UNIVERSIDADE FEDERAL DO ESTADO DE SÃO PAULO

GUILHERME AGUIAR CHRIST

PROJETO DE UMA MÁQUINA DE ESTADOS

Máquina de Mealy

São José Dos Campos - Brasil

2021

GUILHERME AGUIAR CHRIST
RA: 140592

PROJETO DE UMA MÁQUINA DE ESTADOS

Máquina de Mealy

Relatório apresentado ao curso: Laboratório de Sistemas Computacionais: Circuitos Digitais.

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
2021

RESUMO

O projeto teve como objetivo representar uma máquina de estados de Mealy, essa máquina apresenta dez estados, os quais são alternados a partir dos comandos: manter (exibe o estado atual), crescente (exibe o próximo estado), decrescente (exibe o estado anterior) e apagado. O circuito combina o estado de entrada com as entradas *up* e *down*, determinando qual será o próximo estado que será novamente combinado com as entradas *up* e *down*. A combinação do estado atual com as entradas também é utilizada para determinar a saída que será exibida no *display* de sete segmentos da placa Intel FPGA (*Field Programmable Gate Array*), acessada pelo laboratório remoto. O período de transição de estados foi determinado por um divisor de frequências, que aumenta o período do *clock* interno da placa de vinte nanosegundos para um segundo, por meio de circuitos digitais feitos no software ***Quartus Prime 20.1 Lite Edition***.

Palavras-chaves: Máquina de estados, mealy, Intel FPGA e Quartus Prime 20.1 Lite Edition.

LISTA DE IMAGENS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

No mundo atual, o cotidiano de grande parte da população tem uma grande relação com vários tipos de aparelhos eletrônicos, principalmente durante a pandemia do *COVID-19*, pois durante esse período o uso de computadores, celulares e outros dispositivos eletrônicos, se tornaram instrumento de trabalho para aqueles que adotaram o regime *home office*, e instrumento de estudos para os alunos no método de ensino a distância. Porém, mesmo usando esses dispositivos todos os dias, a maioria das pessoas, que os utilizam, não sabem o funcionamento por trás de todas as ações que tais dispositivos executam. Para que haja o funcionamento desses aparelhos, é necessário que existam pessoas que se aprofundem na área de circuitos digitais, para que a tecnologia presente nessas seja devidamente desenvolvida e manipulada.

Nesse relatório, é desenvolvido um projeto de uma máquina de estado de Mealy, que possui diversas aplicações no cotidiano da sociedade atual. Para isso, é usada uma sequência de número pré-definida e é identificado todos os passos para a construção do projeto.

2 Objetivos

2.1 Gerais

Desenvolver uma máquina de estados de Mealy com dez estados, que alterne entre eles no intervalo de um segundo. Essas alterações devem ocorrer de quatro formas diferentes, sendo elas: manter, crescente, decrescente e apagado. A sequência de estados que deve ser produzida é (5 - 4 - 7 - 8 - 0 - 1 - 2 - 6 - 3).

2.2 Específicos

Desenvolver os seguintes circuitos principais em *black boxes*:

- Divisor de frequência: utilizado para atrasar o sinal do *clock* de 50MHz para uma frequência de 1Hz, resultando em período de transição de 1 segundo.
- Circuito combinacional de entrada: utilizado para, a partir da entrada de estado atual e as entradas *up* e *down*, determinar o próximo estado.
- Circuito combinacional de saída: utilizado para, a partir da entrada de estado atual e as entradas *up* e *down*, determinar o número que será exibido no *display* de 7 segmentos.
- Registrador: utilizado para armazenar o estado atual e enviá-lo para os circuitos combinacionais de entrada e saída.
- Decodificador BCD: Utilizado para transformar os sinais de saída do circuito combinacional de saída em um número e exibi-lo no *display* de 7 segmentos.
- Obter e analisar os resultados: Analisar as *waveforms* e analisar as saídas no display da placa Intel FPGA.

3 Fundamentação Teórica

3.1 Funções e Portas Lógicas

Existem três funções primitivas para o desenvolvimento de circuitos combinacionais, sendo elas:

- Função NOT: Apresenta uma entrada A e uma saída S, essa função inverte o valor de A e o envia para saída S.

NOT	
A	S
0	1
1	0

Tabela 1 - Tabela-Verdade função NOT. Fonte: o Autor através do **Paint**.

- Função OR: Apresenta uma entrada A, uma entrada B e uma saída S, caso pelo menos uma das duas entradas esteja em nível alto, apresenta a saída em nível alto, caso contrário apresenta a saída em nível baixo.

OR		
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

Tabela 2 - Tabela-Verdade função OR. Fonte: o Autor através do **Paint**.

- Função AND: Apresenta uma entrada A, uma entrada B e uma saída S, caso ambas as entradas estejam em nível alto, apresenta a saída em nível alto, caso contrário apresenta a saída em nível baixo.

AND		
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Tabela 3 - Tabela-Verdade função AND. Fonte: o Autor através do **Paint**.

> Portas ou circuitos lógicos são dispositivos que operam e trabalham com um ou mais sinais lógicos de entrada para produzir uma e somente uma saída, dependente da função implementada no circuito. (WIKIPEDIA, https://pt.wikipedia.org/wiki/Porta_l%C3%B3gica)

A partir da combinação dessas três portas lógicas primitivas são formadas outras portas lógicas, como por exemplo as portas NAND, NOR, XOR, XNOR, elementos de memórias (como o *latch* e o *flip-flop*).

3.2 Divisor de frequência

Um divisor de frequência é implementado a partir de *flip-flops* JK, esse circuito tem a função de dividir a frequência fundamental do *clock*, e assim, possibilitando alterações no período de tempo entre as mudanças de estado.

3.3 Decodificador BCD

Um *display* de 7 segmentos é composto por 8 LEDs, sendo 7 deles os segmentos que podem representar uma letra ou um número e 1 deles que pode representar um ponto, cada um desses LEDs. No kit FPGA DE2-115, o *display* utilizado é do tipo Anodo comum, ou seja, os segmentos de LED que receberem um sinal em nível baixo serão acesos.

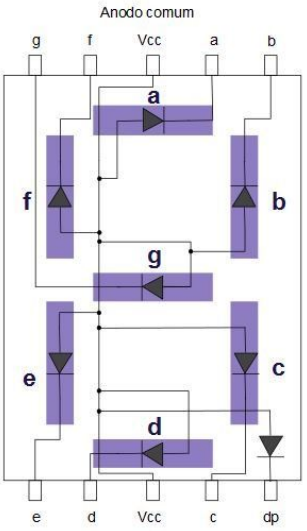

Figura 1 - *Display* de 7 segmentos.

Fonte:http://www.bosontreinamentos.com.br/eletronica/eletronica-digital/como-funciona-um-display-de-leds-de-7-segmentos/

4 Desenvolvimento do Trabalho

A máquina de Estados de Mealy deve apresentar dez estados, e a mudança entre os estados deve ocorrer de acordo com as entradas *up* e *down*, que seguem os seguintes comandos:

Up	Down	Comando
0	0	Manter
0	1	Crescente
1	0	Decrescente
1	1	Apagado

Tabela 4 - Comandos das entradas *up* e *down*. Fonte: o Autor

Para a montagem dos sistemas combinacionais, foi esquematizado um diagrama de estados que apresenta a seguinte forma:

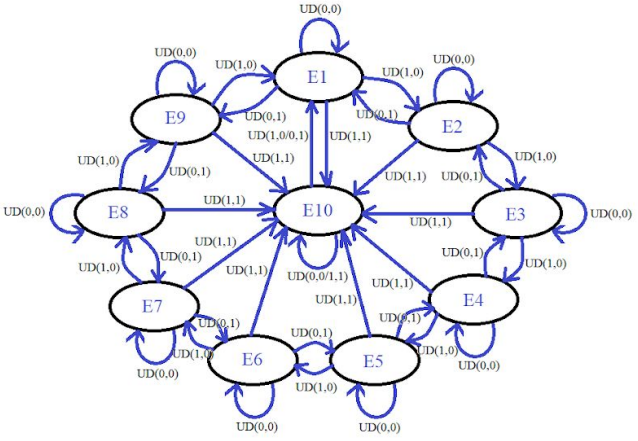

Figura 2 - Diagrama de estados. Fonte: o Autor através do **Paint**.

