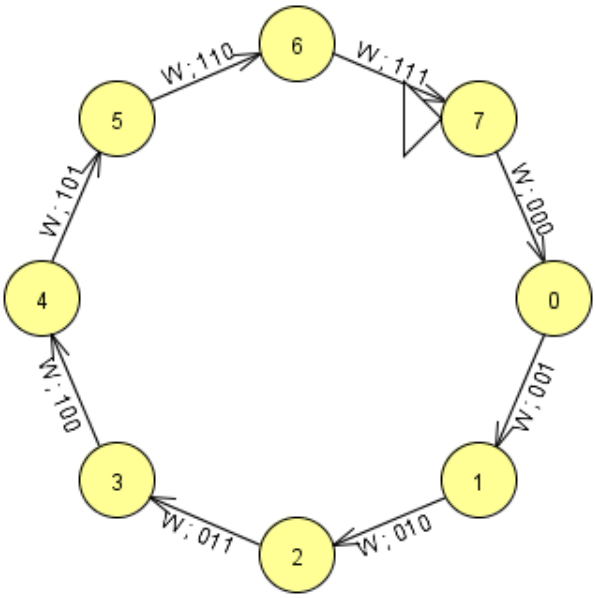


Figura 6 Diagrama de estados feito com o software JFLAP de um contador numérico de 3 bits implementado com Máquina de Mealy. O triângulo indica o estado inicial. Por convenção, temos que, independente da entrada W, o contador será sempre crescente.

4 Desenvolvimento

A tabela 2 apresenta a relação entre as entradas *UP* e *DOWN* e o tipo de contagem ou ação que deve ser realizada.

Tabela 2 Funcionalidades que o contador a ser implementado deverá possuir.

UP (U)	*DOWN (D)*	Contagem
0	0	Mantêm
0	1	Decrescente
1	0	Crescente
1	1	*Blank*

Nesse caso, quando a contagem está em *Blank*, o display de 7 segmentos é apagado. Partindo dessas condições, a figura 7 apresenta o diagrama de estados para contagem numérica que segue o comportamento desejado.

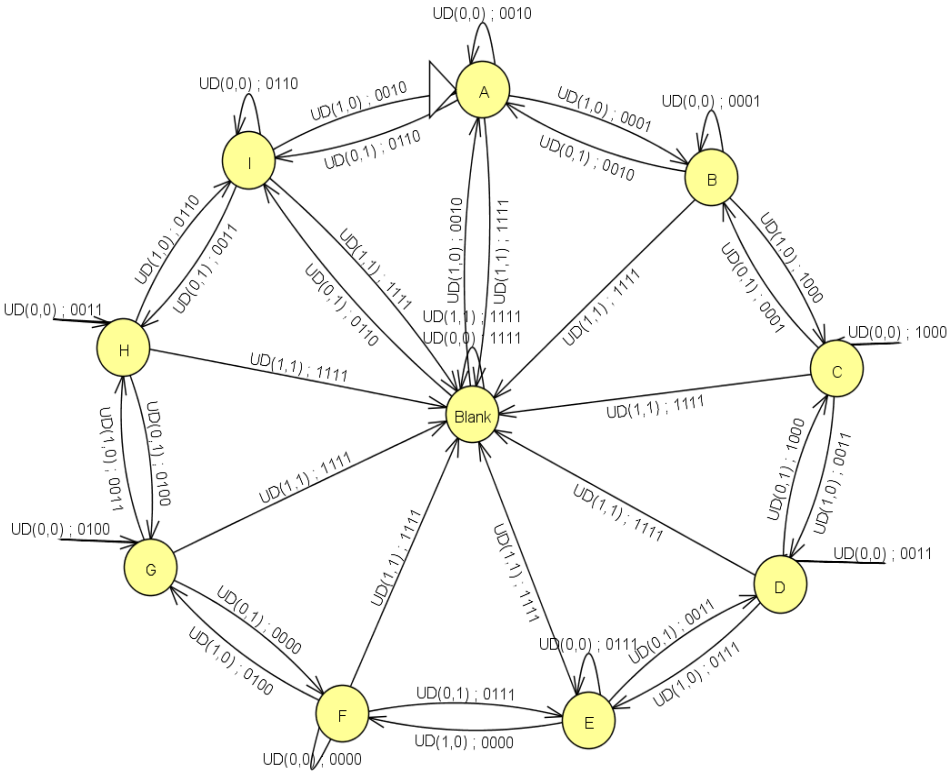


Figura 7 Diagrama de máquina de estados do contador numérico. Perceba que quando o estado é Blank, se a entrada for mudada para crescente o contador começará do A e se for decrescente começará do B. Diagrama elaborado com o software JFLAP

Como pode ser observado, o diagrama apresentado contempla a sequência de contagem 2-1-8-3-7-0-4-3-6. A tabela 3 apresentam as tabelas-verdade correspondente ao diagrama da máquina de estados da figura 7.

