Universidade de São Paulo
Escola de Engenharia de São Carlos
Depto. de Engenharia Elétrica e de Computação

Introdução a VHDL

Aula 7

Professora Luiza Maria Romeiro Codá

Aula 7: Introdução a VHDL

Conteúdo:

- Tipos em VHDL:
  - Tipos pré definidos
  - Tipos definidos pelo usuário
- Máquina de estados:
  1. Moore
  2. Mealy
- Prática nº 10: semáforo de cruzamento

Tipos em VHDL

- Tipos pré-definidos:
  - biblioteca padrão
  - biblioteca importada pelo usário (ex: IEEE)

- Tipos definidos pelo projetista:

  usa a palavra reservada TYPE;

Tipos em VHDL: pré-definidos na biblioteca work:

- BIT : assume valor '0' ou '1'
- BIT_VECTOR: designa um conjunto de bits. Ex: "10101" ou x"00FF"
- BOOLEAN: assume valores { true, false} (Útil apenas para descrições abstratas, onde um sinal só pode assumir dois valores)

  Obs:Em VHDL os valores booleanos (false and true) não são idênticos ao lógico '0' e '1'.
- REAL : Ex: -1.0 / +2.35 / 37.0 / -1.5E+23

  (Utilizado durante desenvolvimento da especificação)
- INTEIRO: são números que variam de $(-2^{31}-1) \leq x \leq (2^{31}-1)$.

  Ex: +1 / 5 / 1232 / -1234
- CHARACTER : é definido por caracter entre aspas simples "a", "x", "A"

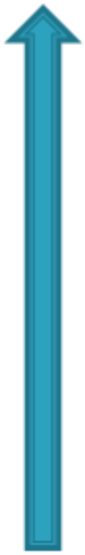
  VHDL não é "case sensitive", exceto para o tipo Character.
- STRING: tipo que designa um conjunto de caracteres.

  Ex: "vhdl"

Tipos em VHDL: pré-definidos na biblioteca na biblioteca IEEE:

- STD_LOGIC
- STD_LOGIC_VECTOR

Precedência



Valor	Estado Lógico
U	Não Inicializado
X	Desconhecido Forte
0 | 1	Nível Baixo Forte | Nível Alto Forte
W	Desconhecido Fraco
L | H	Nível Baixo Fraco | Nível Alto Fraco
Z	Alta Impedância
-	Não Importa

Tipos em VHDL: definidos pelo usuário:

A linguagem VHDL tem a flexibilidade de proporcionar que o projetista defina seus próprios tipos de dados possibilitando:

- Que o código seja mais fácil de modificação;
- Com os próprios tipos de dados minimiza-se erros de definições de tipos e ranges os quais são comuns em projetos grandes

Tipos em VHDL definidos pelo usuário :

Utiliza-se a palavra reservada TYPE

A definição de tipos de dados podem ser :

- Baseados em inteiros
- Baseados em tipos enumerados
- Baeados em arranjos (Array)

Tipos em VHDL definidos pelo usuário :

- **Baseados em inteiros:**

Sintaxe:

```
TYPE <type_name> IS RANGE <type_range>;
```

A definição do tipo INTEGER no pacote STD é:

TYPE INTEGER IS RANGE – 2147483674 TO 2147483674

Exemplo definições pelo usuário de tipos para hora, minuto, segundo:

```
TYPE hora IS RANGE 0 TO 23; -- sinal do tipo hora só podem assumir valores de 0 a 23
TYPE minuto IS RANGE 0 TO 59;
TYPE segundo IS RANGE 0 TO 59;
```

Tipos em VHDL definidos pelo usuário :

- **Baseados em tipos enumerados:**

Sintaxe:

```
TYPE <type_name> IS (state_names);
```

Utilizados para definir tipos de sinais baseados em estados :Ex máquinas de Estados

A definição do tipo BOOLEAN na bibiloteca STD é:

```
TYPE BOOLEAN IS (false,true);
```

A definição do tipo BIT na bibiloteca work é:

```
TYPE BIT IS ('0', '1');
```

Exemplo: 1. definições pelo usuário de tipos para máquina de estado psrs um robô

```
TYPE maquina_estado IS  ( parado, frente, tras direita esquerda );
```

2. Declaração do tipo enumerado "temperatura"

```
TYPE temperatura IS (baixa, media, alta);
```

Exemplo de atribuição de valores de sinal de tipo "temperatira"

```
Temp_Forno <= media;
```

Tipos em VHDL definidos pelo usuário :

- **Baeados em arranjos (Array):**

  Sintaxe:

```
TYPE <type_name> IS ARRAY (array_range) OF data_type;
```

A definição do tipo STD_LOGIC_VECTOR na bibiloteca STD é:

```
TYPE STD_LOGIC_VECTOR IS ARRAY (NATURAL RANGE<>) OF STD_LOGIC;
--NATURAL RANGE<> significa arange indefinido, ou seja, que o usuário define o
-- RANGE no tipo natural
```

Exemplo: 1. definição pelo usuário do tipo enumerado temperatura

```
TYPE maquina_estado IS  ( parado, frente, tras direita esquerda );
```

2. Declaração do tipo "vetor_temperatura", vetor com 8 elementos do tipo "temperatura“ (TYPE temperatura IS (baixa, media, alta);

```
TYPE vetor_temperatura IS ARRAY (0 TO 7) OF temperatura;
```

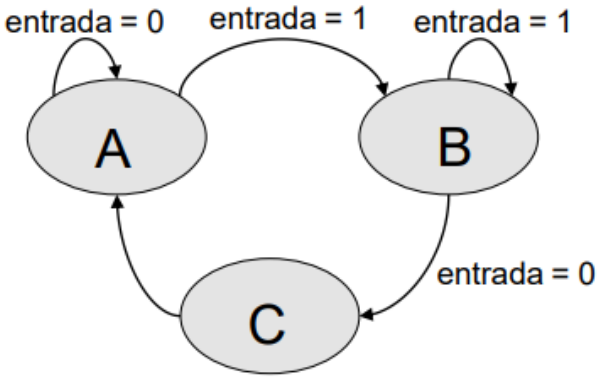
Máquina de estados

- uma máquina de estados é um circuito sequencial síncrono, que transita numa sequência predefinida de estados;
- A transição entre os estados é comandada por um sinal de controle(clock);
- O estado atual é definido por um elemento de memória e o estado futuro é determinado com base no estado atual e nas condições das entradas.

