Universidade de Brasília
Departamento de Ciência da Computação

Laboratório de Circuitos Lógicos - 10º Experimento

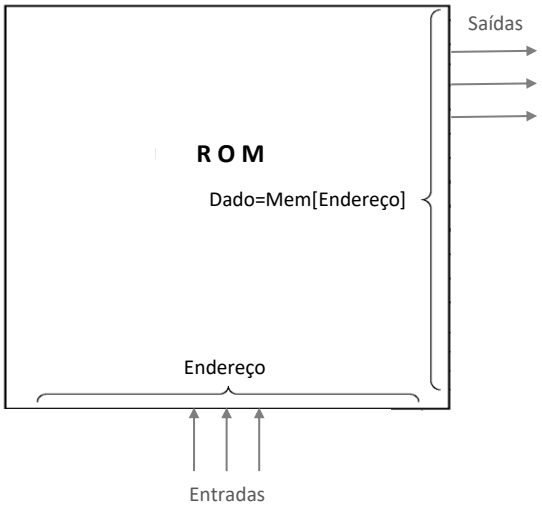
MÁQUINA DE ESTADOS

OBJETIVO: Projetar e montar uma máquina de estados do tipo Moore que atenda aos requerimentos de um controle semafórico de trânsito de carros e pedestres.

1. INTRODUÇÃO TEÓRICA

1.1. IMPLEMENTAÇÃO DE CIRCUITOS COMBINACIONAIS E MÁQUINAS DE ESTADO COM MEMÓRIA ROM

Qualquer função lógica combinacional pode ser sintetizada com o uso da técnica LUT (*Look Up Table*), na qual é necessária apenas a programação de uma memória ROM (*Read Only Memory*) com a tabela verdade desejada. Os sinais de entrada definem o endereçamento da memória, e seu conteúdo define os sinais de saída.



(a)

Entradas			Saídas		
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0

(b)

```
module funcao_logica(
   input [2:0] entradas,
   output [2:0] saidas );

   always @(*)
   casex (entradas)
      3'b 000: saidas = 3'b 000;
      3'b 001: saidas = 3'b 100;
      3'b 010: saidas = 3'b 001;
      3'b 011: saidas = 3'b 101;
      3'b 100: saidas = 3'b 110;
      3'b 101: saidas = 3'b 010;
      3'b 11x: saidas = 3'b 000; // don't care
      default: saidas = 3'b xxx;
   endcase
endmodule
```

(c)

Figura 1: Implementação de funções lógicas por meio de memória ROM e em Verilog

A linguagem de descrição de hardware Verilog permite a implementação de tabelas verdades de forma bastante simplificada, conforme listagem apresentada na **Figura 1(c)** o qual pode ser compilado e criado um bloco para ser utilizado em desenhos esquemáticos.

A **Figura 2** apresenta o diagrama em blocos de uma máquina de estados, onde as funções lógicas de próximo estado e saídas, bem como o tamanho do registrador de estado atual devem ser definidos de acordo com o problema dado.

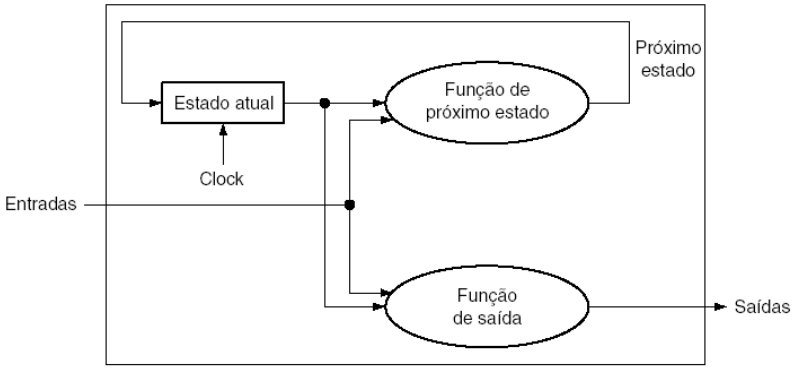


Figura 2: Implementação de uma máquina de estados através da especificação das funções de transição e de saída

Considerando que qualquer função lógica combinacional pode ser implementada de maneira simples com o uso de uma tabela, pode-se construir qualquer máquina de estados com o uso de uma memória ROM. Reduzindo o esforço de projeto dos circuitos de transição e de saída a uma simples programação desta memória, conforme mostrado na figura 3.