Cada estado representado no diagrama de estados (Figura 2) tem sua própria representação em binário:

Estado	Binário
E1	0001
E2	0010
E3	0011
E4	0100
E5	0101
E6	0110
E7	0111
E8	1000
E9	1001
E10	1111

Tabela 5 - Representação dos estados em binário. Fonte: o Autor.

Portanto, a partir da representação dos estados em binário (Tabela 5) e do diagrama de estados (Figura 2), foi gerada a tabela verdade para os circuitos combinacionais de entrada e saída:

Estado Atual	Próximo Estado				Saída			
(A,B,C,D)	(q3,q2,q1,q0)				(S3,S2,S1,S0)			
	UP(0,0)	UP(0,1)	UP(1,0)	UP(1,1)	UP(0,0)	UP(0,1)	UP(1,0)	UP(1,1)
E1(0001)	0001	1001	0010	1111	0101	0011	0100	1111
E2(0010)	0010	0001	0011	1111	0100	0101	0111	1111
E3(0011)	0011	0010	0100	1111	0111	0100	1000	1111
E4(0100)	0100	0011	0101	1111	1000	0111	0000	1111
E5(0101)	0101	0100	0110	1111	0000	1000	0001	1111
E6(0110)	0110	0101	0111	1111	0001	0000	0010	1111
E7(0111)	0111	0110	1000	1111	0010	0001	0110	1111
E8(1000)	1000	0111	1001	1111	0110	0010	0011	1111
E9(1001)	1001	1000	0001	1111	0011	0110	0101	1111
E10(1111)	1111	0001	0001	1111	1111	0101	0101	1111

Tabela 6 - Tabela-Verdade dos circuitos combinacionais de entrada e saída. Fonte:o Autor através do **Paint**.

4.1 Circuito combinacional de entrada

Para a montagem da *black box* foram utilizadas entradas *up* e *down* e as entradas (A,B,C,D) onde "A" representa o bit mais significativo e "D" o bit menos significativo. Portanto, as entradas da *black box* foram organizadas da seguinte forma:

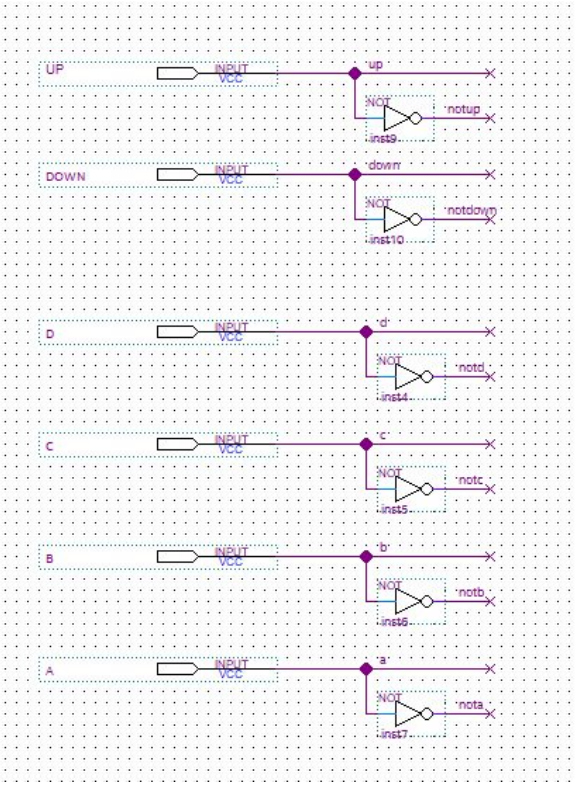

Figura 3 - Entradas da *black box* do circuito combinacional de entrada. Fonte: o Autor.

As saídas da *black box* do circuito combinacional de entrada estão representadas como (q3,q2,q1,q0), onde "q3" é o bit mais significativo e "q0" é o bit menos significativo. A partir da tabela-verdade deste circuito (Tabela 6) e do site "http://www.32x8.com/index.html" foi gerada a expressão booleana para cada uma das saídas. Nessas expressões, o sinal "+" representa a função "OR", o sinal "*" representa a função "AND" e o sinal "~" representa a função "NOT".

Para a saída "q0" foi obtida a seguinte expressão booleana:

$$q0 = (up * down) + (A * C) + (down * (\sim D)) + (up * (\sim D)) + (up * A) + ((\sim up) * (\sim down) * D) + ((\sim up) * (\sim A) * (\sim B) * (\sim C))$$

Portanto, o circuito que calcula a saída q0, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

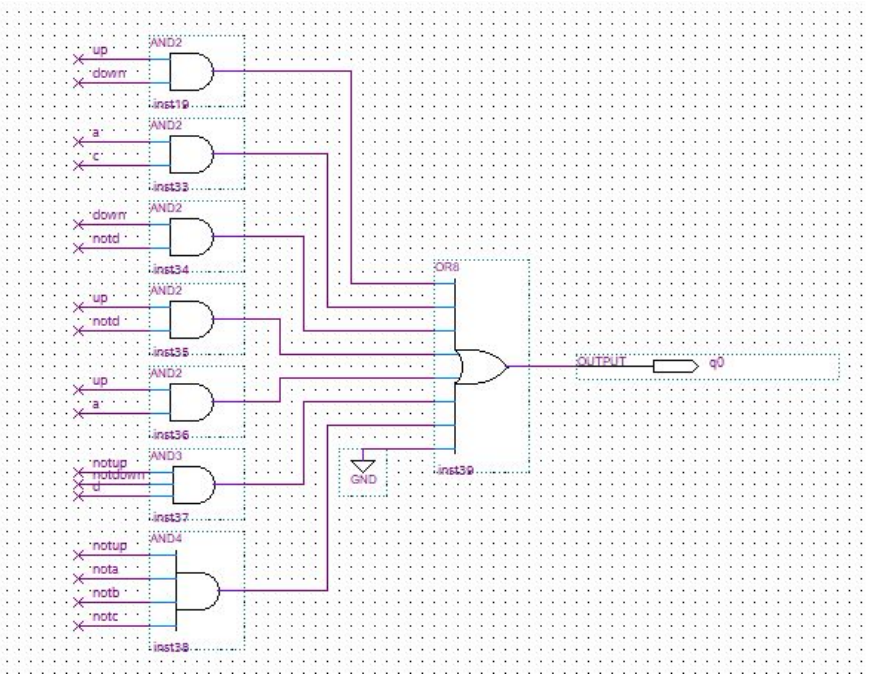

Figura 4 - Circuito que calcula a saída q0 da *black box* do circuito combinacional de entrada. Fonte: o Autor.

