RA: 136124

Contador Crescente e Decrescente Implementado por Máquina de Estados Finitos de Moore

São José dos Campos - Brasil

Outubro de 2019

RA: 136124

Contador Crescente e Decrescente Implementado por Máquina de Estados Finitos de Moore

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. 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 2019

Resumo

Este projeto tem como finalidade a implementação de uma Máquina de Estados Finito do tipo Moore que faça uma contagem não usual pré-determinada. Este processo pode ser feito em ordem crescente e decrescente e também possui funções de manter o número e apagar o display. Deseja-se contar com uma frequência de 1Hz que deve ser reproduzida a partir de um *clock* nativo de 50MHz produzido pela placa FPGA (*Field Programmable Gate Array*) usada no ambiente de testes. O número de saída deve ser decodificado para ser mostrado em um *display* de 7 segmentos e também possui uma função de *reset*, com a finalidade de começar a contagem novamente.

Palavras-chaves: Contador, Máquina de Moore, FPGA.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O estudo de circuito digitais é de suma importância no mundo atual em que vivemos, nessa era tecnológica, uma grande parte de tudo o que fazemos em nosso dia a dia tem relação com algum aparelho eletrônico. Apesar de todos que usam tais dispositivos saberem quais são suas funcionalidades, muitos não sabem de que forma essa função é executada. A lógica por trás de todas essas diretrizes, são feitas por pessoas que optaram por aprofundar seus conhecimentos nesta área ainda tão desconhecida para muitos. Portanto a experiência ao se obter com a pesquisa em circuitos digitais, é tamanha ao ponto de poder desenvolver projetos e dispositivos que podem revolucionar a sociedade.

Começando o estudo da base, e apreendendo como se da o funcionamento de circuitos lógicos pode levar as pessoas para o melhor entendimento dos aparelhos atuais, como por exemplo, entender como um temporizador de micro-ondas funciona. Visando ajudar e facilitar o mundo tecnológico que vivenciamos, os aparelhos digitais são de grande ajuda para a integralização das pessoas ao nosso redor.

Um contador numérico que não siga uma sequência conhecida pode ser um projeto implementado por uma Máquina de Estados Finitos, um conceito complexo porém útil para a projeção de problemas com alta dificuldade. Portanto, neste relatório implementaremos um contador cuja sequência aleatória será pré-definida, identificaremos todos os passos para sua construção e iremos testá-la em um ambiente apropriado.

2 Objetivos

2.1 Geral

Este projeto tem como finalidade produzir uma contagem numérica não usual pré-determinada (9 - 2 - 5 - 7 - 8 - 1 - 3 - 4 - 6) através de uma Máquina de Estados do tipo Moore, com um tempo de transição de um segundo, e com funções de crescente, decrescente, pausar a contagem e apagar o *display*.

2.2 Específico

- Projetar um divisor de frequência para o *clock* de 50MHz;
- Desenvolver um diagrama de estados e uma tabela para o auxilio do projeto da Máquina de Moore para o contador;
- Projetar o circuito da Máquina de Estados;
- Projetar o circuito de decodificação para o *display* de 7 segmentos;
- Combinar os circuitos produzidos;
- Obter e analisar os resultados desejados em laboratório.

3 Fundamentação Teórica

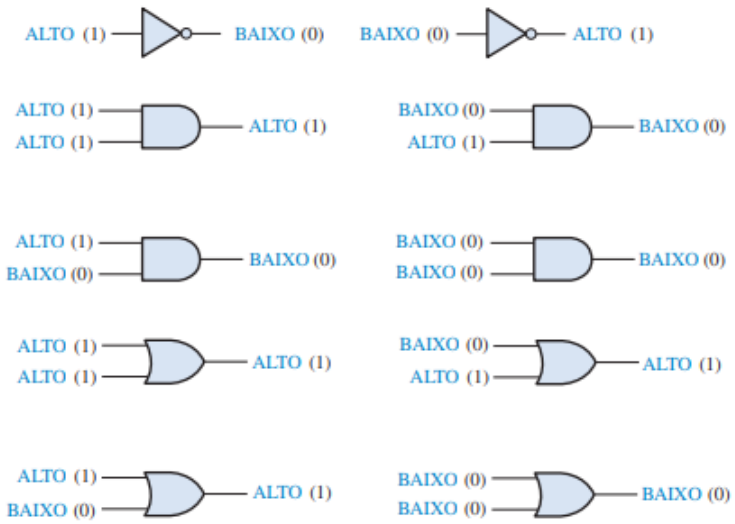
3.1 Elementos básicos de Circuitos Digitais

3.1.1 Portas Lógicas

Os circuitos digitais são compostos de diversos elementos que produzem a lógica necessária para que, o meio que o aplique, funcione corretamente. Podendo ser do tipo combinacional e sequencial, cada um possui suas características mas o que difere um do outro é que dados armazenados de saídas anteriores podem alterar as próximas.

As portas lógicas são as bases para todos os circuitos lógicos, são elas que fazem com que o nosso dispositivo funcione através de um fundamento. Como sabemos que muitos dos aparelhos utilizados funcionam com dados binários, ou seja, sinal alto e baixo, temos algumas operações que podemos fazer com os mesmos.