Tabela 3 Tabela verdade do próximo estado e da saída da máquina de Mealy. Tanto o botão crescente quanto decrescente irão para o primeiro estado, quando estiver no blank.

ESTADO ATUAL	**PRÓXIMO ESTADO(XYWZ) / SAÍDA(X'Y'W'Z')**			
(x,y,w,z)	**UD=00**	**UD=01**	**UD=11**	**UD=10**
A(0,0,0,0)	A(0000)/0010	I(1000)/0110	N(1111)/1111	B(0001)/0001
B(0,0,0,1)	B(0001)/0001	A(0000)/0010	N(1111)/1111	C(0010)/1000
C(0,0,1,0)	C(0010)/1000	B(0001)/0001	N(1111)/1111	D(0011)/0011
D(0,0,1,1)	D(0011)/0011	C(0010)/1000	N(1111)/1111	E(0100)/0111
E(0,1,0,0)	E(0100)/0111	D(0011)/0011	N(1111)/1111	F(0101)/0000
F(0,1,0,1)	F(0101)/0000	E(0100)/0111	N(1111)/1111	G(0110)/0100
G(0,1,1,0)	G(0110)/0100	F(0101)/0000	N(1111)/1111	H(0111)/0011
H(0,1,1,1)	H(0111)/0011	G(0110)/0100	N(1111)/1111	I(1000)/0110
I(1,0,0,0)	I(1000)/0110	H(0111)/0011	N(1111)/1111	A(0000)/0010
N(1,1,1,1)	N(1111)/1111	A(0000)/0110	N(1111)/1111	A(0000)/0010

É importante ressaltar que foi atribuído o rótulo (1111) ao estado afim de gerar expressões mais simples, já que estamos permitindo a variação de mais bits no Mapa de Karnaugh. Utilizando a ferramenta computacional *Karnaugh Map Minimizer*, para síntese das expressões booleanas, temos que a expressão, para saída, será (expressão 2):

$$\begin{cases} X' = UD + \bar{y}z(\bar{w}U + wD) + \bar{U}\bar{D}(xz + \bar{y}w\bar{z}) \\ Y' = UD + \bar{U}\bar{D}(x + y\bar{z}) + \bar{x}z(wU + yD) + \bar{x}(yzU + \bar{y}\bar{w}\bar{z}D) \\ W' = x + \bar{w}(D + \bar{z}\bar{U}) + w(U + z\bar{D}) \\ Z' = wU(\bar{y} + \bar{z}) + D(y\bar{w} + x\bar{y} + U) + \bar{y}(\bar{x}\bar{z}U + z\bar{U}\bar{D} + w\bar{z}D) + \bar{U}(wz\bar{D} + y\bar{w}\bar{z}) \end{cases} \quad (2)$$

e para o próximo estado temos (expressão 3):

$$\begin{cases} X = UD + x\bar{U}\bar{D} + \bar{x}\bar{y}\bar{w}\bar{z}D + \bar{x}ywzU \\ Y = \bar{D}y(\bar{w} + \bar{U}) + yw\bar{z} + D(x\bar{y} + U) + z(\bar{x}y\bar{U} + \bar{y}wU) \\ W = z(\bar{w}U + \bar{x}w\bar{U}) + w\bar{D}(\bar{z} + \bar{U}) + D(U + x\bar{y} + y\bar{w}\bar{z}) \\ Z = U(D + \bar{x}\bar{z}) + z\bar{U}\bar{D} + D(\bar{z}(w + y) + x\bar{y}) \end{cases} \quad (3)$$

O circuito digital correspondente à expressão 2 é (figura 8):

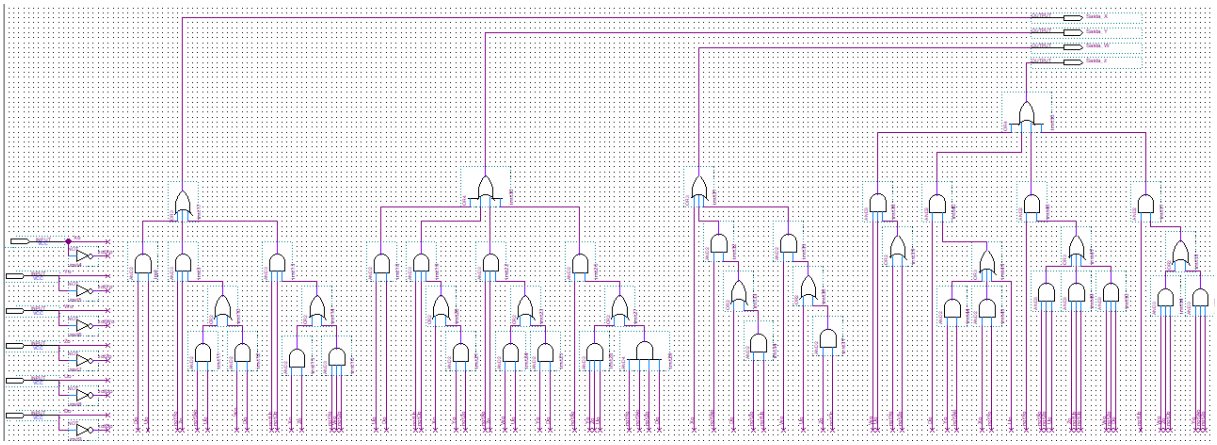


Figura 8 Circuito combinacional da saída implementado no software Quartus Prime.

