ID: .....

Máquina de Estados Finitos: Máquina de Moore

São José dos Campos - Brasil

31 de Outubro de 2018

ID: .....

Máquina de Estados Finitos: Máquina de Moore

Relatório Técnico na aplicação de circuitos digitais em uma Máquina de Moore.

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 - Brasil

31 de Outubro de 2018

Resumo

Neste trabalho foi realizada a confecção de uma Máquina de Estados Finito com parâmetros definidos de forma aleatória por sorteio. A função da Máquina de Moore sorteada é percorrer a sequencia de números (4-5-6-4-8-0-1-9-2) na ordem original ou inversa em uma frequência de 0,5Hz. O circuito também deverá manter o estado atual, apagar o *display* e voltar ao primeiro número da sequencia. Todas as ações são controladas por três sinais de entrada: *up, down* e *reset*. A implementação foi realizada através do *Software* Quartus e da placa FPGA DE2-115.

Palavras-chaves: Máquina de Moore. Máquina de Estados Finitos. Circuitos Digitais.

Lista de ilustrações

Sumário

1 Introdução

Circuitos Digitais são de suma importância no mundo atual, praticamente tudo o que produzimos hoje em dia possui um circuito atribuído a ele. O aprendizado dessa disciplina ajuda muito o futuro da sociedade e o desenvolvimento do raciocínio lógico nas pessoas.

Para isso, vamos realizar a confecção de uma Máquina de Estados na disciplina de Laboratório de Circuitos Digitais.

Essa Máquina de Estados deverá obedecer certos parâmetros sorteados em classe e seguir um modelo disponibilizado pelo professor.

2 Objetivos

2.1 Geral

Utilizando máquina de estados finitos de Moore, desenvolver um contador numérico para a sequência 4-5-6-4-8-0-1-9-2. O contador deverá operar a uma frequência de 0.5 Hz e utilizar três sinais de controle (*up, down* e *reset*) para manter o estado atual, desligar o *display* e percorrer a sequência de maneira normal ou inversa.

2.2 Específicos

- Utilizar o software ***Quartus Prime*** para a implementação dos circuitos.
- Implementar um circuito conversor de 50 MHz para 0.5 Hz.
- Implementar a Máquina de Moore e seus sub-circuitos, incluindo:
  - Circuito de Transição.
  - Circuito de Estado.
  - Circuito de Saída.
- Implementar um circuito conversor de BCD para *Display*.
- Combinar os circuitos criados para realizar a função desejada.
- Analisar os arquivos *Waveform* dos circuitos criados.

3 Fundamentação Teórica

3.1 Máquina de Estados Finitos de Moore

Uma máquina de estados finitos (*FSM*, do inglês *Finite State Machine*) nada mais é um modelo matemático que representa um circuito ou algoritmo que possui um número finito de estados, podendo estar em apenas em um estado de cada vez. Esse tipo de modelo pode ser dividido em quatro categorias: Aceitadores/Reconhecedores, Classificadores, Transdutores e Geradores.

Os Aceitadores possuem a função de analisar uma entrada e emitir um sinal binário que indica se a entrada é valida ou não. Classificadores são similares aos Aceitadores porém com mais de dois estados terminais. Geradores recebem apenas um parâmetro como argumento, eles produzem apenas uma sequência única, que pode ser visto como uma saída de um Aceitador ou Classificador.

Os Transdutores geram uma saída baseada no estado atual do circuito e na entrada. Eles são muito importantes na área de controle computacional. Existem dois principais tipos de Transdutores:

- Máquina de Moore:

  A Máquina de Moore é uma *FSM* Transdutora que sua saída depende apenas do estado atual de seus registradores. Ela foi proposta por Edward F. Moore em 1956 (1).

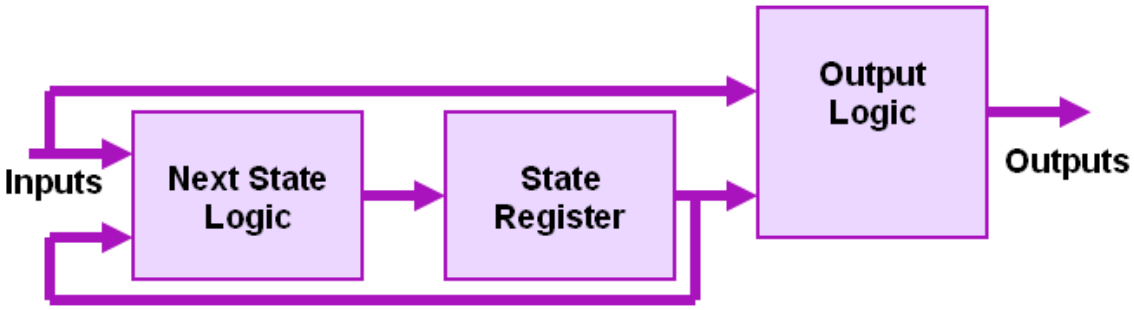
Figura 1 – Modelo Máquina de Moore

Fonte: *http://electrosofts.com/verilog/fsm.html* (2)

- Máquina de Mealy:

  A Máquina de Mealy é uma *FSM* Transdutora que sua saída depende do estado de seus registradores e de sua entrada atual. Ela foi proposta por George H. Mealy em 1955 (3).