Máquina de estados

Máquinas de Estados podem ser representadas por meio de um diagrama de estados.

Exemplo:



Estado	Saída
A	0
B	0
C	1

Cada transição ocorre em um evento do relógio, tanto de um estado para outro quanto para o mesmo estado

Máquina de estados

Há dois tipos de máquinas de estados: **Moore** e **Mealy.**

- ✓ Em máquinas **Moore**, as saídas(estados futuros) dependem apenas do **estado atual**.

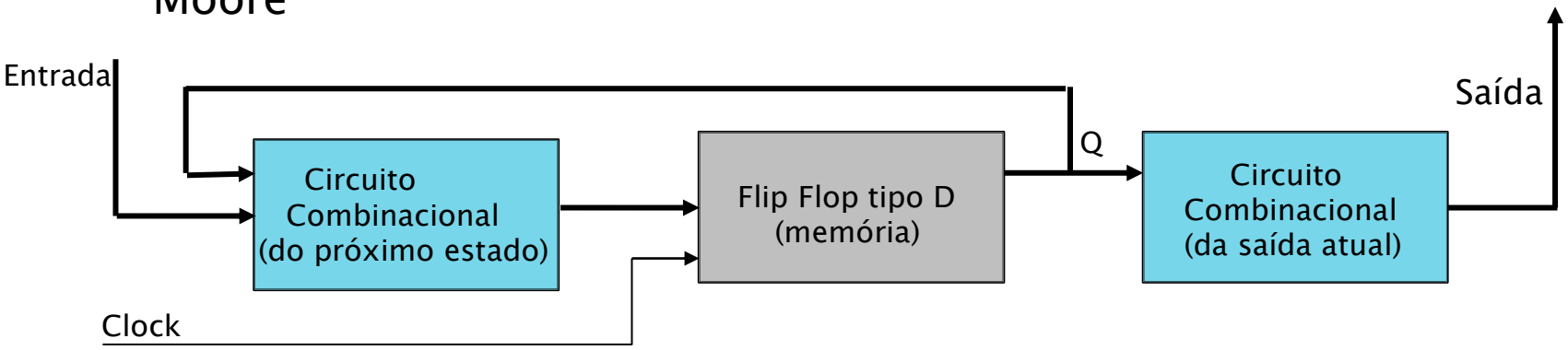
$$Moore: Saídas = f(Estado\ Atual,)$$

- ✓ Em máquinas **Mealy**, as saídas (estados futuros) dependem do **estado atual** e das **entradas**.

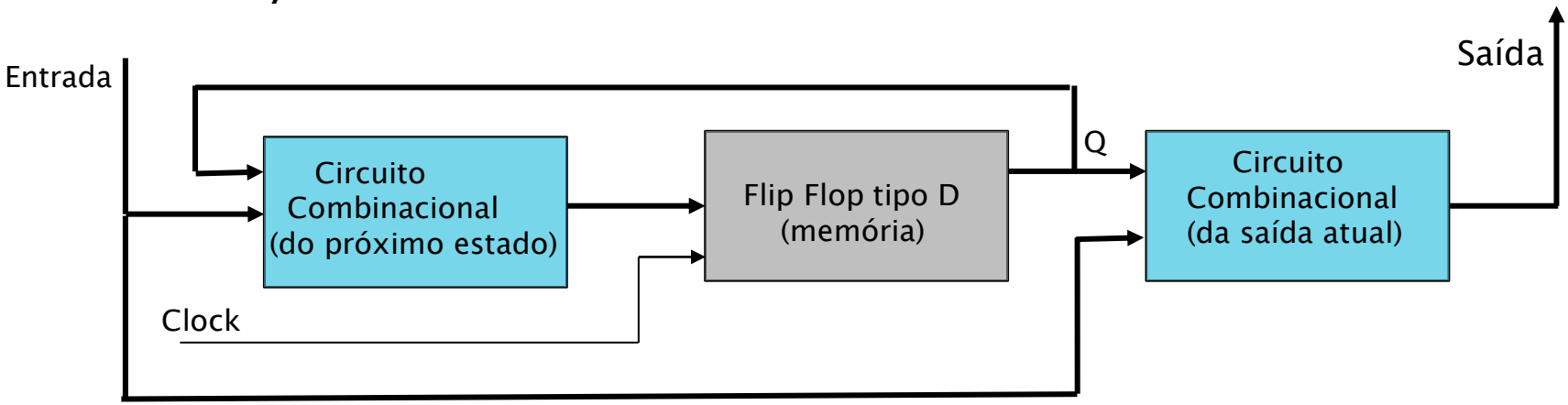
$$Mealy: Saídas = f(Estado\ Atual, Entradas)$$

Máquina de estados

Moore



Mealy



Máquina de estados

A ferramenta de síntese, no caso o *software* Quartus II- Altera, identifica que uma descrição é de uma máquina de estados se a descrição contém um **objeto que armazena o estado atual**, uma **definição de transição de estados** e uma **especificação de valores de saída**.

É possível e aconselhável definir um novo tipo enumerado com os nomes dos estados, a fim de facilitar a leitura do código e a documentação. A ferramenta de síntese gera, neste caso, uma codificação para cada estado. As codificações mais comuns são apresentadas na tabela a seguir.

