# 数字系统设计

郑海永

上课时间地点：周 2/9 10 节/3308

电子信息科学与技术 2010 级和 2011 级

中国海洋大学 电子工程系

2013 年 4 月

# VHDL

# 目录

# 内容提要

# 内容提要

# 半加器

# 半加器

| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

# 半加器

| A | B | S | C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$$S = A \oplus B \tag{1}$$

$$C = A \cdot B \tag{2}$$

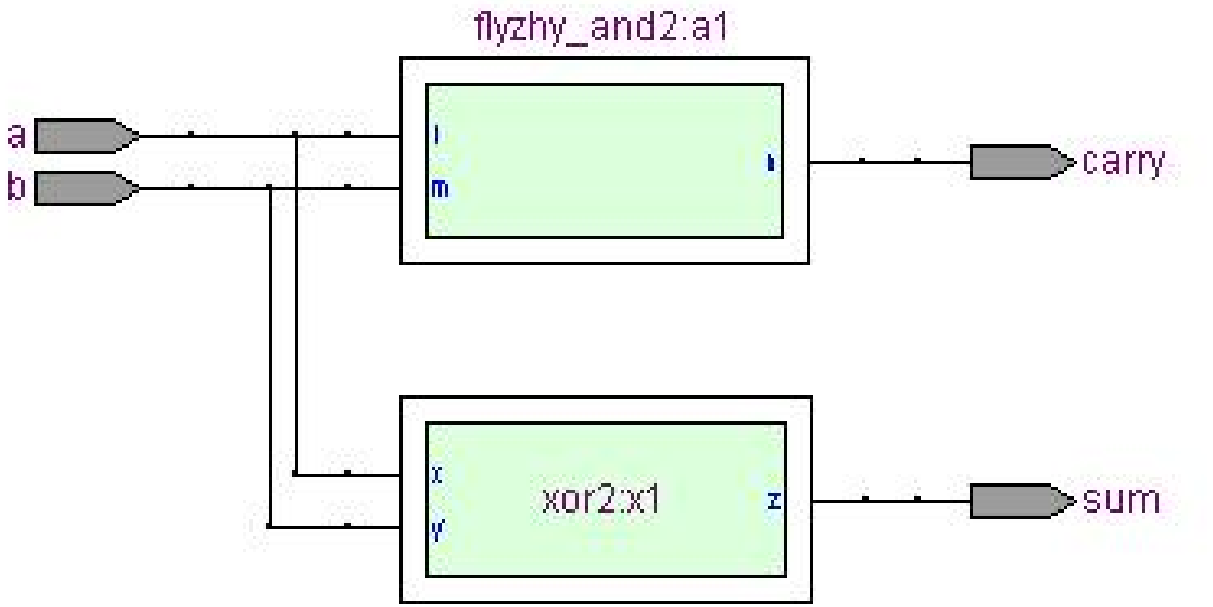
# 实体声明

```
1 ENTITY half_adder IS
2   PORT(a, b: IN Bit;
3        sum, carry: OUT Bit);
4 END half_adder;
```

# 结构体—结构描述 ⇔ 逻辑图

```
1  ARCHITECTURE ha_structure OF half_adder IS
2    COMPONENT xor2
3      PORT(x, y: IN Bit;
4           z: OUT Bit);
5    END COMPONENT;
6    COMPONENT and2
7      PORT(l, m: IN Bit;
8           n: OUT Bit);
9    END COMPONENT;
10 BEGIN
11   x1: xor2 PORT MAP(a, b, sum);
12   a1: and2 PORT MAP(a, b, carry);
13 END ha_structure;
```

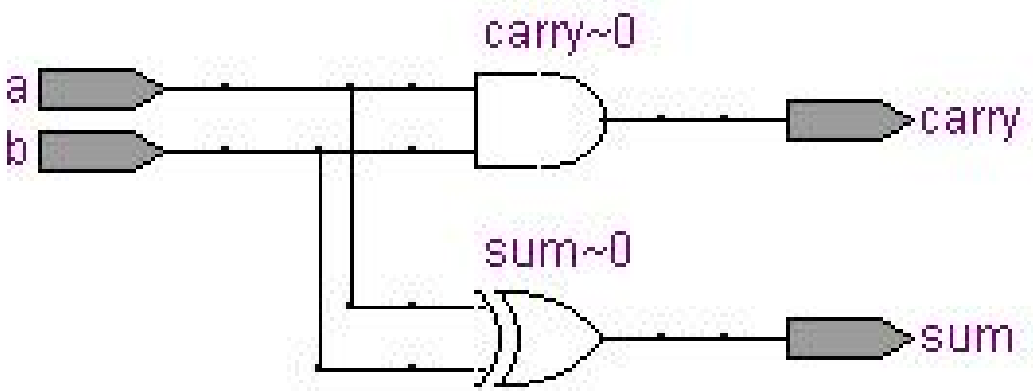
# 半加器结构描述



# 结构体—数据流描述 ⇔ 逻辑表达式（布尔方程）

```
1 ARCHITECTURE ha_concurrent OF half_adder IS
2 BEGIN
3   sum <= a XOR b AFTER 8 ns;
4   carry <= a AND b AFTER 4 ns;
5 END ha_concurrent;
```