Figura 2 – Modelo Máquina de Mealy



Fonte: *http://electrosofts.com/verilog/fsm.html* (2)

3.2 Temporizador

Um temporizador nada mais é que um retardador para o circuito, dessa maneira conseguimos um novo *Clock* com o valor desejado. Definindo um γ a partir da frequência original f_0 e da frequência desejada f (Equação 3.1). Podemos achar o número de *flip-flops* do tipo D (N_{FF}) necessários para a implementação do temporizador utilizando a Equação 3.2, note que é necessário um *flip-flop* a mais para manter a mudança de estado após o pulso.

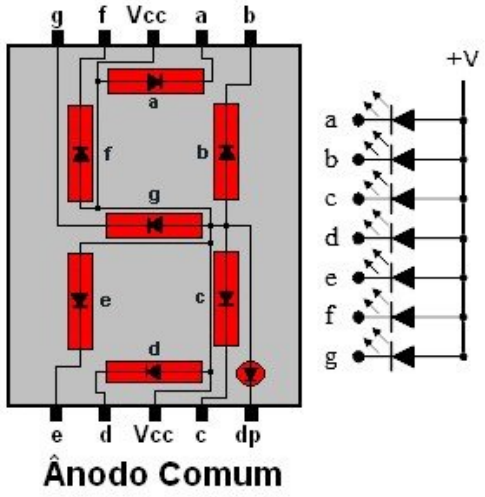
$$\gamma = \frac{f_0}{2 \cdot f} \tag{3.1}$$

$$N_{FF} = \lceil \log_2 \gamma \rceil + 1 \tag{3.2}$$

3.3 *Display* de 7 segmentos

O *Display* de 7 segmentos é um método muito comum de representação visual de dados digitais. O circuito apresenta 8 LEDs, 7 segmentos e 1 ponto, que podem ser acionados individualmente para representar um grande número de informações visuais para o usuário, geralmente sendo números e algumas letras.

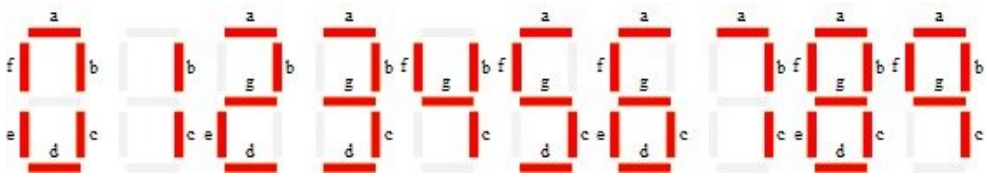
Figura 3 – Circuito Interno de um *Display* de 7 Segmentos



Fonte: *https://www.electronica-pt.com/eletronica-digital/display-7-segmentos*

Como podemos observar na Figura 3, os LEDs são controlados por sinais nomeados de "a"até "f"e "dp". Ligando e desligando esses sinais podemos formar números para representar dados do circuito ou sistema analisado, como podemos observar na Figura 4.

Figura 4 – Escrevendo Números com os Sinais de Controle



Fonte: *https://www.electronica-pt.com/eletronica-digital/display-7-segmentos*

Note na Figura 3 que, devido ao circuito possuir Ânodo Comum, os LEDs são acessos apenas se o sinal de controle for 0.

4 Desenvolvimento

4.1 Temporizador

Devido aos parâmetros do projeto, necessitamos gerar um *Clock* de 0.5 Hz a partir do *Clock* de 50 MHz da placa FPGA utilizada. Aplicando a Equação 3.1 podemos obter o tempo (γ) entre as transções de estado do novo *Clock* a ser produzido:

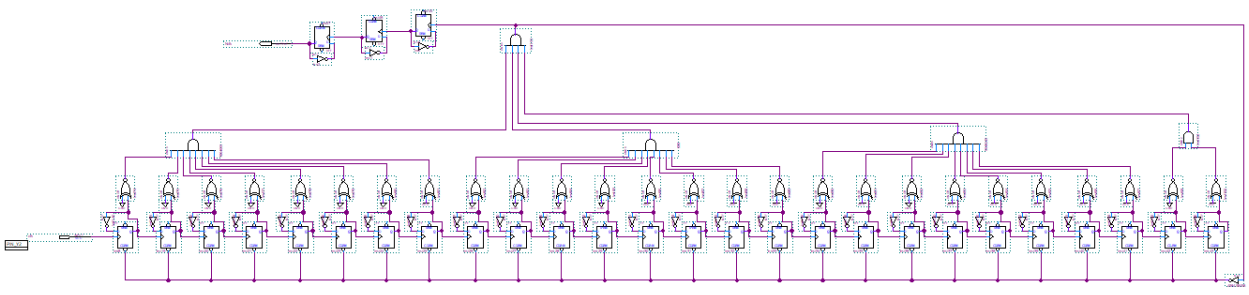
$$\gamma = \frac{50\text{MHz}}{2 \cdot 0.5\text{Hz}} = 50\text{MHz}$$

Com isso teremos uma mudança de estado no novo *Clock* a cada 50 MHz. Para descobrirmos o número de *flip-flops* ($\mathsf{N_{FF}}$) a serem utilizados:

$$\mathsf{N_{FF}} = \left\lceil \log_2 \ 50 \cdot 10^6 \right\rceil + 1 = 27$$

Portanto, necessitamos utilizar 27 *flip-flops*, 1 para manter o sinal após o pulso e 26 para contar até γ. Quando o valor de γ for atingido o contador deverá reiniciar e inverter no estado do *Clock* de saída. Portanto o circuito ficará desta maneira:

Figura 5 – Circuito conversor de 50 MHz para 0.5 Hz



4.2 Conversor de BCD para *Display*

O conversor irá transformar sinais digitais em visuais, para ser possível a saída de dados do circuito. Seguindo a tabela da Figura 6, podemos criar circuitos para gerar sinais visuais.

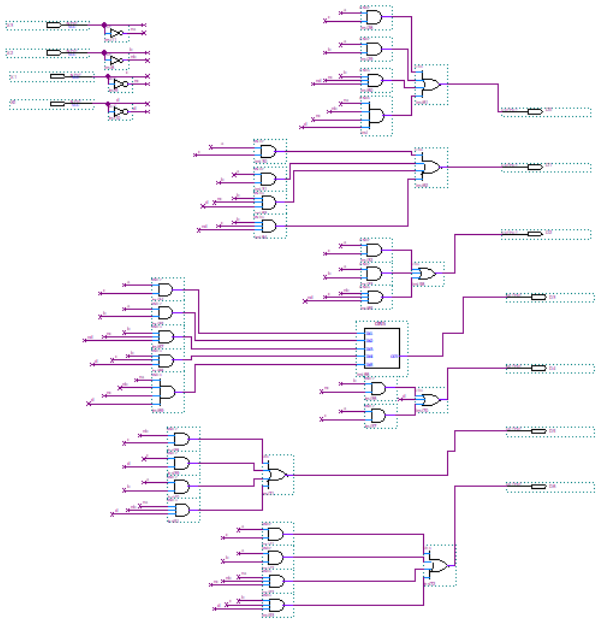
Figura 6 – Tabela verdade de conversão para *Display*

Binary Inputs	Decoder Outputs	7 Segment Display Outputs
D C B A	a b c d e f g	
0 0 0 0	1 1 1 1 1 1 0	0
0 0 0 1	0 1 1 0 0 0 0	1
0 0 1 0	1 1 0 1 1 0 1	2
0 0 1 1	1 1 1 1 0 0 1	3
0 1 0 0	0 1 1 0 0 1 1	4
0 1 0 1	1 0 1 1 0 1 1	5
0 1 1 0	1 0 1 1 1 1 1	6
0 1 1 1	1 1 1 0 0 0 0	7
1 0 0 0	1 1 1 1 1 1 1	8
1 0 0 1	1 1 1 1 0 1 1	9

Fonte: *https://quickgrid.wordpress.com/2015/03/22/7-segment-decoder-implementation-truth-table-logisim-diagram/*

A partir desta tabela podemos criar o circuito abaixo:

Figura 7 – Circuito BCD para *Display*

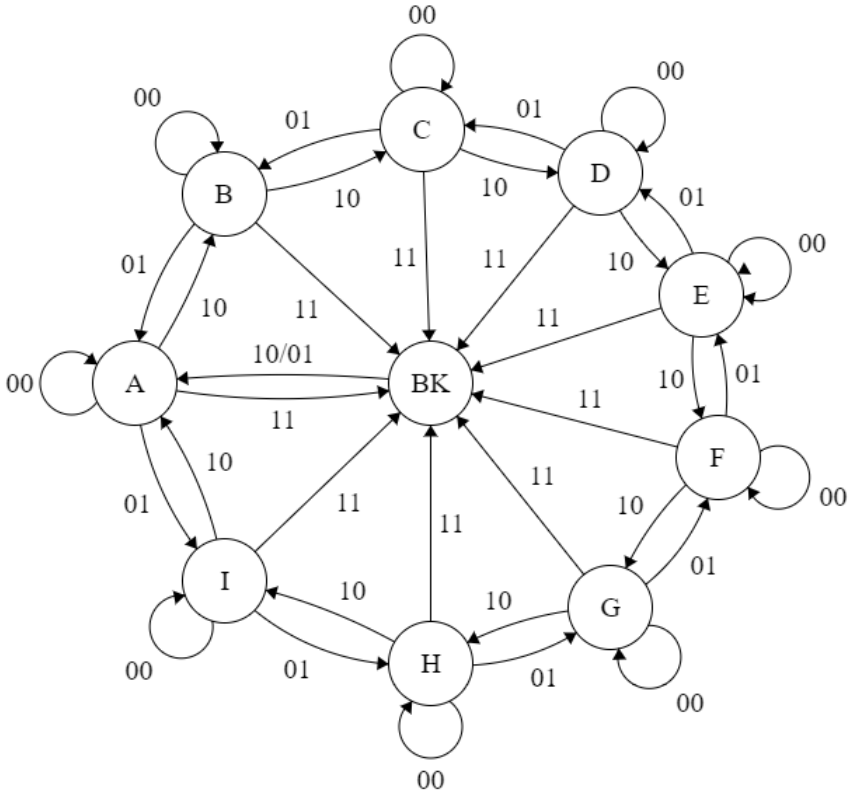


Fonte: *https://quickgrid.wordpress.com/2015/03/22/7-segment-decoder-implementation-truth-table-logisim-diagram/*

4.3 Máquina de Estados

Podemos contruir nosso grafo de estados com base nas posições das chaves *up* e *down*.

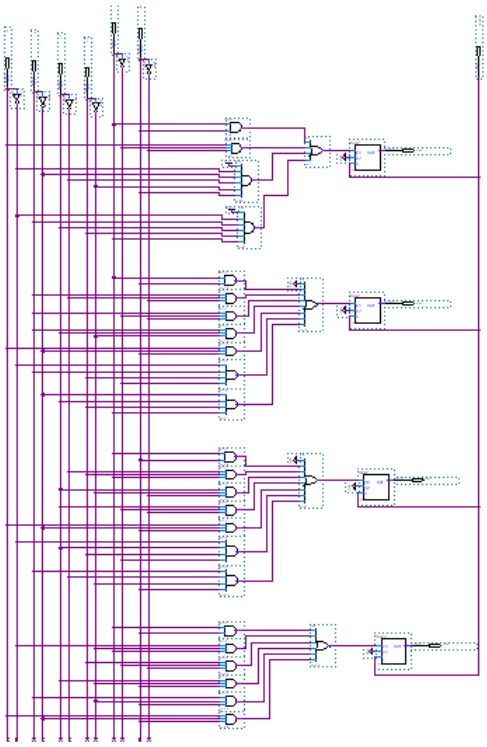
Figura 8 – Máquina de Estados



Os números representam, em ordem, as chaves *up* e *down*.

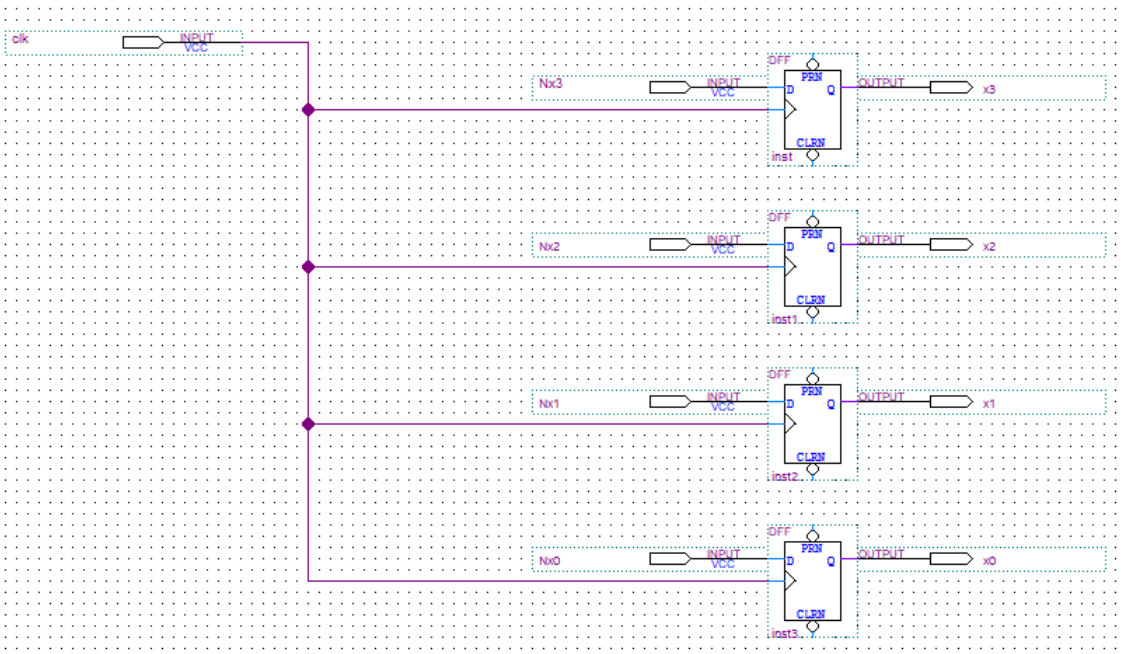
Após definir-mos o diagrama da máquina de estados, necessitamos criar as 3 partes fundamentais: Circuito de transição, Estado Atual e Decodificador. Com base no diagrama e utilizando mapas de Karnaugh, podemos chegar no seguinte Circuito de transição:

Figura 9 – Circuito de Transição de Estados



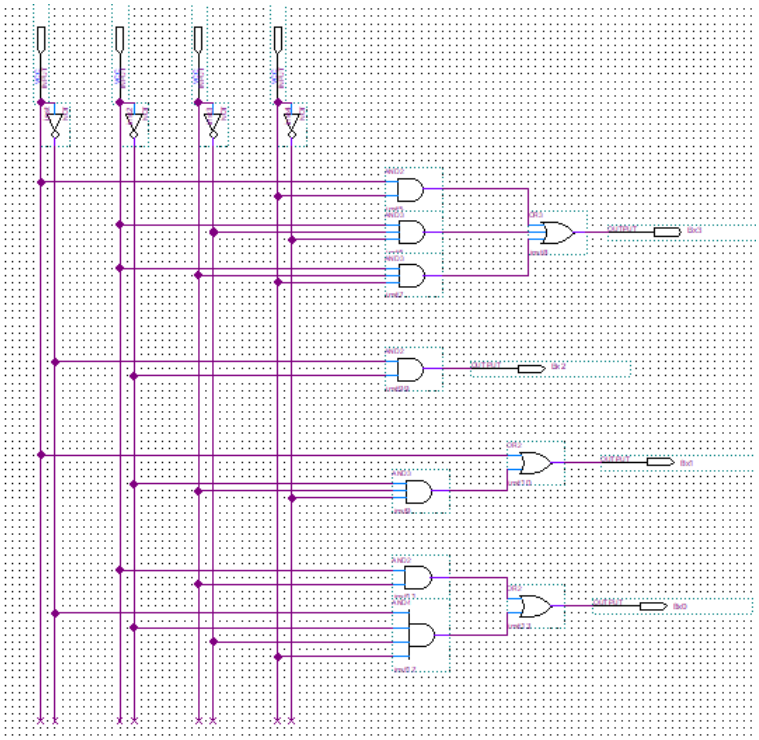
O circuito de estado atual é um simples banco de registradores feito a partir de flip-flops D.

Figura 10 – Circuito de Estado Atual



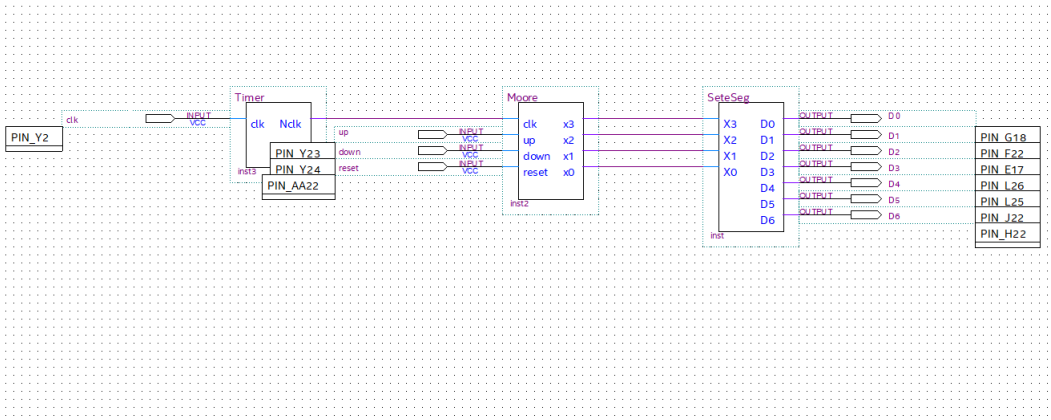
Utilizando o estado atual, decodificamos o estado em seu número correspondente utilizando o seguinte circuito:

Figura 11 – Circuito de Decodificação



Portanto, combinando todos os circuitos, conseguimos montar um circuito que realiza seu objetivo:

Figura 12 – Projeto Completo



5 Resultados Obtidos e Discussões

O circuito construído na placa FPGA funcionou perfeitamente como visto nas Waveforms abaixo:

Figura 13 – Waveform Up=1,Down=0

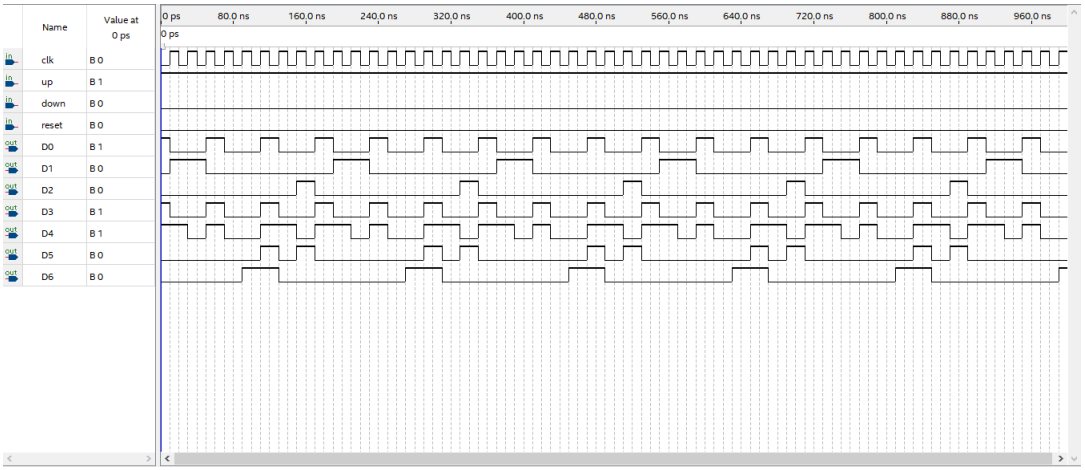


Figura 14 – Waveform Up=1,Down=0 com ativação do Reset

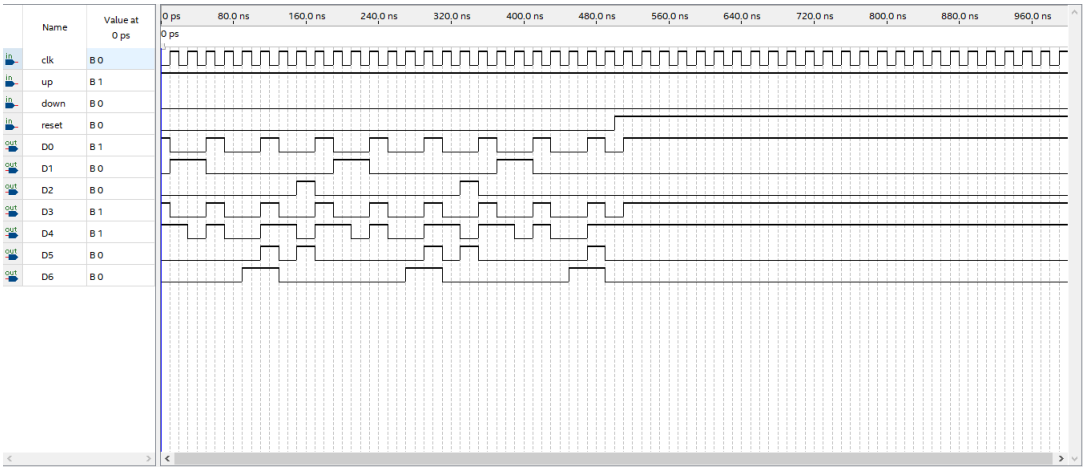


Figura 15 – Waveform Up=1,Down=0 com ativação do Down

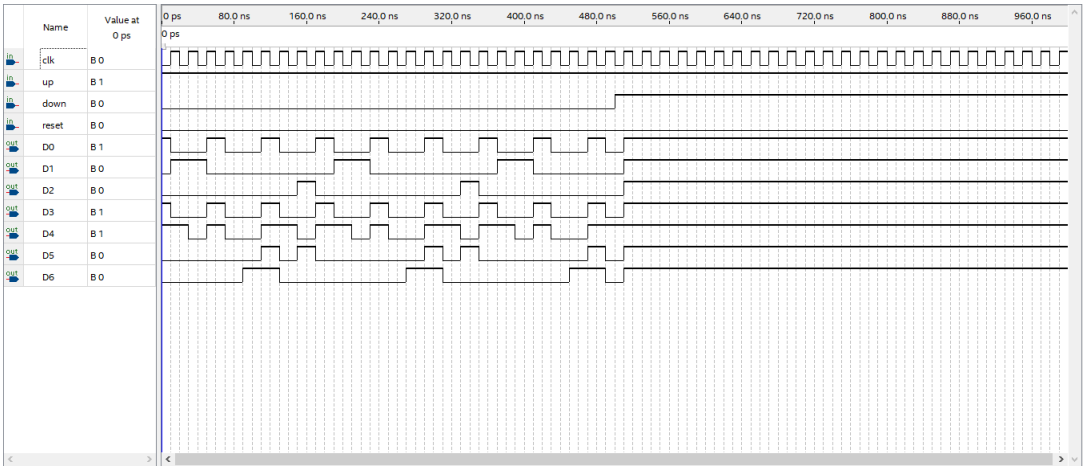


Figura 16 – Waveform Up=0,Down=1

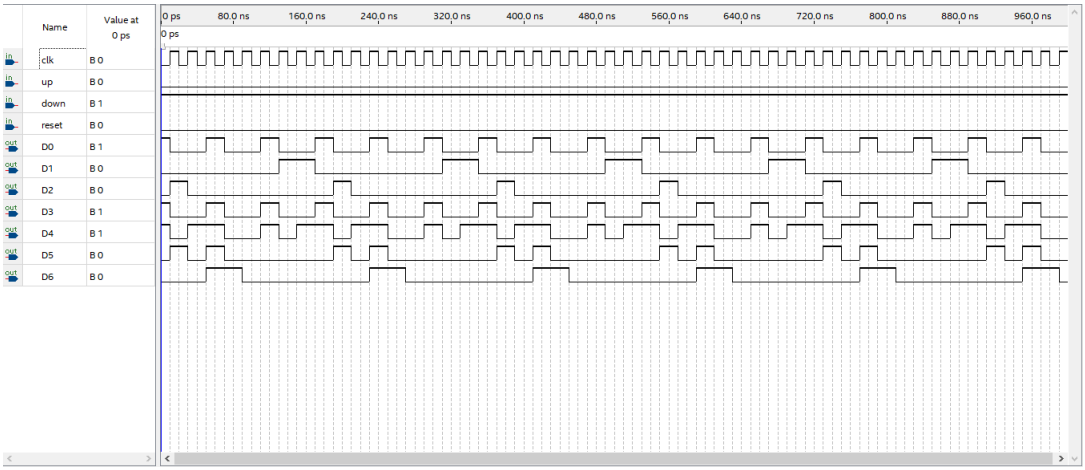


Figura 17 – Waveform Up=0,Down=1 com ativação do Reset

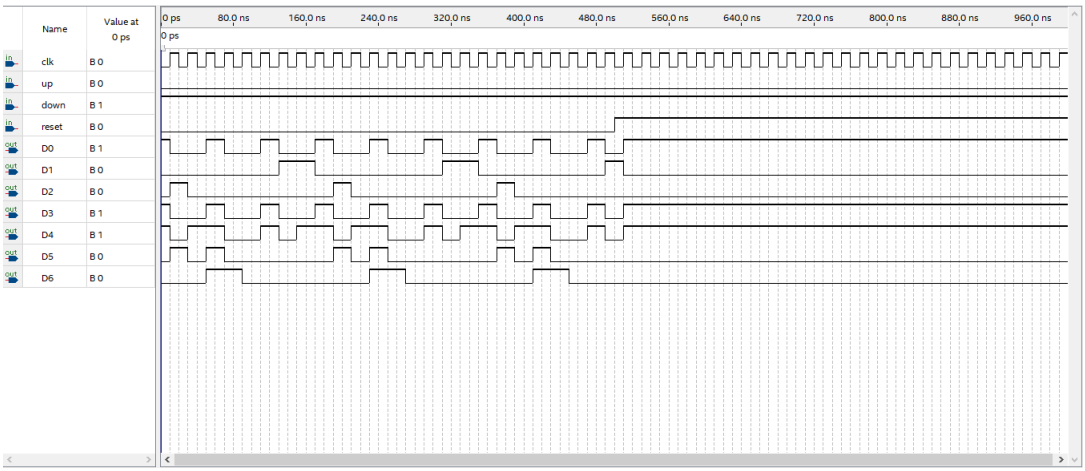


