دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

درس طراحی کامپیوتری سیستمهای دیجیتال

مقدمه ای بر VHDL

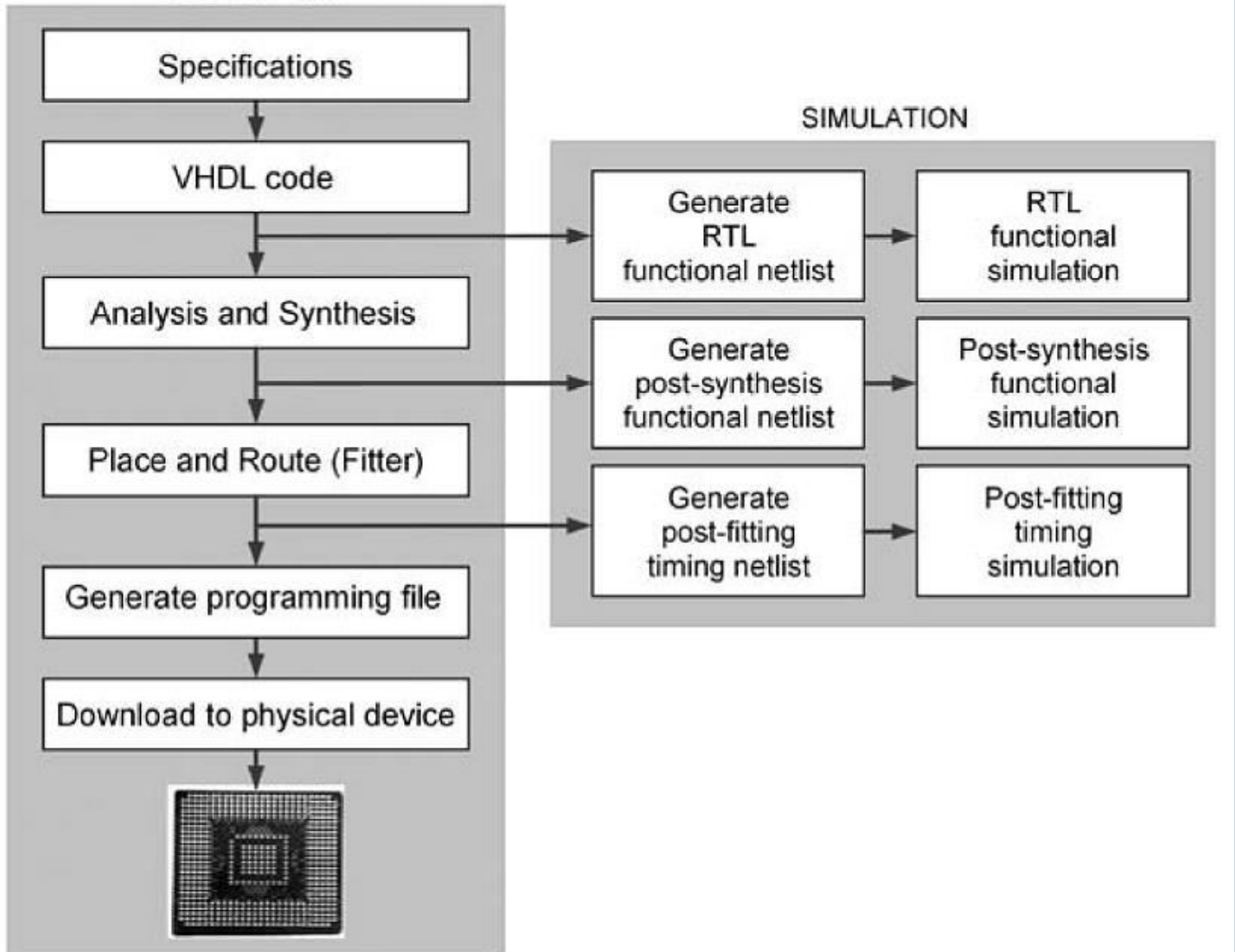
نسخه های مختلف VHDL

- نسخه اول VHDL 87 در سال 1987
- نسخه VHDL 93 در سال 1993
- نسخه VHDL 2002 در سال 2002
- نسخه VHDL 2008 در سال 2008

کاربردهای VHDL

- سنتز(تبدیل سورس برنامه به المانهای سخت افزار)
- شبیه سازی(آیا خروجی سنتز همان عملکرد مورد انتظار ما را انجام می دهد؟)

مراحل طراحی با VHDL



ابزار های EDA
(Electronic Design Automation)

- From Altera: Quartus II(for synthesis and graphical simulation)
- From Xilinx: ISE(XST for synthesis, ISE Simulator for simulation)
- From Mentor Graphics: Precision RTL and Leonardo Spectrum(synthesis), ModelSim(simulation)
- From Synopsys/Symplicity: Design Compiler Ultra and Synplify Pro/Premier(synthesis), VCS(simulation)
- From Cadence: NC-Sim(simulation)
- From Aldec: Active-HDL(simulation).

مثال یک برنامه vhdl

- 3 sections to a piece of VHDL code
- File extension for a VHDL file is .vhd
- Name of the file should be the same as the entity name (nand_gate.vhd) [OpenCores Coding Guidelines]

