Laboratório de Eletrônica Digital
SEL-EESC-USP

EESC · USP

Introdução a VHDL

Aula 7

Professora Luiza Maria Romeiro Codá

TYPE – Conceito

Usando a palavra reservada TYPE é possível definir tipos personalizados. Por exemplo, pode-se definir tipos **enumerados** (Enumerated) e tipos **compostos** (Array).

Tipos enumerados definem uma lista de valores, e são especialmente úteis na descrição de **Máquinas de Estados**.

```
-- Declaração do tipo enumerado "temperatura"
TYPE temperatura IS (baixa, media, alta);
-- Atribuição de valores a uma variável do tipo "temperatura"
Temp_Forno <= media;
```

Tipos compostos são usados para definir vetores n-dimensionais. O tipo BIT_VECTOR, por exemplo, é um tipo declarado no pacote padrão (work) VHDL, e é um tipo composto por elementos do tipo BIT.

```
-- Declaração do tipo "vetor_temperatura", vetor com 8 elementos do tipo "temperatura"
TYPE vetor_temperatura IS ARRAY (0 TO 7) OF temperatura;
```

TYPE – Declaração e uso de tipos compostos (vetores)

Um objeto da classe ARRAY obedece a mesma sintaxe já vista para objetos do tipo BIT_VECTOR e STD_LOGIC_VECTOR. A declaração de um vetor contém seu **nome**, os **limites do índice**, e o **tipo**.

```
-- Declaração geral de um vetor
TYPE <nome> IS ARRAY <limites_índice> OF <tipo>;
```

Exemplo de aplicação:

```
ENTITY teste IS
END teste;

ARCHITECTURE teste_v OF teste IS
    -- Limites especificados. Vetor 4x1
  TYPE vetor1 IS ARRAY (0 TO 3) OF BIT;
    -- Limites não especificados. Vetor ?x4
  TYPE vetor2 IS ARRAY (NATURAL RANGE <>) OF BIT;-- arranjo de vetor de 1 bit com
                                                 --limite em aberto

    SIGNAL sinal1 : vetor1;
    SIGNAL sinal2 : vetor2(7 DOWNTO 0); -- Definindo limites. Vetor 8x4

BEGIN
    sinal1 <= "0101"; sinal2 <= "11110000";
END         ;
```

TYPE – Declaração e uso de tipos compostos: vetores compostos de elementos do tipo vetor

Também é possível declarar vetores compostos por elementos do tipo vetor, obtendo-se, efetivamente, vetores n-dimensionais. O referenciamento aos elementos do conjunto são feitos com índices adicionais: vetor(índice1)(índice2).

```
ENTITY teste_v IS
END teste_v;

ARCHITECTURE a OF teste_v IS
    -- Vetor bidimensional
    TYPE vetor_2Da IS ARRAY (0 TO 1) OF BIT;
    TYPE vetor_2Db IS ARRAY (0 TO 2) OF BIT_VECTOR(3 DOWNTO 0);
    TYPE vetor_3D  IS ARRAY (0 TO 1) OF vetor_2Db; -- Vetor tridimensional,
-- vetor_3D é composto de 2 elementos do tipo vetor_2Db
    SIGNAL sinal_2D : vetor_2Da;
    SIGNAL sina2_2D : vetor_2Db;
    CONSTANT const_a :vetor_2Db :=(0 TO 2 =>(“0000”);
        --const_a = (“0000,”0000”,”0000”)
    CONSTANT const_b :vetor_2Db :=(0 TO 1 => “0000”, OTHERS => ‘0’,‘1’,‘1’,‘1’);
--const_a = (“0000,”0000”,”0111”)
BEGIN
    sinal_2D <= "11";
    sina2_2D <= constante_2D;
    sinal_2D <= ("0001","0010","0100"); --(sinal_2D(0),sinal_2D(1),sinal_2D(2))
END a;
```

TYPE – Declaração e uso de tipos compostos (vetores)