Para a saída "q1" foi obtida a seguinte expressão booleana:

q1 = (up * down) + ((~up) * (~down) * C) + ((~down) * C * (~D)) + (down * (~C) * (~D)) + ((~up) * (~A) * C * D) + (up * (~A) * (~C) * D)

Portanto, o circuito que calcula a saída q1, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

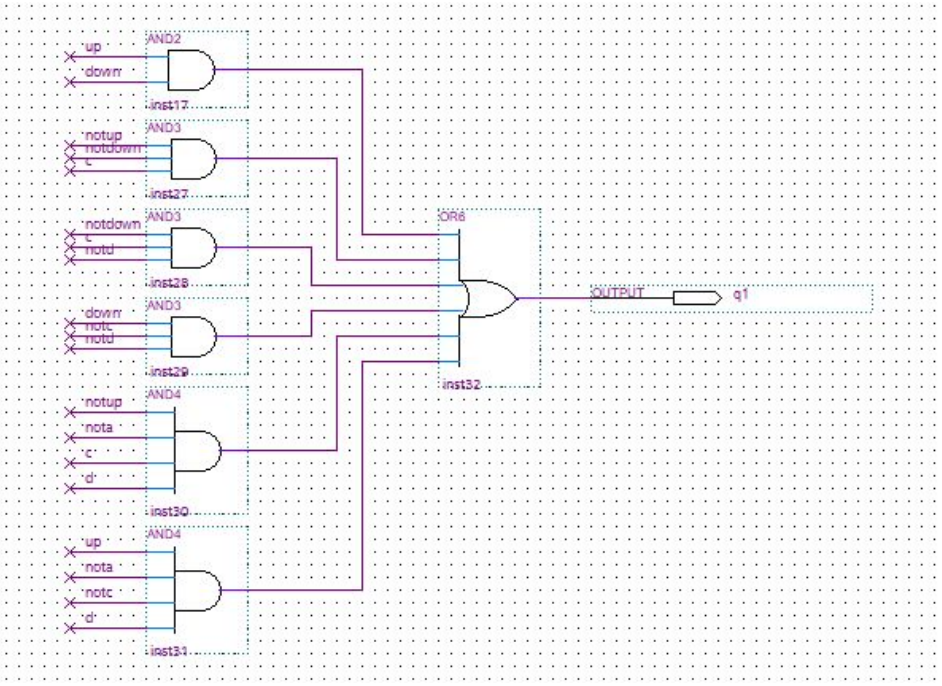

Figura 5 - Circuito que calcula a saída q1 da *black box* do circuito combinacional de entrada. Fonte: o Autor.

Para a saída “q1” foi obtida a seguinte expressão booleana:

q2 = (up * down) + ((~up) * (~down) * B) + ((~down) * B * (~C)) + (B * C * (~D)) + (down * A * (~D)) + ((~up) * (~A) * B * D) + (up * (~B) * C * D)

Portanto, o circuito que calcula a saída q2, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

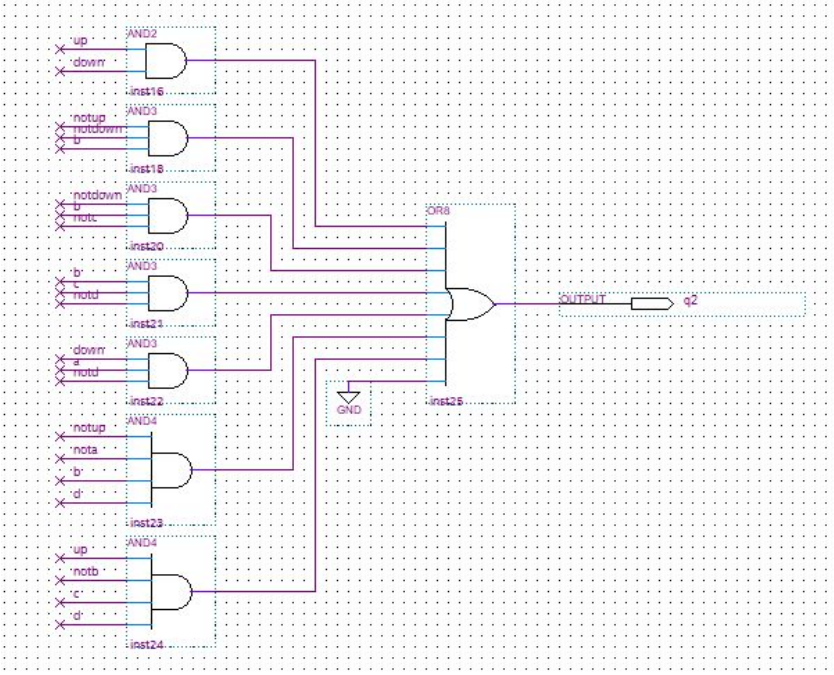

Figura 6 - Circuito que calcula a saída q2 da *black box* do circuito combinacional de entrada. Fonte: o Autor.

Para a saída “q3” foi obtida a seguinte expressão booleana:

q3 = (up * down) + (down * (~B) * (~C) * D) + ((~down) * (~B) * (~C) * (~D)) + ((~up) * (~down) * A * D) + (up * (~A) * B * C * D)

Portanto, o circuito que calcula a saída q3, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

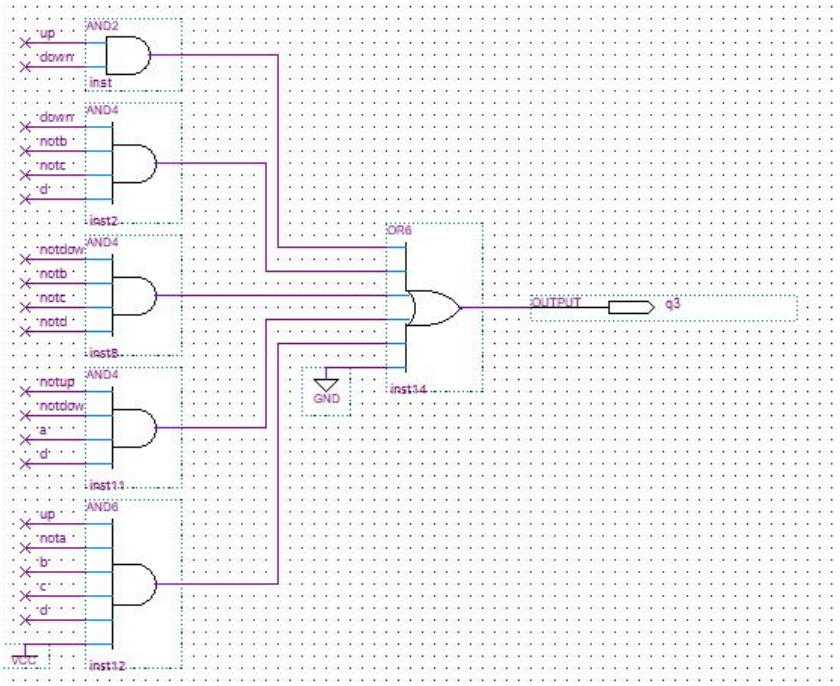

Figura 7 - Circuito que calcula a saída q3 da *black box* do circuito combinacional de entrada. Fonte: o Autor.

Então, a saída dessa *box* vai ser composta por esses 4 bits (q3,q2,q1,q0), que representam os 4 bits do próximo estado.

