Arquitetura e Projeto VLSI I

UFRGS – PPGC - 2003

Parte de Projeto - Marcelo Johann

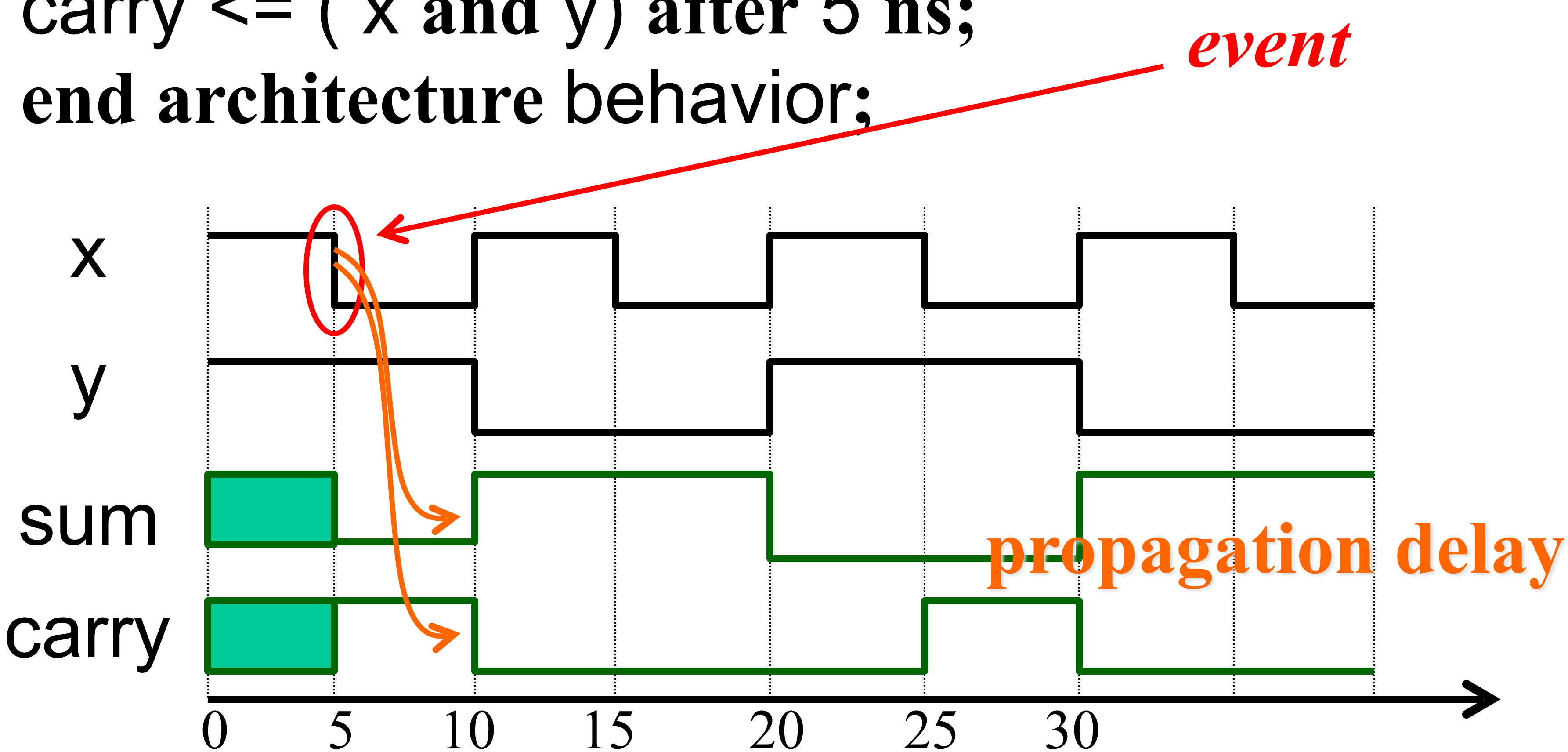
AULA 3

Conteúdo da Aula

- Signals and delay models
  - signal versus variable
  - inertial, *reject*, transport and delta delays
  - resolved and unresolved signals
- Concurrent signal assignment statemens
- Conditional signal assignment
  - order, uncovered, unaffected
- Selected signal assignment
  - order, covered, others

Signal assignment and inertial delay

```
architecture behavior of half_adder is
begin
sum <= ( x xor y) after 5 ns;
carry <= ( x and y) after 5 ns;
end architecture behavior;
```