Vetores de vetores são estruturas extremamente úteis na síntese de memórias. Seria simples, por exemplo, declarar um **sinal** do tipo vetor com 256 elementos do tipo BIT_VECTOR(7 DOWNTO 0), usado como memória **RAM** 256x8. Declarando-se uma **constante** do tipo vetor com 256 elementos do tipo BIT_VECTOR(7 DOWNTO 0), cria-se uma memória **ROM** 256x8;

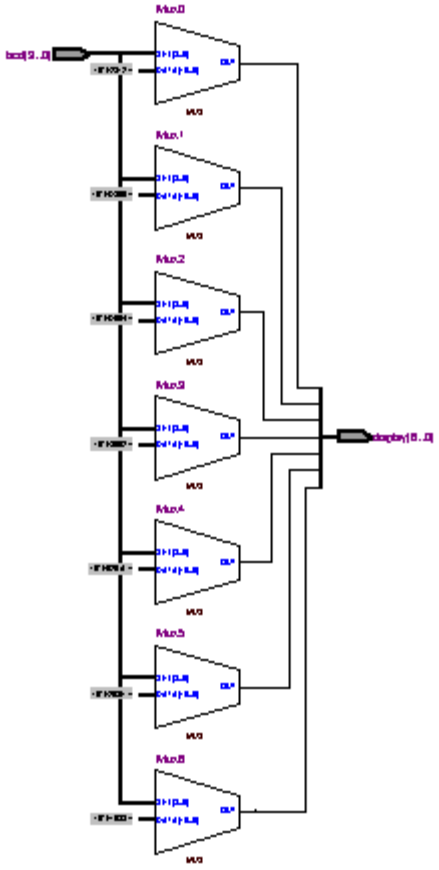
Apesar de a declaração de um vetor de vetores efetivamente criar um objeto multidimensional, o vetor ainda é considerado um objeto unidimensional. Conjuntos propriamente multidimensionais normalmente não são suportados pelas ferramentas de síntese.

TYPE – Exemplo: Decodificador BCD/7 Segmentos

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.numeric_std.all;
USE ieee.std_logic_unsigned.all;
-- Criar um decodificador bcd 7 segmentos usando matriz
-- Criar um tipo
ENTITY display_matriz IS
    PORT(bcd     : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);
         display : OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END  display_matriz;