4.2 Circuito combinacional de saída

Similarmente ao circuito combinacional de entrada, para a montagem da *black box* do circuito combinacional de saída foram usadas as entradas *up* e *down* e as entradas (W,Z,Y,X) onde "W" representa o bit mais significativo e "X" o bit menos significativo. As entradas da *black box* foram dispostas da seguinte forma:

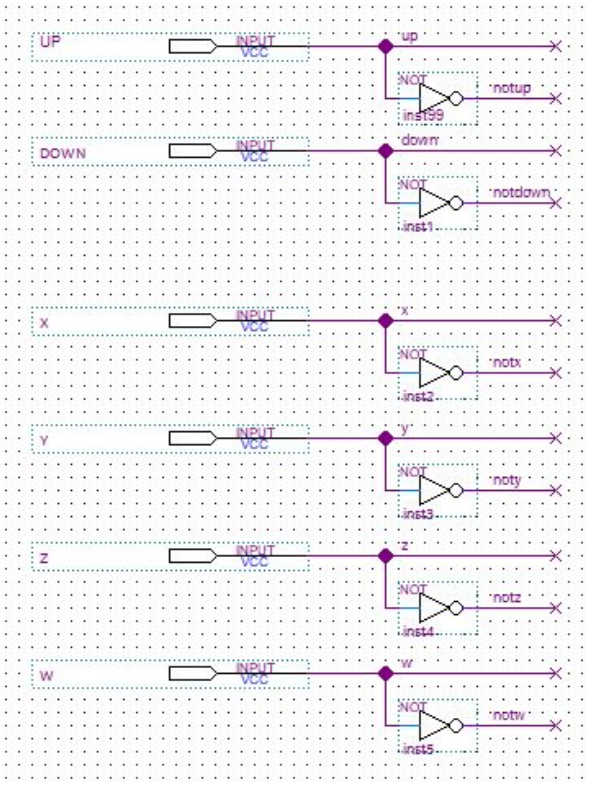

Figura 8 - Entradas da *black box* do circuito combinacional de saída. Fonte: o Autor.

As saídas da *black box* do circuito combinacional de entrada estão representadas como (S3,S2,S1,S0), onde "S3" é o bit mais significativo e "S0" é o bit menos significativo. Para obter as expressões booleanas de cada saída, foi usado o mesmo método utilizado no circuito combinacional de entrada. Porém, neste circuito, ao invés de obtermos os 4 bits que representam o próximo estado, obteremos 4 bits que representam um número binário, o qual será enviado ao decodificador BCD e será exibido no *display* de 7 segmentos em sua representação decimal.

Para a saída "S0" foi obtida a seguinte expressão booleana:

S0 = (up * down) + (A * C) + (up * A) + (up * (~B) * (~D)) + (down * B * C * D) + ((~up) * (~A) * (~B) * (~C)) + ((~up) * (~down) * (~B) * D) + (down * (~A) * (~B) * (~D)) + (down * (~A) * (~C) * (~D)) + (up * B * (~C) * D) + ((~up) * (~down) * B * C * (~D))

Portanto, o circuito que calcula a saída S0, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

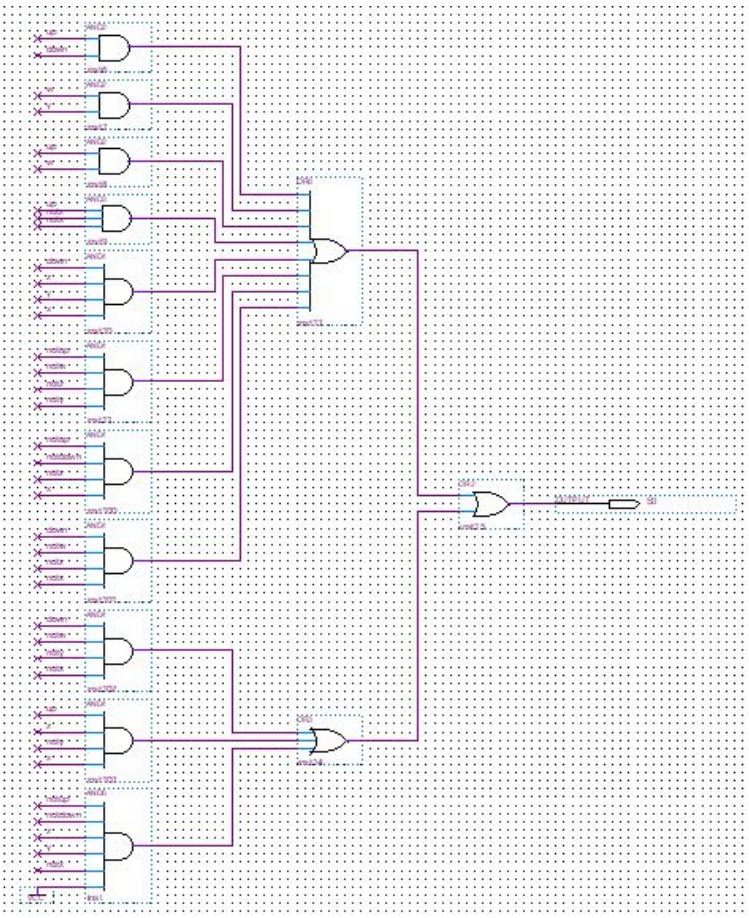

Figura 9 - Circuito que calcula a saída S0 da *black box* do circuito combinacional de saída. Fonte: o Autor.

Para a saída “S1” foi obtida a seguinte expressão booleana:

S1 = (up * down) + ((~up) * (~down) * A) + (down + (~B) * (~C)) + (down * (~C) * (~D)) + (up * (~B) * (~D)) + (up * (~A) * B * C) + ((~up) * (~down) * C * D)

Portanto, o circuito que calcula a saída S1, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

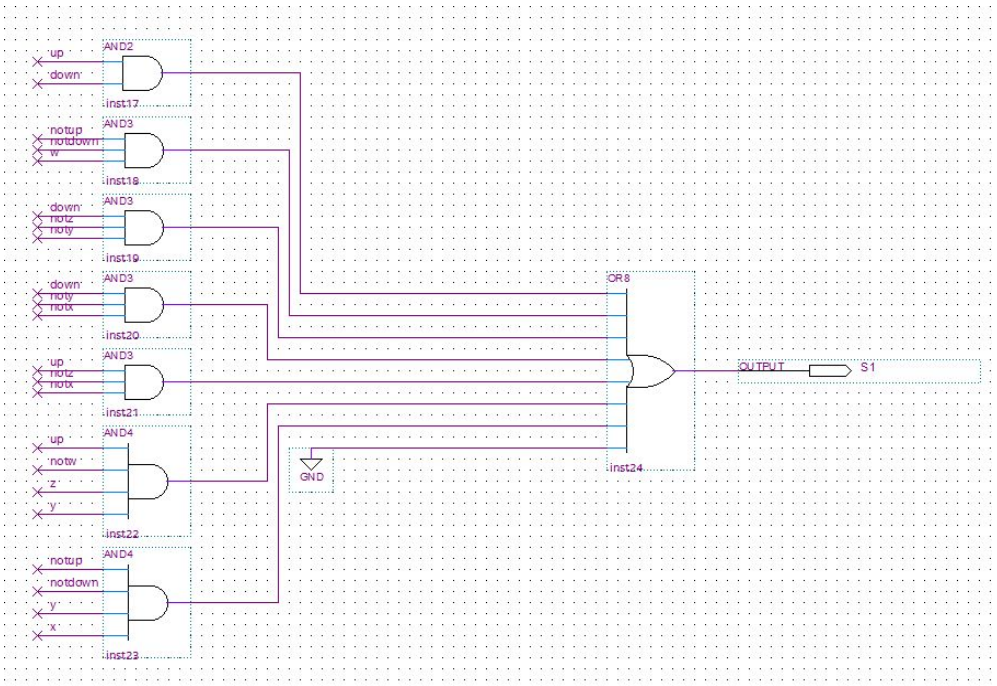

Figura 10 - Circuito que calcula a saída S1 da *black box* do circuito combinacional de saída. Fonte: o Autor.