Figura 1 – Portas Lógicas



Fonte: Sistemas Digitais (1)

As três operações mais simples e mais usadas são a NOT, que faz com que o sinal de entrada se inverta na saída, a porta AND, que só possui sinal alto se ambas as entradas estejam em estado alto, e a OR, que tem sinal alto quando qualquer uma das duas possuem sinal alto. Podemos fazer um paralelo dessas lógicas com a matemática que conhecemos e então aplicá-las em uma álgebra chamada de Booleana, que nos ajudará a resolver e

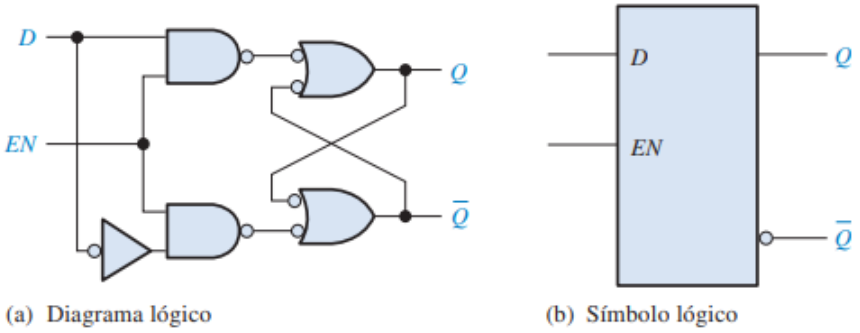
também simplificar circuitos muito complexos. A porta AND, indicada por $(\cdot)$, pode ser relacionada com uma multiplicação e a OR, indicada por $(+)$, é comparado a soma.

3.1.2 Elementos de Memória

Outro tipo de dispositivo muito usado são aqueles que armazenam valores, chamados de memória e também mas conhecidos como *Latches* e *Flip-Flops*. Ambos têm a capacidade de armazenar um bit que normalmente são usados para realimentar a entrada e formar os conhecidos circuitos sequenciais, que são aqueles que as saídas interferem na lógica inicial.

O *Latch* é o componente de memória mas simples entre os dois, o seu funcionamento é devido a um circuito combinacional compacto porém com uma capacidade de armazenar um determinado bit, e o seu funcionamento ocorre segunda a ativação de sua chave de *enable*. O mesmo pode ser encontrado em diversos tipos, onde os mais vistos são o SR, que possui modos de setar (definir sinal alto na saída) ou resetar (definir sinal baixo), e também o tipo D, que será explicado juntamente ao próximo elemento.

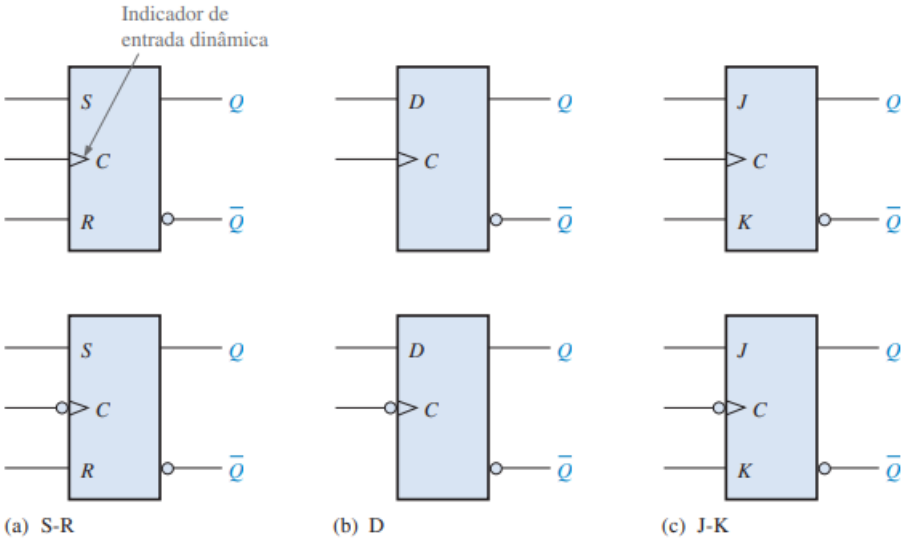
Figura 2 – *Latch* Tipo D



Fonte: Sistemas Digitais (1)

O mais conhecido e também mais usado são os *Flip-Flops*, que são parecidos com os *Latches*, porém seu funcionamento é sensível a uma entrada que chamamos de *clock*, que nada mais é uma onda que assume valores altos e baixos determinados por uma frequência que o mesmo possui. A sua ativação pode ser na subida (baixo para alto) ou na descida (alto para baixo) desses valores e além disso, os *Flip-Flops* existem de diversas funcionalidades. O do tipo D conduz a entrada para saída, ou seja, quando a entrada for 0 teremos esse valor na saída e o mesmo acontece para o bit 1, e também temos o do tipo SR que segue a mesma lógica do *Latch*. Já o JK, é mais completo, possuindo as funções de manter, setar e resetar, e também a de *Toggle* que também dá nome ao do tipo T, que nada mais é que a alternação do valor o mesmo possui.

