Descrição de Circuitos Combinatórios em VHDL

Prof. Mário Véstias

Índice

- Descrição de circuitos combinatórios em VHDL
- Descrição estrutural VHDL
- Simulação de circuitos combinatórios

Descrição Estrutural

- A descrição de um circuito combinatório em VHDL pode compreender apenas uma função lógica (circuitos combinatórios simples).
- Um circuito combinatório é geralmente constituído por vários circuitos mais simples interligados entre si.
- A linguagem VHDL permite descrever o circuito como uma interligação de vários componentes – descrição estrutural;
- Descreve-se cada um dos componentes em funções lógicas e depois descreve-se como os blocos estão interligados;

Declaração de Componentes

- Uma determinada entidade pode ser usada como um **componente** dentro de outra entidade;
- O componente deve ser **declarado** e depois **instanciado** uma ou mais vezes;
- Uma determinada entidade pode ser definida como a interligação de vários componentes;
- Uma determinada entidade pode ser definida como a interligação de componentes e atribuições lógicas;

Declarar Componente em VHDL

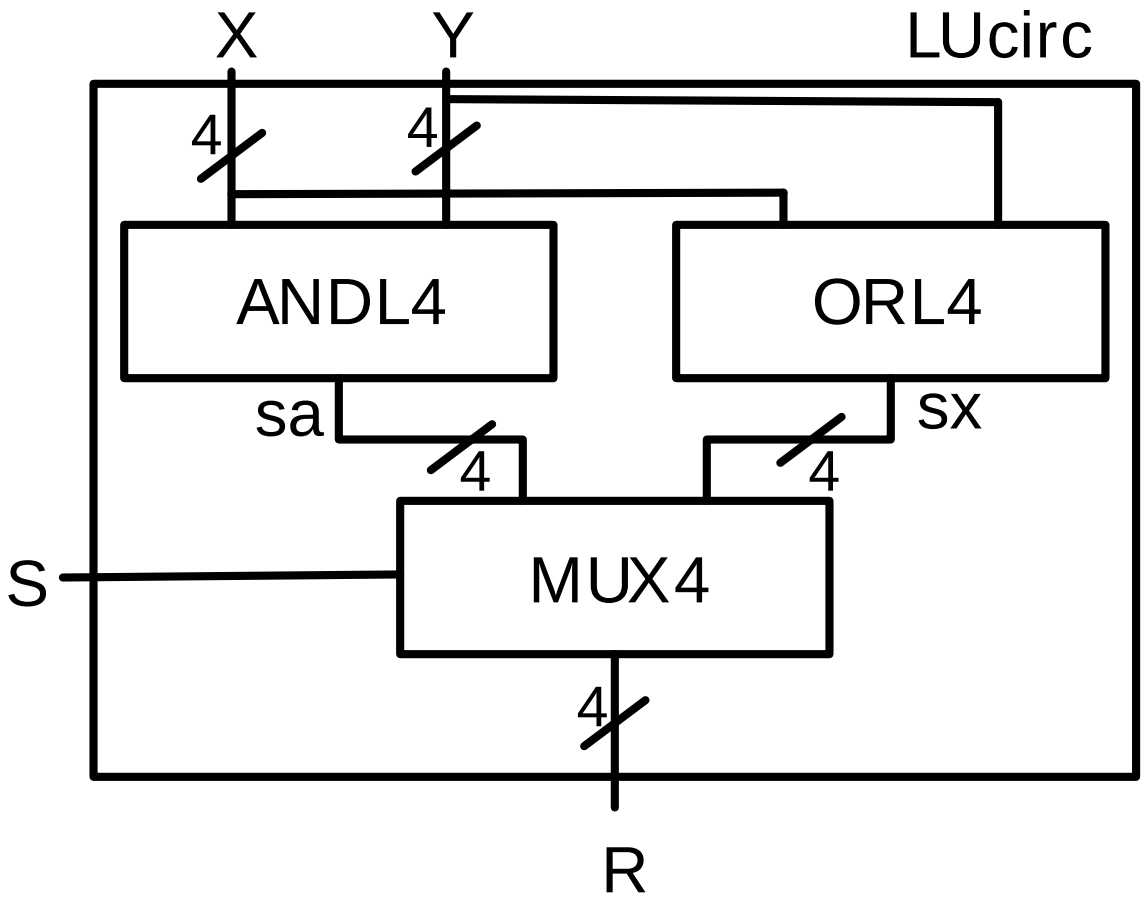
```
entity circuitoTopo is
  port (...
  );
end circuitoTopo;

architecture nome_arquitetura of circuitoTopo is

COMPONENT NOME_DO_COMPONENTE
port (...
  );
END COMPONENT;

-- declaração de sinais

begin
-- descrição do circuito digital
end nome_arquitetura;
```