Tabela 1 - Exemplo de codificação de estados pela ferramenta de síntese para uma máquina de estados com 5 estados

Estado definido no código	Binário	Código Gray	"Único um" ou "*one-hot*"
reset	000	000	00001
estado_A	001	001	00010
estado_B	010	011	00100
estado_C	011	010	01000
estado_D	100	110	10000

Máquina de Estados

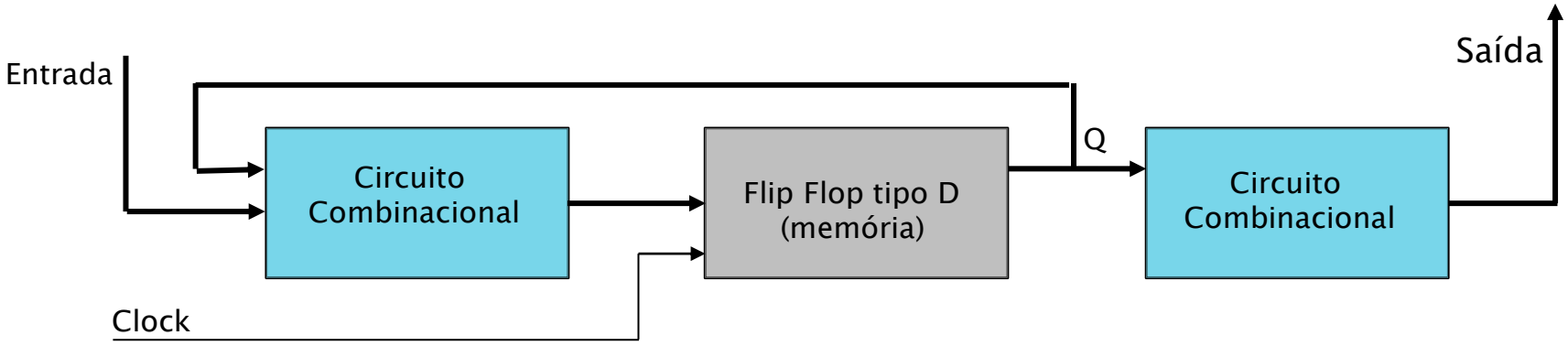
Sequência de passos a seguir para realizar o projeto:

- Inicialmente define-se um tipo enumerado contendo o nome dos estados e os sinais para cada tipo:

```
TYPE tipo_estado IS (parado, decisao, leitura, escrita);
SIGNAL estado_atual, próximo_estado : tipo_estado;
```

- Cada estado pode ser transcrito utilizando-se CASE;
- A transição de estados é realizada através de IF-ELSE;

Máquina de Moore

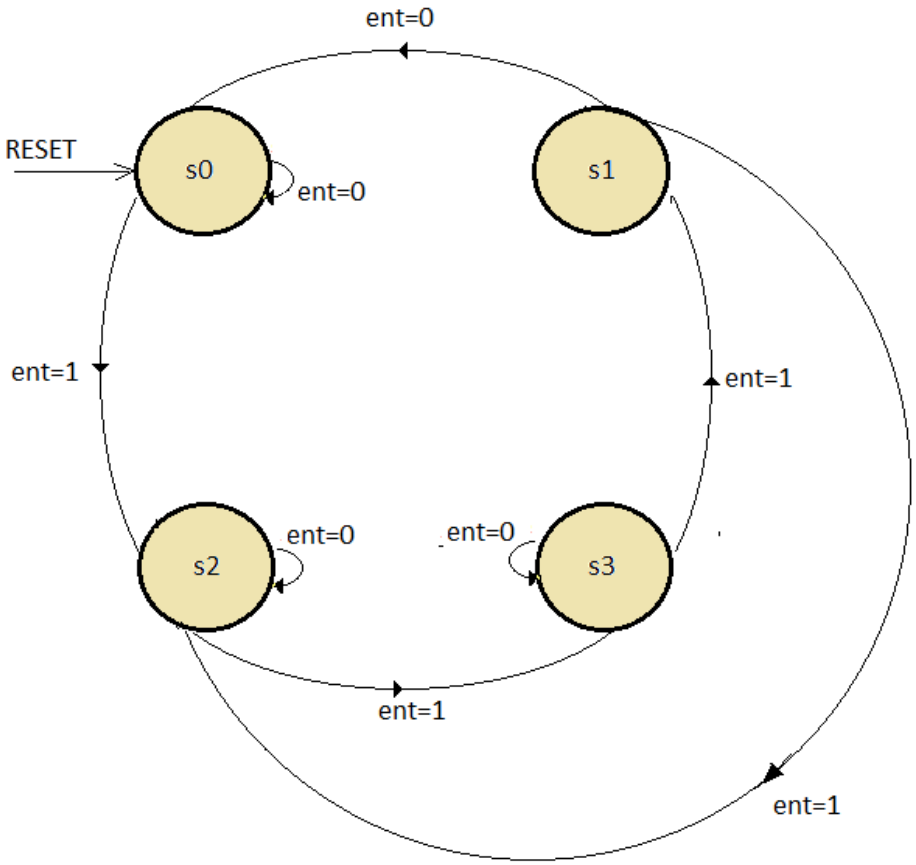


$$Saídas = f(Estado\ Atual,)$$

Descrição de Máquina de Moore

com 4 estados, 1 entrada e 1 saída.

Máquina de estados a ser descrita



Descrição de Máquina de Moore (continuação)

com 4 estados, 1 entradas e 1 saída.