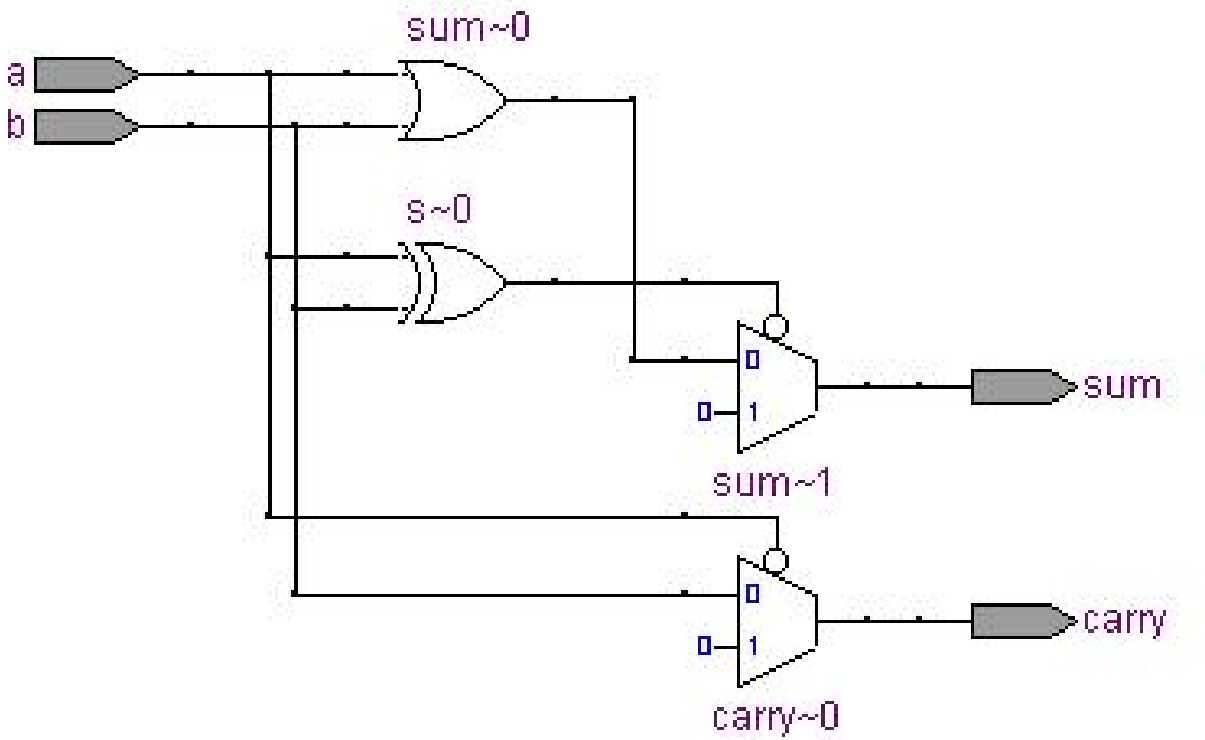
# 半加器数据流描述

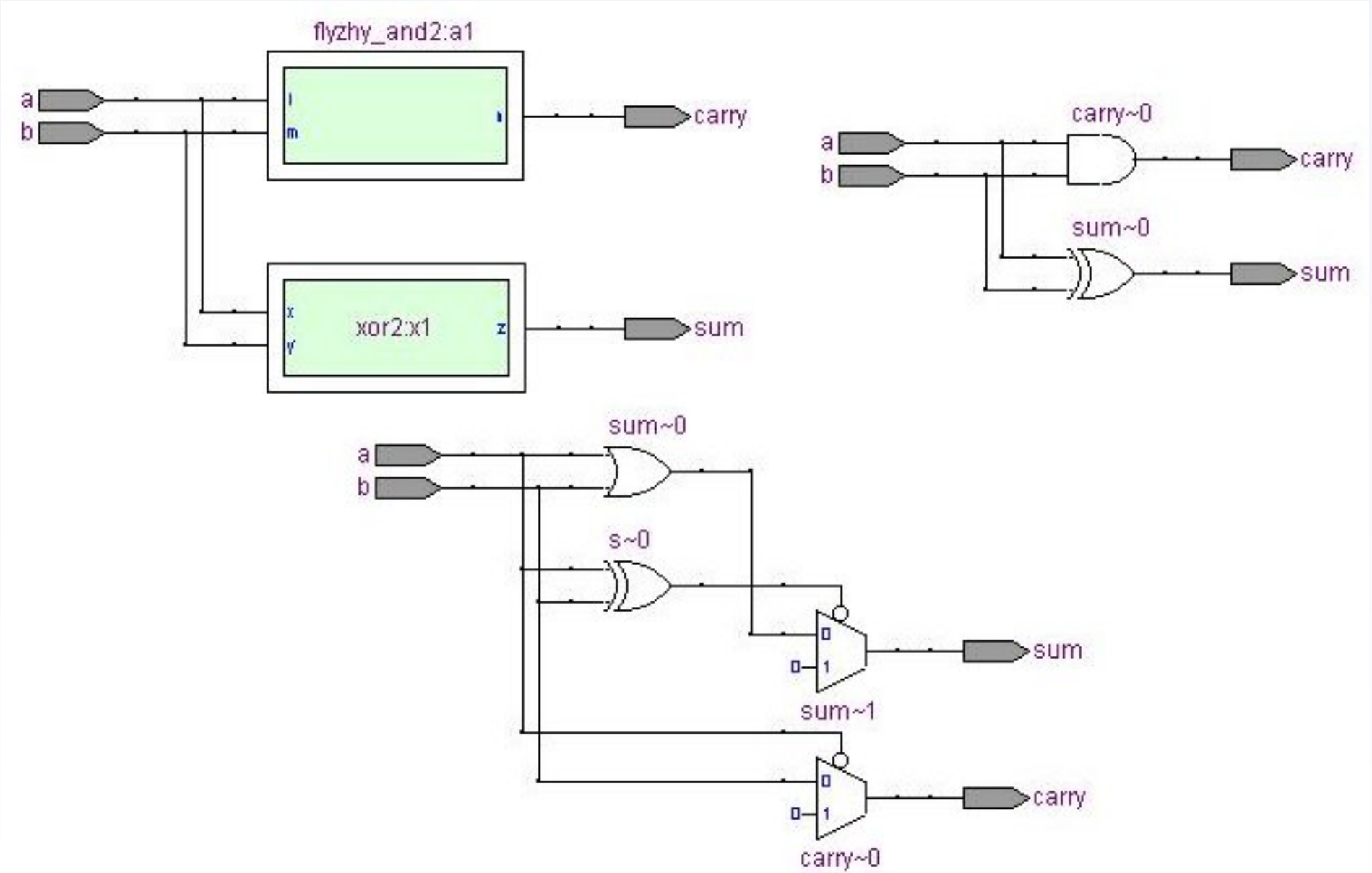


# 结构体—行为描述 ⇔ 真值表（状态图）

```
1  ARCHITECTURE ha_behavioral OF half_adder IS
2  BEGIN
3    s: PROCESS(a, b)
4    BEGIN
5      IF a = b THEN
6        sum <= '0';
7      ELSE
8        sum <= (a OR b);
9      END IF;
10   END PROCESS;
11   c: PROCESS(a, b)
12   BEGIN
13     CASE a IS
14       WHEN '0' => carry <= a;
15       WHEN '1' => carry <= b;
16     END CASE;
17   END PROCESS;
18 END ha_behavioral;
```

# 半加器行为描述




flyzhy_and2:a1
a
b
l
m
carry
x
y
xor2:x1
z
sum
carry~0
a
b
carry
sum~0
sum
sum~0
a
b
s~0
0
1
sum
sum~1
0
1
carry
carry~0

# 内容提要

# 2 选 1 多路选择器

# 2 选 1 多路选择器

$$out1 = \begin{cases} a & \text{if } c = 0, \\ b & \text{if } c = 1. \end{cases}$$

