Aula 03 – Linguagem de hardware e síntese de circuitos

Tópicos da aula

- **Operadores Lógicos e Aritméticos**
- **Tipos de dados**

Sinais INTERNOS e EXTERNOS

- Na aula anterior criamos um **componente** (porta lógica AND) com duas entradas e uma saída
- Na ENTIDADE definiu-se quais **sinais** “entram” ou “saem” deste componente
- **Pergunta**: É possível ter “vários” componentes dentro de um mesmo componente?

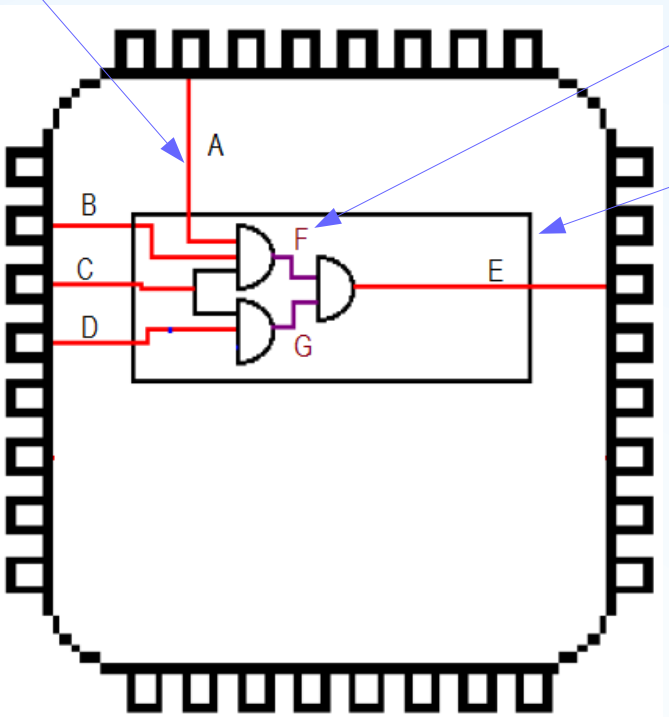
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

ENTITY and2 IS

   PORT(
         A          : IN STD_LOGIC;
         B          : IN STD_LOGIC;
         C          : OUT STD_LOGIC
   );
END and2;

ARCHITECTURE behavioral OF and2 IS

BEGIN

   C <= A and B;

END behavioral;
```

Sinais INTERNOS e EXTERNOS

Sinais **externos** do componente (A, B, C, D, E)

Sinais **internos** do componente (F, G)

Componente

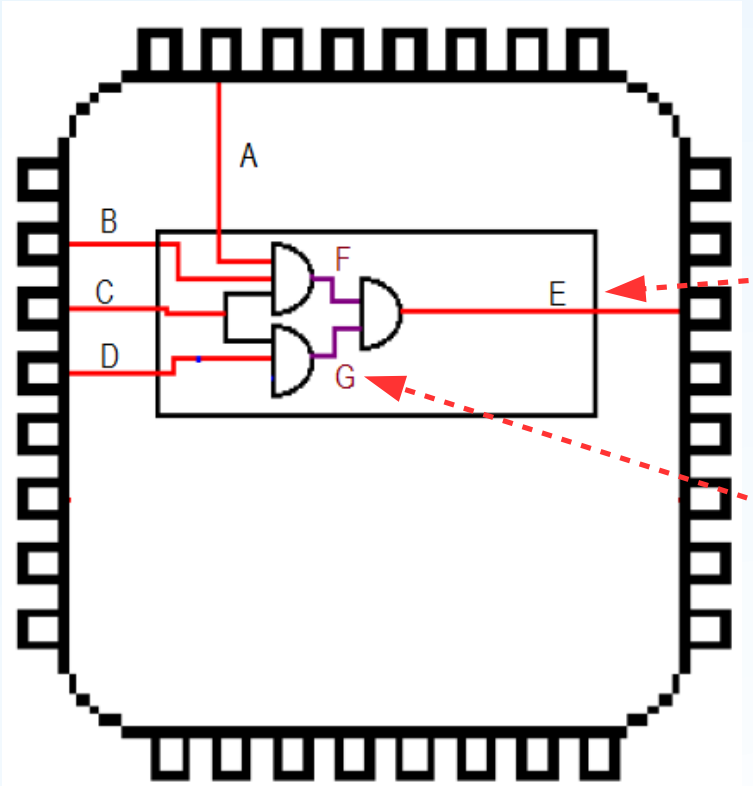


Sinais INTERNOS e EXTERNOS

- **EXTERNOS:** são sinais de entrada e saída do componente (na declaração da ENTIDADE)

- **INTERNOS:** são sinais usados para conectar os pinos de dois ou mais componentes dentro do FPGA, ou conectar sinais internos de um componente interno ao FPGA (estão dentro da ARQUITETURA)

Exemplo: declaração de sinais