Figura 3 – *Flip-Flops*

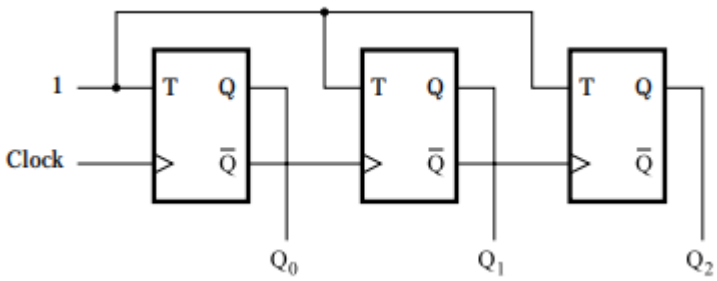


Fonte: Sistemas Digitais (1)

3.2 Divisor de Frequência

Esse circuito tem a finalidade de poder obter uma frequência menor a partir de um certo *clock* de entrada, e é montado usando elementos de memórias do tipo T (*Toggle*) em sequência. Seu funcionamento está ligado a definição de divisão, pois teremos que o próximo *Flip-Flop* terá a metade de frequência do anterior já que o mesmo só alterará seu valor após o antecedente fazê-lo duas vezes. Analisando-o podemos perceber que o mesmo é um contador binário, onde o bit mais significativo é o que possui o maior tempo de mudança de estado e é o que normalmente queremos em um divisor de frequência.

Figura 4 – Divisor de Frequência



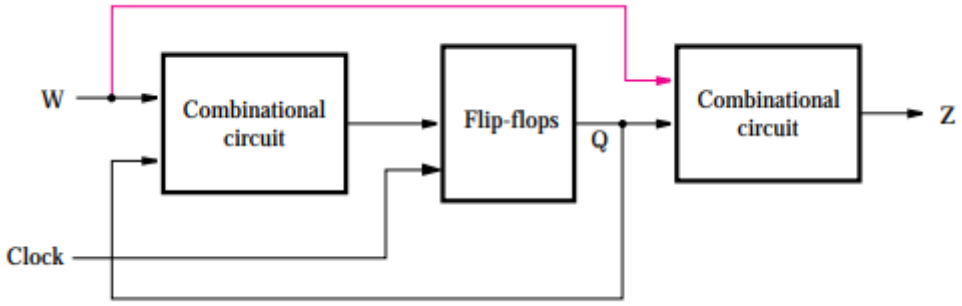
Fonte: Fundamentals of Digital Logic with VHDL Desgin (2)

3.3 Máquina de Estados Finitos

A Máquina de Estados Finitos é um circuito sequencial em que consiste em simplificar um determinado problema ou algoritmo através da notação de estados que são implementados por *Flip-Flops*, a fim de proporcionar que a máquina esteja em apenas um por determinado valor de *clock* e que se altere apenas em sua subida ou descida. Podendo ser do tipo Moore ou de Mealy, a única diferença entre eles é que na segunda a entrada também influencia o valor de saída.

A máquina é formada por um circuito combinacional de entrada, que definirá o próximo estado, um conjunto de *Flip-Flops* para armazenar o atual, e outro circuito combinacional que diz a lógica da saída que pode ou não depender da entrada.

Figura 5 – Máquina de Estados Finitos



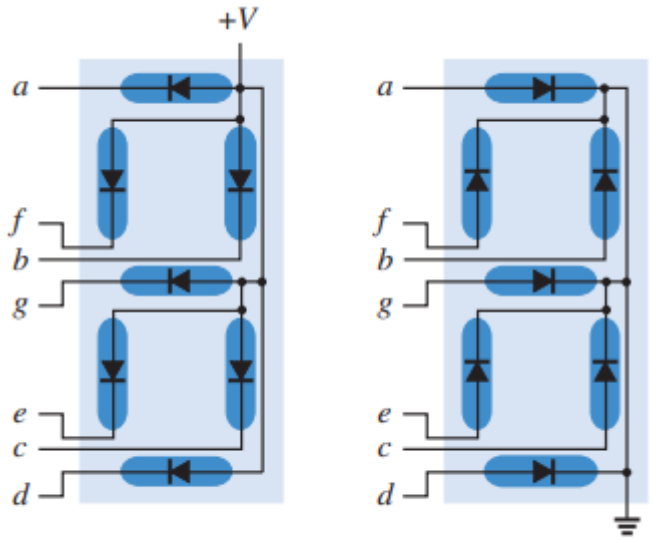
Fonte: Fundamentals of Digital Logic with VHDL Desgin (2)

Ao longo do desenvolvimento do projeto devemos ser capazes de identificar cada passo para a implementação de uma Máquina de Estados.

3.4 *Display* de 7 segmentos

O *Display* de 7 segmentos é conjunto de LEDs que juntos podem formar números e letras de acordo com uma certa entrada. Possui 7 segmentos e 1 ponto, o que faz com que a variedade de informação a ser projetada seja grande. O display pode ser do tipo cátodo ou ânodo comum, o que significa que ou os LEDs estão com os terminais positivos em comum ou então os negativos, portanto para cada tipo, o valor de entrada para ligá-los é diferente.

Figura 6 – *Display* de 7 segmentos



(a) Anodo comum (b) Catodo comum

Fonte: Sistemas Digitais (1)

4 Desenvolvimento

A proposta do projeto é construir um contador numérico que seguisse uma certa sequência designada para cada aluno usando uma Máquina de Estados Finitos com um tipo também pré-determinado desenvolvido no software Quartus Prime, e que tivesse funções de crescente e decrescente, *reset*, pausar a contagem e apagar o *Display* e então ser apresentada em aula em um FPGA da Altera modelo DE2-115.

A sequência a ser percorrida é 9 - 2 - 5 - 7 - 8 - 1 - 3 - 4 - 6 e o tipo de Máquina de Estados será Moore. Para melhor compreensão de sua projeção, foi dividida em três subproblemas: Projetar um divisor de frequência para a contagem de 1 segundo, projetar um conjunto de circuitos necessários da Máquina de Estados e projetar um decodificador BCD para o *Display* de 7 segmentos.

