Aula 12 – Sinais e Variáveis

Tópicos da aula

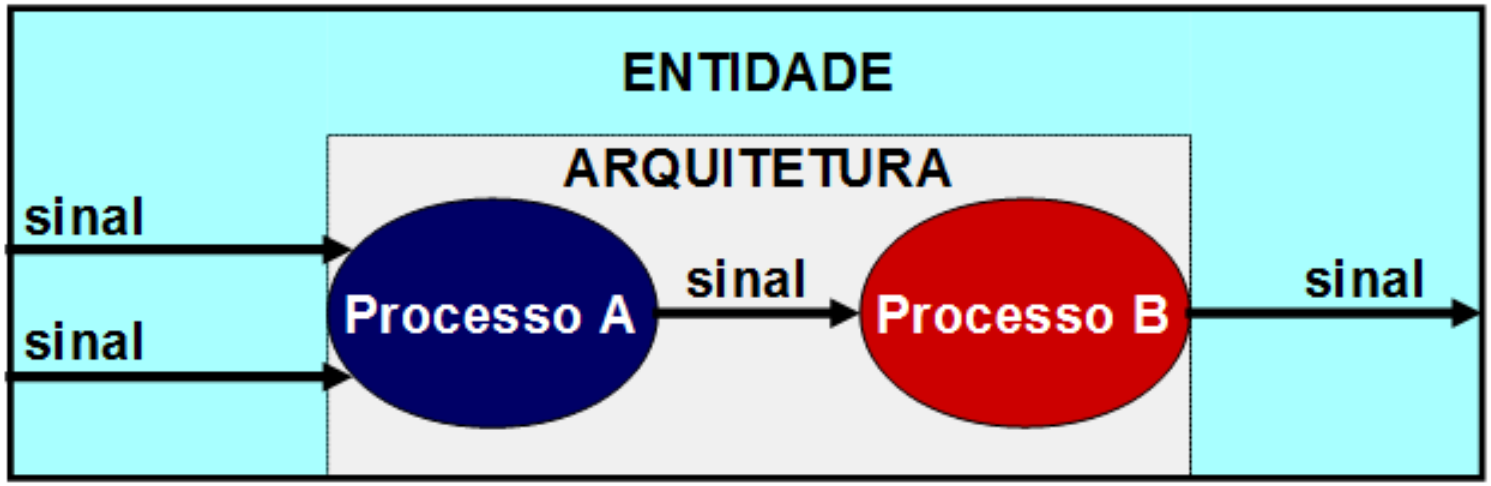
- **Sinais**
- **Processos implícitos**
- **Processos explícitos**
- **Variáveis**

Sinais

- Representam interconexão física (fio) que realizam a comunicação de informações entre os processos
- Declaração

```
SIGNAL <signal_name>: <type>;
```

- Declarados na entidade, na arquitetura ou em um pacote

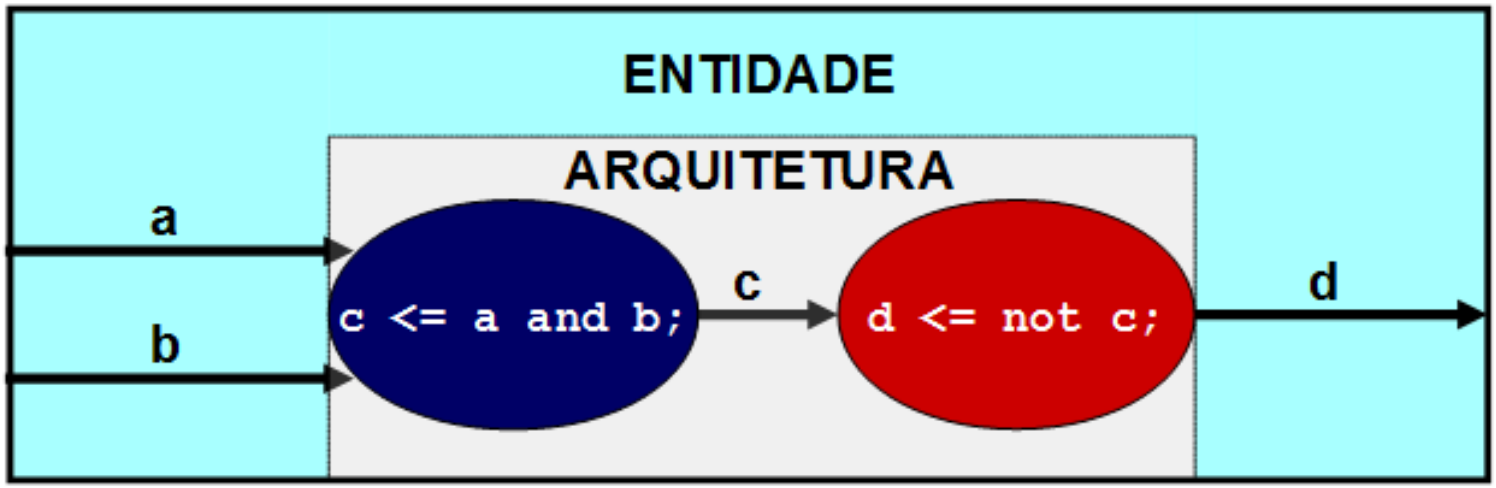


Atribuindo valores a sinais (processos implícitos)