# 实体声明

```
1  ENTITY inverter IS
2    PORT(a: IN Bit; b: OUT Bit);
3  END inverter;
4  ARCHITECTURE inv_arch OF inverter IS
5  BEGIN
6    b <= NOT a;
7  END inv_arch;

9  ENTITY nand2 IS
10   PORT(a, b: IN Bit; c: OUT Bit);
11 END nand2;
12 ARCHITECTURE nand2_arch OF nand2 IS
13 BEGIN
14   c <= a NAND b;
15 END nand2_arch;

17 ENTITY mux IS
18   PORT(a, b, c: IN Bit; out1: OUT Bit);
19 END mux;
```

# 结构体—结构描述 ⇔ 逻辑图

```
1  ARCHITECTURE mux_arch1 OF mux IS
2    SIGNAL a_n, b_n, c_not: Bit;
3    COMPONENT inverter
4      PORT(a: IN Bit;
5           b: OUT Bit);
6    END COMPONENT;
7    COMPONENT nand2
8      PORT(a, b: IN Bit;
9           c: OUT Bit);
10   END COMPONENT;
```

```
1  BEGIN
2    U0: inverter PORT MAP(a => c, b => c_not);
3    U1: nand2 PORT MAP(a => a, b => c_not, c => a_n);
4    U2: nand2 PORT MAP(a => b, b => c, c => b_n);
5    U3: nand2 PORT MAP(a => a_n, b => b_n, c => out1);
6  END mux_arch1;
```

```
1  BEGIN
2    U0: inverter PORT MAP(c, c_not);
3    U1: nand2 PORT MAP(a, c_not, a_n);
4    U2: nand2 PORT MAP(b, c, b_n);
5    U3: nand2 PORT MAP(a_n, b_n, out1);
6  END mux_arch1;
```

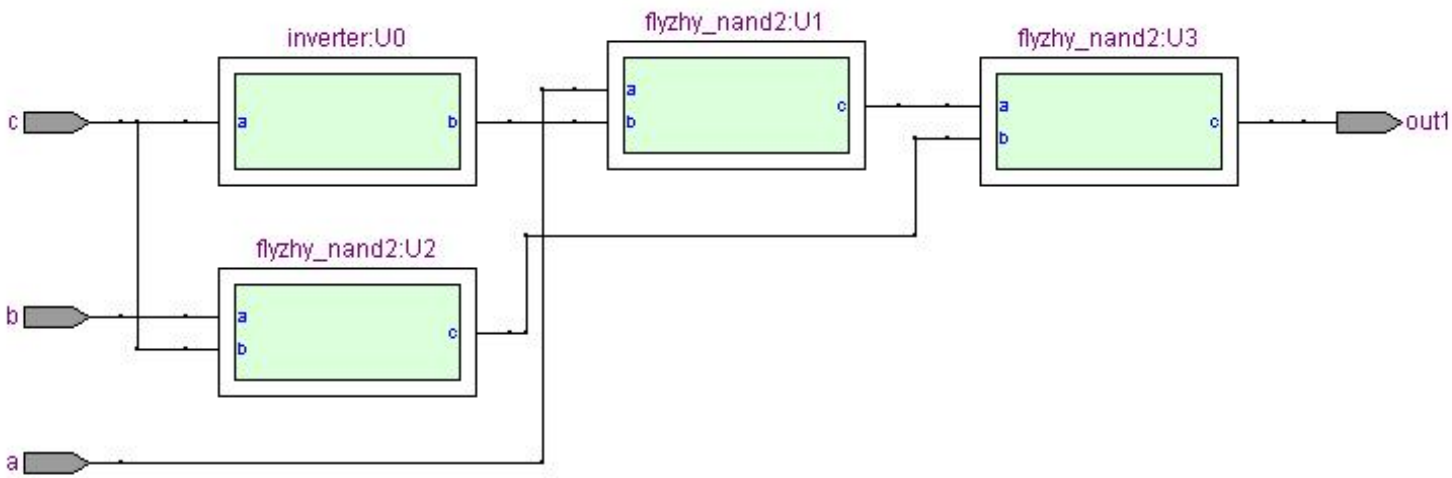
# 结构体—结构描述 ⇔ 逻辑图

```
1  ARCHITECTURE mux_arch1 OF mux IS
2    SIGNAL a_n, b_n, c_not: Bit;
3    COMPONENT inverter
4      PORT(a: IN Bit;
5           b: OUT Bit);
6    END COMPONENT;
7    COMPONENT nand2
8      PORT(a, b: IN Bit;
9           c: OUT Bit);
10   END COMPONENT;
```

```
1  BEGIN
2    U0: inverter PORT MAP(a => c, b => c_not);
3    U1: nand2 PORT MAP(a => a, b => c_not, c => a_n);
4    U2: nand2 PORT MAP(a => b, b => c, c => b_n);
5    U3: nand2 PORT MAP(a => a_n, b => b_n, c => out1);
6  END mux_arch1;
```

```
1  BEGIN
2    U0: inverter PORT MAP(c, c_not);
3    U1: nand2 PORT MAP(a, c_not, a_n);
4    U2: nand2 PORT MAP(b, c, b_n);
5    U3: nand2 PORT MAP(a_n, b_n, out1);
6  END mux_arch1;
```

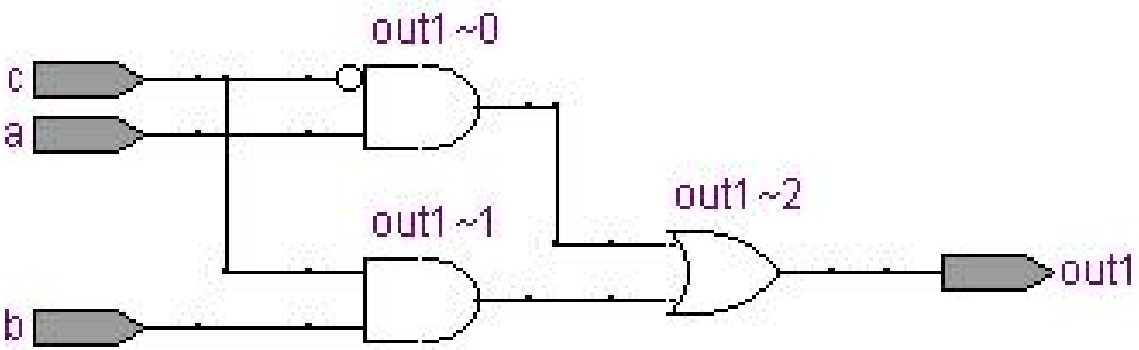
# 2 选 1 多路选择器结构描述



# 结构体—数据流描述 ⇔ 逻辑表达式（布尔方程）

```
1 ARCHITECTURE mux_arch2 OF mux IS
2 BEGIN
3   out1 <= (NOT c AND a) OR (c AND b);
4 END mux_arch2;
```