PARTE 1

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY est_maq_moore IS
    PORT(clk, reset       : IN  STD_LOGIC;
          ent : IN  STD_LOGIC;  --  entrada
         saida     : OUT STD_LOGIC); -- Saída
END est_maq_moore;

ARCHITECTURE a OF est_maq_moore IS
    -- Definição de novo tipo enumerado
    TYPE estado_tipo IS (s0, s1, s2, s3); --definição do 4 estados
    SIGNAL estado: estado_tipo; -- Cria o sinal estado cujo tipo é ESTADO_TIPO
BEGIN
    -- Processo de Controle de estados da máquina de Moore
    PROCESS(clk, reset)
    BEGIN
      IF reset = '1' THEN
            estado <= s0;

      ELSIF (clk'EVENT AND clk = '1') THEN -- Verifica borda de subida do clock
          -- Máquina de estados usando CASE
          --Lógica combinacional que decide qual será o próximo estado
          --está no próximo slide

          END IF;
END PROCESS;
END a;
```

Descrição de Máquina de Moore (continuação)

com 4 estados, 1 entrada e 1 saída.

PARTE 2

```
-- Máquina de estados
--Lógica combinacional que decide qual será o próximo estado
--na máquina de moore o valor da saída independe da entrada

CASE estado IS
    WHEN s0 =>
           saida <= '0';
        IF ent = '1' THEN
            estado <= s2;
        END IF;
    WHEN s1 =>
           saida <= '1';
        IF ent = '0' THEN
            estado <= s0;
        ELSE
           estado <= s2;
        END IF;
    WHEN s2 =>
           saida <= '1';
        IF ent = '1' THEN
            estado <= s3;
        END IF;
     WHEN s3 =>
           saida <= '0';
        IF ent = '1' THEN