Para a saída "S2" foi obtida a seguinte expressão booleana:

S2 = (up * down) + (A * B) + ((~A) * (~B) * (~D)) + (up * A * D) + ((~up) * (~B) * C) + (down *A * D) + (up * B * C * D) + (down *(~A) * (~C) * (~D)) + ((~down) * (~A) * (~B) * (~C)) + ((~up) * (~down) * A * (~D))

Portanto, o circuito que calcula a saída S2, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

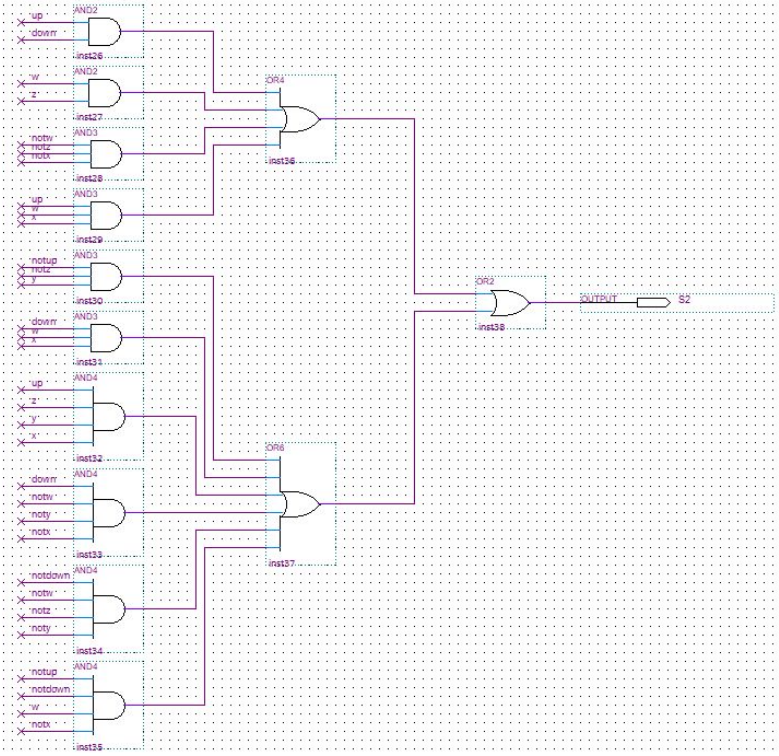

Figura 11 - Circuito que calcula a saída S2 da *black box* do circuito combinacional de saída. Fonte: o Autor.

Para a saída "S3" foi obtida a seguinte expressão booleana:

S3 = (up * down) + ((~up) * (~down) * A * C) + (down * B * (~C) * D) +
(up * (~B) * C * D) + ((~up) * (~down) * (~A) * (~C) * (~D)

Portanto, o circuito que calcula a saída S3, gerado a partir da expressão booleana, ficou implementado na *black box* da seguinte forma:

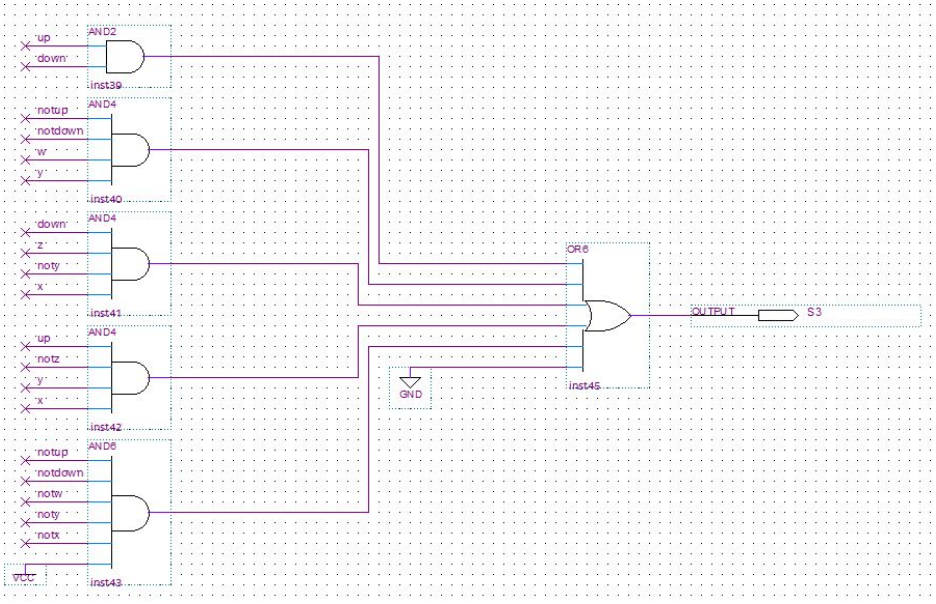

Figura 12 - Circuito que calcula a saída S3 da *black box* do circuito combinacional de saída. Fonte: o Autor.

Os sinais recebido por esse circuito vem do registrador, que tem como função armazenar o estado atual e as entradas *up* e *down*. O registrador envia essas entradas e o estado atual para o circuito combinacional de saída que converte essas entradas para um 4 bits que representam o número binário que será enviado ao decodificador BCD e será exibido no *display* de 7 segmentos na sua representação em decimal.

4.3 Registrador

O Registrador tem como entrada 4 bits, que vêm do circuito combinacional de entrada, esses bits representam o estado atual, junto com esses bits, o registrador tem uma entrada para o *clock* e uma entrada para o *reset*, essa última entrada tem como função retornar a contagem para o estado atual, caso seu valor esteja em nível alto. Para a montagem do circuito foram utilizados quatro *flip-flops* do tipo D, que armazenam os 4 bits de entrada do circuito. A organização do circuito dentro da *black box* ficou da seguinte forma:

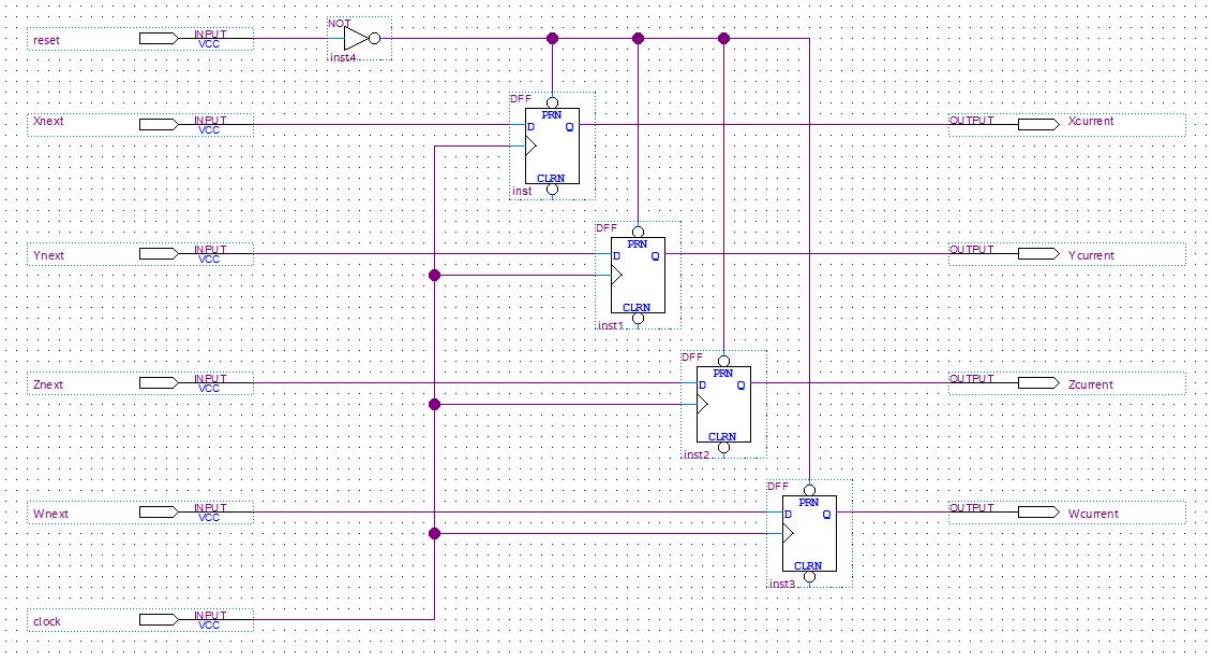

Figura 13 - Circuito que representa a *black box* do registrador. Fonte: o Autor.