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
     PORT(
          a    : IN STD_LOGIC;
          b    : IN STD_LOGIC;
          z    : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
     z <= a NAND b;
END model;
```

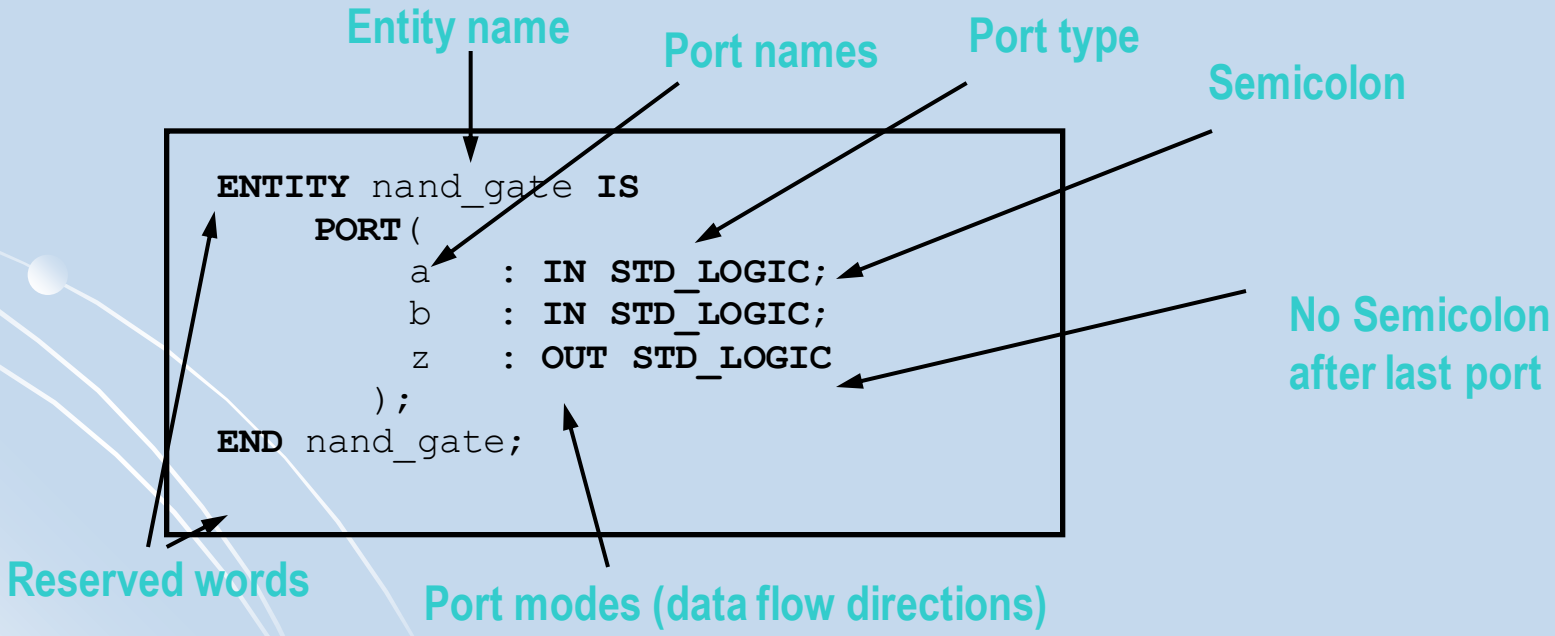
LIBRARY DECLARATION

ENTITY DECLARATION

ARCHITECTURE BODY

Entity Declaration

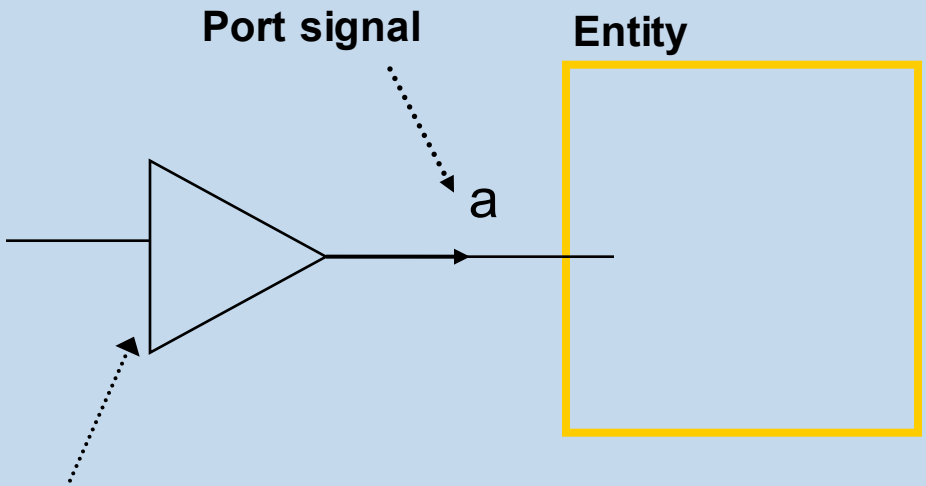
- Entity Declaration describes the interface of the component, i.e. input and output ports.



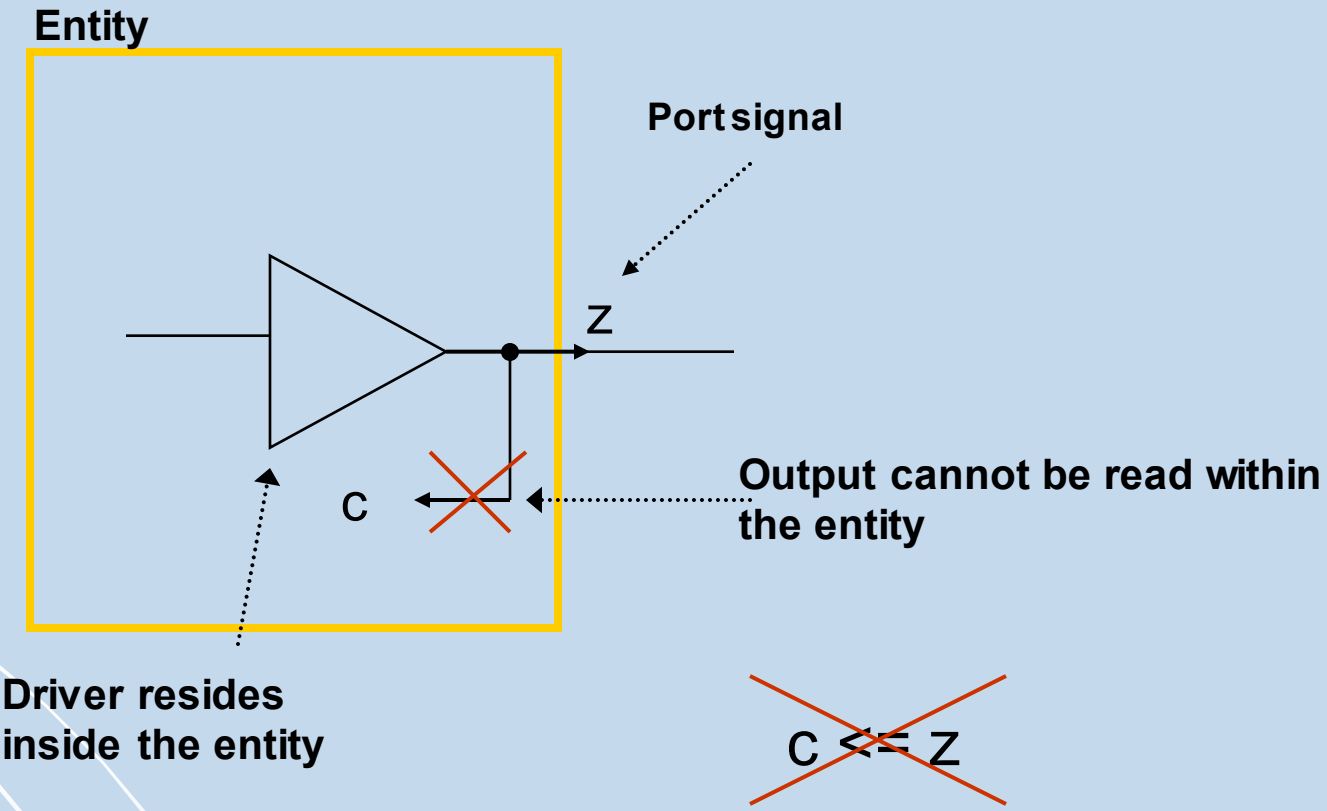
Entity declaration – simplified syntax