O circuito digital correspondente à expressão 3 é (figura 9):

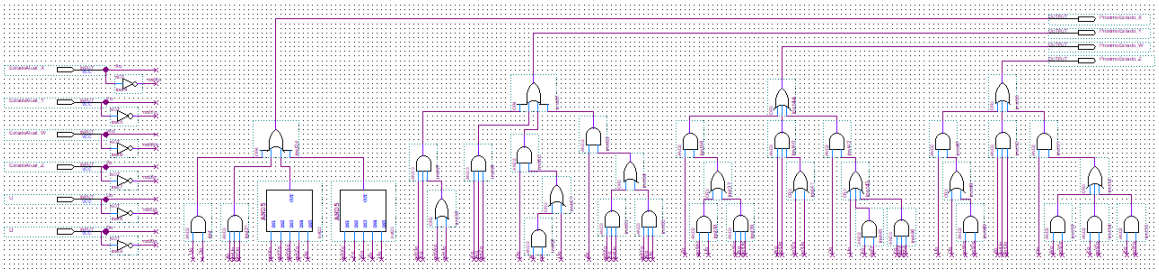


Figura 9 Circuito combinacional do próximo estado implementado no software Quartus Prime.

O banco de registradores implementado para máquina de Mealy é (figura 10):

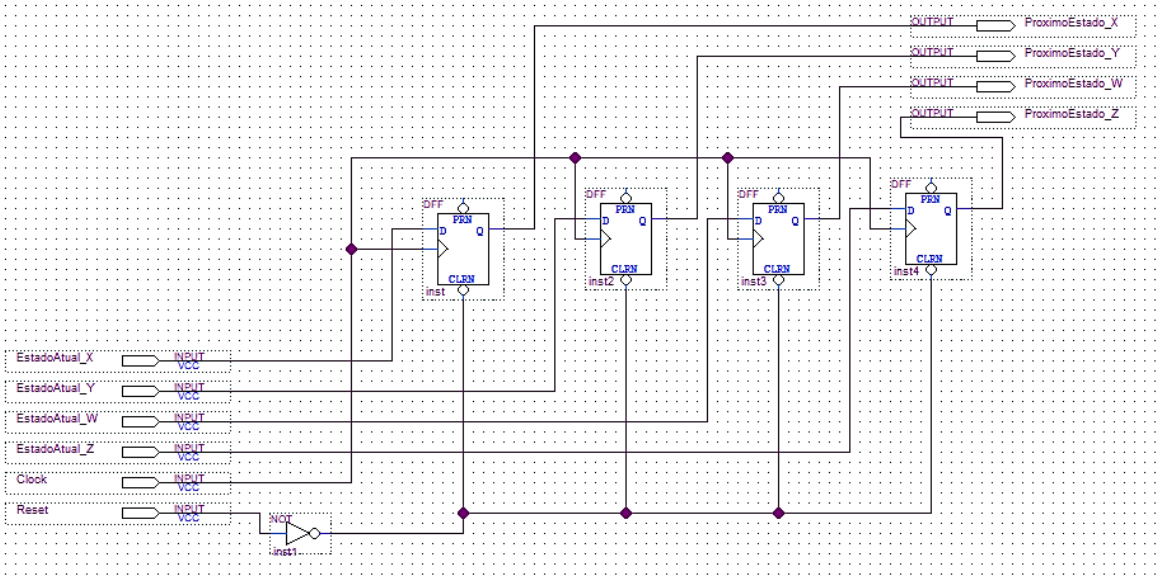


Figura 10 Banco de registradores para Máquina de Mealy implementado no Quartus Prime com flip flop's D. O reset insere 0 no flip flop D quando acionado e independe do clock (reset assíncrono).

Esses circuitos são então convertidos em blocos (circuitos particulares) e tem-se o esquemático final para máquina de Mealy para o contador numérico é (figura 11):

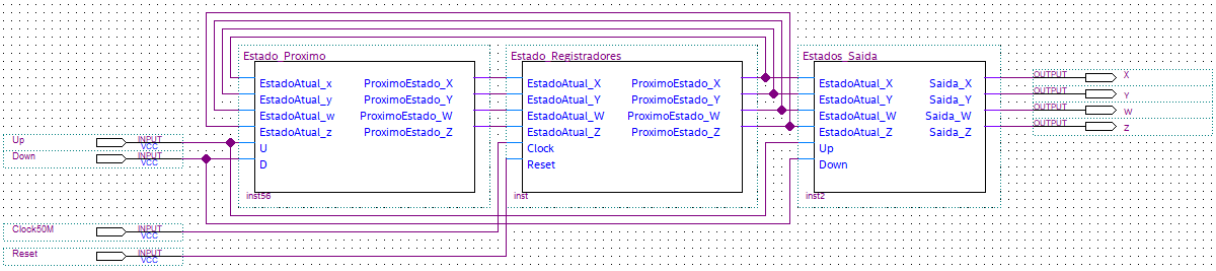


Figura 11 Esquemático da máquina de estados de Mealy implementada no software Quartus Prime. Nesse caso, tem-se três blocos (Estado_Próximo,Estado_Registradores e Estado_Saída).