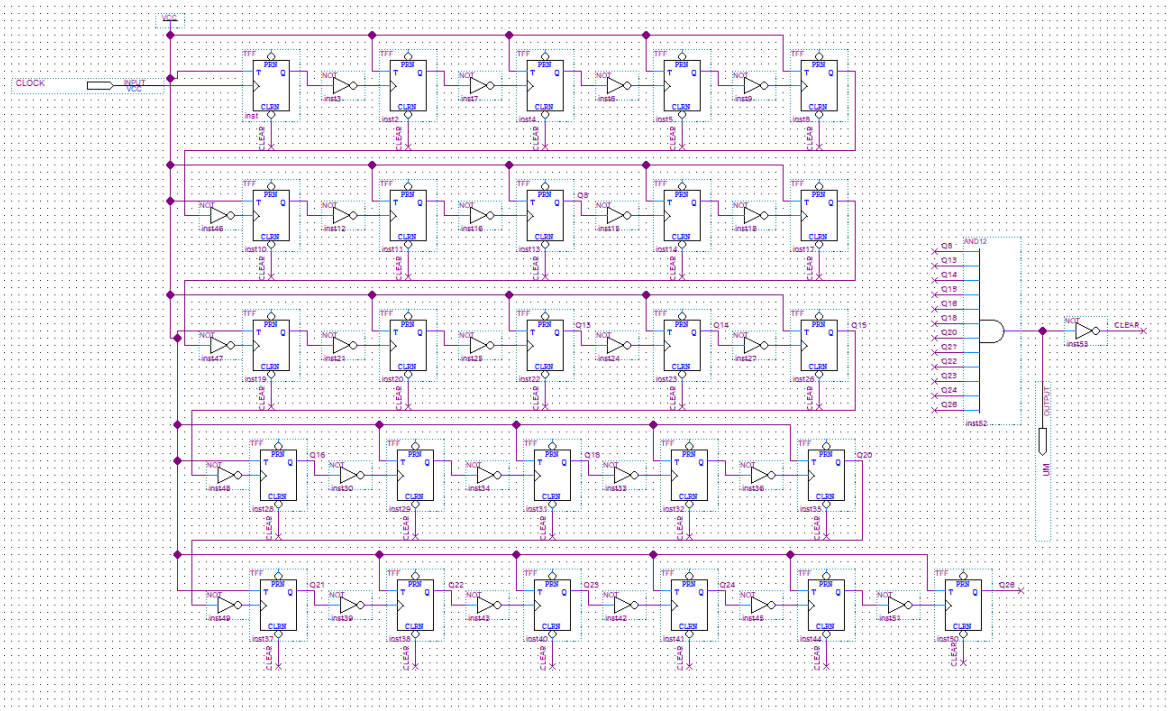
4.1 Divisor de Frequência

A contagem deve ser alterada a cada segundo, portanto devemos produzir um *clock* de 1Hz para termos um período de um segundo. Como o FPGA possui um *clock* nativo de 50MHz e a cada *Flip-Flop* teremos a metade da frequência do anterior, pela Equação 4.1 vemos que necessitaremos de 26 unidades de memórias.

$$T = \frac{1}{F} \Rightarrow 1 = \frac{2^n}{50000000} \Rightarrow \log_2 50000000 = n \Rightarrow n \approx 26 \tag{4.1}$$

Para atingir um tempo ainda mais preciso de 1 segundo, além de apenas cascatear os *Flip-Flops*, quando tivermos 50 milhões (10111110101111000010000000 em binário) no número deste contador é quando teremos uma precisão maior. Logo se fizermos uma operação *AND* com todos os uns de seus respectivos elementos de memórias obtemos o circuito da Figura 7. Além de precisarmos dar *clear*, ou seja, resetar todos os *Flip-Flops* para que a contagem até 1 segundo recomece.

Figura 7 – Divisor de Frequência do projeto



4.2 Decodificador BCD para *Display* de 7 segmentos

Para esse circuito, devemos projetar um que satisfaça a projeção de números de 0 a 9 no *Display* de 7 segmentos. Em nosso ambiente de teste, o FPGA possui *Displays* do tipo ânodo comum, vide Figura 6, logo para que o led acenda devemos colocar valor baixo na entrada. Com isso mente montamos então a Tabela 1.

Tabela 1 – Saídas para o *Display* de 7 segmentos

ENTRADAS				SAÍDAS PARA O DISPLAY DE 7 SEGMENTOS (ÂNODO COMUM)							
W	X	Y	Z	a	b	c	d	e	f	g	NÚMERO
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**

Com a tabela produzida, podemos usar o site 32x8.com (3) para gerar com maior facilidade o nosso circuito combinacional de decodificação. As expressões da Figura 8 representam um *Display* do tipo cátodo comum, logo necessitamos negar as saídas para usá-las como ânodo comum, as mesmas são expressas em funções das entradas e suas saídas