```
ENTITY entity_name IS
   PORT (
      port_name :  port_mode  signal_type;
      port_name :  port_mode  signal_type;
      ……..
      port_name :  port_mode  signal_type);
END entity_name;
```

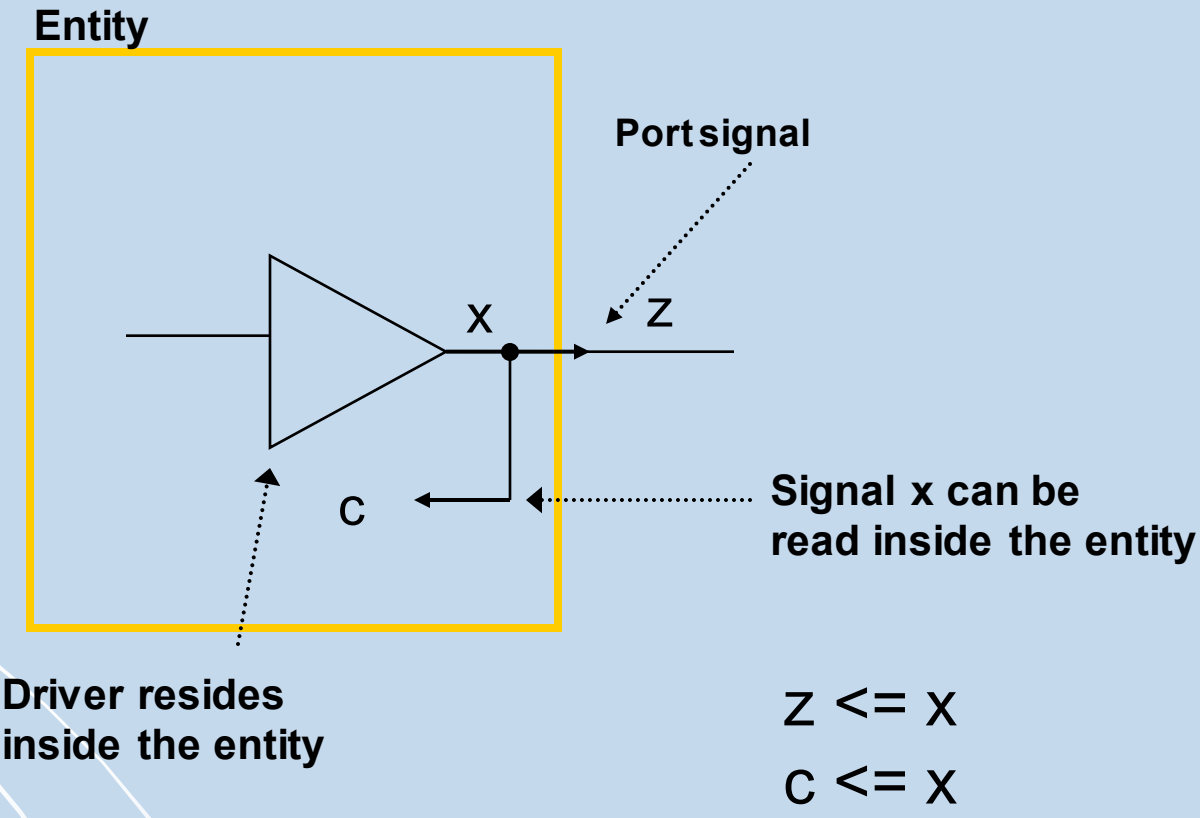
Port Mode IN



Port Mode OUT



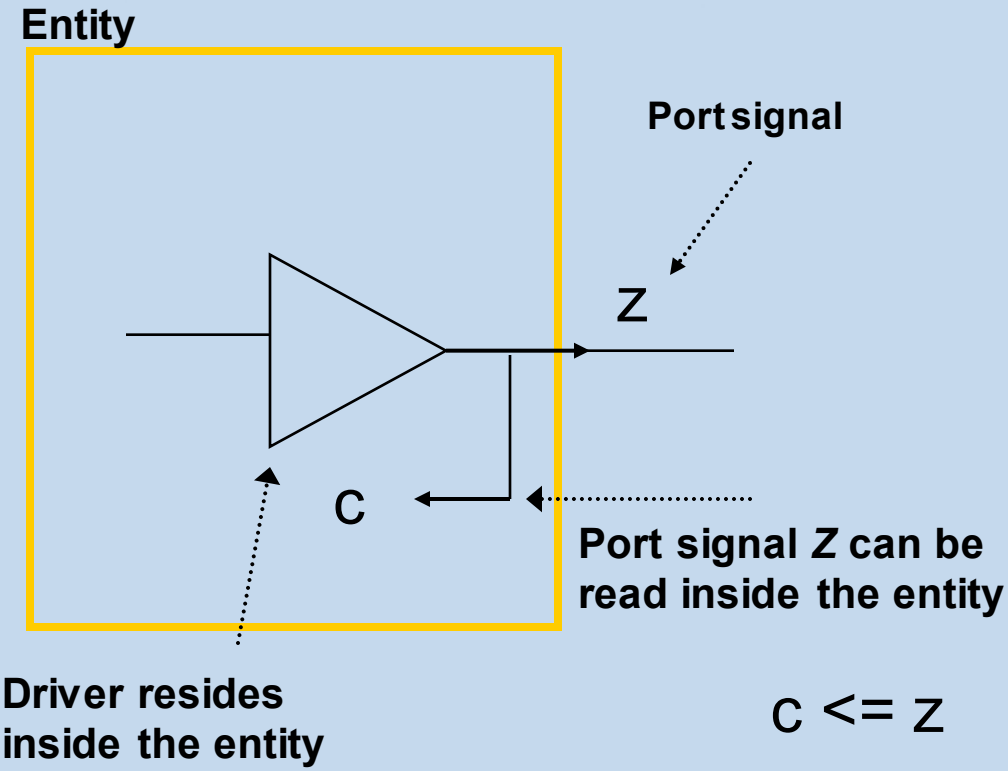
Port Mode OUT (with extra signal)



z <= x

c <= x

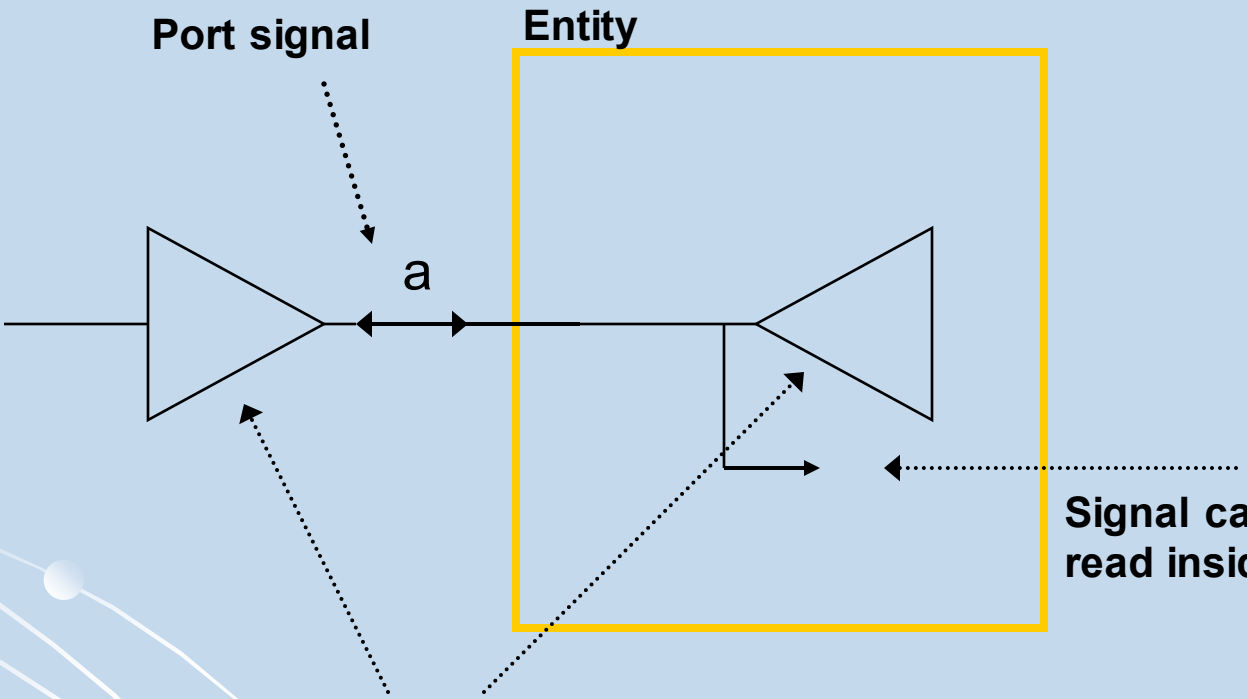
Port Mode BUFFER



Not recommended by OpenCores Coding Guidelines.

Port of mode buffer can not be connected to other types of ports so buffer mode will propagate throughout the entire hierarchical design. Problems reported with synthesis of designs using these ports.

Port Mode INOUT



Port Modes - Summary

The *Port Mode* of the interface describes the direction in which data travels with respect to the c*omponent*

- **In**: Data comes into this port and can only be read within the entity. It can appear **only on the right side** of a signal or variable assignment.
- **Out**: The value of an output port can only be updated within the entity. **It cannot be read**. It can only appear **on the left side** of a signal assignment.
- **Inout**: The value of a bi-directional port can be read and updated within the entity model. It can appear on **both sides** of a signal assignment.
- **Buffer:** Used for a signal that is an output from an entity. The value of the signal can be used inside the entity, which means that in an assignment statement the signal can appear on the left and right sides of the <= operator. Not recommended to be used in the synthesizable code.

Architecture (Architecture body)

- Describes an implementation of a design entity
- Architecture example:

```
ARCHITECTURE model OF nand_gate IS
BEGIN
     z <= a NAND b;
END model;
```

Architecture – simplified syntax

```
ARCHITECTURE architecture_name OF entity_name IS
   [ declarations ]
BEGIN
    code
END architecture_name;
```

Library Declarations

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_gate IS
     PORT(
          a    : IN STD_LOGIC;
          b    : IN STD_LOGIC;
          z    : OUT STD_LOGIC);
END nand_gate;

ARCHITECTURE model OF nand_gate IS
BEGIN
     z <= a NAND b;
END model;
```

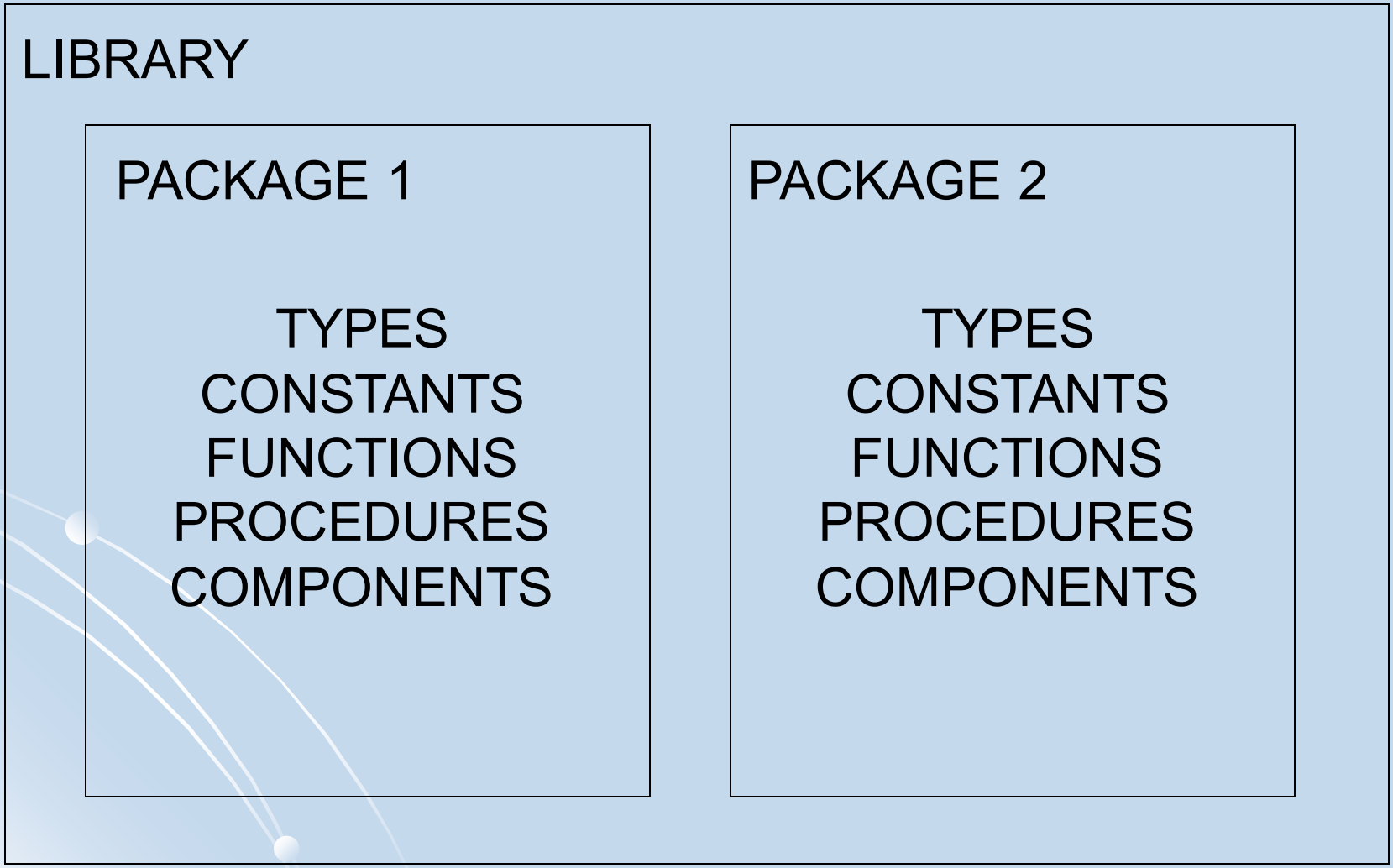
Library declaration

Use all definitions from the package std_logic_1164

Library declarations - syntax

```
LIBRARY  library_name;
USE library_name.package_name.package_parts;
```

Fundamental parts of a library



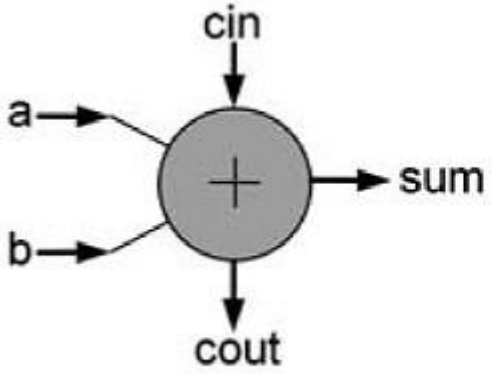
GENERIC

- نوشتن کد به صورت پارامتری
- در بخش Entity و قبل از تعریف پورتها، معرفی میشود

```
-----------------------------------------
ENTITY my_entity IS
GENERIC (m: INTEGER := 8;
n: BIT_VECTOR(3 DOWNTO 0) := "0101");

PORT (...);
END my_entity;
-----------------------------------------
```

مثال تمام جمع کننده



a	b	cin	sum	cout
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

```
ENTITY full_adder IS
  PORT (a, b, cin: IN BIT;
        sum, cout: OUT BIT);
END full_adder;
-------------------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
  sum <= a XOR b XOR cin;
  cout <= (a AND b) OR (a AND cin) OR (b AND cin);
END dataflow;
```

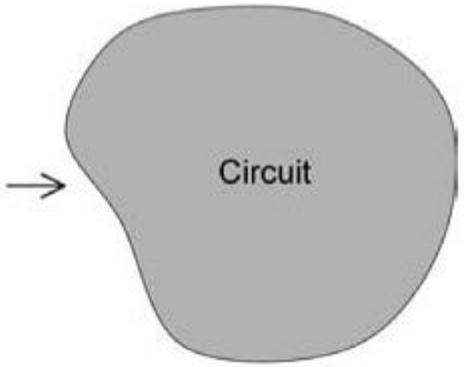


Figure 1.3
VHDL code for the full-adder unit of figure 1.2.

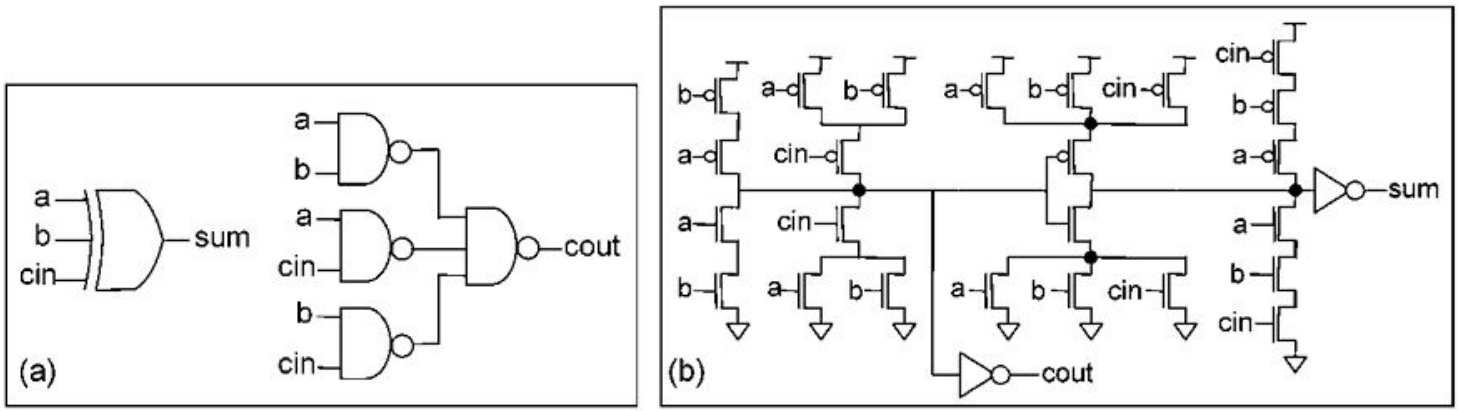


Figure 1.4
Implementation examples for the full-adder circuit of figure 1.2: (a) With conventional gates; (b) At transistor level (with CMOS logic).