Inertial delay with reject pulse width

```
architecture behavior of half_adder is
begin
sum <= reject 2 ns inertial ( x xor y) after 5 ns;
carry <= reject 2 ns inertial ( x and y) after 5 ns;
end architecture behavior;
```

Transport delay

```
architecture wire_delay of half_adder is
signal s1, s2: std_logic;
begin
s1 <= ( x xor y) after 2 ns;
s2 <= ( x and y) after 2 ns;
sum <= transport s1 after 4 ns;
carry <= transport s2 after 4 ns;
end architecture wire_delay ;
```

Delta delays

```
library IEEE;
use IEEE.std_logic_1164.all;
entity combi is
port ( in1, in2: in std_logic;
       out: out std_logic );
end entity combi;
```

Delta delays

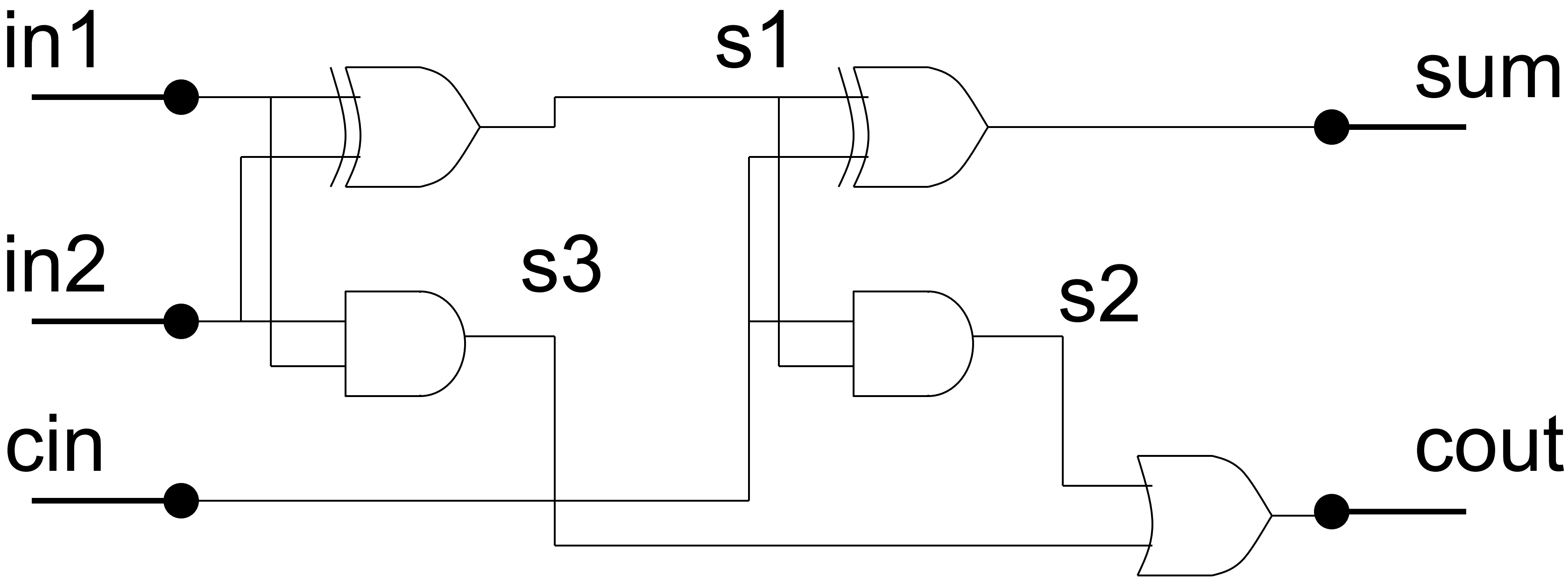
```
architecture delta_delay of combi is
signal s1, s2, s3, s4: std_logic := '0';
begin
s1 <= not in1 ;
s2 <= not in2 ;
s3 <= not ( s1 and in2) ;
s4 <= not ( s2 and in1) ;
out <= not s2 ( s3 and s4) ;
-- after 0 ns
end architecture delta_delay ;
```

Exemplo de somador completo

```
library IEEE;
use IEEE.std_logic_1164.all;

entity full_adder is
port ( in1, in2, cin: in std_ulogic;
       sum, cout: out std_ulogic );
end entity full_adder;
```