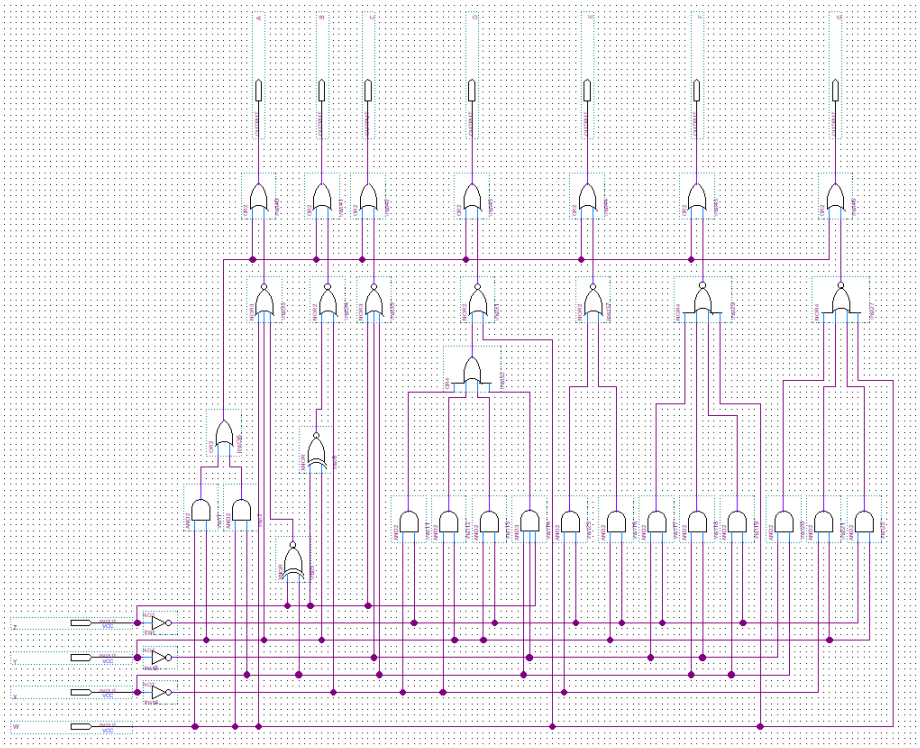
são o valor de cada bit a ser colocado no display. As operações ANDs são representadas pelas entradas juntas, os ORs por (+) e a negação pelo (') após a letra.

Figura 8 – Expressões para o decodificador

```
a = Y + W + X´Z´ + XZ
b = X´+ Y´Z´ + YZ
c = Y´+ Z + X
d = W + X´Z´ + X´Y + YZ´ + XY´
e = X´Z´ + YZ´
f = Z + Y´Z´ + XY´ + XZ´
g = Z + Y´X + YX´ + YZ´
```

Com isso em mente geramos o circuito junto a um habilitador para que números maiores que 9 deixem o *Display* apagado, obtendo a Figura 9,

Figura 9 – Decodificador BCD para o *Display* de 7 segmentos



4.3 Máquina de Estados

Uma Máquina de Estados Finitos de Moore pode ser dividada em três pequenos setores de circuitos, a primeira contendo um circuito combinacional de transição, a segunda um banco de registradores para armazenar os bits necessários para a representação dos estados do problema, e por fim um circuito combinacional de saída. Podemos ver pela Figura 5 que a Máquina de Estados é um circuito sequencial pois a sua saída interfere diretamente na entrada.

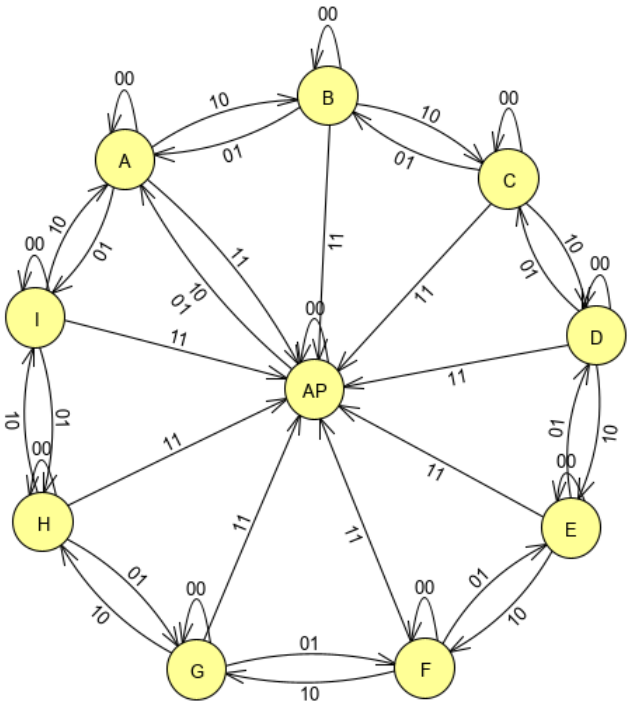
Para este projeto, temos quatro opções de interações a cada número que estão representadas na Tabela 2, e podem ser selecionados pelas chaves de entradas UP (U) e DOWN (D).

Tabela 2 – Entradas para o contador

ENTRADAS		
U	D	SAÍDA
0	0	MANTER
0	1	DECRESCER
1	0	CRESCER
1	1	APAGAR

Portanto como primeiro passo, devemos construir um diagrama de estados para que possamos compreender e conseguir analisar todas as utilidades do projeto. Representaremos cada saída possível em estados, identificando-os por letras e siglas. Os números da contagem foram apresentados de forma alfabética em sequência, logo o número 9, que é o primeiro, assume a letra A, o 2 assume a letra B, e assim por diante até o último número da sequência 6 como I. O *blank*, que siginifica o *display* apagado será AP. As transições são mostradas por setas que ligam os estados e os números acima delas são as entradas UP e DOWN que interagem com o contador. Com essas informações reunidas e usando o software chamado de JFLAP (4) podemos gerar o diagrama da Figura 10.