Com base na tabela 1, e utilizando o software ***Karnaugh Map Minimizer***, tem-se as seguintes expressões booleanas (expressão 4) para o decodificador do display.

$$\begin{cases} a = \bar{X}W + \bar{X}YZ + X\bar{Y}\bar{W} + \bar{Y}\bar{W}\bar{Z} \\ b = \bar{X}\bar{Y} + \bar{Y}\bar{W} + W\bar{X}Z + \bar{W}\bar{X}\bar{Z} \\ c = \bar{X}Y + \bar{X}Z + \bar{Y}\bar{W} \\ d = \bar{X}(\bar{Y}W + \bar{Z}W + \bar{Z}\bar{Y} + \bar{W}YZ) + X\bar{Y}\bar{W} \\ e = \bar{Z}(\bar{Y}\bar{W} + \bar{X}W) \\ f = \bar{X}(\bar{W}\bar{Z} + Y\bar{W} + Y\bar{Z}) + X\bar{Y}\bar{W} \\ g = X\bar{Y}\bar{W} + \bar{X}(Y\bar{W} + \bar{Y}W + Y\bar{Z}) \end{cases} \quad (4)$$

A figura 12 apresenta o circuito decodificador implementado no ***Quartus Prime***.

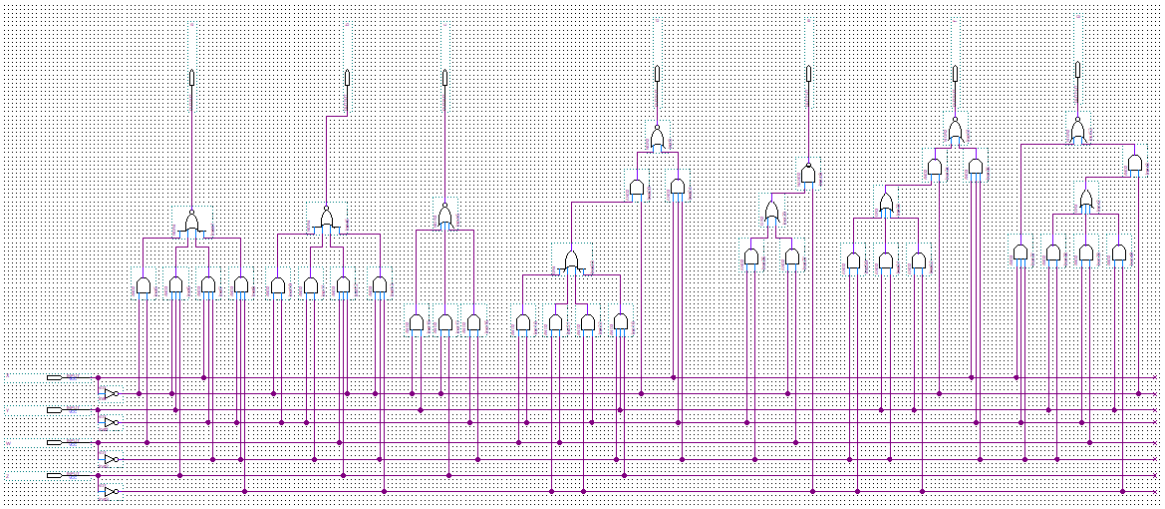


Figura 12 Esquemático do circuito decodificador para o display de 7 segmentos implementado no Quartus Prime.

A tabela verdade referente à máquina de Mealy que implementa um temporizador é apresentada na tabela 4.

Tabela 4 Tabela verdade da Máquina de Mealy para o temporizador.

ESTADO ATUAL	**PRÓXIMO ESTADO (XYZ)/SAÍDA(X',Y',Z')**
(x,y,z)	**W=0 / W=1**
A(1,1,1)	B(0,0,0) / 000
B(0,0,0)	C(0,0,1) / 001
C(0,0,1)	D(0,1,0) / 010
D(0,1,0)	E(0,1,1) / 011
E(0,1,1)	F(1,0,0) / 100
F(1,0,0)	G(1,0,1) / 101
G(1,0,1)	H(1,1,0) / 110
H(1,1,0)	A(1,1,1) / 111

Com o software ***Karnaugh Map Minimizer***, as expressões do próximo estado, para o temporizador, são:

$$\begin{cases} X = \bar{Z} \\ Y = Y \oplus Z \\ Z = X(\bar{Y} + \bar{Z}) + \bar{X}YZ \end{cases} \quad (5)$$

Ressalta-se que o valor da saída será o próprio rótulo do estado (XYW).

