Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT
Circuitos Digitais

MÁQUINA DE MEALY

ELE2715 - Laboratório 09

Isaac de Lyra Junior

Natal, 28 de março de 2021

Isaac de Lyra Junior

MÁQUINA DE MEALY

Projeto da disciplina de Circuitos Digitais do Departamento de Engenharia Elétrica da Universidade do Rio Grande do Norte para relatório das atividades.

Docente: Samaherni Morais Dias

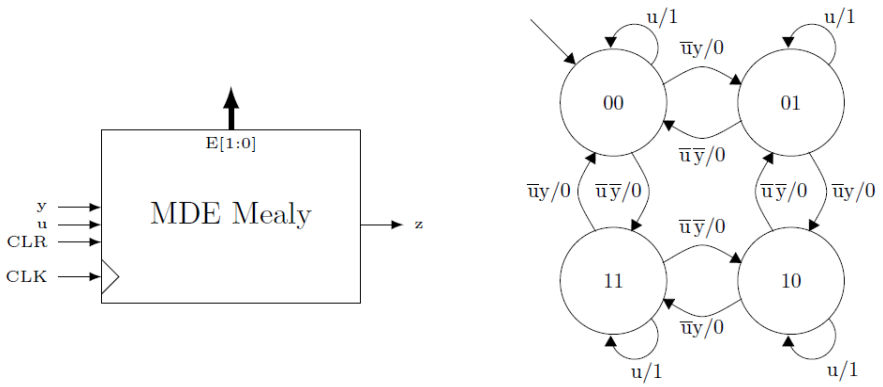
Natal, 28 de março de 2021

SUMÁRIO

1 DESENVOLVIMENTO

A problemática do laboratório é implementar uma máquina de estados de Mealy, que deverá possuir uma entrada de clock, uma entrada de clear (CLR) que quando CLR = 1 faz com que a MDE imediatamente no estado 00, uma entrada para *u* e outra para *y*. O circuito deve apresentar o estado atual da MDE através de uma saída E[1:0] e saída da MDE deve ser direcionada para a saída z. O diagrama da MDE do tipo Mealy está apresentado na Figura 1.

Figura 1 - Diagrama de estados



Fonte: Dados do problema.

O primeiro passo é transformar o diagrama de estados visto na Figura 1 em uma tabela que define quando ocorrerão as mudanças entre os quatro estados existentes com base nas entradas e também a saída z da máquina. O resultado da tabela criada é vista na Figura 2.

Figura 2 - Tabela de transição de estados da máquina

ESTADO ATUAL		ENTRADAS		ESTADO FUTURO		SAÍDA
E1	E0	y	u	E1+	E0+	z
0	0	X	1	0	0	1
0	0	0	0	1	1	0
0	0	1	0	0	1	0
0	1	X	1	0	1	1
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	X	1	1	0	1
1	0	0	0	0	1	0
1	0	1	0	1	1	0
1	1	X	1	1	1	1
1	1	0	0	1	0	0
1	1	1	0	0	0	0

Fonte: Elaborado pelo autor.

Em seguida foi utilizado o conceito de mapas de Karnaugh para definir as equações booleanas de cada saída da máquina, o resultado dessas equações é visto nas Figuras 3,4 e 5 respectivamente.

Figura 3 - Mapa de Karnaugh para a saída E1+

E1+		y/u			
		00	01	11	10
E1/E0	00	1	0	0	0
	01	0	0	0	1
	11	1	1	1	0
	10	0	1	1	1
E1+	(E1'E0'y'u') + (E1'E0yu') + (E1E0y') + (E1E0'y) + (E1u)				

Fonte: Elaborado pelo autor.

Figura 4 - Mapa de Karnaugh para a saída E0+

E0+		y/u			
		00	01	11	10
E1/E0	00	1			1
	01		1	1	
	11		1	1	
	10	1			1
E0+		(E0'u') + (E0u)			

Fonte: Elaborado pelo autor.

Figura 5 - Mapa de Karnaugh para a saída z

z		y/u			
		00	01	11	10
E1/E0	00		1	1	
	01		1	1	
	11		1	1	
	10		1	1	
z		u			

Fonte: Elaborado pelo autor.