- A atribuição é feita usando <=

```
<signal_name> <= <expression>;
```

- Uma atribuição a um sinal possui um processo implícito associado



Atribuindo valores a sinais

- Exemplo

```
SIGNAL temp : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

- Atribuindo valor a todos os bits (usar aspas duplas)

```
temp <= "10101010";
temp <= X"AA";
```

- Atribuindo valor a um único bit (usar aspas simples)

```
temp(7) <= '1';
```

- Atribuindo valor a um grupo de bits

```
temp (7 downto 4)<= "1010";
```

Tipos de atribuição a sinais concorrentes. Ex.: mux 2x1

- Atribuição simples

```
s <= (a AND NOT sel) OR (b AND sel);
```

- Atribuição condicional

```
s <= a WHEN sel='0' ELSE
     b;
```

- Atribuição selecionada

```
WITH sel SELECT
 s <= a WHEN '0'
      b WHEN OTHERS;
```

No caso de `sel` ser do tipo `STD_LOGIC`, `WHEN OTHERS` garante um valor definido para `s` (ou seja `b`) quando `sel` possuir qualquer outro valor diferente de '0'

Processos Explícitos

- **Necessários na implementação de funções compostas por atribuição seqüenciais (ex. `if-then-else`)**

```
<process_label>:                                    -- (optional)
PROCESS (<sensitive_list>)
  -- Constant declarations
  -- Type  declarations
  -- Variable declarations
BEGIN
  -- Signal Assignment Statement (optional)
  -- Variable Assignment Statement (optional)
  -- Procedure Call Statement (optional)
  -- If Statement (optional)
  -- Case Statement (optional)
  -- Loop Statement (optional)
END PROCESS <process_label>;
```

Processos Explícitos

- **Um processo é executado infinitamente se não for quebrado por uma declaração `WAIT` ou por uma lista de sensibilidade**
  - Uma lista de sensibilidade infere uma declaração `WAIT` no final do processo
  - Um processo pode ter múltiplos `WAIT`, mas não pode ter um `WAIT` e uma lista de sensibilidade
- **As atribuições a sinais em um processo são efetivadas ao término da sua execução**

```
ARCHITECTURE behavior OF mux_2x1 IS
BEGIN
  PROCESS(a,b,sel)
  BEGIN
    IF (sel='0') THEN
      s <= a;
    ELSE
      s <= b;
    END IF;
  END PROCESS;
END behavior;
```

```
ARCHITECTURE behavior OF mux_2x1 IS
BEGIN
  PROCESS
  BEGIN
    IF (sel='0') THEN
      s <= a;
    ELSE
      s <= b;
    END IF;
    WAIT(a,b,sel);
  END PROCESS;
END behavior;
```

Variáveis

- Declaradas dentro de processos e constituem-se em objetos de armazenamento temporário
- Declaração

```
VARIABLE <variable_name>: <type>;
```

- A atribuição é feita usando “:=”

```
<variable_name> := <expression>;
```

- São atualizadas no momento da atribuição (não ao término do processo, como os sinais)

Atribuindo valores a variáveis

- Exemplo

```
ARCHITECTURE behavior OF mux_2x1 IS
VARIABLE temp := BIT;
BEGIN
  PROCESS(a,b,sel)
  BEGIN
    IF (sel='0') THEN
      temp := a;
    ELSE
      temp := b;
    END IF;

  s <= temp;
  END PROCESS;
END behavior;
```

Sinais x Variáveis

	Sinal	Variável
Atribuição	<=	:=
Utilidade	Representa um fio no circuito	Representa um local de armazenamento temp.
Escopo	Global (comunicação entre processos)	Local (dentro do processo)
Comportamento	Atualizado no final do processo	Atualizada imediatamente

Atribuindo valores a variáveis

- Exemplo

```
VARIABLE temp : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

- Atribuindo valor a todos os bits (usar aspas duplas)

```
temp := "10101010";
temp := X"AA";
```

- Atribuindo valor a um único bit (usar aspas simples)

```
temp(7) := '1';
```

- Atribuindo valor a um grupo de bits

```
temp (7 downto 4):= "1010";
```

FIM AULA XII