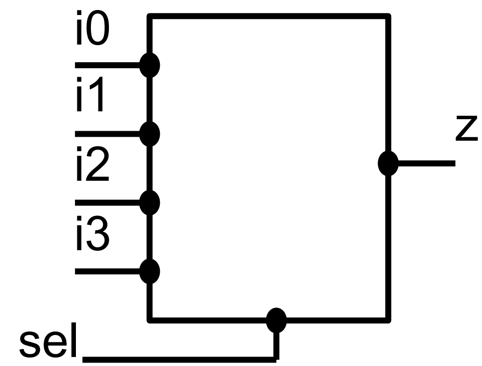
Exemplo de somador completo



Exemplo de somador completo

```
architecture dataflow of full_adder is
signal s1, s2, s3: std_ulogic;
constant gate_delay: Time := 5 ns;
begin
L1: s1 <= ( in1 xor in2) after gate_delay;
L2: s2 <= ( cin and s1) after gate_delay;
L3: s3 <= ( in1 and in2) after gate_delay;
L4: sum <= ( x xor y) after gate_delay;
L5: cout <= ( x or y) after gate_delay;
end architecture dataflow;
```

Exemplo de multiplexador 4 para 1




```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux4 is
port ( i0, i1 : in std_logic_vector (7 downto 0);
        i2, i3: in std_logic_vector (7 downto 0);
        s0,s1: in std_logic;
        z : out std_logic_vector (7 downto 0)
        );
end entity mux4;
```

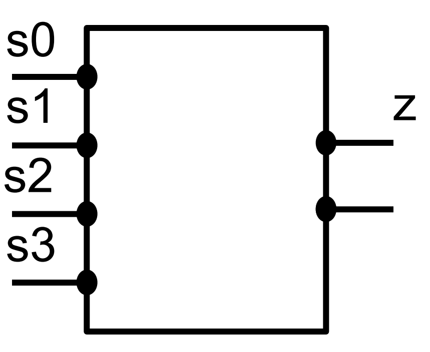
Exemplo de multiplexador 4 para 1

```
architecture behavior of mux4 is
begin
Z <=  i0 after 5 ns when s0='0' and s1='0' else
      i1 after 5 ns when s0='0' and s1='1' else
      i2 after 5 ns when s0='1' and s1='0' else
      i3 after 5 ns when s0='1' and s1='1' else
      "00000000" after 5 ns;
end architecture behavior ;
```

Exemplo de codificador de prioridade

```
library IEEE;
use IEEE.std_logic_1164.all;

entity priority_encoder is
port ( s0,s1,s2,s3 : in std_logic;
        z : out std_logic_vector (1 downto 0)
        );
end entity priority_encoder ;
```



Exemplo de codificador de prioridade

```
architecture behavior of priority_encoder is
begin
Z <=  "00" after 5 ns when s0='1' else
      "01" after 5 ns when s1='1' else
      "10" after 5 ns when s2='1' else
      "11" after 5 ns when s3='1' else
      "00" after 5 ns;
end architecture behavior ;
```

Unaffected gera lógica não combinacional

```
architecture behavior of priority_encoder is
begin
Z <=  “00” after 5 ns when s0=‘1’ else
      “01” after 5 ns when s1=‘1’ else
      unaffected when s2=‘1’ else
      “11” after 5 ns when s3=‘1’ else
      “00” after 5 ns;
end architecture behavior ;
```

Exemplo de banco de registradores

```
library IEEE;
use IEEE.std_logic_1164.all;

entity rom_reg_file is
-- read only register file
port ( addr : in std_logic_vector (2 downto 0);
       out : out std_logic_vector (31 downto 0)
       );
end entity rom_reg_file ;
```

Exemplo de banco de registradores

```
architecture behavior of rom_reg_file is
signal r0,r1,r2,r3: std_logic_vector (2 downto 0)
        := x“12345678”; begin
with addr select
out <=      r0 after 5 ns when ‘000’ ,
            r1 after 5 ns when ‘001’ ,
            r2 after 5 ns when ‘010’ ,
            r3 after 5 ns when ‘011’ ,
            r3 after 5 ns when others ;
end architecture behavior ;
```

Exemplo de banco de registradores

```
architecture behavior of rom_reg_file is
signal r0,r1,r2,r3: std_logic_vector (2 downto 0)
        := x“12345678”; begin
with addr ( 1 downto 0 ) select
out <=      r0 after 5 ns when ‘00’ ,
            r1 after 5 ns when ‘01’ ,
            r2 after 5 ns when ‘10’ ,
            r3 after 5 ns when ‘11’ ,
            r3 after 5 ns when others ;
end architecture behavior ;
```

Para próxima semana

Ver tutorial EVITA de 1 a 5

Exemplos VHDL

Iniciar uso de simulador VHDL