            estado <= s1';
        END IF;
END CASE;
```

Descrição de Máquina de Moore (continuação)

com 4 estados, 1 entrada e 1 saída.

PARTE 3

Circuito Gerado:

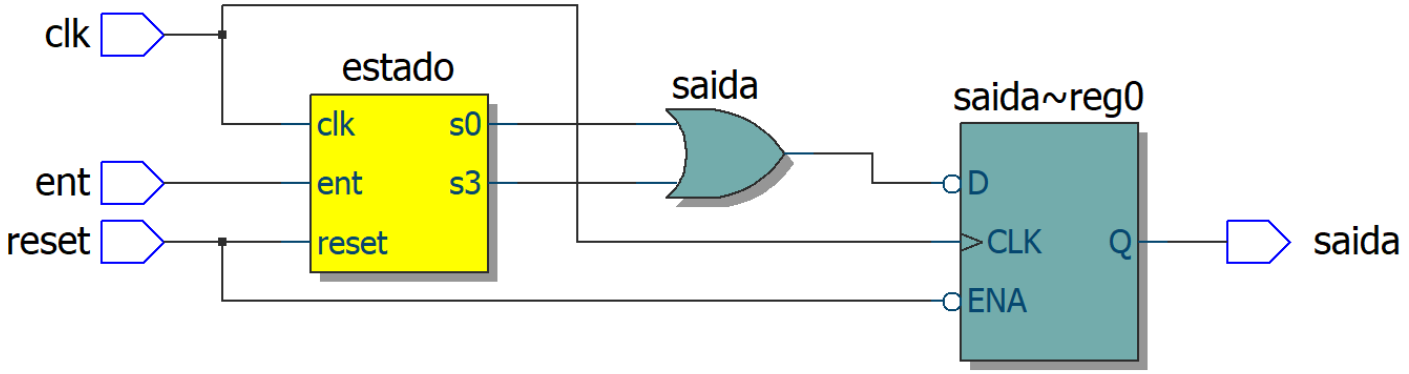
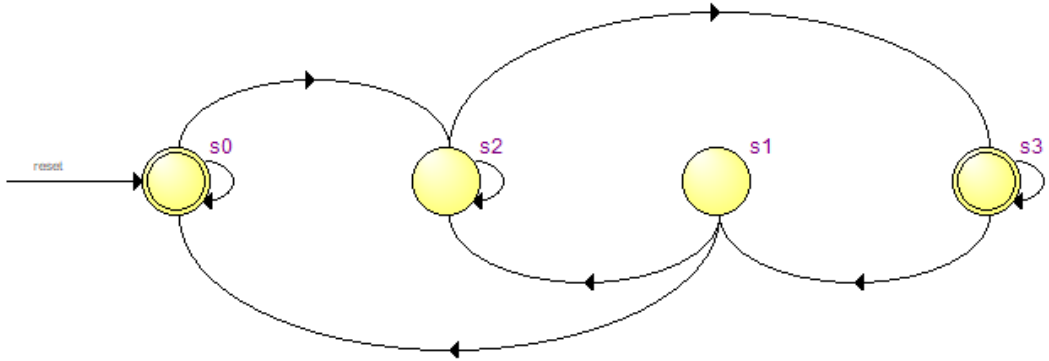


Diagrama de Estados Gerado:

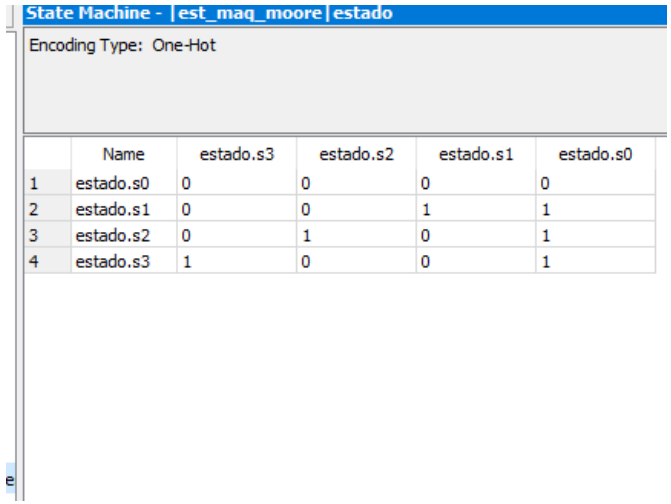


Descrição de Máquina de Moore (continuação)

com 4 estados, 1 entrada e 1 saída.

Codificação dos estados:

Após a compilação, a ferramenta de síntese atribui valores aos estados criados na descrição do projeto "est_maq_moore“.



State Machine - |est_maq_moore|estado

Encoding Type: One-Hot

	Name	estado.s3	estado.s2	estado.s1	estado.s0
1	estado.s0	0	0	0	0
2	estado.s1	0	0	1	1
3	estado.s2	0	1	0	1
4	estado.s3	1	0	0	1

s0 = "0000", s1= "0011" , s2 = “0101“ e s3 = “1001”

Estado s0 recebeu o valor "0000" porque foi o primeiro valor listado na declaração “TYPE”.

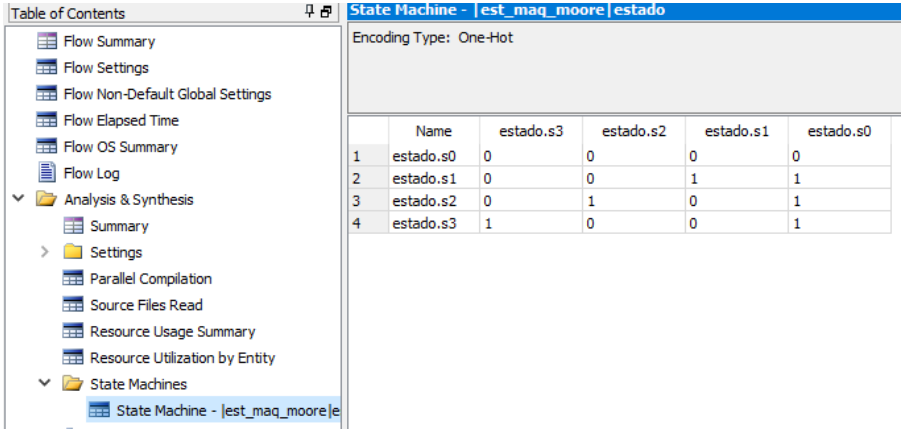
Descrição de Máquina de Moore (continuação)

com 4 estados, 1 entrada e 1 saída.

Codificação dos estados:

Após a compilação, a ferramenta de síntese atribui valores aos estados criados na descrição do projeto. Ela utiliza a codificação "One-Hot", para atribuir valores a cada estado, nessa codificação a quantidade de FFs utilizada é a mesma da quantidade de estados, deixando o circuito mais rápido, porém consumindo mais lógica.

Esses valores podem ser visualizados, após compilação na janela **Table of Content/Analysis & Synthesis/State Machines** :



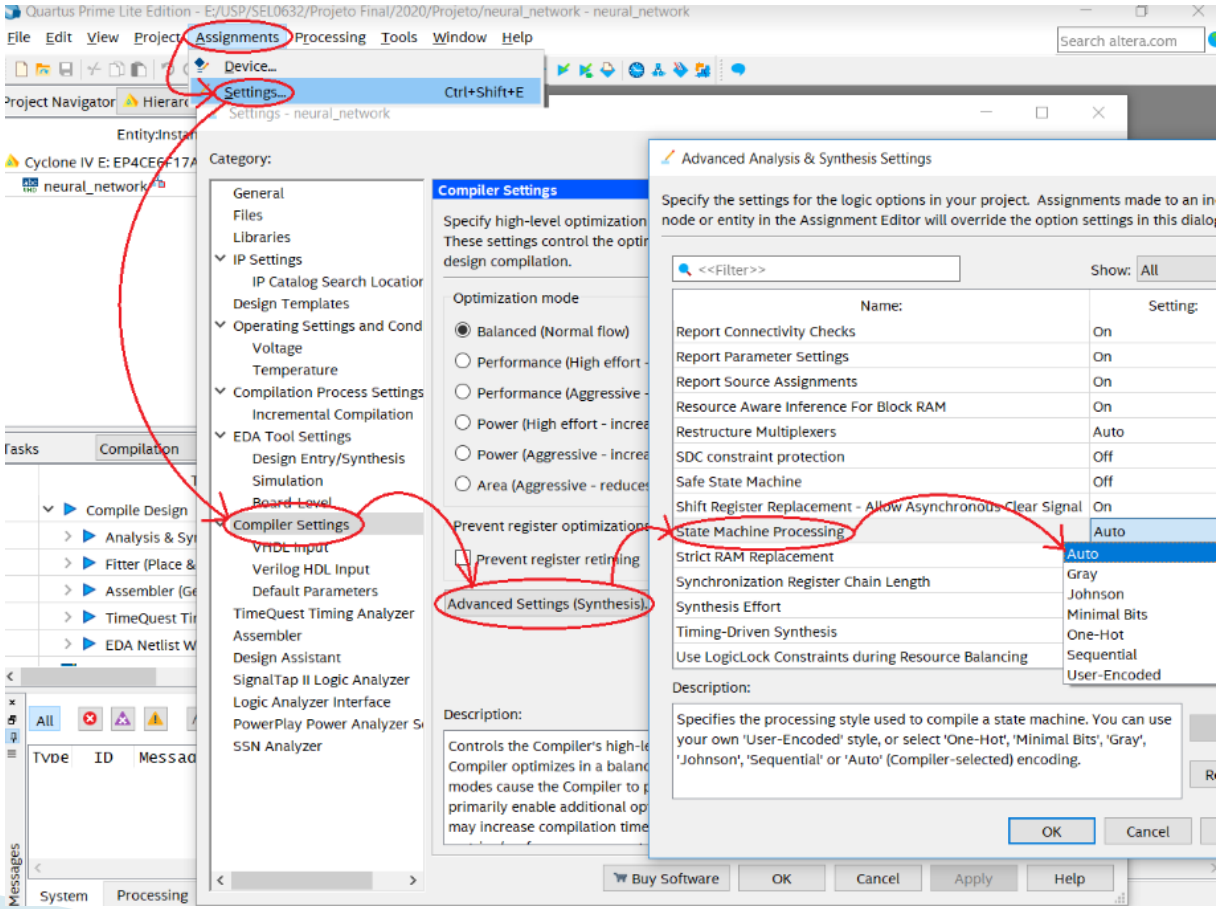
Para mudar o código de “one-Hot” para outro segue os passos:
Assignment/Setting

Descrição de Máquina de Moore (continuação)

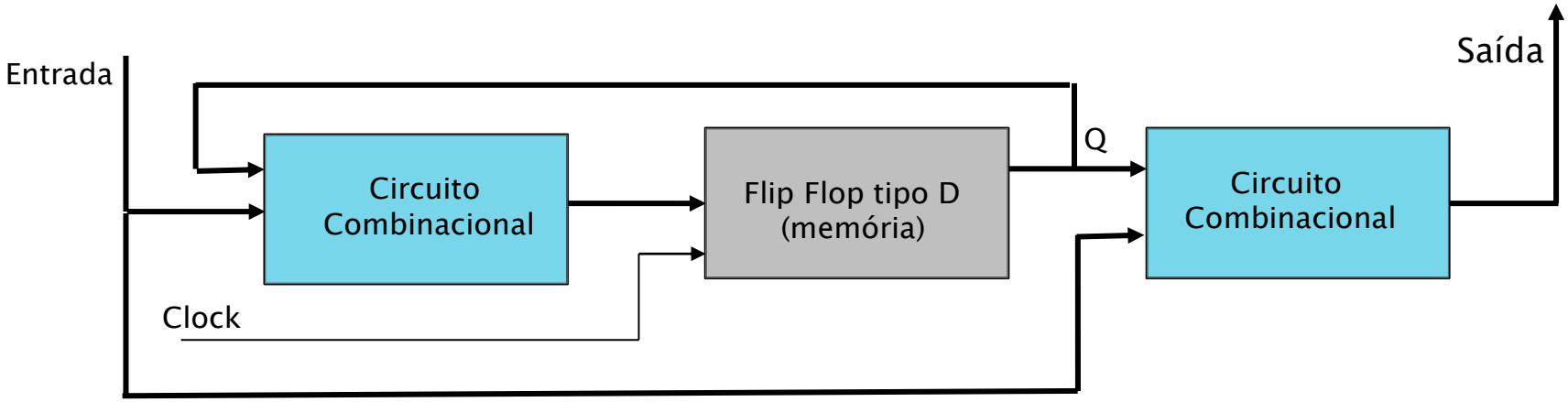
com 4 estados, 1 entrada e 1 saída.

Codificação dos estados:

Utilizar o código de "one-Hot" para codificação de estados não é crucial caso o circuito projetado seja pequeno, porém se quiser alterar esse código para códigos que consumam menor lógica, ex. código Gray, seguir os passos da Figura:



Máquina de Mealy

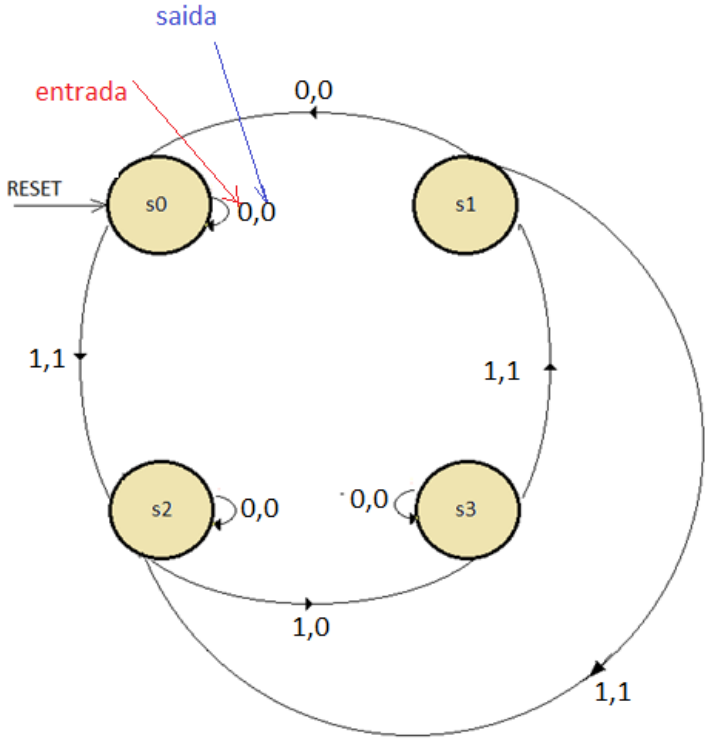


$$Saídas = f(Estado\ Atual, Entradas)$$

Descrição de Máquina de Mealy

com 4 estados, 1 entrada e 1 saída.

Máquina de estados a ser descrita



Descrição de Máquina de Mealy
com 4 estados, 1 entrada e 1 saída.

PARTE 1

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY maq_mealy IS
     PORT (  ck          : IN std_logic;
                 reset     :IN std_logic;
                 entrada : IN std_logic;
                 saida    : OUT std_logic  );
END maq_mealy;

ARCHITECTURE a OF maq_mealy IS
-- Definição de novo tipo enumerado
TYPE state_type IS (s0, s1, s2, s3);
SIGNAL estado: state_type;
BEGIN
-- Processo de controle de estado da máquina de Mealy
     PROCESS (ck, reset)
     BEGIN
          IF reset = '1' THEN
          estado <= S0;
          ELSIF ck'EVENT  AND ck = '1' THEN
          -- Máquina de Estados
          --está no slide seguinte
          END IF;
    END PROCESS;
-- Processo de controle de saída
-- o valor da saída depende da entrada
saida <=  '1' WHEN  entrada = '1' AND  (estado = s1 OR estado = s3) ELSE
          '0';
END a;
```