نتیجه شبیه سازی تمام جمع کننده

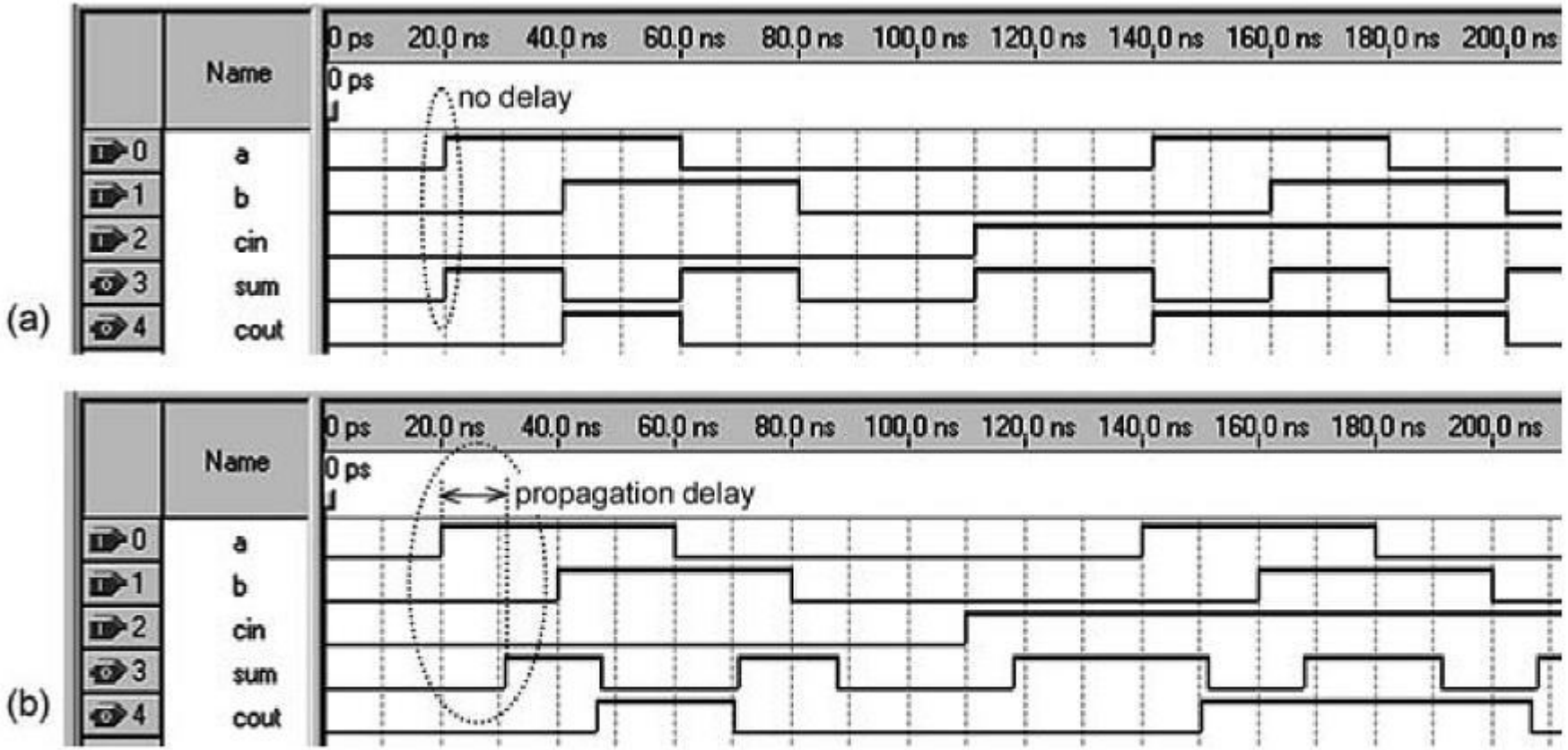
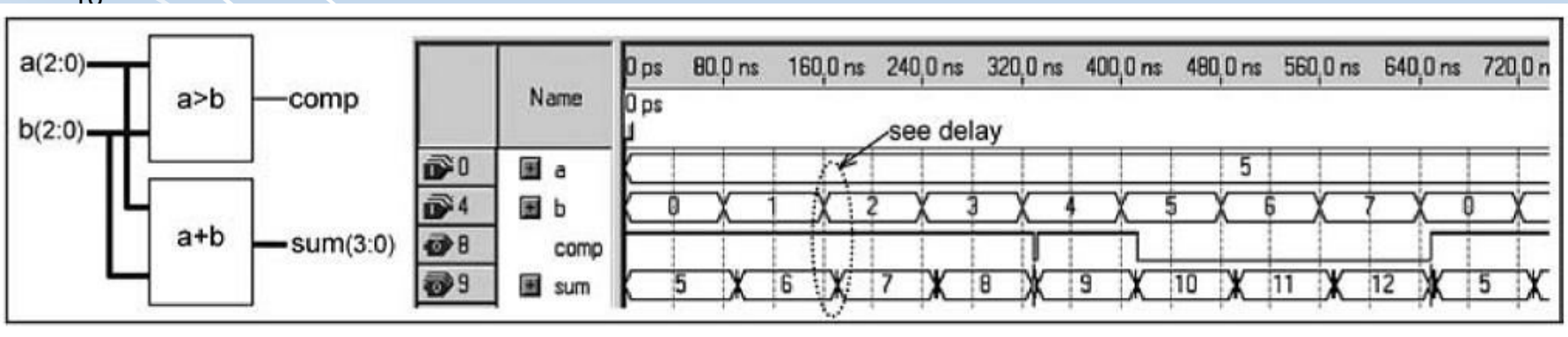


Figure 1.5
(a) Functional versus (b) timing simulation results obtained from the VHDL code of figure 1.3.

نمونه برنامه ها

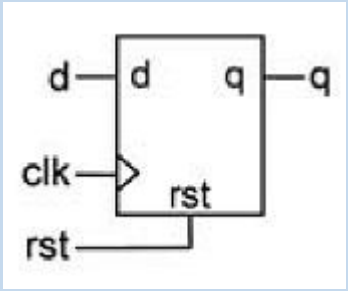
- مقایسه و جمع

```
1 ----------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ----------------------------------------5
ENTITY comp_add IS
6 PORT (a, b: IN INTEGER RANGE 0 TO 7;
7           comp: OUT STD_LOGIC;
8           sum: OUT INTEGER RANGE 0 TO 15);
9 END ENTITY;
10 ----------------------------------------
11 ARCHITECTURE circuit OF comp_add IS
12 BEGIN
13    comp <= '1' WHEN a>b ELSE '0';
14    sum <= a + b;
15 END ARCHITECTURE;
16 ----------------------------------------
```