Figura 10 – Diagrama de estados



Após a formação do diagrama de estados, temos que gerar uma tabela que abranja

todas as possibilidades de transições para que possamos gerar o circuito que nos dirá o próximo estado a ser tomado, assim como também a saída que o display deve apresentar em função do estados atual. Então, usando a Tabela 3 devemos conseguir atribuir uma expressão para os bits que representam os próximos estados, identificados como E3' (mais significativo), E2', E1', E0' (menos significativo), e também equações para cada bit de saída representado por S3 até S0, seguindo o mesmo padrão de significância dos de próximo estado. Note que as (') na Tabela 3 para os bits de transição não representam negação, e sim uma diferenciação do estado atual.

Tabela 3 – Tabela para circuito de transição e de saída

	ESTADO ATUAL				PRÓXIMO ESTADO (E3´, E2´, E1´, E0´)				SAÍDA			
ESTADO	E3	E2	E1	E0	U,D = 0,0	U,D = 0,1	U,D = 1,0	U,D = 1,1	S3	S2	S1	S0
A	0	0	0	0	0,0,0,0	1,0,0,0	0,0,0,1	1,0,0,1	1	0	0	1
B	0	0	0	1	0,0,0,1	0,0,0,0	0,0,1,0	1,0,0,1	0	0	1	0
C	0	0	1	0	0,0,1,0	0,0,0,1	0,0,1,1	1,0,0,1	0	1	0	1
D	0	0	1	1	0,0,1,1	0,0,1,0	0,1,0,0	1,0,0,1	0	1	1	1
E	0	1	0	0	0,1,0,0	0,0,1,1	0,1,0,1	1,0,0,1	1	0	0	0
F	0	1	0	1	0,1,0,1	0,1,0,0	0,1,1,0	1,0,0,1	0	0	0	1
G	0	1	1	0	0,1,1,0	0,1,0,1	0,1,1,1	1,0,0,1	0	1	0	0
H	0	1	1	1	0,1,1,1	0,1,1,0	1,0,0,0	1,0,0,1	0	0	1	1
I	1	0	0	0	1,0,0,0	0,1,1,1	0,0,0,0	1,0,0,1	0	1	1	0
AP	1	0	0	1	1,0,0,1	0,0,0,0	0,0,0,0	1,0,0,1	1	1	1	1

Para gerar essas expressões, usaremos novamente o site 32x8.com (3), para melhor aproveitamento do mesmo. Note que para a representação das expressões as (') representaram negação e os bits de transição estão identificados como A, para o mais significativo, até D para o menos, e os bits E3 até E0 representam os estados atuais. Para melhor compreensão do dados da tabela as operações de AND são ( ). Com isso em mente obtemos as expressões da Figura 11.

Figura 11 – Expressões para os bits de transições e de saída

```
A = U D + E3 U' D' + E2 E1 E0 U + E3' E2' E1' E0' D
B = E2 E1' D' + E2 E0' D' + E2 E0 U' + E2 E1 U' + E3 E0' U' D + E2' E1 E0 U D'
C = E1 E0' D' + E1 E0 U' + E3 E0' U' D + E3' E1' E0 U D' + E2 E1' E0' U' D
D = U D + E3' E0' U + E0 U' D' + E1 E0' D + E2 E0' D + E3 E0' D

S3 = E3 E0 + E3' E1' E0'
S2 = E3 + E2' E1 + E1 E0'
S1 = E3 + E2 E0 + E1 E0
S0 = E2' E1 + E2 E0 + E3 E0 + E3' E2' E0'
```

Com essas expressões devemos então reproduzi-las no esquemático e fazer um circuito de transição e de saída. Para os bits 3 e 2 de próximo estado temos a Figura 12, para o 1 e 0 temos a Figura 13 e por fim para o circuito de saída temos a Figura 14.