O *clock* recebido pelo circuito deve ter uma frequência de 1Hz para que o período de transição entre os estados seja de 1 segundo, porém a frequência nativa do *clock* interno da placa FPGA é de 50 MHz, portanto é necessário um divisor de frequência para alcançarmos a frequência desejada.

4.4 Divisor de Frequência

Um divisor de frequência é um circuito que apresenta uma sequência de *flip-flops* do tipo JK. Esse circuito tem como função dividir a frequência do *clock* de entrada por dois a cada *flip-flop* que é inserido nele. No casa da placa FPGA, o *clock* interno presente nela tem uma frequência nativa de 50MHz, para que o tempo de transição entre os estados da nossa máquina seja de 1 segundo, precisamos dividir a frequência nativa do *clock* interno para que ela seja de 1Hz. Sendo "*f*" a frequência que queremos obter, "*fn*" a frequência nativa do *clock* interno e "*n*" o número de *flip-flops* que devem ser usados, temos que:

$$f = \frac{fn}{2^n} \Rightarrow 1 = \frac{50.10^6}{2^n} \Rightarrow 2^n = 50.10^6 \Rightarrow n.log_2(2) = log_2(50.10^6)$$

$$\Rightarrow n \approx 25,57$$

Portanto, para obtermos uma frequência de 1Hz são necessários 26 *flip-flops* do tipo JK, no circuito. Então, o circuito montado, dentro da *black box* do divisor de frequência ficou da seguinte forma:

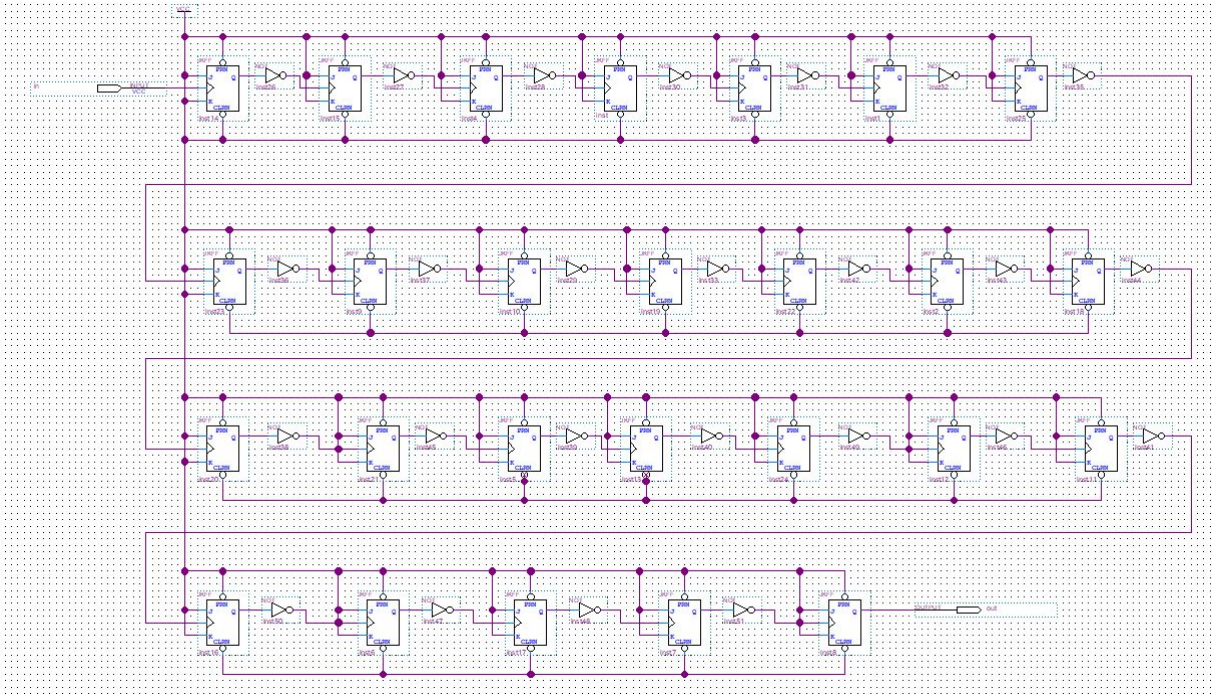

Figura 14 - Circuito completo do divisor de frequência. Fonte: o Autor.

4.5 Decodificador BCD

O decodificador BCD é responsável por transformar os 4 bits de saída do circuito combinacional de saída em um número decimal que será exibido no *display* de 7 segmentos da placa FPGA. Cada saída do circuito será exibido da seguinte forma:

Saída Estado Atual (S3,S2,S1,S0)	**Decimal**
E1(0101)	5
E2(0100)	4
E3(0111)	7
E4(1000)	8
E5(0000)	0
E6(0001)	1
E7(0010)	2
E8(0110)	6
E9(0011)	3
E10(1111)	15

Tabela 7 - Representação da saída de cada estado em decimal. Fonte: o Autor.

Esse circuito apresenta 4 entradas (W,X,Y,Z), onde "W" é o bit mais significativo do número binário de entrada e "Z" é o bit menos significativo, e 7 saídas (a,b,c,d,e,f,g), cada saída representa seu respectivo led no *display* de 7 segmentos, como pode-se observar na Figura 1. Por se tratar de *display* do tipo anodo comum, as saídas que estiverem em nível baixo estarão acesas e as em nível alto estarão apagadas. Tomemos como exemplo o número 0101(b), para essa situação, as saídas (a,b,c,d,e,f,g) terão seus respectivos níveis como (0,1,0,0,1,0,0). Para realizar esta função foi montada uma *black box* com o seguinte sistema:

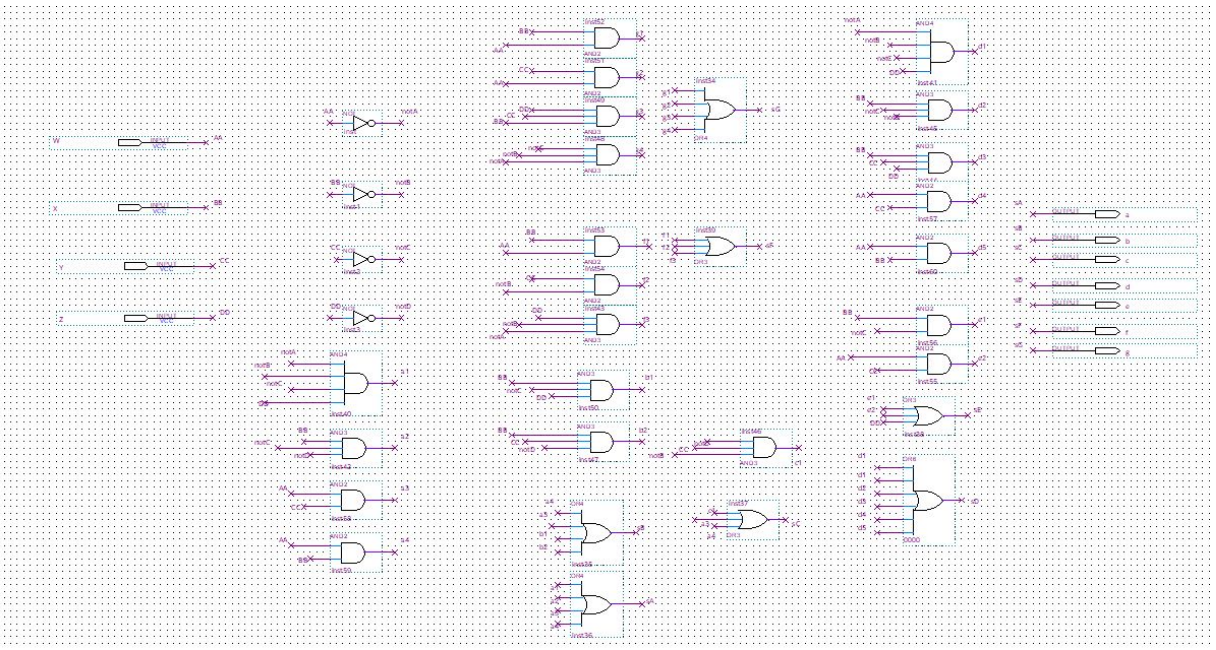

Figura 15 - Circuito referente a *black box* do decodificador BCD para o *display* de 7 segmentos. Fonte: o Autor.