Com todas as equações booleanas definidas, foi possível implementar a entidade de lógica combinacional da máquina de mealy, tal entidade recebe os bits do estado atual e as entradas y e u da máquina de estados de mealy para gerar as saídas E1P e E0P que são os bits do estado futuro e também a saída z. O resultado pode ser visto na Figura 6.

Figura 6 - Entidade lógica combinacional

```
ENTITY LOGICA_COMB IS
   PORT( E1,E0,y,u: in bit;
         E1P, E0P, z: out bit);
END LOGICA_COMB;

ARCHITECTURE CKT OF LOGICA_COMB IS

BEGIN

   E1P <= (not E1 and not E0 and not y and not u) or (not E1 and E0 and y and not u) or
          (E1 and E0 and not y) or (E1 and not E0 and y) or (E1 and u) ;

   E0P <= (not E0 and not u) or (E0 and u);

   z <= u;

END CKT;
```

Fonte: Elaborado pelo autor.

Em seguida, com a entidade de lógica combinacional feita, foi possível implementar a entidade da máquina de estados de mealy, ela recebe o clock através da entrada CLK, as entradas y e u que são definidas pelo usuário e a entrada CLR que é responsável por resetar o registrador

de estados quando estiver em nível lógico alto, tal máquina terá como saída um número binário de 2 bits que irá mostrar o estado atual da máquina e também a saída z. Na Figura 7 é possível visualizar a definição desta entidade e também a lógica dela, que utiliza como componentes a entidade do Flip Flop D e também a entidade da lógica combinacional explicada anteriormente.

Figura 7 - Entidade da máquina de estados mealy

```
ENTITY mealy_machine IS
   PORT(CLK, y, u, CLR: in bit;
        E: out bit_vector(1 downto 0);
        z: out bit);
END mealy_machine;

ARCHITECTURE CKT OF mealy_machine IS

COMPONENT ffd IS
   PORT (clk, D, P, C: IN BIT ;
        q: OUT BIT );
END COMPONENT;

COMPONENT LOGICA_COMB IS
   PORT( E1,E0,y,u: in bit;
        E1P, E0P, z: out bit);
END COMPONENT;

signal E1, E0, E1P, E0P, clear: bit;

BEGIN
   clear <= not CLR;

   FFD1: ffd port map (CLK, E1P, '1', clear, E1);
   FFD2: ffd port map (CLK, E0P, '1', clear, E0);

   LC: LOGICA_COMB port map( E1, E0, y, u, E1P, E0P,z);

   E(0) <= E0;
   E(1) <= E1;

END CKT;
```

Fonte: Elaborado pelo autor.

2 RESULTADOS

Para realizar os testes da entidade da máquina de estados de mealy foi utilizado um arquivo de extensão ".do" para simular no software *ModelSim* o código VHDL escrito que força as entradas da entidade para realizar todos os testes necessários e visualizar o comportamento da máquina em suas saídas, o resultado da escrita desse arquivo é visto na Figura 8.

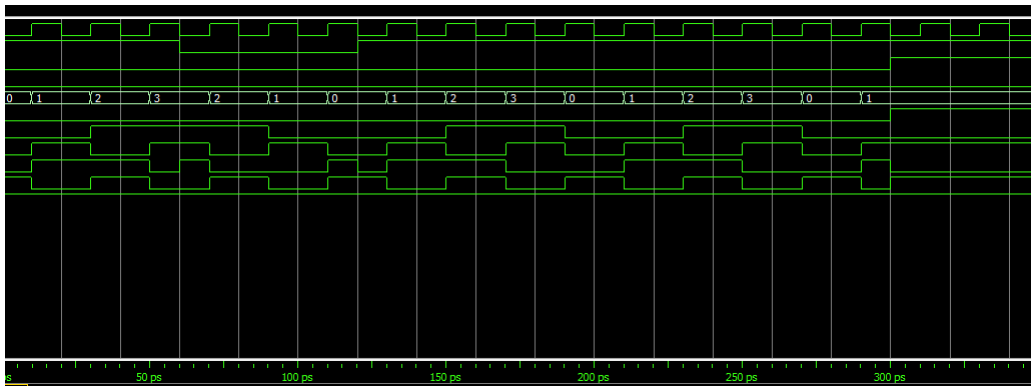
Figura 8 - Conteúdo do arquivo .do

```
vsim mealy_machine

add wave *

force CLK 0 0, 1 10 -repeat 20
force y 1 0, 0 60, 1 120
force u 1 300, 0 600
force CLR 1 1000

run 1200
```

Fonte: Elaborado de autor.