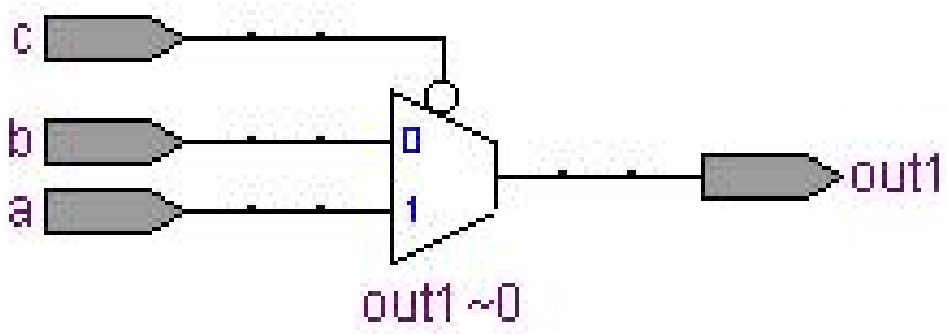
# 2 选 1 多路选择器数据流描述

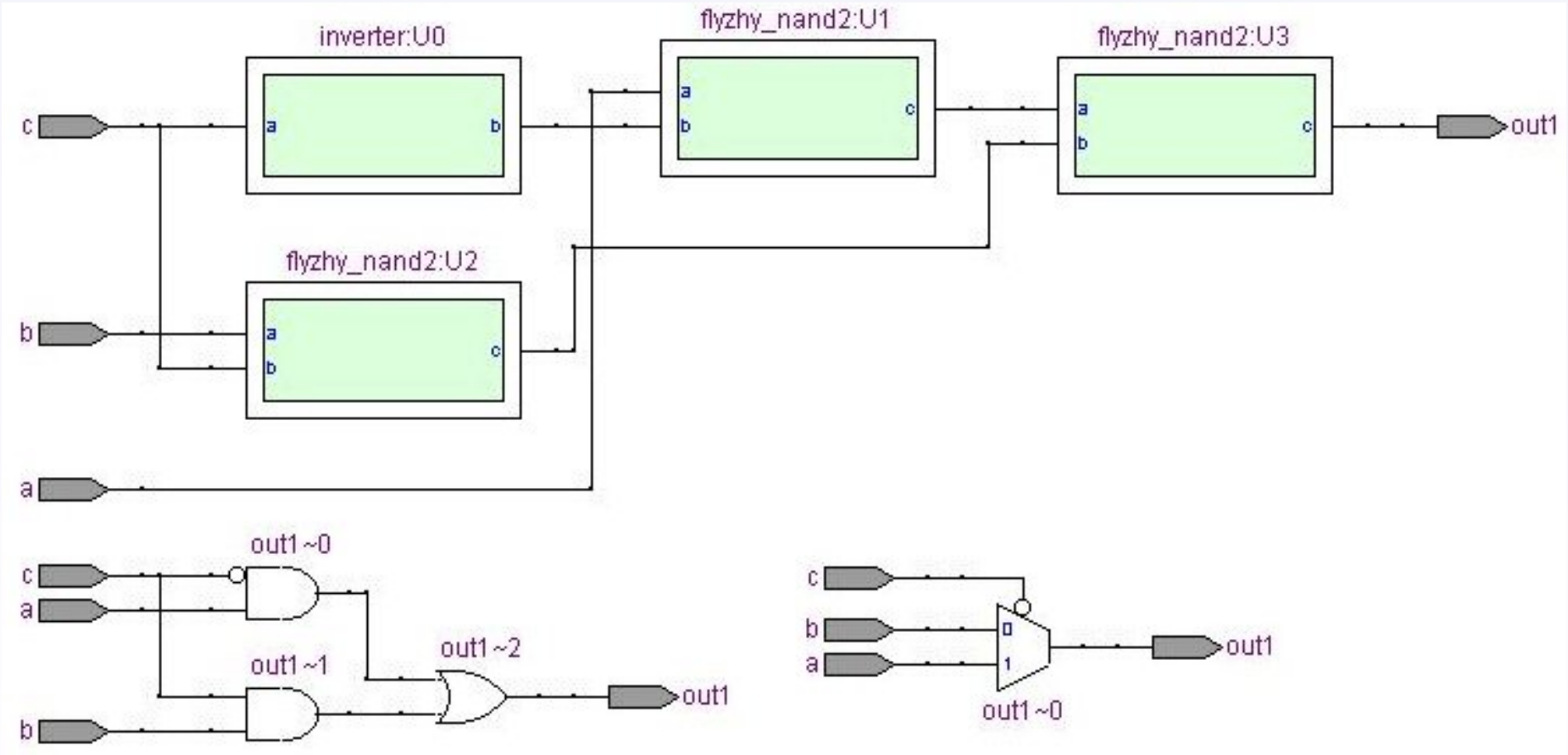


# 结构体—行为描述 ⇔ 真值表（状态图）

```
1  ARCHITECTURE mux_arch3 OF mux IS
2  BEGIN
3    PROCESS(a, b, c)
4    BEGIN
5      IF c='0' THEN
6        out1 <= a;
7      ELSE
8        out1 <= b;
9      END IF;
10   END PROCESS;
11 END mux_arch3;
```

# 2 选 1 多路选择器行为描述




inverter:U0
flyzhy_nand2:U1
flyzhy_nand2:U3
flyzhy_nand2:U2
c
b
a
out1
out1~0
out1~1
out1~2
out1

# 内容提要

# 2-4 解码器

# 2-4 解码器

| enable | a | b | z(0) | z(1) | z(2) | z(3) |
|---|---|---|---|---|---|---|
| 0 | x | x | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| | 0 | 1 | 1 | 0 | 1 | 1 |
| | 1 | 0 | 1 | 1 | 0 | 1 |
| | 1 | 1 | 1 | 1 | 1 | 0 |