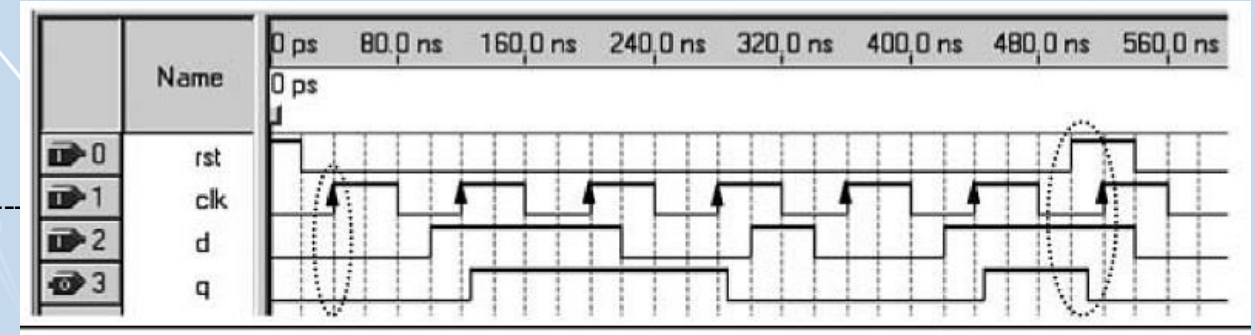


نمونه برنامه ها

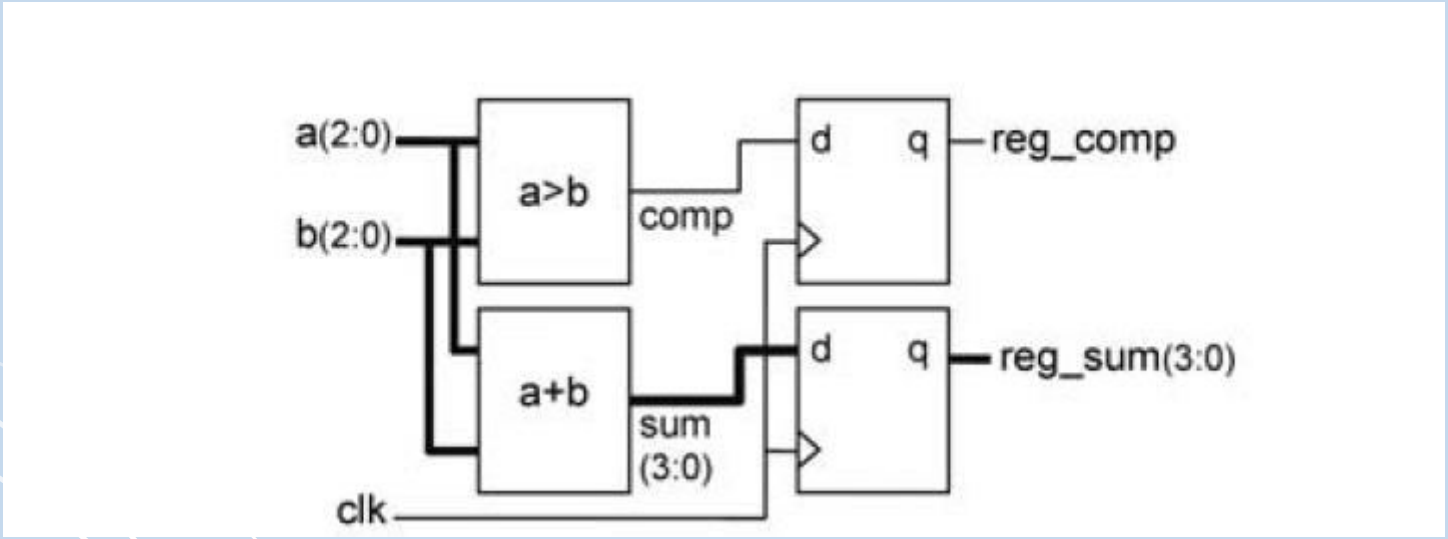
- فلیپ فلاپ نوع D




```
1 ------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------
5 ENTITY  flip_flop IS
6 PORT (d, clk, rst: IN STD_LOGIC;
7           q: OUT STD_LOGIC);
8 END ENTITY;
9 ------------------------------------
10 ARCHITECTURE  flip_flop OF flip_flop IS
11 BEGIN
12  PROCESS (clk, rst)
13  BEGIN
14      IF (rst='1') THEN
15          q <= '0';
16      ELSIF (clk'EVENT AND clk='1') THEN
17          q <= d;
18      END IF;
19  END PROCESS;
20 END ARCHITECTURE;
21 ------------------------------------
```