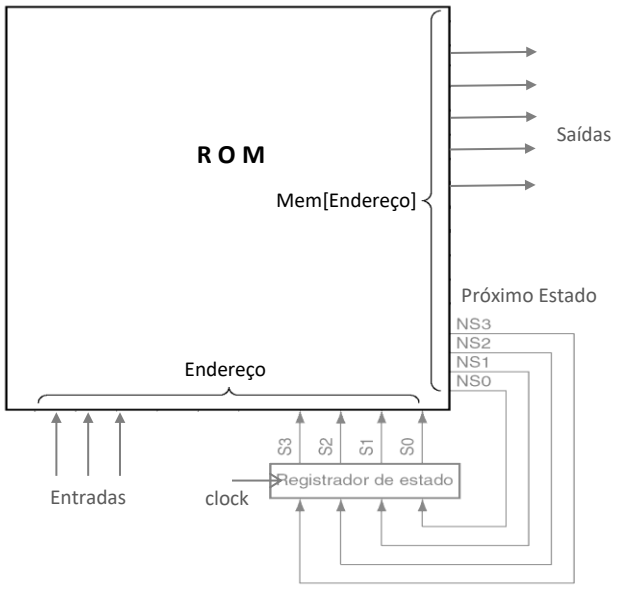


Figura 3: Implementação de uma máquina de estados através da programação de uma memória ROM

No exemplo mostrado na **Figura 3**, utiliza-se um registrador de estados de 4 bits, possibilitando a definição de uma máquina de até 16 estados. Está definido também 3 sinais de entradas (X,Y,Z) e 5 sinais de saída (A,B,C,D,E). Os sinais NS correspondem ao próximo estado (*Next State*) a ser assumido pela máquina, dadas as entradas e o estado atual (*S*). A memória ROM possuirá então $2^{(3+4)}$=128 posições de 9 bits em cada endereço, sendo que apenas algumas delas serão efetivamente utilizadas dependendo do problema.

Uma das formas de se realizar a temporização desejada para as transições é através da definição da frequência do sinal de *clock*.

1.2. CLOCK E TEMPORIZAÇÃO

Os sistemas síncronos implementam máquinas de estado tendo um sinal de relógio (*clock*) como referência para as mudanças de estados. Em experimentos anteriores foram vistos como fazer a divisão de um *clock* a fim de gerar sinais com frequência menor. Os kits DE2 e DE-70 possuem um sinal de *clock* nativo de 50MHz (CLOCK_50) que está disponível ao FPGA. É fornecido no Moodle a implementação em Verilog de um divisor de frequências programável, listado na figura 4.

```
1   module fdiv (
2       input clkin,
3       output reg clkout
4   );
5
6   integer cont;
7   initial cont=0;
8
9   always @(posedge clkin)
10    begin
11      if (cont==25000000)
12        begin
13            cont<=0;
14            clkout<=~clkout;
15        end
16      else
17        begin
18            cont<=cont+1;
19        end
20    end
21  endmodule
22
```

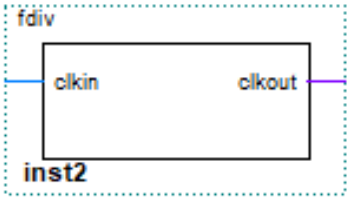
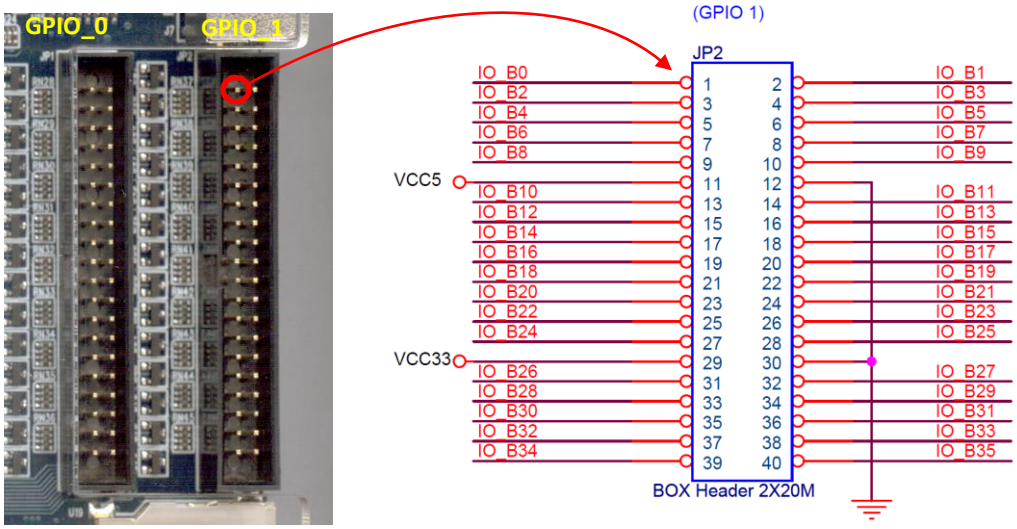


Figura 4: Divisor de Frequência em Verilog

Neste experimento será necessária a implementação de temporizadores que utilizam frequências de *clock* muito menores que 50MHz. Deste modo o valor a ser colocado na linha 11 deve ser calculado de acordo com o valor da frequência de entrada clkin e frequência desejada clkout.