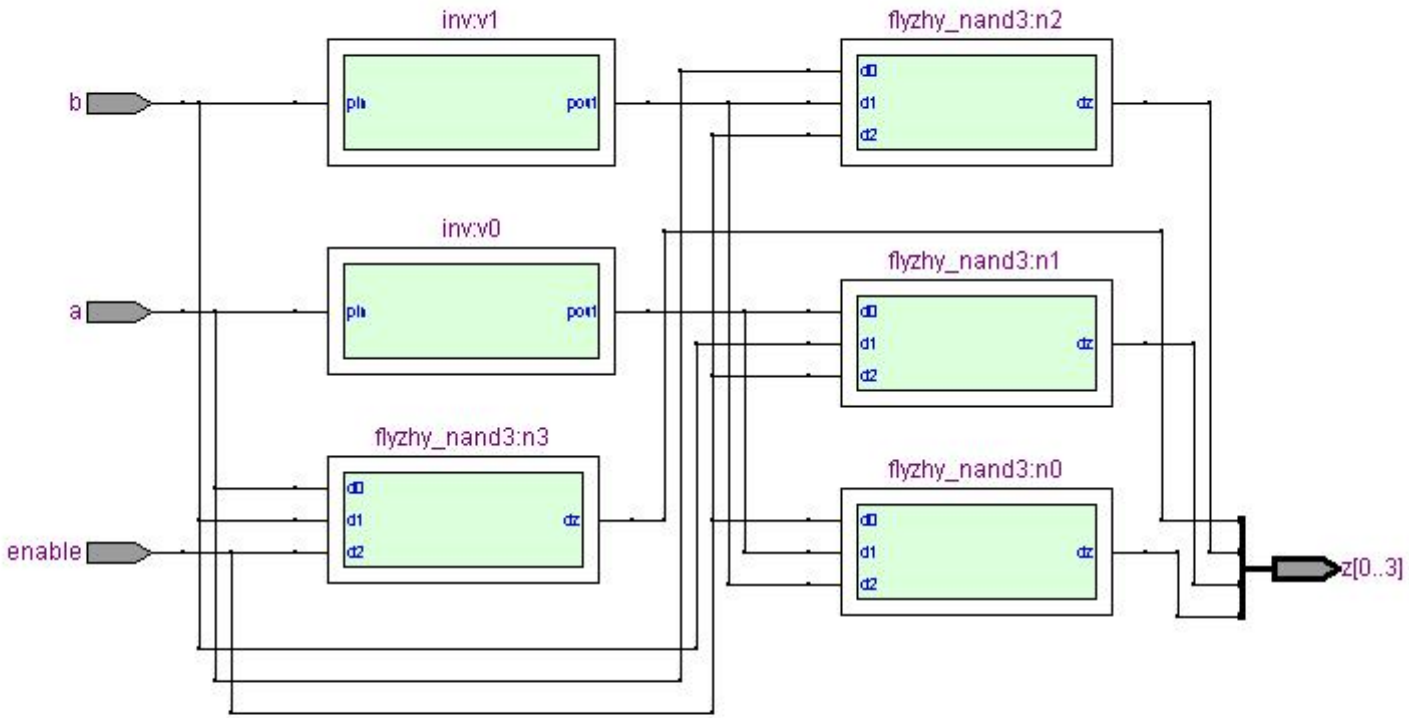
# 实体声明

```
1 ENTITY decoder2x4 IS
2   PORT(a, b, enable: IN Bit;
3        z: OUT Bit_Vector(0 TO 3));
4 END decoder2x4;
```

# 结构体—结构描述 ⇔ 逻辑图

```
1  ARCHITECTURE dec_str OF decoder2x4 IS
2    COMPONENT inv
3      PORT(pin: IN Bit; pout: OUT Bit);
4    END COMPONENT;
5    COMPONENT nand3
6      PORT(d0, d1, d2: IN Bit; dz: OUT Bit);
7    END COMPONENT;
8    SIGNAL abar, bbar: Bit;
9  BEGIN
10   v0: inv PORT MAP(a, abar);
11   v1: inv PORT MAP(b, bbar);
12   n0: nand3 PORT MAP(enable, abar, bbar, z(0));
13   n1: nand3 PORT MAP(abar, b, enable, z(1));
14   n2: nand3 PORT MAP(a, bbar, enable, z(2));
15   n3: nand3 PORT MAP(a, b, enable, z(3));
16 END dec_str;
```

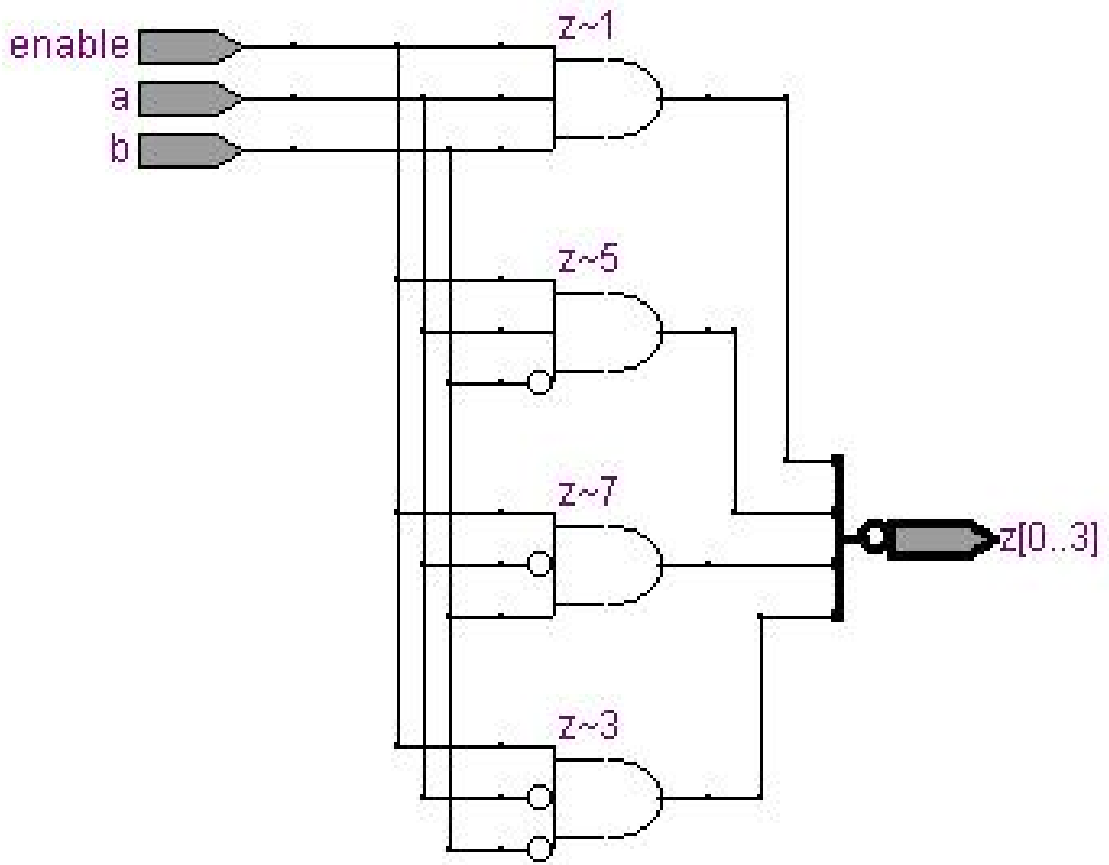
# 2-4 解码器结构描述


inv:v1
flyzhy_nand3:n2
b
inv:v0
flyzhy_nand3:n1
a
flyzhy_nand3:n3
flyzhy_nand3:n0
enable
z[0..3]

