Laboratório de Sistemas Computacionais: Circuitos Digitais

Projeto 04 - Máquina de Estados Finitos

Aluna: Thauany Moedano

Universidade Federal de São Paulo - UNIFESP
Instituto de Ciência e Tecnologia, São José dos Campos

1. Introdução

Projetos de circuitos digitais são extremamente importantes para a implementação de processadores, hardwares, circuitos integrados e tecnologias com sistemas digitais em geral. Desde um simples relógio digital a supercomputadores, o primeiro passo é pensar no projeto lógico do circuito a partir de suas tabelas verdade, simplificação até a construção do circuito em si. Este trabalho apresenta a construção e implementação de um circuito digitial que mostra em um display a sequência **4-2-5-2-0-1-4-3-5** em ordem crescente, ordem descrecente, capaz de manter um único número aparecendo no display e também mostrar o display apagado de acordo com as entradas.

A sessão 2 disserta sobre a fundamentação teórica utilizada para desenvolver o trabalho. A sessão 3 apresenta o desenvolvimento, simplificação das expressões lógicas e implementação do circuito. A próxima sessão discute os resultados obtidos com a implementação do circuito.

2. Fundamentação Teórica

Esta sessão aborda os principais conceitos utilizados para projetar o circuito lógico. A primeira fase do projeto tem como objetivo montar a tabela verdade das expressões e simplificá-las por meio de mapas de Karnaugh. A segunda fase consiste na construção da máquina de estados utilizando circuitos sequenciais. A terceira fase do projeto conecta a máquina de estados a um decodificador para a saída em um display de sete segmentos e utiliza um temporizador como clock automático. Por fim, o funcionamento do circuito é testado em uma placa FPGA.

2.1. Tabela verdade e Mapas de Karnaugh

A maneira mais simples de encontrar expressões para um circuito digital é utilizando a tabela verdade. A tabela verdade consiste em um arranjo de entradas e saídas do programa no qual se preenche com os binários zero e um.

O Mapa de Karnaugh foi proposto em 1953 por Maurice Karnaugh como uma forma de simplificar expressões sem fazer muitos cálculos, representando graficamente a tabela verdade. [Karnaugh 1953]

O Mapa de Karnaught é um diagrama com quadrados onde cada quadrado representa um mini termo. As variáveis são dispostas no diagrama de acordo com os princípios do código de Gray onde apenas uma variável muda em quadrados adjacentes. [Hung 2011] Cada linha da tabela verdade é um quadrado do Mapa de Karnaugh que será preenchido com zeros, uns ou X (*Don't Care*). Uma condição *Don't Care* pode surgir por várias razões: A mais comum é a existência de algumas situações nas quais certas combinações de entrada não podem nunca ocorrer e portanto, não existem saída para

estas condições [Tocci et al. 2003]. Após preencher o Mapa de Karnaugh é necessário agrupar os termos com 1s adjacentes em grupos pares (2, 4 ou 8 dependendo do tamanho do Mapa). Os termos X (*Don't Care*) podem ser agrupados ou não. A escolha depende se o agrupamento de Xs irá ajudar na simplificação.

Após agrupar os termos, a variável que aparece nas formas complementada e não-complementada dentro de um grupo deve ser eliminada da expressão. Variáveis que não mudam para todos os quadrados devem aparecer na expressão final. [Tocci et al. 2003]

Mapas de Karnaugh com mais de quatro variáveis são complexos e foram feitos com ajuda de um programa online [map]

2.2. Flip Flops

Em determinados projetos de circuitos lógicos, como a Máquina de Estados Finitos, é necessário armazenar informação para que o circuito funcione adequadamente. Flip Flops são estruturas de memórias em circuitos digitais capazes de armazenar um bit. Os Flip Flops operam de modo síncrono, ou seja: seu estado é atualizado somente na transição do clock que pode ser tanto na borda de subida ou descida.

Existem vários tipos de Flip Flop mas este trabalho irá apresentar apenas o Flip Flop D (FFD). O FFD possui apenas uma entrada síncrona, D. A saída corresponde a entrada que ocorrer em D: Se D estiver em zero, a saída irá para zero na transição de clock. Se D estiver em um, a saída irá para um na transição de clock.

Na maioria das aplicações do FFD, a saída deve assumir os valores da entrada em determinados instantes de tempo precisamente definidios [Tocci et al. 2003]. É o que ocorrerá com a Máquina de Estados Finitos que será apresentada posteriormente.