1.3. General Purpose Input Output – GPIO

Geralmente os modernos sistemas digitais possuem interfaces que facilitam sua conexão com o mundo externo. A GPIO é um exemplo que interface que permite acesso direto aos pinos do chip FPGA através de circuitos de proteção internos. O kit de desenvolvimento DE2 possui dois conectores padrão IDE para este tipo de conexão, apresentado na figura 5, denominados GPIO_0 e GPIO_1.



(a) Conectores IDE 40 pinos da DE2 **(b) Diagrama de ligação com o FPGA**

Figura 5: Terminais GPIO_1 do kit DE2

Cada interface possui 40 pinos, sendo um +5V (VCC5), +3.3V (VCC33) e dois terras, disponibilizando 36 pinos para IO do circuito implementado na FPGA. Assim, o pino indicado na **Figura 5(a)** corresponde ao pino IO_B0 que é mapeado para o nome GPIO_1[0] no Quartus-II.

1.4. Diodo Emissor de Luz (LED)

Os LEDs são diodos que, quando diretamente polarizados, emitem luz. O símbolo e correta utilização de um LED é mostrado na **Figura 6**.

É importante notar que o componente **queimará** caso seja percorrido por uma corrente superior a 20 mA. Logo, deve-se usar um resistor limitador que é calculado de acordo com a tensão Vcc, considerando uma queda de tensão de aproximadamente 2V sobre o LED. Use um resistor de 470Ω para obter maior brilho.

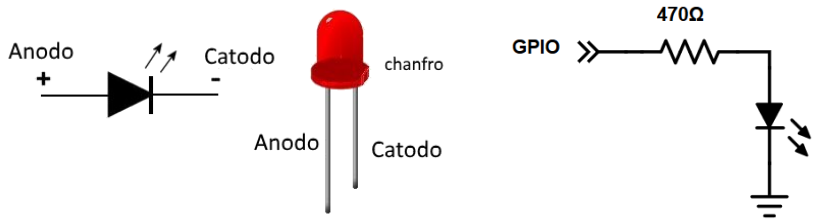
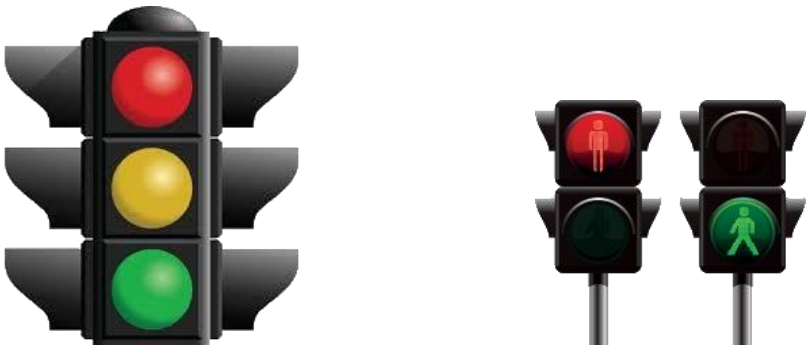


Figura 6: Diodo Emissor de Luz
LEMBRE-SE DE LIGAR O TERRA DO CIRCUITO AO TERRA DA DE2 (pino 12)!!!

2. PARTE EXPERIMENTAL

Projete e implemente um controlador de semáforo de carros e pedestres com as seguintes características:

O controle possui cinco sinais de saída, três lâmpadas para o semáforo de carros (Verde, Amarelo e Vermelho) e duas lâmpadas para o semáforo de pedestre (Verde e Vermelho), de acordo com a **Figura 7**, um sinal de entrada A que indica o estado de funcionamento dos semáforos (Normal ou Atenção) e um botão de *Reset* que reinicializa o sistema. Use um sinal de *clock* de 1Hz.



(a) (b)

Figura 7: Semáforos (a) carros (b) pedestre

Se o sinal A = 0, os semáforos apresentam funcionamento normal, com a seguinte sequência de acendimento das luzes em cada Etapa: 1, 2, 3, 4, 1, 2, 3, 4, 1, 2,...:

Etapa	Semáforo de carros	Semáforo de pedestres	Tempo
1	Verde	Vermelho	6 segundos
2	Amarelo	Vermelho	1 segundos
3	Vermelho	Verde	4 segundos
4	Vermelho	Vermelho piscando	5 segundos
5	Amarelo piscando	Apagado	indefinido

Se o sinal A = 1 for acionado, o semáforo de carros apresenta o funcionamento de atenção, onde apenas a lâmpada Amarela permanece piscando indefinidamente com frequência de **1 Hz**, e o semáforo de pedestre permanece apagado, conforme Etapa 5.