Descrição de Máquina de Mealy (continuação)

com 4 estados, 1 entrada e 1 saída.

PARTE 2

```
-- Máquina de estados ( Mealy)
--Lógica combinacional que decide qual será o próximo estado

CASE estado IS
        WHEN s0 =>
                IF entrada = '1' THEN
                estado <= s2;
                END IF;

        WHEN s1 =>
                IF entrada = '0' THEN
                estado <= s0;
                ELSE
                estado <= s2;
                END IF;

        WHEN s2 =>
                IF entrada = '1' THEN
                estado <= s3;
                END IF;

        WHEN s3 =>
                IF entrada = '1' THEN
                estado <= s1;
                END IF;
END CASE;
-- Processo de controle de saída
```

Descrição de Máquina de Mealy (continuação)

com 4 estados, 1 entrada e 1 saída.

PARTE 3

Circuito Gerado:

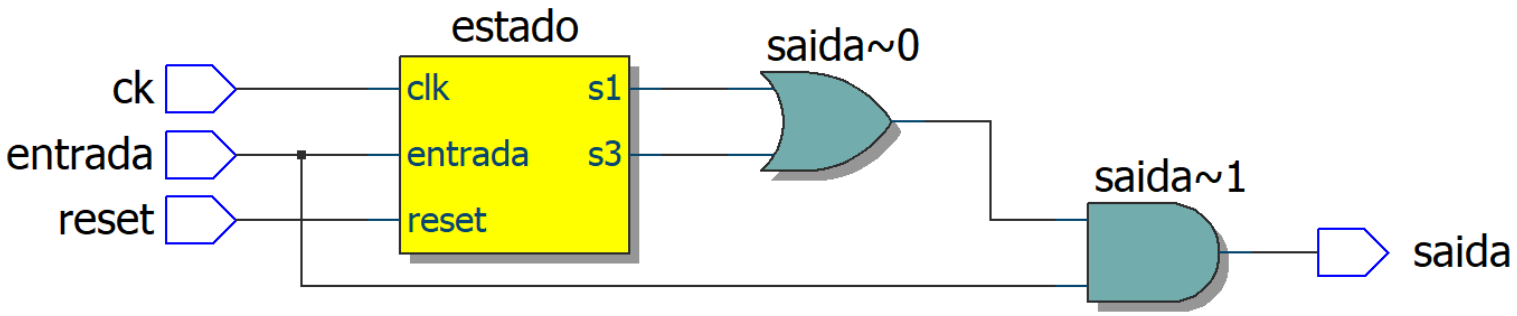
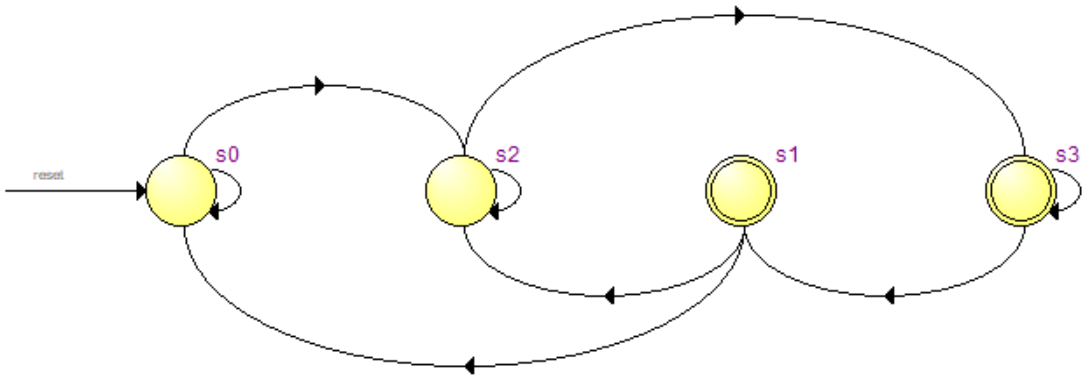
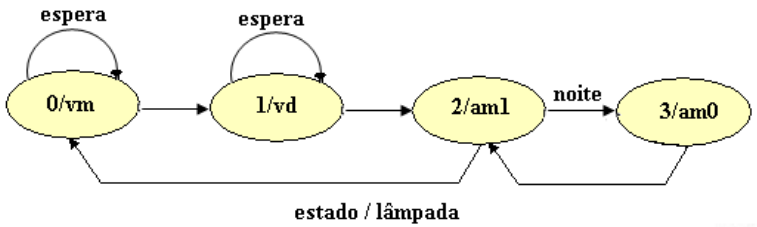


Diagrama de estados gerado:

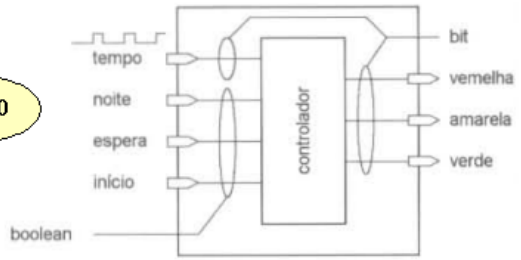
Visualizar em Tools/Netlist Viewer/State Machine Viewer



Pratica 10: semáforo de cruzamento usando Máquina de estados



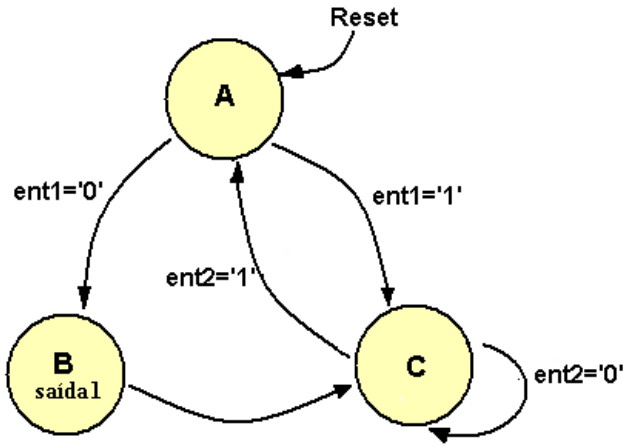
vm = vermelha acesa
vd = verde acesa
am1 = amarela acesa
am0 = amarela apagada



Exemplo 2: Descrição de Máquina de Moore

com 3 estados, 2 entradas e 1 saída.