Figura 18 – Waveform Up=0,Down=1 com ativação do Up

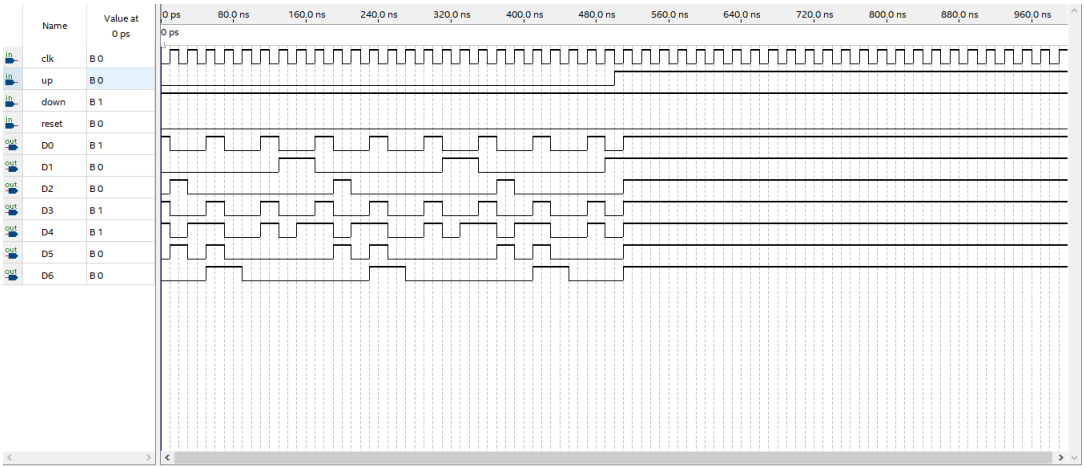
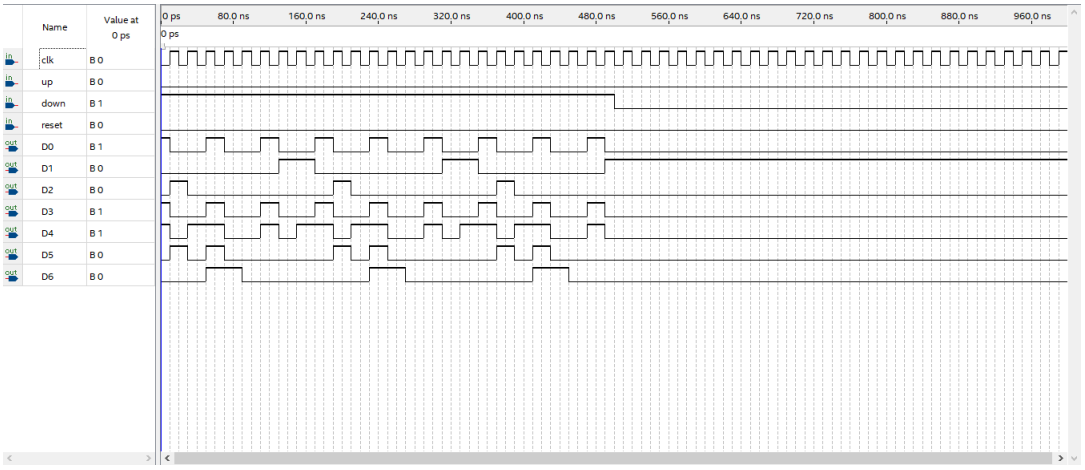


Figura 19 – Waveform Up=0,Down=1 com desativação do Down



Como podemos observar, todos os casos de teste satisfazem o proposto na atividade.

6 Considerações Finais

Apesar de o circuito ter funcionado, a implementação de circuitos digitais utilizando diagramas é muito imprática pois, além de tomar grandes proporções e gastar muito tempo para a realização de algo simples, qualquer erro pode ser muito difícil de ser achado em diagramas.

Portanto a implementação ideal de circuitos digitais é a partir de HDLs (Hardware Description Language), pois facilitam a portabilidade, rendimento, resolução de erros e simplicidade do circuitos.

A resolução de problemas complexos pode ser resolvida com circuitos relativamente simples porém extensos. A utilização de máquina de estados facilita muito a resolução desses problemas.

Referências

1 MOORE, E. F. Gedanken Experiments on Sequential Machines. In: *Automata Studies*. [S.l.]: Princeton U., 1956. p. 129–153. Citado na página 9.

2 PERLA, H. *Designing Finite State Machines (FSM) using Verilog*. 2006. Disponível em: <http://electrosofts.com/verilog/fsm.html>. Citado 2 vezes nas páginas 9 e 10.

3 MEALY, G. H. A method for synthesizing sequential circuits. *The Bell System Technical Journal*, v. 34, n. 5, p. 1045–1079, Sept 1955. ISSN 0005-8580. Citado na página 10.