A figura 13 apresenta o temporizador implementado no ***Quartus Prime***.

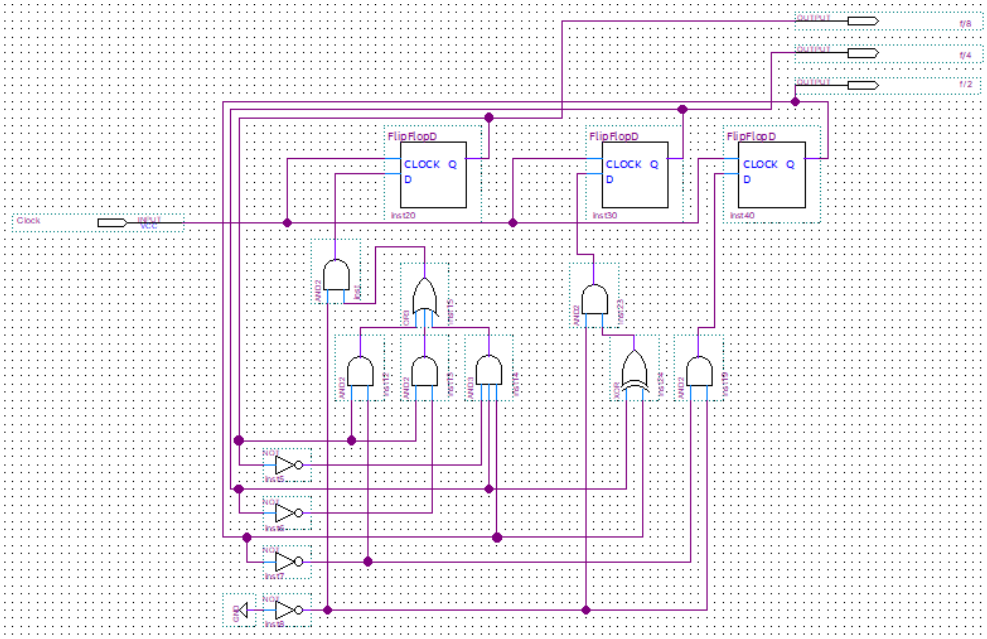


Figura 13 Esquemático do temporizador implementado no Quartus Prime.

O temporizador implementado consegue dividir a frequência em 2^3. Uma vez que desejamos obter o valor mais próximo de 2 segundos e a frequência do *clock* que o FPGA disponibiliza é 50MHz, usando a equação 1, temos que será necessário:

$$2 = \frac{2^{qb}}{50 * 10^6} \rightarrow qb = 1 + \log_2(50 * 10^6) \cong 26{,}57$$

Podemos utilizar 26 bits ou 27 bits para obter o tempo desejado, sendo assim:

$$t_1'(s) = \frac{2^{26}}{50 * 10^6} = 1{,}34\ s$$

$$t_2'(s) = \frac{2^{27}}{50 * 10^6} = 2{,}68\ s$$

Devemos então escolher a aproximação com menor erro relativo ao aproximar os segundos para 2, sendo esse erro definido como

$$Erro\% = \frac{|2 - t'(s)|}{2} \qquad (6)$$

Sendo assim,

$$Erro(26)\% = \frac{|2 - 1{,}34\ |}{2} = 33\%$$

$$Erro(27)\% = \frac{|2 - 2{,}68\ |}{2} * 100 = 34\%$$

Nesse caso, é mais conveniente utilizar 26 bits pelo erro ser menor.

A figura 14 apresenta o esquemático do temporizador de 2 segundos.

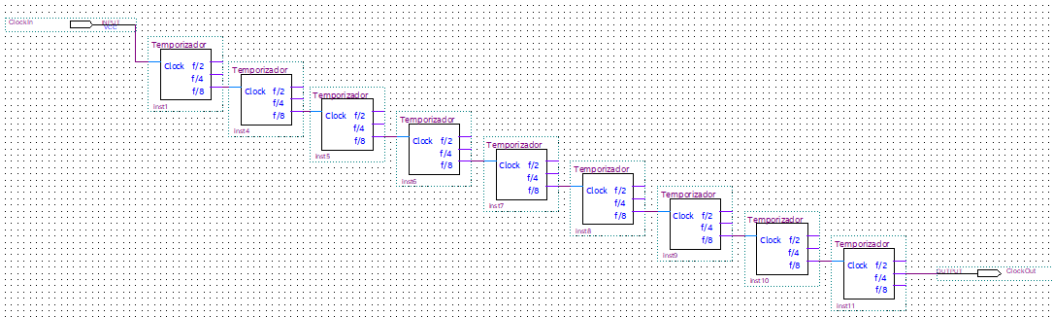


Figura 14 Circuito temporizador de 2 segundos implementado no software Quartus Prime.