2.3. Máquinas de Estados Finitos

Uma Máquina de Estados Finitos é um circuito lógico que pode ser divididos em estados, como o próprio nome sugere. A saída de cada estado é definida por um estado anterior (no caso de uma Máquina de Moore) e também pelas entradas do circuito (No caso da máquina de Mealy). Portanto, a máquina de Mealy tem os estados atualizados imediatamente quando as entradas mudam ao contrário da Máquina de Morre que só faz transições a cada ciclo de clock.

O primeiro passo para a construção de uma Máquina de Estados Finitos é o diagrama de estados. No diagrama de estados, os estados são representados indicando qual a saída correspondente e para qual estado deve-se ir de acordo com as entradas. Desse diagrama de estados obtem-se a tabela verdade do projeto lógico que deve conter as seguintes informações: O estado atual, o próximo estado, as entradas e a saída correspondente.

Deve-se notar que as combinações de bits não utilizadas podem ser marcados como *Don't Care* para facilitar as simplificações via mapa de Karnaugh.

Cada bit de saída deve ser armazenado em um flip flop D. Portanto, o número de flip flops corresponde ao número de bits da saída da Máquina de Estados. Por exemplo, neste trabalho usou-se quatro Flip Flops D pois a saída tinha quatro bits.

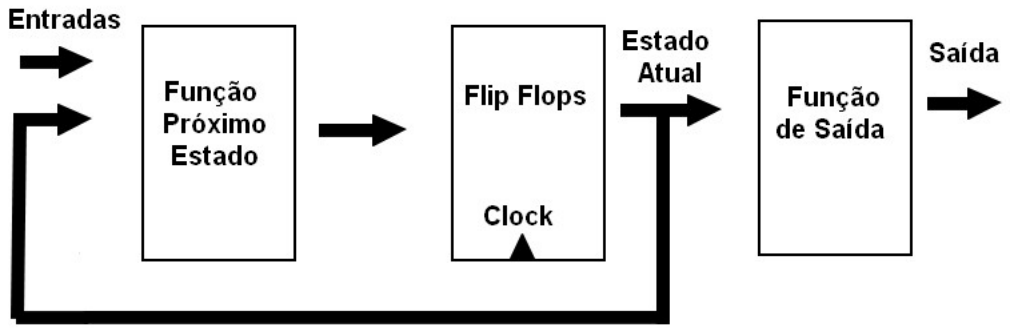

Figura 1. Esquemático de uma Máquina de Estados Finitos

A Máquina de Estados Finitos pode ter opcionalmente uma função de reset. Esta função deve levar imediatamente ao estado inicial. A implementação da funçãor reset depende de como o seu estado inicial está representado. Por exemplo, se o estado inicial é representado pelos bits 00, o reset deve ser capaz de forçar os flip flops a saírem 00. É simples implementar um reset: basta utilizar portas AND e OR entre o reset e a entrada D do flip flop de acordo a saída desejada.

2.4. Temporizador e Display de Sete Segmentos

O clock da Máquina de Estados Finitos pode ser controlado a partir de um temporizador.

2.5. Testes em FPGA

3. Desenvolvimento

O primeiro passo para desenvoler o projeto é desenhar o diagrama de estados de acordo com as entradas. As entradas são **Up** e **Down** e ditam o que devem acontecer com cada estado seguindo a tabela:

Up	Down	Contagem
1	0	Crescente
0	1	Descrecente
0	0	Mantém
1	1	Blank

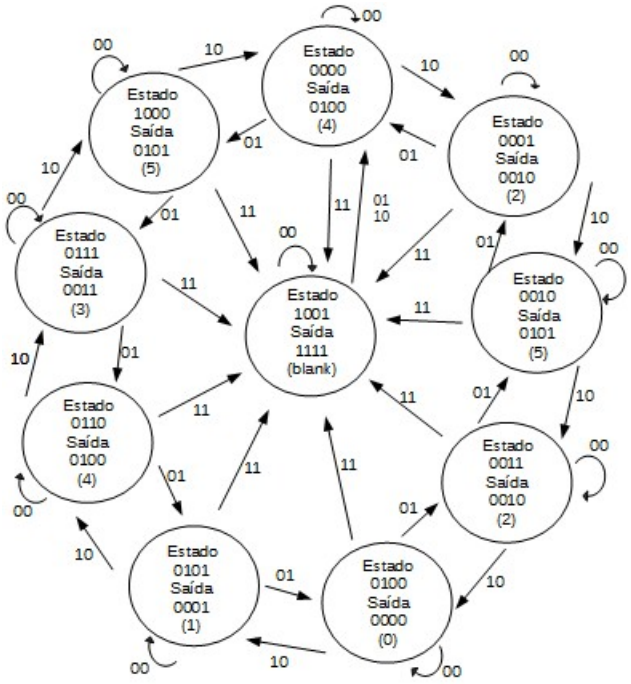

Figura 2. Diagrama de Estados

Note que o estado blank representa um estado inválido das entradas, ou seja: o display deve aparecer aparecer apagado quando se manter nesse estado. Isto foi representado com a saída 1111 no display cujo saída será todos os leds apagados.