# 结构体—数据流描述 ⇔ 逻辑表达式（布尔方程）

```
1  ARCHITECTURE dec_dataflow OF decoder2x4 IS
2    SIGNAL abar, bbar: Bit;
3  BEGIN
4    z(3) <= NOT (a AND b AND enable);
5    z(0) <= NOT (abar AND bbar AND enable);
6    bbar <= NOT b;
7    z(2) <= NOT (a AND bbar AND enable);
8    abar <= NOT a;
9    z(1) <= NOT (abar AND b AND enable);
10 END dec_dataflow;
```

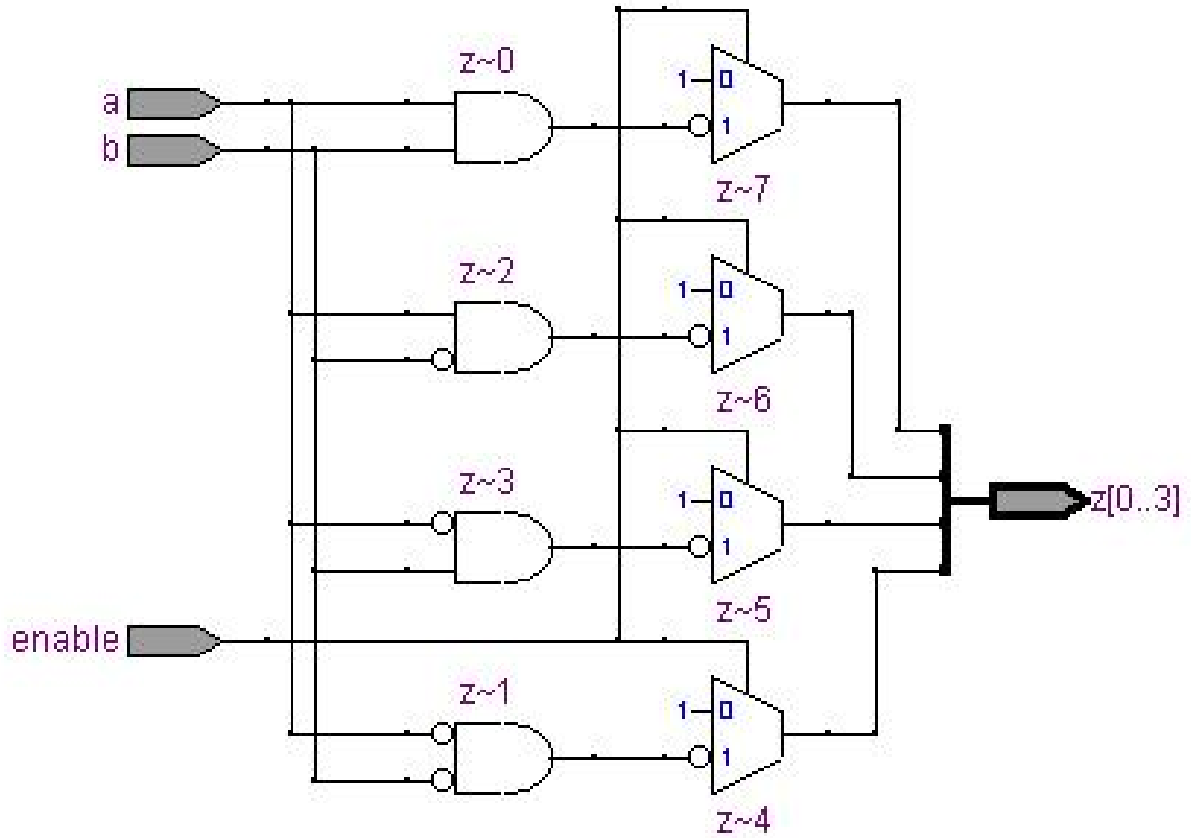
# 2-4 解码器数据流描述



# 结构体—行为描述

```
1  ARCHITECTURE dec_sequential OF decoder2x4 IS
2  BEGIN
3    PROCESS(a, b, enable)
4      VARIABLE abar, bbar: Bit;
5    BEGIN
6      abar := NOT a;
7      bbar := NOT b;
8      IF enable = '1' THEN
9        z(3) <= NOT (a AND b);
10       z(0) <= NOT (abar AND bbar);
11       z(2) <= NOT (a AND bbar);
12       z(1) <= NOT (abar AND b);
13     ELSE
14       z <= "1111";
15     END IF;
16   END PROCESS;
17 END dec_sequential;
```

# 2-4 解码器行为描述



# 结构体—行为描述 ⇔ 真值表（状态图）

```
1  ARCHITECTURE dec_sequential OF decoder2x4 IS
2  BEGIN
3    PROCESS(a, b, enable)
4      VARIABLE temp: Bit_Vector(1 DOWNTO 0);
5    BEGIN
6      temp := a & b;
7      IF enable = '1' THEN
8        CASE temp IS
9          WHEN "00" => z <= "0111";
10         WHEN "01" => z <= "1011";
11         WHEN "10" => z <= "1101";
12         WHEN "11" => z <= "1110";
13         WHEN OTHERS => z <= (OTHERS => '1');
14       END CASE;
15     ELSE
16       z <= (OTHERS => '1');
17     END IF;
18   END PROCESS;
19 END;
```