Agora que todos os componentes necessários para o nosso sistema digital foram implementados (e transformados em blocos), temos que o circuito final será (Figura 15):

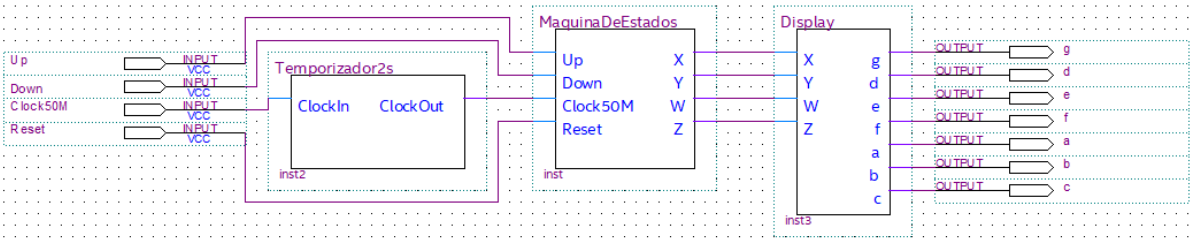


Figura 15 Sistema digital para o contador numérico da sequência 2-1-8-3-7-0-4-3-6. Tem-se três blocos (Temporizador2s, MáquinaDeEstados e Display).

5 Resultados obtidos e discussões

A figura 16 mostra a forma de onda (waveform) do contador que implementa o temporizador. Como é observado, a sua funcionalidade está de acordo com esperado.

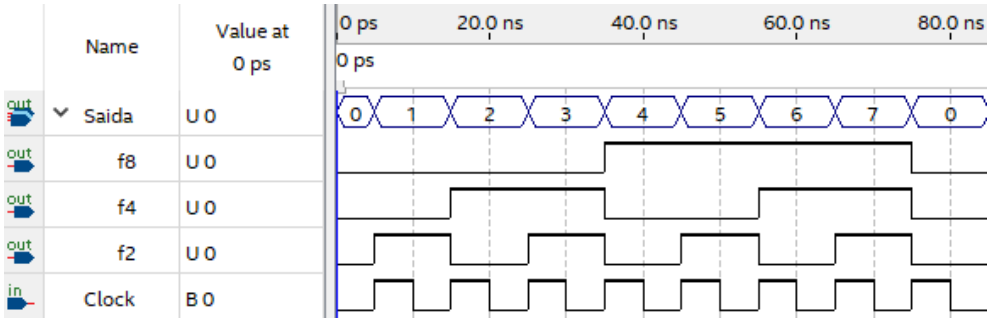


Figura 16 Waveform do contador que implementa o temporizador implementado com Quartus Prime.

Por motivos desconhecidos, não foi possível gerar a forma de onda para o temporizador de 2 segundos. O erro gerado é apresentado na figura 17.

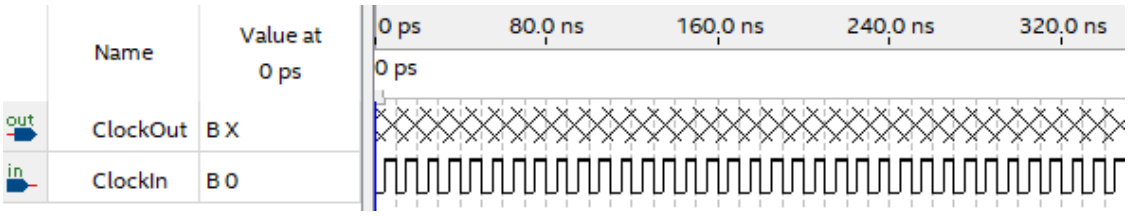


Figura 17 Erro ao gerar a forma de onda para o temporizador de 2 segundos no Quartus Prime.

A figura 18 apresenta a forma de onda do decodificador para o display de 7 segmentos. Como pode ser observado, as respectivas saídas estão condizentes com a tabela 1 e portanto, não houve quaisquer erros na implementação.

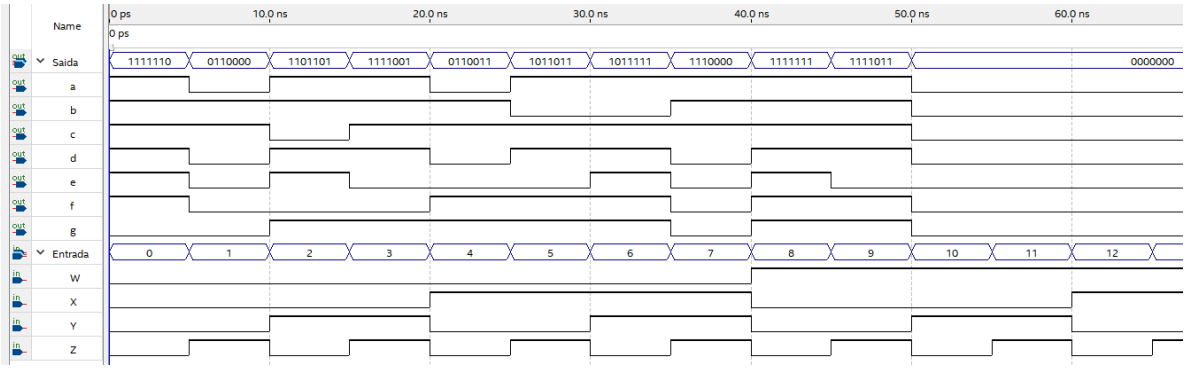


Figura 18 Forma de onda do decodificador para o display de 7 segmentos implementado no Quartus Prime. No FPGA, é necessário inverter as saídas do display para que seja apresentado o resultado corretamente.