O próximo passo é construir a tabela verdade de acordo com as entradas e saída anotadass no diagrama de estados. Os casos de combinação que não foram utilizados no diagrama foram tratados como *Don't Care*

As expressões foram obtidas com ajuda de um programa online [map]. Entradas do Estado Atual foram chamadas de Q_0 Q_1 Q_2 Q_3 e Q_4. As saídas do Próximo Estado foram chamadas de D_0 D_1 D_2 D_3 D_4. As saídas para o display foram chamadas de S_0 S_1 S_2 S_3

Função de próximo estado:

$$D_0 = UP.DOWN + Q_0\overline{UPDOWN} + Q_1Q_2Q_3UP + \overline{Q_0Q_1Q_2Q_3DOWN}$$

$$D_1 = Q_1\overline{Q_2DOWN} + Q_1\overline{Q_3DOWN} + Q_1Q_3\overline{UP} + Q_1Q_2\overline{UP} + Q_0\overline{Q_3UP}DOWN + \overline{Q_1}Q_2Q_3UP\overline{DOWN}$$

$$D_2 = Q_2\overline{Q_3DOWN} + Q_2Q_3\overline{UP} + Q_0\overline{Q_3UP}DOWN + \overline{Q_0Q_2}Q_3UP\overline{DOWN} + \overline{Q_0Q_2}Q_3UP\overline{DOWN} + Q_1\overline{Q_2Q_3UP}DOWN$$

$$D_3 = UPDOWN + \overline{Q_0Q_3}UP + Q_3\overline{UPDOWN} + Q_2\overline{Q_3}UP + Q_1\overline{Q_3}DOWN + Q_0\overline{Q_3}UP$$

Tabela 1. Tabela verdade do diagrama de estados

Estado Atual	Up—Down	Próximo Estado	Saída
0000	00	0000	0100
0000	01	1000	0100
0000	10	0001	0100
0000	11	1001	0100
0001	00	0001	0010
0001	01	0000	0010
0001	10	0010	0010
0001	11	1001	0010
0010	00	0010	0101
0010	01	0001	0101
0010	10	0011	0101
0010	11	1001	0101
0011	00	0011	0010
0011	01	0100	0010
0011	10	0100	0010
0011	11	1001	0010
0100	00	0100	0000
0100	01	0011	0000
0100	10	0101	0000
0100	11	1001	0000
0101	00	0101	0001
0101	01	0100	0001
0101	10	0110	0001
0101	11	1001	0001
0110	00	0110	0100
0110	01	0101	0100
0110	10	0111	0100
0110	11	1001	0100
0111	00	0111	0011
0111	01	0101	0011
0111	10	1000	0011
0111	11	1001	0011
1000	00	1000	0101
1000	01	0111	0101
1000	10	1001	0101
1000	11	1111	0101
1001	00	1001	1111
1001	01	0000	1111
1001	10	0000	1111
1001	11	1001	1111

A próxima tabela indica todos os casos de combinações tratados de acordo com o diagrama de estados:

O circuito foi projetado utilizando o software *Quartus II*. Utilizou-se um esquemático AND-OR para projetar as entradas e as saídas:

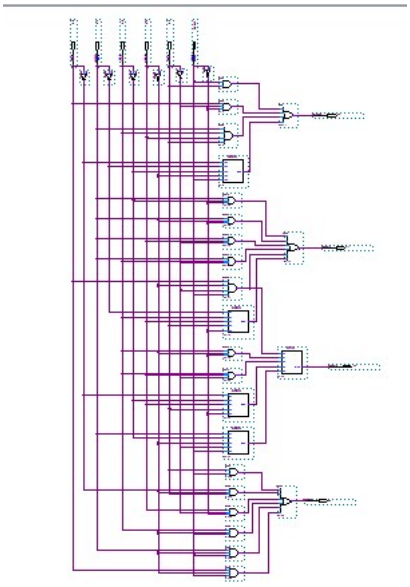

Figura 3. Esquemático AND-OR da função de próximo estado

A corretude do circuito pode ser observada na forma de onda gerada pelo circuito:

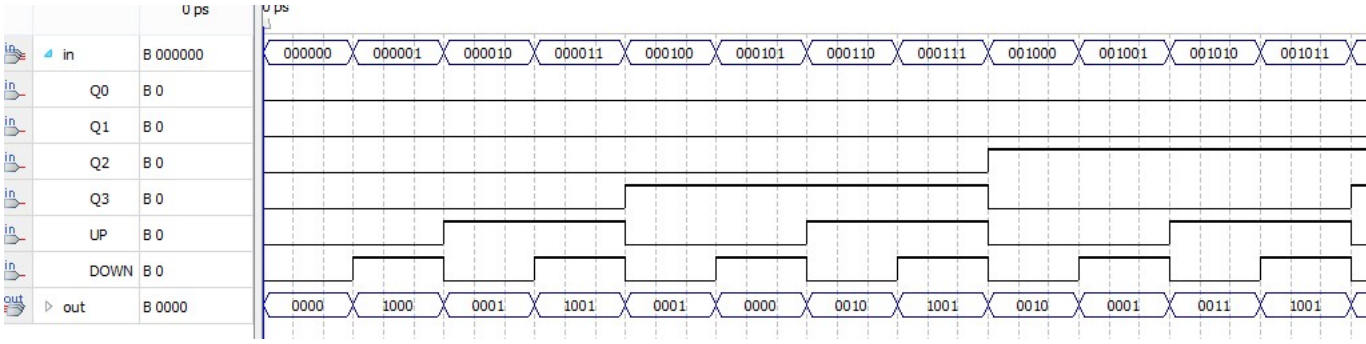

Figura 4. Forma de onda gerada pela função de próximo estado

As saídas da função de próximo de estado devem ser armazenadas e realimentadas via Flip-Flops. Neste circuito também foi implementado uma opção de **Reset**. Quando Reset está um, a máquina de estados deve realizar a contagem normalmente. Quando Reset está em zero, a máquina deve obrigatoriamente permanecer no seu estado inicial (no caso, o estado 0000 cujo a saída é quatro no display). O Reset foi implementado por meio de ANDs nas saídas dos flip flops.

Os flips flops já programados do Quartus foram substituídos por flip flops criados à mão que tem o seguinte circuito:

A função de saída foi obtida com base no estado atual, apenas.