Instanciar Componente em VHDL

```
architecture nome_arquitetura of circuitoTopo is

COMPONENT NOME_DO_COMPONENTE
port (...
  );
END COMPONENT;

-- declaração de sinais

begin

NOME_INSTANCIA: NOME_DO_COMPONENTE port map(
  PORTO_COMPONENTE => SINAL_CIRCUITO;
  PORTO_COMPONENTE => SINAL_CIRCUITO;
  ...
);
end nome_arquitetura;
```

Exemplo de Descrição Estrutural em VHDL

Circuito Lógico - LUcirc



Descrição do Circuito ANDL4

ISEL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;

entity ANDL4 is
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
       B : in STD_LOGIC_VECTOR(3 downto 0);
       O : out STD_LOGIC_VECTOR(3 downto 0)
  );
end ANDL4;

architecture arq_andl4 of ANDL4 is

begin
O(0) <= A(0) and B(0);
O(1) <= A(1) and B(1);
O(2) <= A(2) and B(2);
O(3) <= A(3) and B(3);
end arq_andl4;
```

Descrição do Circuito ORL4

ISEL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;

entity ORL4 is
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
       B : in STD_LOGIC_VECTOR(3 downto 0);
       O : out STD_LOGIC_VECTOR(3 downto 0)
  );
end ORL4;

architecture arq_orl4 of ORL4 is

begin
O(0) <= A(0) or B(0);
O(1) <= A(1) or B(1);
O(2) <= A(2) or B(2);
O(3) <= A(3) or B(3);
end arq_orl4;
```

Descrição do Circuito MUX4

ISEL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;

entity MUX4 is
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
       B : in STD_LOGIC_VECTOR(3 downto 0);
       sel : in std_logic;
       O : out STD_LOGIC_VECTOR(3 downto 0)
  );
end MUX4;

architecture arq_mux4 of mux4 is

begin
O(0) <= (not sel and A(0)) or (sel and B(0));
O(1) <= (not sel and A(1)) or (sel and B(1));
O(2) <= (not sel and A(2)) or (sel and B(2));
O(3) <= (not sel and A(3)) or (sel and B(3));
end arq_mux4;
```

Descrição do Circuito Lucirc – Versão 1

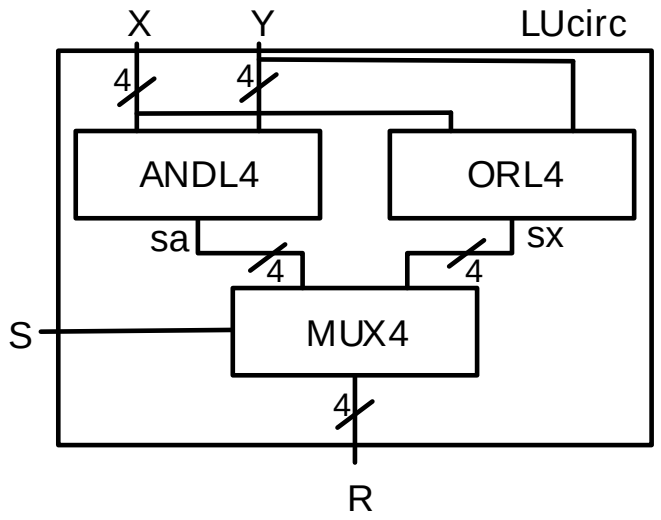
ISEL

```
entity LUcirc is
  port(X, Y : in STD_LOGIC_VECTOR(3 downto 0);
          S : in STD_LOGIC;
          R : out STD_LOGIC_VECTOR(3 downto 0)
  );
end LUcirc;

architecture arq_LUcirc of LUcirc is

component ANDL4
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
          B : in STD_LOGIC_VECTOR(3 downto 0);
          O : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component ORL4
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
          B : in STD_LOGIC_VECTOR(3 downto 0);
          O : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component MUX4
  port(A : in STD_LOGIC_VECTOR(3 downto 0);
          B : in STD_LOGIC_VECTOR(3 downto 0);
          sel : in std_logic;
          O : out STD_LOGIC_VECTOR(3 downto 0));
end component;
```

Descrição do Circuito LUcirc – Versão 1 (cont.)

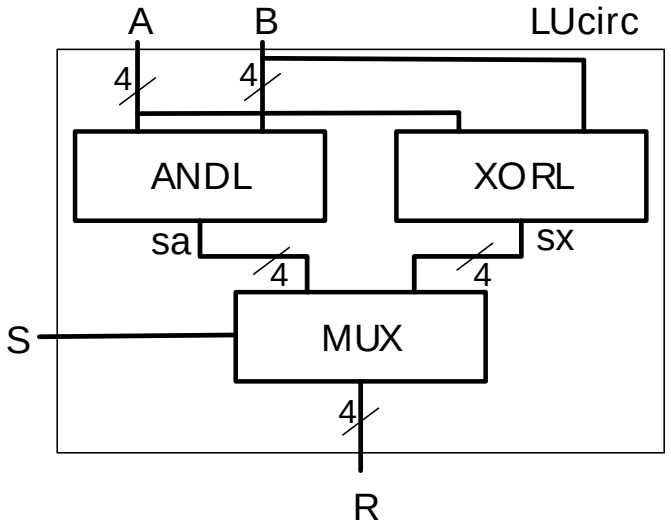
ISEL