4.6 Circuito Final

Após o desenvolvimento de todos os circuitos em *black boxes*, foi feito o circuito final, combinando todas as *boxes* e implementando os sinais de entrada *up*, *down*, *clock* e *reset*, e tendo como saída os sinais (a,b,c,d,e,f,g) que serão utilizado para a exibição dos estados no *display* de 7 segmentos. O circuito final ficou organizado da seguinte forma:

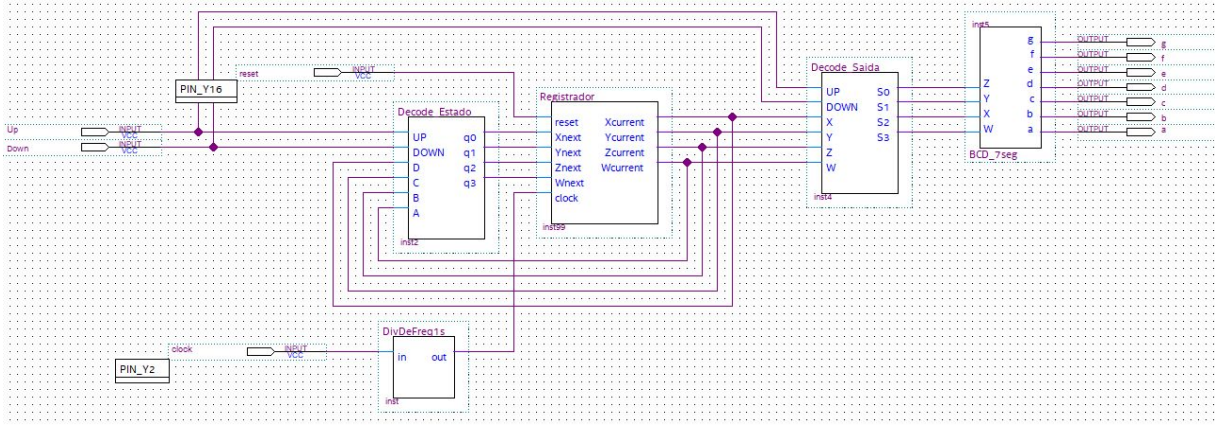

Figura 16 - Circuito final da máquina de estados de Mealy. As *boxes* Decode_Estado e Decode_Saida representam, respectivamente, os circuitos combinacionais de entrada e saída, a *box* registrador representa o próprio registrador, a *box* DivDeFreq1s representa o divisor de frequência e a *box* BCD_7seg representa o decodificador de 7 segmentos. Fonte: o Autor.

5 Resultados Obtidos e Discussão

Para a análise dos resultados, foram geradas as *waveforms* pelo software ***Quartus Prime 20.1 Lite Edition***. A partir delas, é possível analisar quais serão as saídas (a,b,c,d,e,f,g) que acenderão seus respectivos LEDs, representados na Figura 1, para cada configuração das entradas *up*, *down* e *reset*. Para a análise, foi retirado o divisor de frequência e o decodificador BCD, para que o número exibido do *display* de 7 segmentos já esteja representado na imagem da *Waveform*.

Para a disposição das entradas *up* = 1 e *down* = 0, teremos a seguinte *waveform*:

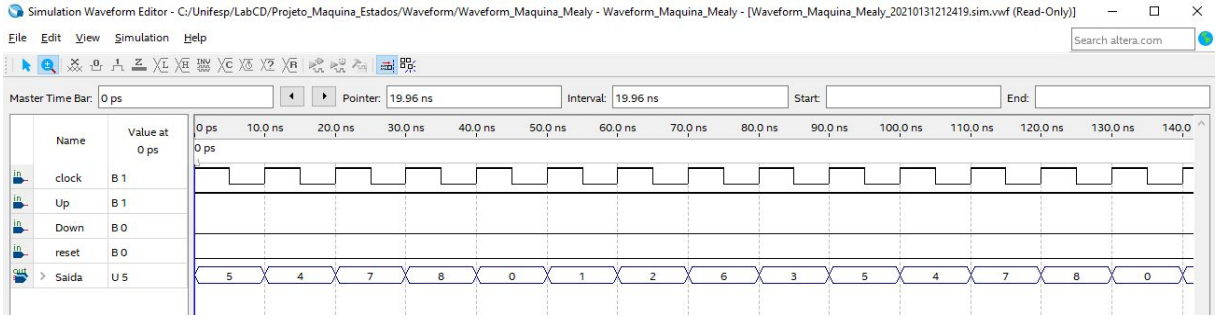

Figura 17 - *Waveform* para a disposição UD(1,0). Fonte: o Autor.

Então, para essa disposição UD(1,0), a sequência (5-4-7-8-0-1-2-6-3) é seguida de forma crescente, como era esperado.

Para a disposição das entradas *up* = 0 e *down* = 1, teremos a seguinte *waveform*:

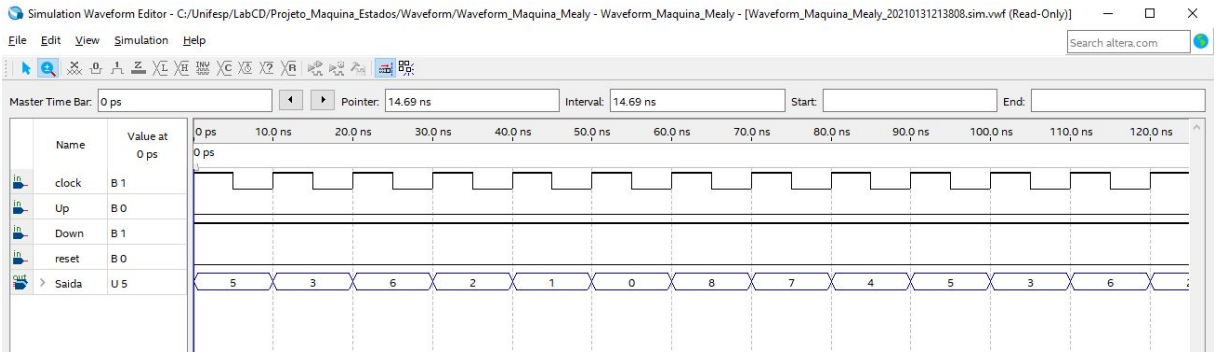

Figura 18 - *Waveform* para a disposição UD(0,1). Fonte: o Autor.

Essa disposição UD(0,1), apresenta, devidamente, a sequência (5-4-7-8-0-1-2-6-3) na forma de decrescente, assim como era esperado.