A figura 19 mostra a forma de onda (*waveform*) para sequência crescente (*Up =1 e Down = 0*), na qual o reset é acionado em 110 ns.

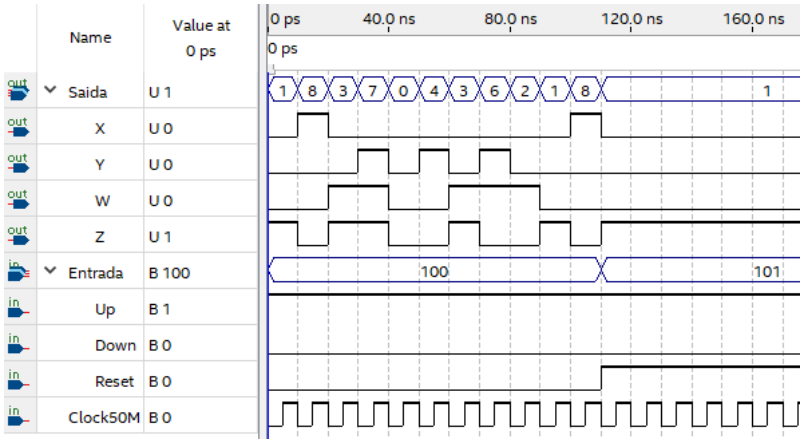


Figura 19 Forma de onda gerado pelo Quartus Prime para Up = 1 e Down = 0 (Contagem crescente).

A figura 20 mostra a forma de onda para sequência decrescente (*Up = 0 e Down = 1*), na qual o reset é acionado em 110 ns.

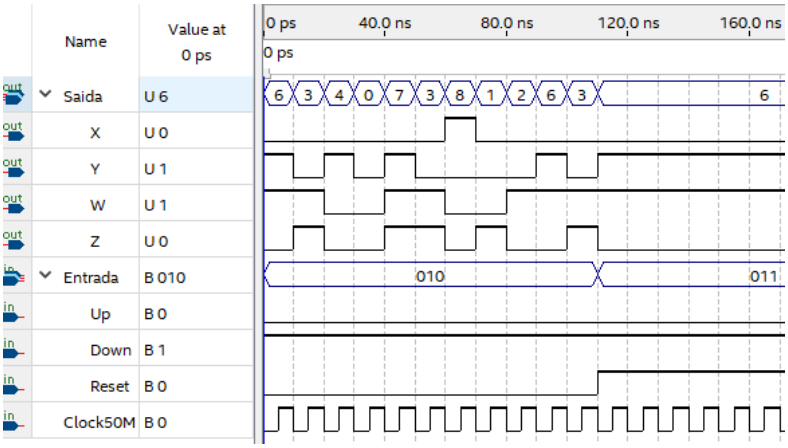


Figura 20 Forma de onda gerado pelo Quartus Prime para Up = 0 e Down = 1 (Contagem decrescente).

Observa-se que na contagem decrescente (figura 20), o resultado está de acordo com o esperado, inclusive ao pressionar o reset (que volta para o 6), porém, na crescente (figura 19) o mesmo não se observa, pois ele começa contando do 1 e ao reiniciar ele volta para o 1. O motivo disso é que, uma vez que estamos utilizando Mealy, esse resultado é o esperado já que, embora esteja no primeiro estado, ele está mostrando o resultado do segundo e assim sucessivamente. Para contornar esse problema, seria necessário colocar estado I como inicial.

A figura 21 mostra a forma de onda com o *blank* (Up = 1 e Down = 1) acionado em 45 ns em uma contagem crescente.

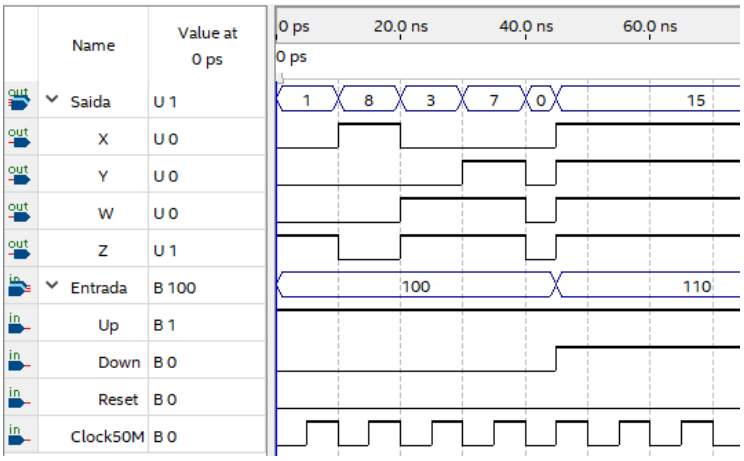


Figura 21 Forma de onda gerada pelo Quartus Prime exemplificando o blank acionado (Up = 1 e Down = 1) em 45 ns em uma contagem crescente.