Durante qualquer uma das Etapas, os semáforos voltam ao estado de inicial (Etapa 1) caso o botão de *Reset* seja pressionado.

Projete e implemente um modelo desses semáforos de trânsito utilizando uma máquina de estados e flip-flops do tipo D.

2.1) Desenhe o diagrama de estados completo, apresente a listagem do conteúdo da memória ROM. Desenhe o esquemático do circuito e sua simulação temporal e funcional por forma de onda no Quartus-II (**Pré-Projeto 1**).

Dica: para a simulação ocorrer em um intervalo de 100µs retire o divisor de frequências, de modo que seja usada a frequência base de 50MHz e aumente o tempo de simulação Edit/Set End Time...

2.2) Implemente o controle no Kit de desenvolvimento DE2 ou DE2-70 com acionamento de LEDs externos (*protoboard*) à placa através do conector GPIO_1, nos pinos 0, 1, 2, 3 e 4. Considere a chave de entrada A como SW[0] e o botão de *reset* como o KEY[0] de acordo com o diagrama em blocos sugerido abaixo. Filme o funcionamento em todas as situações e coloque o link no relatório (**Pós-Experimento 1**).

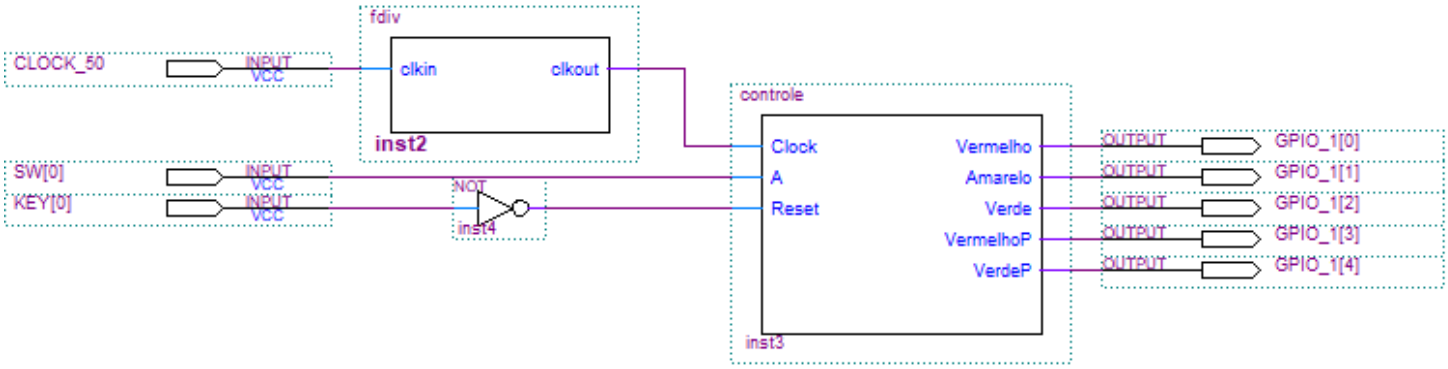


Figura 8: Sugestão de implementação de interfaceamento com o kit DE2

3. SUMÁRIO

Neste experimento os conceitos de máquina de estado e temporizadores são aplicados na construção de um sistema controlador de semáforo de trânsito para carros e pedestres.

4. EQUIPAMENTOS E MATERIAL

- *Software Quartus-II v13.0 SP1*
- Kit desenvolvimento FPGA DE2 ou DE2-70 Intel
- *Protoboard* externo
- *Pendrive* com o projeto
- 5 resistores 470Ω 1/4W
- LEDs de 5mm: 2 Vermelhos, 2 Verdes e 1 Amarelo
- 6 Fios com conectores macho-fêmea (figura ao lado, para facilitar a ligação do conector GPIO da DE2 com o protoboard)

5. TESTE DE AUTOAVALIAÇÃO

Nos itens abaixo marque V (verdadeiro) ou F (falso).

1. ( ) Em uma máquina de Moore as saídas dependem unicamente das entradas.
2. ( ) Em uma máquina de Mealy os estados dependem das entradas
3. ( ) Em uma máquina de Moore o próximo estado depende unicamente do estado atual.
4. ( ) Em uma máquina de Mealy as saídas dependem unicamente da transição entre os estados.
5. ( ) Uma máquina de Mealy sempre produz um circuito mais complexo que uma máquina de More para um mesmo problema.
6. ( ) O diagrama de estados define o tipo de máquina a ser usada na sua implementação.
7. ( ) Em uma máquina Mealy as saídas dependem do estado atual e das saídas passadas.
8. ( ) O diagrama de estados de uma máquina Mealy as transições entre os estados são relacionadas às saídas.
9. ( ) As transições em uma máquina Moore são definidas apenas pelas entradas.
10. ( ) Todo sistema digital síncrono pode ser sintetizado tanto por Mealy quanto por Moore.