Figura 12 – Circuito combinacional para os bits 3 e 2 de próximo estado

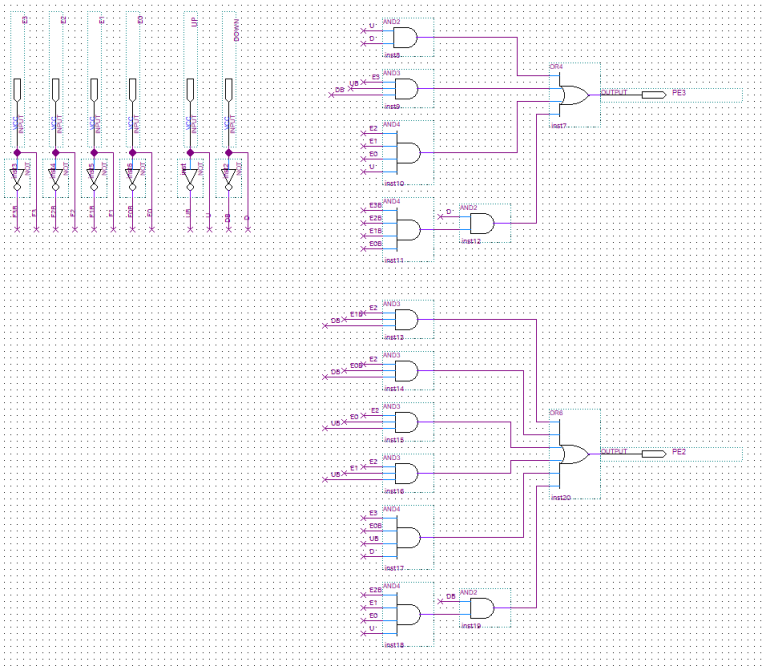


Figura 13 – Circuito combinacional para os bits 1 e 0 de próximo estado

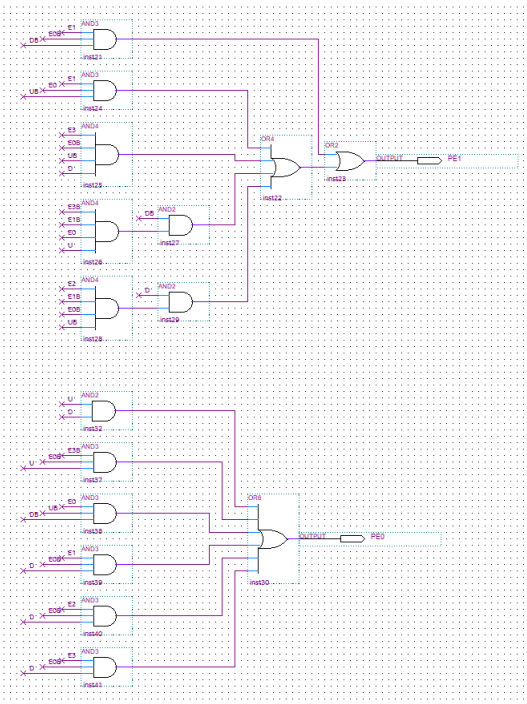
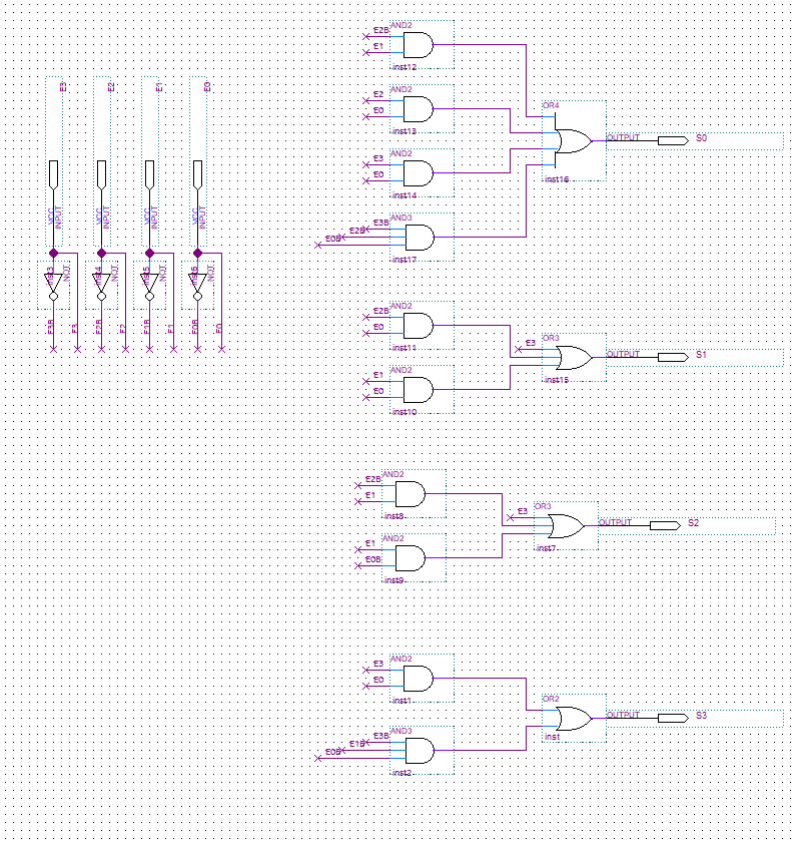
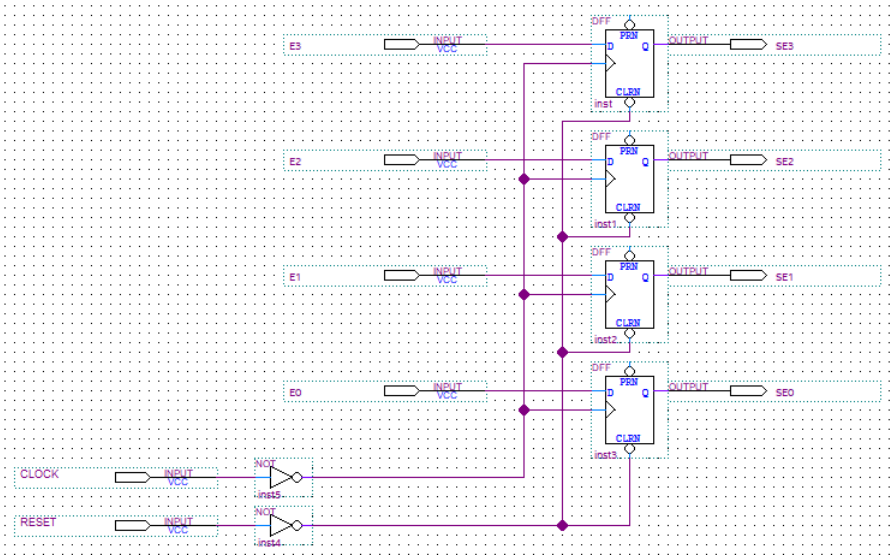


Figura 14 – Circuito combinacional de saída



O único setor restante da Máquina de Estados é o banco de resgistrados usados para armazenar o estados atual, como temos 4 bits de estados necessitaremos de 4 *Flip-Flops* do tipo D. Esses elementos de memória terão o *clock* de 1 segundo que produzimos anteriormente e também a chave de *reset* que ligará no *clear* de cada bit que fará com que voltemos para o estado inicial 0000, ou seja, para o primeiro número da sequência, produzindo então o circuito da Figura 15.