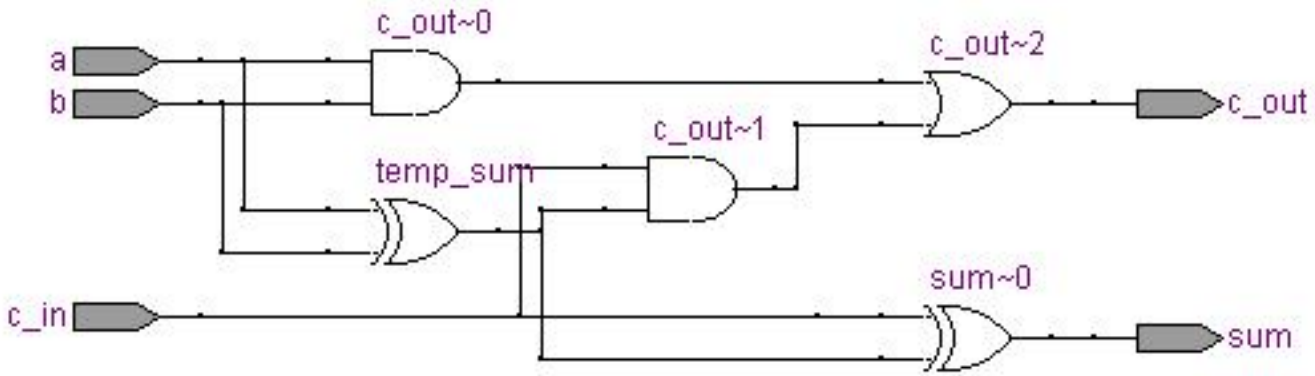
# 内容提要

# 数据流描述 ⇔ 逻辑表达式（布尔方程）

```
1  ENTITY full_adder IS
2    PORT(a, b, c_in: IN Bit;
3         sum, c_out: OUT Bit);
4  END full_adder;

6  ARCHITECTURE rtl OF full_adder IS
7    SIGNAL temp_sum: Bit;
8  BEGIN
9    temp_sum <= a XOR b AFTER 20 ns;
10   sum <= temp_sum XOR c_in AFTER 20 ns;
11   c_out <= (a AND b) OR (temp_sum AND c_in) AFTER 22 ns;
12 END rtl;
```

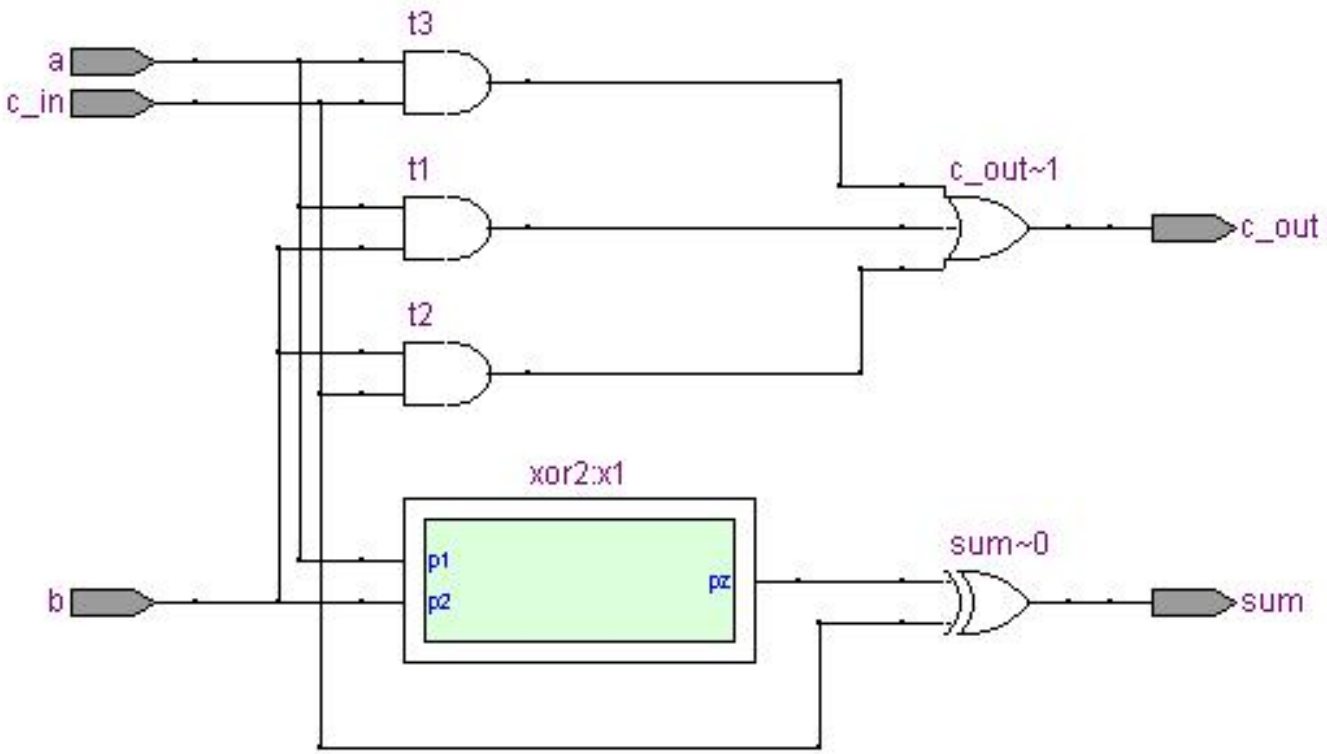
# 1 位全加器数据流描述



# 混合描述

```
1  ENTITY full_addr IS
2    PORT(a, b, c_in: IN Bit; sum, c_out: OUT Bit);
3  END full_addr;
4
5  ARCHITECTURE fa_mixed OF full_addr IS
6    COMPONENT xor2
7      PORT(p1, p2: IN Bit; pz: OUT Bit);
8    END COMPONENT;
9    SIGNAL s1: Bit;
10 BEGIN
11   x1: xor2 PORT MAP(a, b, s1); --Structural description
12   PROCESS(a, b, c_in)          --Behavioral description
13     VARIABLE t1, t2, t3: Bit;
14   BEGIN
15     t1 := a AND b;
16     t2 := b AND c_in;
17     t3 := a AND c_in;
18     c_out <= t1 OR t2 OR t3;
19   END PROCESS;
20   sum <= s1 XOR c_in;          --RTL description
21 END fa_mixed;
```