Primeiramente é definido que *y* está em nível lógico alto e *u* em nível lógico baixo no instante 0, isso deve fazer com que a máquina alterne entre os estados de forma crescente. No instante 60 é definido que *y* tenha nível lógico baixo e *u* permaneça em nível lógico baixo, o comportamento esperado é que a máquina alterne de maneira decrescente entre os estados. Em seguida foi definido *u* com nível lógico alto e é esperado que a máquina não alterne os estados. Por fim, foi testada a entrada CLR que é definida para nível lógico alto, e é esperado que o estado volte para 0 quando e permaneça lá enquanto esta entrada CLR estiver em nível lógico alto.

Figura 9 - Simulação da máquina



Fonte: Elaborado pelo autor.

3 CONCLUSÃO

O relatório tinha como finalidade apresentar a implementação deu uma máquina de estados do tipo Mealy, que possui uma entrada para o clock, uma entrada CLR que faz a máquina retornar para o estado 00 quando estiver em nível lógico alto, a entrada *u* e a entrada *y,* também possui uma saída E de dois bits que exibirá o estado atual e a saída z. Para implementação de tal máquina foi utilizado o diagrama disponibilizado na atividade, foi feito uma tabela de transição de estados e utilizado o conceito de mapas de Karnaugh para definir as equações booleanas das saídas da MDE do tipo mealy. Por fim, foi implementado às entidades da lógica combinacional e da própria MDE em VHDL e simulado no software ModelSim.

REFERÊNCIAS

TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L.. **Sistemas Digitais: Princípios e Aplicações**. 11a ed. São Paulo: Pearson, 2011. 844 p.

VAHID, Frank. **Sistemas digitais: projeto, otimização e HDLS**. Rio Grande do Sul: Artmed Bookman, 2008. 558 p.

ANEXO A - CÓDIGO VHDL COMPLETO

```
--==============--
-- Flip-Flop D --
--==============--

ENTITY ffd IS
	port (clk ,D,P,C: IN BIT ;
		q: OUT BIT );
END ffd;

ARCHITECTURE ckt OF ffd IS

	SIGNAL qS: BIT;

BEGIN

	PROCESS (clk ,P,C)
	BEGIN
		IF P = '0' THEN qS <= '1';
		ELSIF C = '0' THEN qS <= '0';
		ELSIF clk ='1' AND clk ' EVENT THEN
		qS <= D;
		END IF;
	END PROCESS ;

	q <= qS;

END ckt;


--======================--
-- LÓGICA COMBINACIONAL --
--======================--

ENTITY LOGICA_COMB IS
	PORT( E1,E0,y,u: in bit;
		E1P, E0P, z: out bit);
END LOGICA_COMB;

ARCHITECTURE CKT OF LOGICA_COMB IS
```

```
BEGIN

	E1P <= (not E1 and not E0 and not y and not u) or (not E1 and E0 and y and not u) or
			(E1 and E0 and not y) or (E1 and not E0 and y) or (E1 and u) ;

	E0P <= (not E0 and not u) or (E0 and u);

	z <= u;

END CKT;

--=====================--
-- MÁQUINA DE ESTADOS --
--=====================--

ENTITY mealy_machine IS
	PORT(CLK, y, u, CLR: in bit;
		E: out bit_Vector(1 downto 0);
		z: out bit);
END mealy_machine;

ARCHITECTURE CKT OF mealy_machine IS

COMPONENT ffd IS
	PORT (clk, D, P, C: IN BIT ;
		q: OUT BIT );
END COMPONENT;

COMPONENT LOGICA_COMB IS
	PORT( E1,E0,y,u: in bit;
		E1P, E0P, z: out bit);
END COMPONENT;

signal E1, E0, E1P, E0P, clear: bit;

BEGIN
	clear <= not CLR;

	FFD1: ffd port map (CLK, E1P, '1', clear, E1);
```

```
        FFD2: ffd port map (CLK, E0P, '1', clear, E0);

        LC: LOGICA_COMB port map( E1, E0, y, u, E1P, E0P,z);

        E(0) <= E0;
        E(1) <= E1;

END CKT;
```