Agora, iniciamos a contagem na disposição das entradas *up* = 1 e *down* = 0, e em seguida mudamos o valor de *up* para 0 e depois, voltamos o valor de *up* para 1, o que gera a seguinte *waveform*:

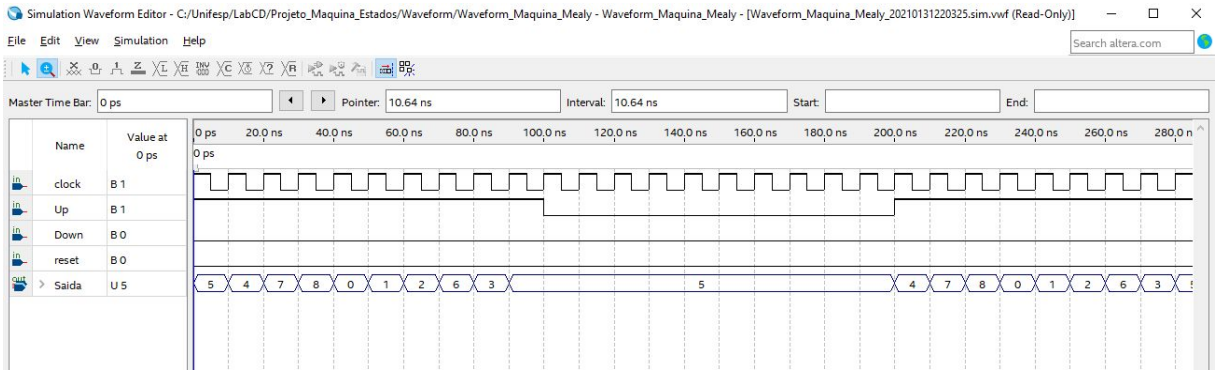

Figura 19 - *Waveform* para a disposição UD(1,0) mudando para a disposição UD(0,0). Fonte: o Autor.

Enquanto a entrada *up* está em nível alto, a sequência (5-4-7-8-0-1-2-6-3) é devidamente percorrida na forma crescente e quando a entrada *up* é posta em nível baixo, o estado atual, que nesse caso é representado pelo número 5, é mantido, até que a entrada *up* volte para o nível e a sequência volte a ser percorrida de forma crescente, como era esperado.

Em seguida, iniciamos a contagem na disposição das entradas *up* = 0 e *down* = 1, e após um tempo, mudamos o valor de *up* para 1 e após um tempo, mudamos o valor de *up* para 0 novamente, o que gera a seguinte *waveform*:

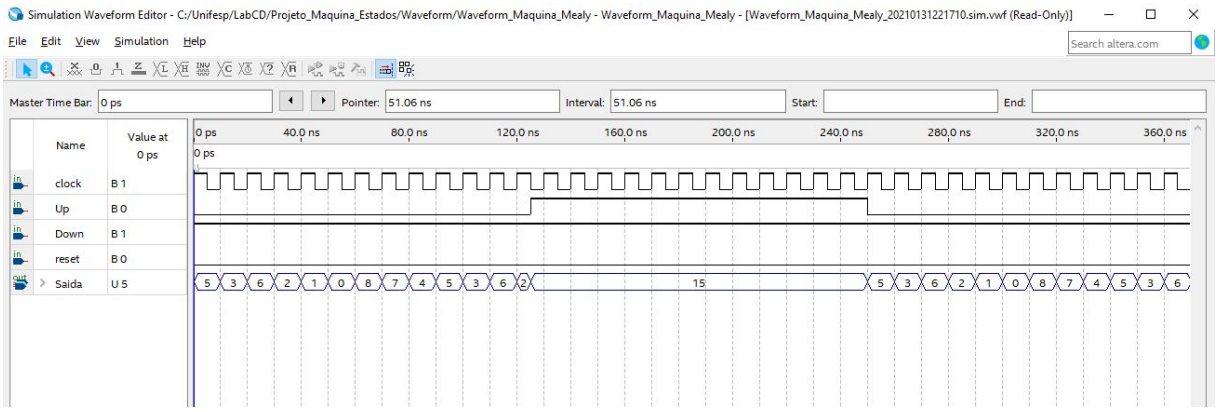

Figura 20 - *Waveform* para a disposição UD(0,1) mudando para a disposição UD(1,1). Fonte: o Autor.

Observe que, enquanto as entradas *up* e *down* estão disposta como UD(0,1) a sequência (5-4-7-8-0-1-2-6-3) é percorrida corretamente na ordem decrescente, quando mudamos a disposição para UD(1,1) é exibido o número 15, que representa o *display* apagado, como previsto na Tabela 7, e quando mudamos a disposição para UD(0,1) a sequência é retomada no seu estado inicial e é percorrida novamente na ordem decrescente, assim como era esperado.

Iniciando novamente na disposição *up* = 1 e *down* = 0, vamos mudar o valor da entrada *reset* para nível alto e depois mudar o valor novamente para nível baixo, isso nos gera a seguinte *waveform*:

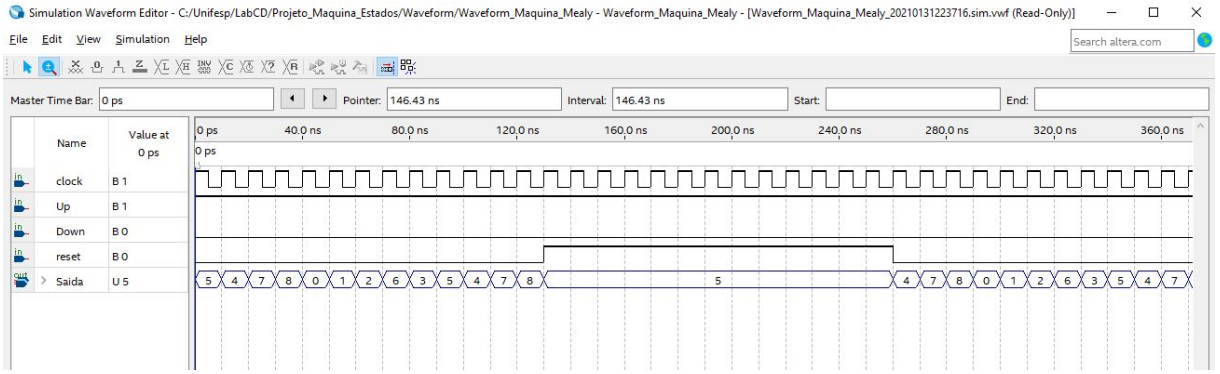

Figura 21 - *Waveform* para a disposição UD(1,0) ligando e desligando o *reset*. Fonte: o Autor.

Note que, antes do *reset* ser acionado, a sequência (5-4-7-8-0-1-2-6-3) é percorrida corretamente na ordem crescente, quando o *reset* é acionado, a contagem volta ao estado inicial e quando o *reset* volta para o nível baixo, a sequência volta a ser percorrida na ordem crescente, como era esperado.

6 Considerações Finais

A máquina de estados de Mealy representou os estados de forma correta, porém sua implementação foi muito complexa, uma vez que há uma grande quantidade de ligações a serem feitas, principalmente nos circuitos combinacionais de entrada e saída. Também, quando o projeto foi descarregado na placa Intel FPGA e testado através do laboratório remoto, notou-se alterações repentinas nos números exibidos no *display* quando há uma mudança de valor das entradas, entretanto isso é esperado, devido ao atraso que o laboratório remoto apresenta. No geral, a máquina de estados de Mealy funcionou perfeitamente e atingiu os objetivos do projeto.

Referências

1. Prof. Dr. Lauro Paulo da Silva Neto.
2. Profª. Dra. Fernanda Quelho Rossi.
3. Lucas Gomes Ferraz.
4. Rafael Moore Corrêa.
5. Sistemas Digitais – Fundamentos e Aplicações. Thomas L. Floyd. Editora Bookman, 2007.
6. http://www.bosontreinamentos.com.br/eletronica/eletronica-digital/como-funciona-um-display-de-leds-de-7-segmentos/.
7. WIKIPEDIA, https://pt.wikipedia.org/wiki/Porta_l%C3%B3gica.
8. https://www.techtudo.com.br/dicas-e-tutoriais/2018/12/como-fazer-formatacao-abnt-no-google-docs.ghtml.