Máquina de estados a ser descrita



Resolução do Exemplo 2 ;Descrição de Máquina de Moore (continuação) com 3 estados, 2 entradas e 1 saída.

PARTE 1

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;

ENTITY est_maq IS
    PORT(clk         : IN  STD_LOGIC;
         reset       : IN  STD_LOGIC;  -- Reset = '1' leva ao estado A
         ent1, ent2  : IN  STD_LOGIC;  -- Duas entradas
         saida1      : OUT STD_LOGIC); -- Saída
END est_maq;

ARCHITECTURE a OF est_maq IS
    -- Definição de novo tipo enumerado
    TYPE state_type IS (state_A, state_B, state_C);
    SIGNAL state: state_type; -- Cria o sinal state cujo tipo é STATE_TYPE
BEGIN
    -- Processo de Controle de estados da máquina de Moore
    PROCESS(clk, reset)
    BEGIN
      IF reset = '1' THEN
            state <= state_A;
    -- Verifica borda de subida do clock

      ELSIF (clk'EVENT AND clk = '1') THEN
```

Resolução do Exemplo 2 ;Descrição de Máquina de Moore (continuação) com 3 estados, 2 entradas e 1 saída.

PARTE 2

```
-- Máquinq de estados
--Lógica combinacional que decide qual será o próximo estado

                CASE state IS
                    WHEN state_A =>
                        IF ent1= '0' THEN
                            state <= state_B;
                        ELSE
                            state <= state_C;
                        END IF;
                    WHEN state_B =>
                        state <= state_C;
                    WHEN state_C =>
                        IF ent2 = '1' THEN
                            state <= state_A;
                        END IF;
                    WHEN OTHERS =>
                        state <= state_a;
                END CASE;
            END IF;
        END PROCESS;
END IF;

END PROCESS;
```

Resolução do Exemplo 2 ;Descrição de Máquina de Moore (continuação) com 3 estados, 2 entradas e 1 saída.

PARTE 3

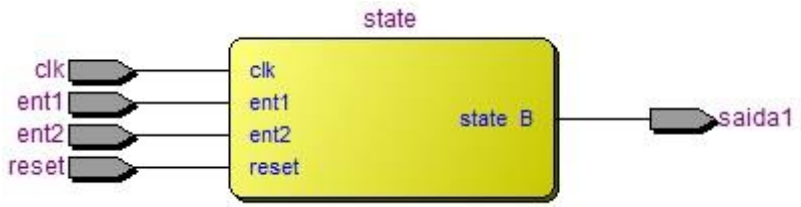
```
-- Define a saída da Máquina de Estado
-- Apenas o estado_B leva a saída ao valor 1

    WITH state SELECT
        saida1 <= '0' WHEN state_A,
                  '1' WHEN state_B,
                  '0' WHEN state_C;
END a;
```

Descrição de Máquina de Moore (continuação)

com 3 estados, 2 entradas e 1 saída.

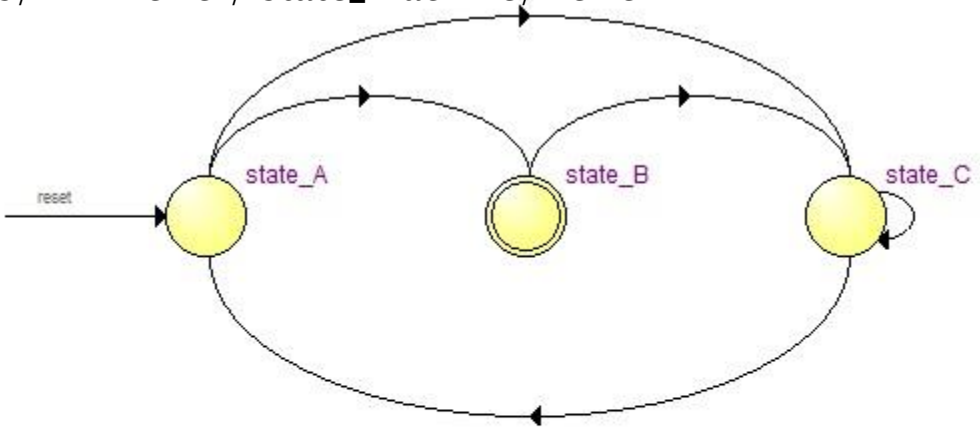
PARTE 5

- RTL Gerado em Tools/Netlist Viewers/RTL viewer



- Diagrama de Estados Gerado:

Em Tools/RTL viewer/ state_machine/viewer



	Source State	Destination State	Condition
1	state_A	state_C	(ent1)
2	state_A	state_B	(!ent1)
3	state_B	state_C	
4	state_C	state_C	(!ent2)
5	state_C	state_A	(ent2)

Resolução do Exemplo 2 ;Descrição de Máquina de Moore (continuação) com 3 estados, 2 entradas e 1 saída.

PARTE 6

Codificação dos estados:

Após a compilação, a ferramenta de síntese gera a mensagem informando qual a atribuição de estados estabelecida para a descrição "est_maq":

state_A = "00", state_B= "01" e state_C = "10"

	Name	state_bit_1	state_bit_0
1	state_A	0	0
2	state_B	1	0
3	state_C	0	1

state_A recebeu o valor "00" porque foi o primeiro valor listado na declaração “TYPE”.