```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;


ENTITY COMPONENTE IS

   PORT(
         A          : IN STD_LOGIC;
         B          : IN STD_LOGIC;
         C          : IN STD_LOGIC;
         D          : IN STD_LOGIC;
         E          : OUT STD_LOGIC
   );
END COMPONENTE;


ARCHITECTURE behavioral OF COMPONENTE IS

   signal F : STD_LOGIC;
   signal G : STD_LOGIC;

BEGIN

   F <= A and B and C;
   G <= C and D;
   E <= F and G;

END behavioral;
```

Outros “Tipos” de sinais internos

- **SIGNAL**
- **CONSTANT**
- **VARIABLE**
- **REAL**
- **...**

Exemplo: Tipos

```
ARCHITECTURE behavioral OF COMPONENTE IS

    signal F : STD_LOGIC;
    signal G : STD_LOGIC;

    constant K : integer := 7;

BEGIN
```

Tipos Escalares

Tipos definidos	Valor	Exemplos
STD_LOGIC	1, 0, Z	
BOOLEAN	Verdadeiro, falso	TRUE, FALSE
INTEGER	$(-2^{31} -1) \leq x \leq (2^{31} -1)$	123
NATURAL	$0 \leq x \leq (2^{31} -1)$	
REAL	$-3,65^{47} \leq x \leq 3,65^{47}$	
TIME	Pico, nano, micro, mili, etc.	1 us

Tipos Compostos

Tipos definidos	Valor	Exemplos
STD_LOGIC_VECTOR	1, 0, Z	“011010010100”
STRING	Tipo caracter	“texto qualquer”

Exercícios

- **Implementar uma porta lógica OR 4x1 usando STD_LOGIC_VECTOR**
- **Implementar uma porta lógica AND 8x1 usando STD_LOGIC_VECTOR**

Definição de novos tipos

- **A linguagem permite a criação de novos tipos**
- **Aplicações**:
  - facilitar a leitura do código: estados de uma máquina
  - definir novos tipos físicos: resistência, capacitância etc.
  - novos tipos compostos: definição de memórias
- **Declaração**: palavra reservada TYPE
- **Exemplo**:

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);
SIGNAL abc : estado := parado;
```

- é definido um novo tipo estado

```
TYPE estado IS (parado, inicio, caso_1, caso_2, caso_3);
```

- declaração de um sinal do tipo estado

```
SIGNAL abc : estado := parado;
```

- valores possíveis para o sinal abc: parado, inicio, caso_1, caso_2, caso_3

Operadores

- **Divididos em classes**:
  - as classes definem a precedência dos operadores
  - operadores de uma mesma classe: igual precedência
- **Maior precedência**: classe diversos
- **Menor precedência**: classe lógicos
- **Operador not**: operador lógico; está na classe diversos devido à precedência

classe	operadores
lógicos	and or nand nor xor xnor
relacionais	= /= < <= > >=
deslocamento	sll srl sla sra rol ror
adição	+ – &
sinal	+ –
multiplicação	* / mod rem
diversos	** abs not

FIM AULA 3