ARCHITECTURE a OF display_matriz IS
SIGNAL indice : INTEGER RANGE 0 TO 15;
-- Criação do tipo "matriz", que é um inteiro contendo 16 vetores
-- de 7 bits cada
TYPE matriz IS ARRAY (INTEGER RANGE 0 TO 15) OF STD_LOGIC_VECTOR(6 DOWNTO 0);
-- Criação de uma constante do tipo "matriz" com a atribuição dos valores
CONSTANT Display_hexa : matriz := -- Segmentos do display abcdefg
("0000001","1001111","0010010","0000110","1001100","0100100","0100000","0001111",
 "0000000","0001100","0001000","1100000","0110001","1000010","0110000","0111000");
BEGIN
          display <= display_hexa(conv_integer(indice));
          indice <= conv_integer (bcd);
END a;
--bcd é um sinal do tipo std_logic_vector de 4 bits. A função CONV_INTEGER recebe o sinal bcd que é
--um std_logic_vector como parâmetro e retorna o sinal inteiro equivalente indice, que é então usado
--como índice para acesso ao vetor constante de conversão de inteiro para formato BCD (decimal
--codificado em binário, em inglês binary coded decimal) em 8 bits. Por exemplo, suponha que o bcd
--contenha 0010. Sua conversão para inteiro (CONV_INTEGER) resulta em 2. Na posição 2 da constante
-- Display_hexa existe "0010010" , o que resulta nos dígitos decimal 2 no display anodo comum
```

TYPE – Exemplo: Circuito Gerado Decodificador BCD/7 Segmentos



Máquina de estados

Uma máquina de estados é um circuito sequencial que transita numa sequência predefinida de estados. A transição entre os estados é comandada por um sinal de controle. 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.

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

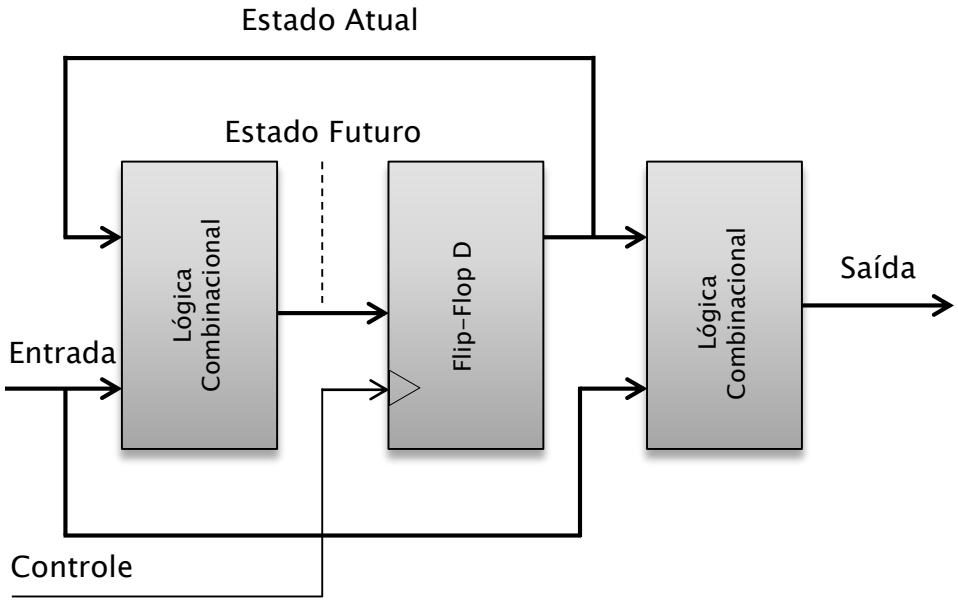
- ✓ Em máquinas **Mealy**, as saídas dependem do **estado atual** e das **entradas.**

- ✓ Em máquinas **Moore**, as saídas dependem apenas do **estado atual**.

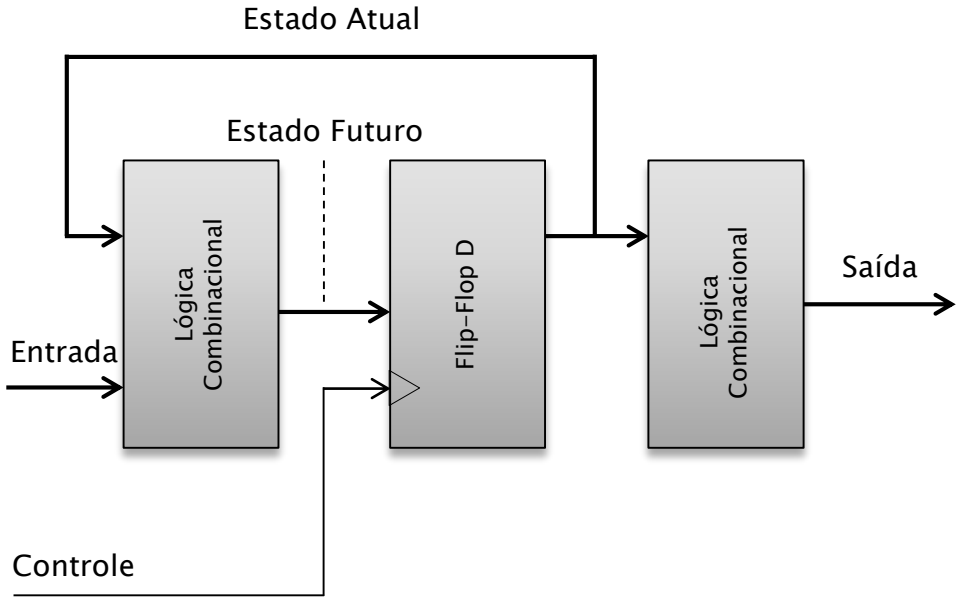
$$Mealy\text{: } Saídas = f(Estado\ Atual, Entradas)$$

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

Máquina de estados



Mealy



Moore

Máquina de estados

O *software* Quartus II 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
Exemplo Parte 1

Descrição de uma máquina de estados de Moore com 3 estados, 2 entradas e 1 saída.