A figura 22 mostra a forma de onda com o mantêm (*Up = 0 e Down = 0*) acionado em 45 ns em uma contagem crescente.

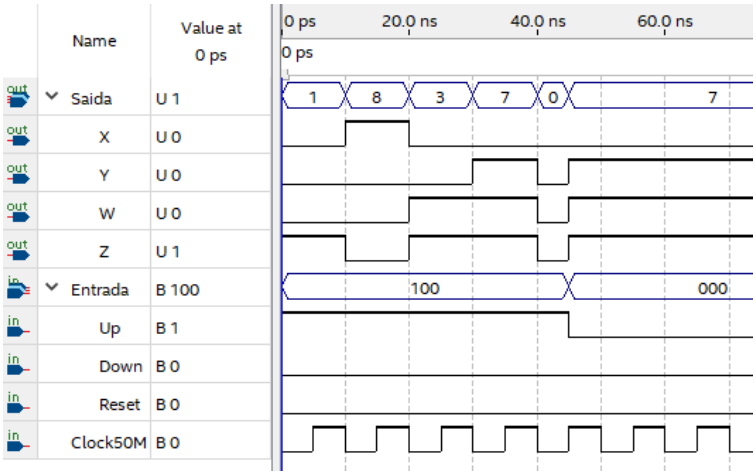


Figura 22 Forma de onda gerada pelo Quartus Prime exemplificando o mantêm acionado (Up = 0 e Down = 0) em 45 ns em uma sequência crescente.

Nota-se que não houve quaisquer instabilidade para função *blank* da máquina de estados (Figura 21), porém, no mantêm (figura 22), dá a impressão de que ele retorna o estado anterior quando é pressionado, nesse caso devemos lembrar que estamos implementando com Mealy e, pelo mesmo motivo do que ocorreu com o início da contagem, embora ele esteja em um estado, ele apresentará a saída do próximo e consequentemente, ao pressionar o mantêm, a saída do estado atual que será mostrada.

6 Considerações finais

Embora a máquina funcionou como projetada, na avaliação dos resultados, uma questão relacionada à implementação do projeto com máquina de Mealy, que é o fato de apresentar a saída do próximo estado enquanto está no atual.

Como a abordagem do Mealy seja não aguardar a subida (ou descida) do clock para apresentar o resultado, não se mostrou adequada para esse problema, uma vez que, embora fosse possível contornar o problema do início da contagem indicando o estado final como inicial, o mesmo não ocorreria com a função mantêm, pois se tentássemos corrigir essa questão para ordem crescente, o "problema" persistiria na ordem decrescente e vice-versa. Essas questão não seriam observadas no Moore.

A princípio, tentei separar as máquinas das funções que deveriam exercer (crescente, decrescente, mantêm e *blank*) e do contador numérico com o auxílio de multiplexadores, porém, houve um *delay* no clock de entre essas máquinas, o que demonstrou uma ideia ineficaz para modelagem do problema.

Mesmo com o auxílio do *software* ***Karnaugh Map Minimizer***, houveram diversos problemas, tanto na geração das expressões quanto na implementação no software ***Quartus Prime***, que comprometeram completamente o projeto.

Com o auxílio da *waveform* e da função de nomear os cabos no ***Quartus Prime***, descobertas posteriormente, esses problemas foram reduzidos consideravelmente. Em particular, sempre que era realizado uma atualização em uma dos blocos, utilizando a função *update symbol or block*, os fios que estavam sobrepostos se uniam e ocasionavam erro de compilação. A nomeação dos fios evitava a sobreposição e então união.

Acabei tendo que utilizar os *flip flop's* D do *Quartus*, pois os implementados, e inclusive utilizados no temporizador, não funcionaram na máquina de estados do contador numérico e não consegui identificar o problema.

Como não havia implementado nenhum projeto com máquina de Mealy anteriormente, tive bastante dificuldade em compreender a lógica e inclusive modelar o problema, acreditando que os resultados obtidos estavam incorretos e que não estava sabendo modelar o problema adequadamente.

Referências

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Grefory L. **Sistemas Digitais: Princípios e Aplicações**, 11ª ed. São Paulo: Pearson, 2011.

GOMES, Ícaro S. F.; FILHO, José A. de B.; HERCULANO, L. C.; MAIA, Lucas R.; JÚNIOR, Ricardo A. O. S. **Eletrônica Digital**. Ceará. Apostila de Eletrônica Digital do Curso de Engenharia Elétrica da Universidade Federal do Ceará, 2014.

PCS-EPUSP. **Utilização de circuitos biestáveis**. São Paulo. Apostila de Laboratório Digital da Escola Politécnica da USP, 2012.