```
SIGNAL sa, sx : std_logic_vector(3 downto 0);

BEGIN

UANDL4: ANDL4 port map (
        A => X,
        B => Y,
        O => sa );

UORL4: ORL4 port map (
        A => X,
        B => Y,
        O => sx );

UMUX4: MUX4 port map (
        A => sa,
        B => sx,
        sel => S,
        O => R);

end arq_Lucirc;
```



Descrição do Circuito LUcirc – Versão 2


```
SIGNAL sa, sx : std_logic_vector(3 downto 0);

BEGIN

UANDL4: ANDL4 port map (
        A => X,
        B => Y,
        O => sa );

UORL4: ORL4 port map (
        A => X,
        B => Y,
        O => sx );

-- funções lógicas do multiplexer
R(0) <= (not S and sa(0)) or (S and sx(0));
R(1) <= (not S and sa(1)) or (S and sx(1));
R(2) <= (not S and sa(2)) or (S and sx(2));
R(3) <= (not S and sa(3)) or (S and sx(3));

end arq_Lucirc;
```



Simulação de Circuitos em VHDL

Como Simular um Circuito VHDL

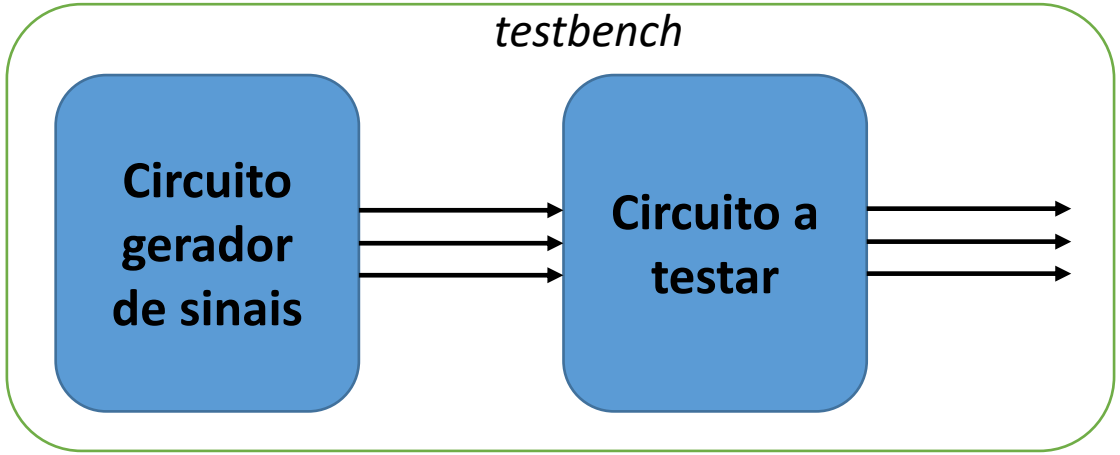
ISEL

- Aplicar sinais à entrada e verificar os sinais de saída;
- É necessário descrever um circuito para gerar os sinais digitais de entrada;
- O circuito deve gerar todas as combinações de entrada ou, caso sejam muitas, as combinações mais “críticas”;

Ficheiro de Simulação - *testbench*

ISEL

- O circuito gerador de sinais de simulação é descrito em VHDL;
- Deve instanciar o circuito a testar;
- Não tem entradas nem saídas.



Descrição do *testbench*

ISEL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;

entity circuito_testbench is
end circuito_testbench;

architecture nome_arquitetura of circuito_testbench is
-- Declaração do componente
-- Declaração de sinais

begin
-- instanciar componente
process
begin
--gerar sinais
wait for 10 ns;
--gerar sinais
wait;
end process;

end nome_arquitetura;
```

Testbench do Exemplo *LUcirc*

ISEL

```
LIBRARY IEEE;
use IEEE.std_logic_1164.all;

entity LUcirc_tb is
end LUcirc_tb;

architecture teste of LUcirc_tb is
component LUcirc
  port(X, Y : in STD_LOGIC_VECTOR(3 downto 0);
       S : in STD_LOGIC;
       R : out STD_LOGIC_VECTOR(3 downto 0)
  );
end component;

signal X, Y : std_logic_vector(3 downto 0);
signal S : std_logic;
signal R : std_logic_vector(3 downto 0);

begin

U0 : LUcirc port map (X=> X, Y => Y,
                      S => S, R => R)
```

```
process
begin
 X <= "0101"; Y <= "1100"; S <= '0';
 wait for 10 ns;
 X <= "0101"; Y <= "1100"; S <= '1';

wait;

end process;
end teste;
```