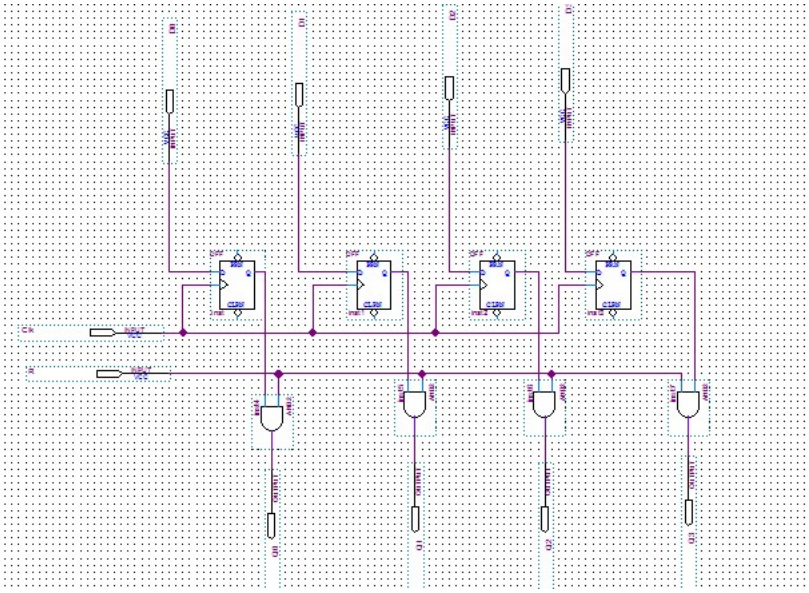

Figura 5. Registradores

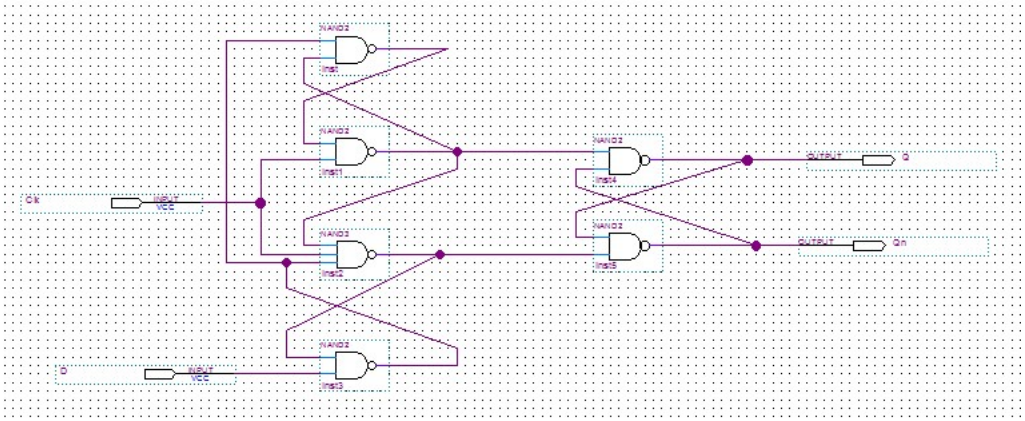

Figura 6. Flip Flop D

Função de saída:

$S_0 = Q_0 Q_3$

$S_1 = Q_0 + \overline{Q_1 Q_3} + Q_2 \overline{Q_3}$

$S_2 = \overline{Q_1} Q_3 + Q_2 Q_3$

$S_3 = Q_0 + Q_2 Q_3 + \overline{Q_1} Q_2 \overline{Q_3}$

Baseando-se nas expressões acima, a função de saída foi desenhada:

Desta maneira, a máquina de estados foi construída utilizando três black boxes (um para cada circuito esquematizado acima)

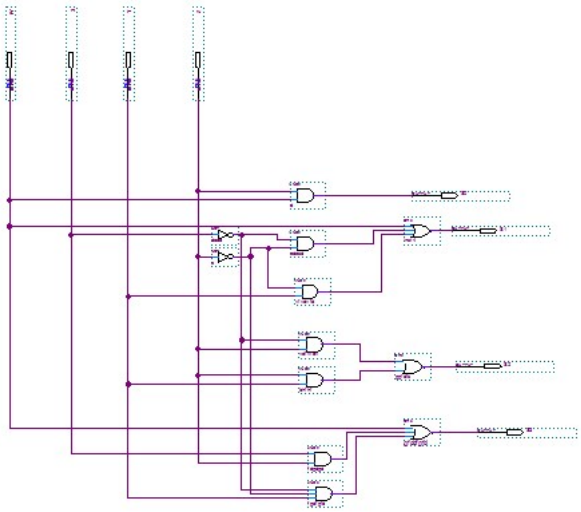

Figura 7. Função de saída

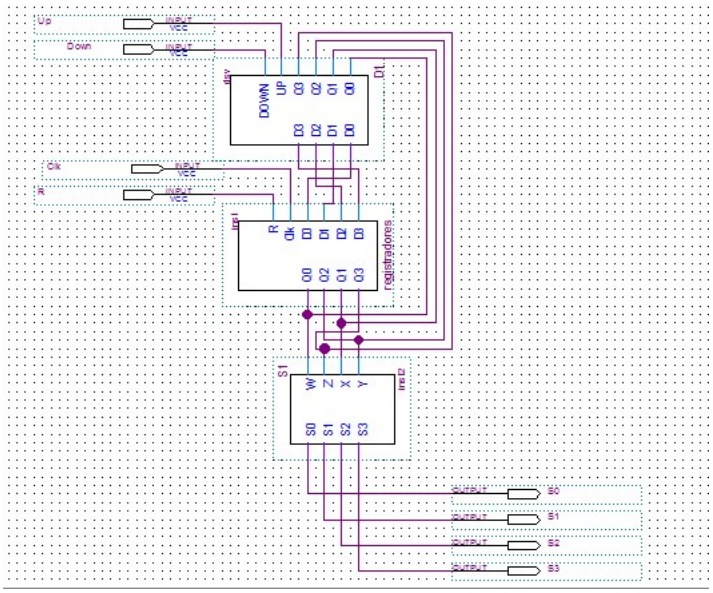

Figura 8. Máquina de Estados Finitos

Referências

Online karnaugh map solver with circuit for up to 8 variables. `www.32x8.com/`. acessado em 18/03/2013.

Hung, Y.-C. (2011). The effects of web-based and face-to-face discussion on computer engineering majors' performance on the karnaugh map. *Journal of Educational Computing Research*, 44(3):345–359.

Karnaugh, M. (1953). The map method for synthesis of combinational logic circuits', commun.

Tocci, R. J., Widmer, N. S., and Moss, G. L. (2003). *Sistemas digitais: princípios e aplicações*, volume 8. Prentice Hall.