# 1 位全加器混合描述



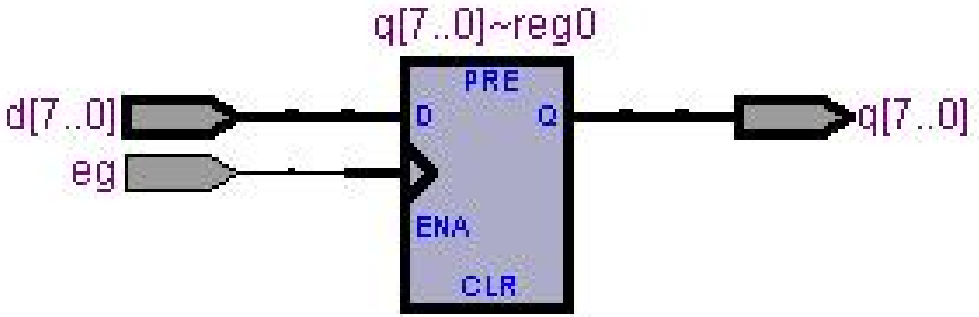
# 内容提要

# 高电平锁存的 n 位锁存器

```
1  ENTITY latch IS
2    GENERIC (n: Positive := 8);
3    PORT (eg: IN Bit;
4          d: IN Bit_Vector(n-1 DOWNTO 0);
5          q: OUT Bit_Vector(n-1 DOWNTO 0));
6  END latch;

8  ARCHITECTURE latch_eg OF latch IS
9  BEGIN
10   PROCESS(eg)
11   BEGIN
12     IF eg='1' THEN
13       q <= d;
14     END IF;
15   END PROCESS;
16 END latch_eg;
```

# 高电平锁存的 n 位锁存器行为描述



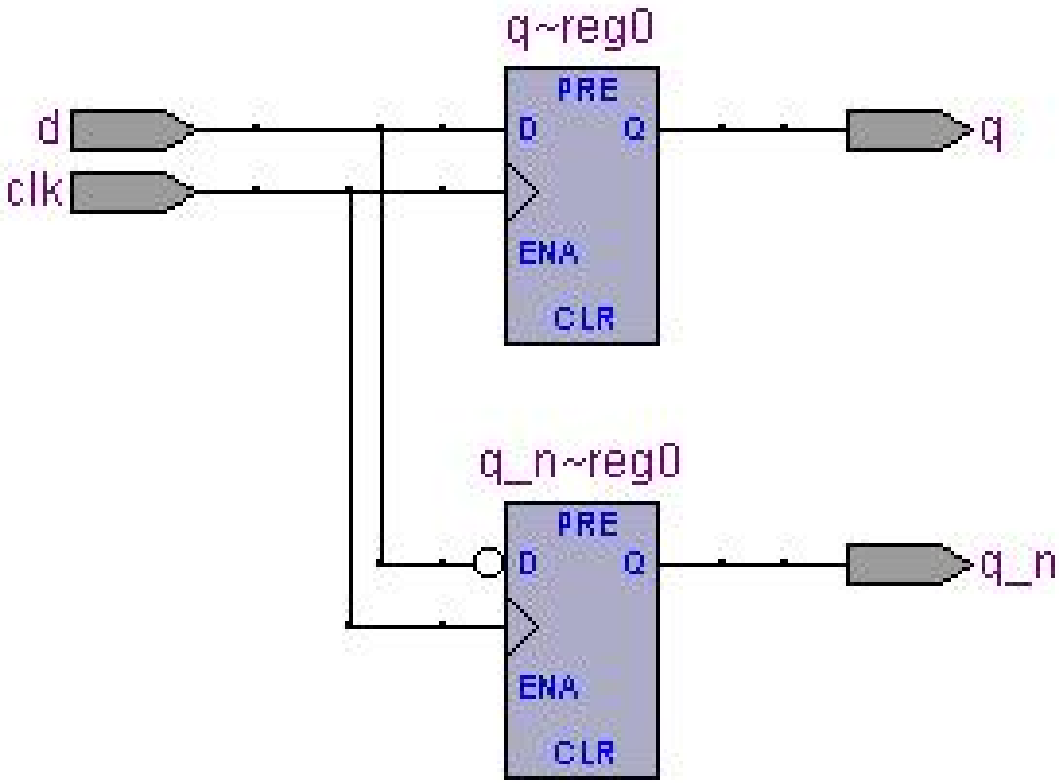
# 内容提要

# 上升沿触发的 D 触发器

```
1  ENTITY dff IS
2    PORT (clk, d: IN Bit;
3          q, q_n: OUT Bit);
4  END dff;

6  ARCHITECTURE dff_bhv OF dff IS
7  BEGIN
8    PROCESS(clk)
9    BEGIN
10     IF clk'Event AND clk='1' THEN
11       q <= d;
12       q_n <= NOT d;
13     END IF;
14   END PROCESS;
15 END dff_bhv;
```

# 上升沿触发的 D 触发器行为描述

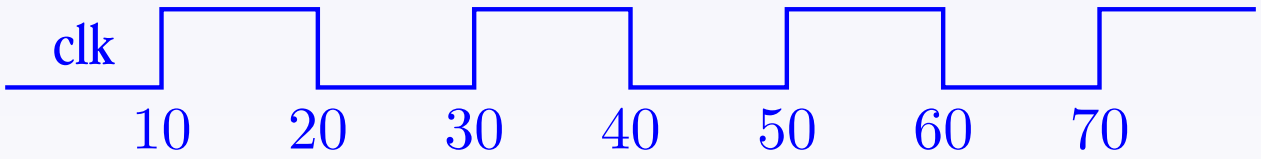


# 内容提要

# 时钟周期信号

## 周期为 20ns 的时钟信号

```
1  clk <= NOT clk AFTER 10 ns;
```



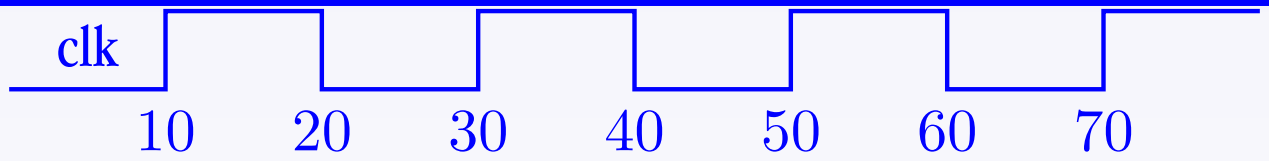

## 不同开关时间的时钟波形

```
1  PROCESS
2  BEGIN
3    clk <= '0';
4    WAIT FOR 20 ns;
5    clk <= '1';
6    WAIT FOR 12 ns;
7  END PROCESS;
```

# 时钟周期信号

## 周期为 20ns 的时钟信号

```
1  clk <= NOT clk AFTER 10 ns;
```




## 不同开关时间的时钟波形

```
1  PROCESS
2  BEGIN
3    clk <= '0';
4    WAIT FOR 20 ns;
5    clk <= '1';
6    WAIT FOR 12 ns;
7  END PROCESS;
```