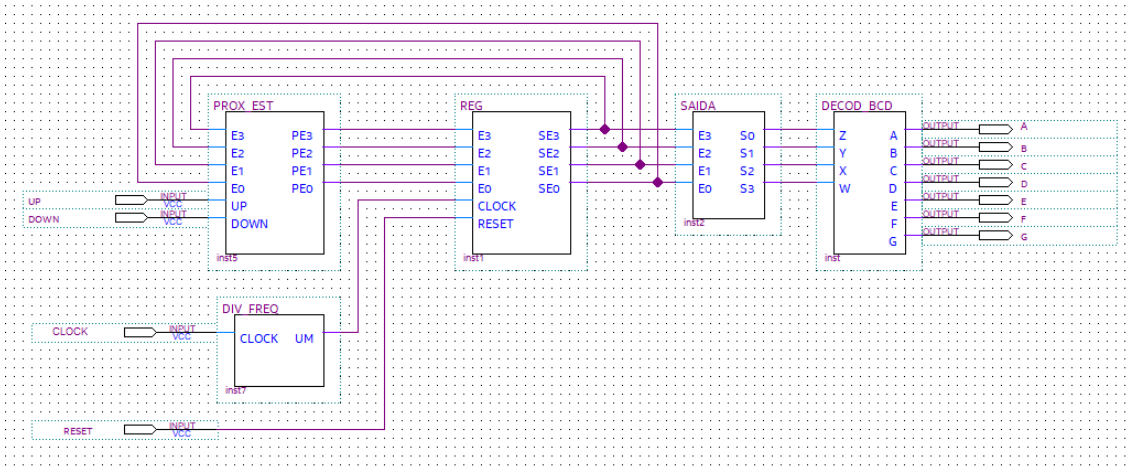
Figura 15 – Circuito de estado atual



4.4 Circuito Final

Com todos os subproblemas propostos resolvidos, tudo o que resta é a junção deles. As entradas U e D entrarão apenas no circuito combinacional de transição de estado que consequentemente se conectará com o banco de registradores junto ao *clock* e o *reset*. A saída do circuito de estado atual realimentará a transição e entrará no circuito de saída da Máquina de Estados, que ligaremos no Decodificador BCD para o *Display* para a apresentação dos resultados.

Figura 16 – Circuito final do contador



5 Resultados Obtidos e Discussões

Algumas etapas durante o projeto foram testados no ambiente virtual oferecido pela UNIFESP, e em sua maioria também foram muito útil para não ter que estar presente no laboratório para testá-los e normalmente eles funcionavam como previsto. Apenas o circuito final que houve um imprevisto, onde ao tentar manipulá-lo no FPGA online, tudo estava funcionando corretamente exceto a interação onde as entradas eram 00, que obteríamos como resultado o número mantido no *Display*, porém ele voltava a mostrar o número inicial da contagem.

Apesar do problema com relação ao ambiente remoto, todos os testes reproduzidos no laboratório físico foram um sucesso. O contador implementado por Máquina de Estados Finitos de Moore funcionou corretamente, contando com um tempo de 1 segundo e demonstrando seguir as sequências tanto na interação de crescer e decrescer, consequentemente o decodificador para o *Display* estava funcionando. A partir de qualquer número poderíamos voltar para o anterior, avançar para o próximo, mantê-lo ou então apagar o *Display*. Quando estávamos com o mesmo apagado, poderíamos tanto colocar a entrada em 11 ou 00 que mantinha-no, e se as chaves de UP e DOWN fossem 10 ou 01 retornaríamos para o início da contagem seguindo o planejado para o projeto.

6 Considerações Finais

Com a conclusão do projeto e os teste em laboratório, podemos observar que o conceito de Máquina de Estados é muito útil, podendo resolver alguns problemas que possuem maior dificuldade, porém sua compreensão e implementação podem ser complexas devido a grande quantidade de informações a serem reproduzidas em formas de circuito lógicos extensos. Em relação ao FPGA em questão, podíamos usar dois tipos de laboratórios e com isso tivemos uma maior interação com o projeto, deixando o mesmo mais compreensível já que não teríamos que apenas produzi-lo no software, que é algo meio abstrato em comparação com a realidade.

Referências

1 FLOYD, T. L. *Sistemas Digitais: Fundamentos e Aplicações*. 9ª edicao. ed. Porto Alegre: Bookman, 2007. Citado 4 vezes nas páginas 11, 12, 13 e 15.

2 BROWN, S. D.; VRANESIC, Z. *Fundamentals of Digital Logic with VHDL Desgin*. 3rd edition. ed. New York: McGraw-Hill, 2009. Citado 2 vezes nas páginas 13 e 14.

3 ONLINE Karnaugh map solver with circuit for up to 8 variables. 2014. Disponível em: <32x8.com>. Citado 2 vezes nas páginas 18 e 21.

4 JFLAP 7.1. 2018. Disponível em: <http://www.jflap.org>. Citado na página 20.