```
ENTITY est_maq IS
    PORT(clk         : IN  BIT;
         reset       : IN  BIT;  -- Reset = '1' leva ao estado A
         ent1, ent2  : IN  BIT;  -- Duas entradas
         saida1      : OUT BIT); -- 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 da máquina de estados 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
```

Máquina de estados
Exemplo Parte 2

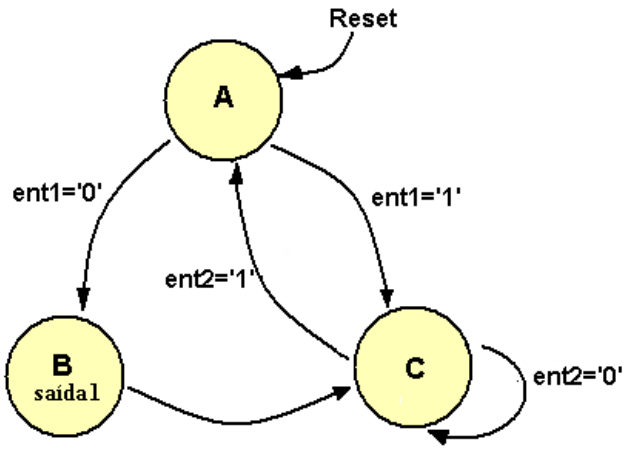
```
-- Lógica combinacional que decide qual será o próximo estado

            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;
```

Máquina de estados
Exemplo 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;
```



Máquina de estados descrita

Máquina de estados
Exemplo – Codificação dos estados

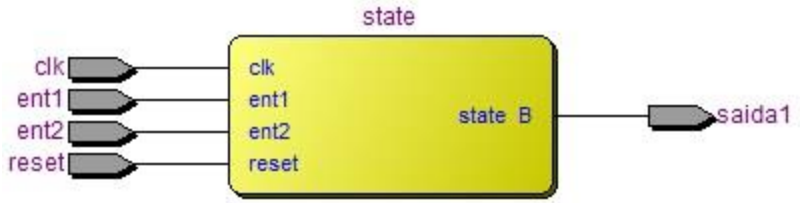
Após a compilação, a ferramenta de síntese gera a mensagem informando qual a atribuição de estados estabelecida para a descrição "est_maq":

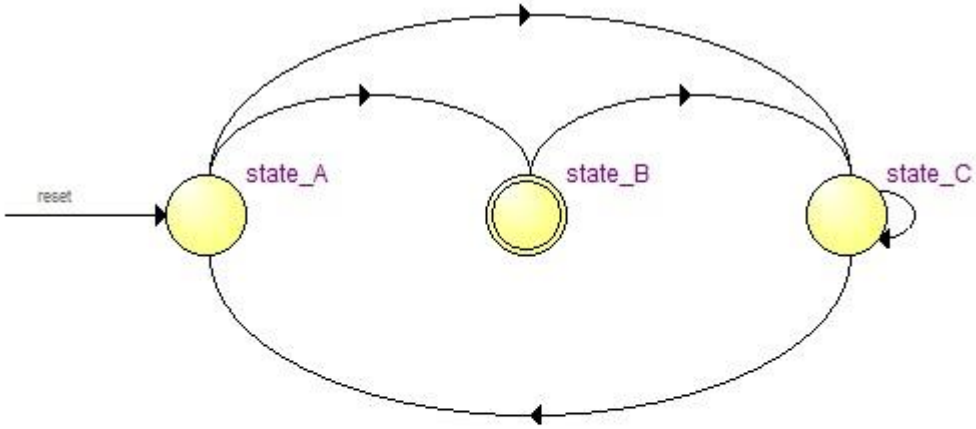
state_A = "00", state_B= "01" e state_C = "10"

	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”.

Máquina de estados
Exemplo – RTL gerado





	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)

Pratica 12: semáforo de cruzamento usando Máquina de estados