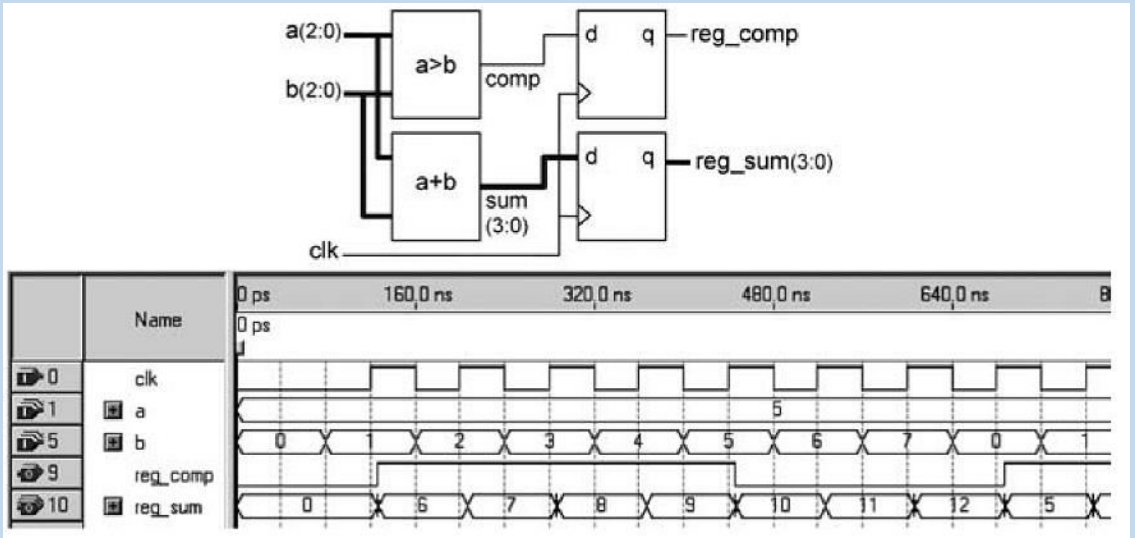


Registered Comp-Add Circuit

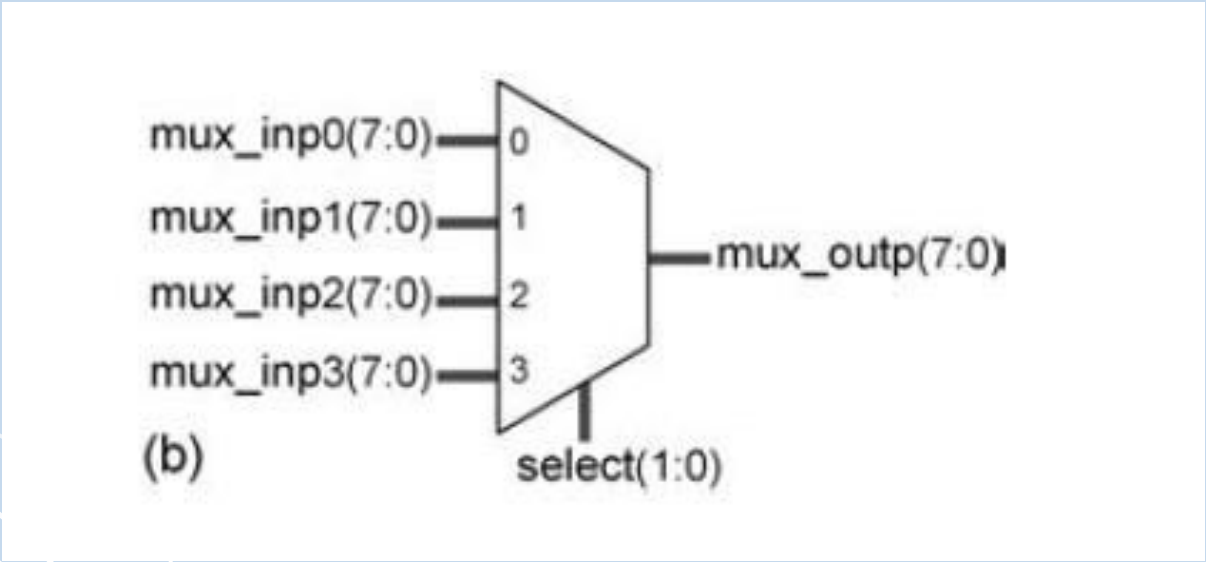


نمونه برنامه ها

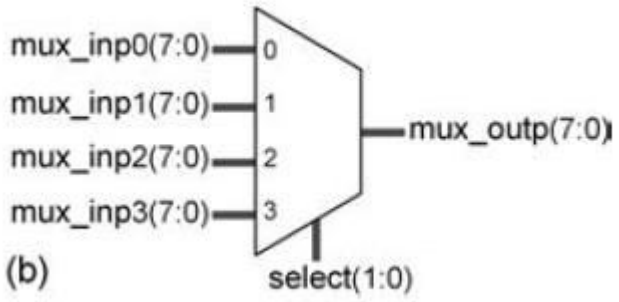
```
1 ----------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ----------------------------------------
5 ENTITY  registered_comp_add IS
6 PORT (clk: IN STD_LOGIC;
7           a, b: IN INTEGER RANGE 0 TO 7;
8           reg_comp: OUT STD_LOGIC;
9           reg_sum: OUT INTEGER RANGE 0 TO 15);
10 END ENTITY;
11 ----------------------------------------
12 ARCHITECTURE  circuit OF registered_comp_add IS
13 SIGNAL comp: STD_LOGIC;
14 SIGNAL sum: INTEGER RANGE 0 TO 15;
15 BEGIN
16    comp <= '1' WHEN a>b ELSE '0';
17    sum <= a + b;
18  PROCESS (clk)
19   BEGIN
20     IF (clk'EVENT AND clk='1') THEN
21        reg_comp <= comp;
22        reg_sum <= sum;
23      END IF;
24  END PROCESS;
25 END ARCHITECTURE;
26 ----------------------------------------
```



Generic Mux


mux_inp0(7:0)
mux_inp1(7:0)
mux_inp2(7:0)
mux_inp3(7:0)
0
1
2
3
mux_outp(7:0)
select(1:0)
(b)

مالتی پلکسر پارامتری



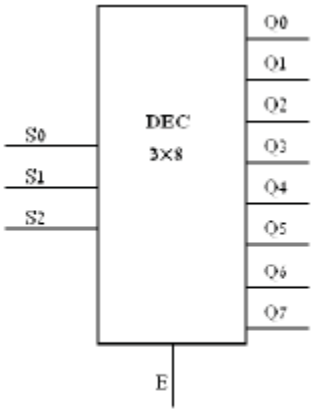

```
1  ----------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------------------
5  ENTITY multiplexer_4x8 IS
6     GENERIC (
7        N: NATURAL := 8;      --bits in in/out signals
8        M: NATURAL := 2);     --bits in select
9     PORT (
10       mux_inp0:    IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
11       mux_inp1:    IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
12       mux_inp2:    IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
13       mux_inp3:    IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
14       select:      IN STD_LOGIC_VECTOR(M-1 DOWNTO 0);
15       mux_outp:    OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
16 END ENTITY;
17 ----------------------------------------------------
19 ARCHITECTURE multiplexer_4x8 OF multiplexer_4x8 IS
20 BEGIN
21    mux_outp <= mux_inp0 WHEN select="00" ELSE
22                mux_inp1 WHEN select="01" ELSE
23                mux_inp2 WHEN select="10" ELSE
24                mux_inp3;
```

```
25 END ARCHITECTURE;
26 ----------------------------------------------------
```

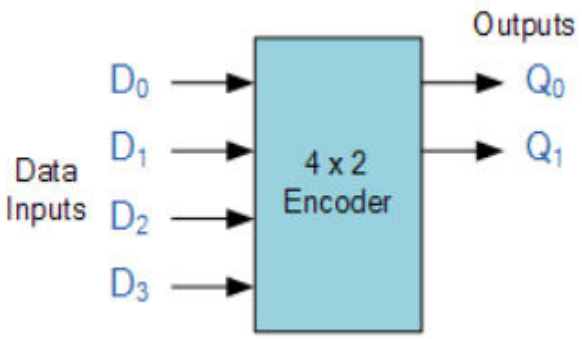
تمرین

نمایش عملکرد دیکودر 3×8

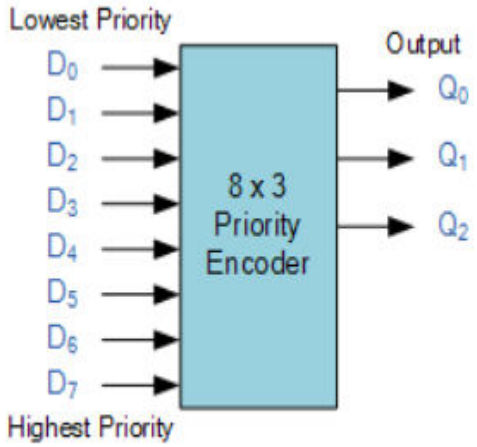
E	$S_2\ S_1\ S_0$	$Q_7Q_6Q_5Q_4Q_3Q_2Q_1Q_0$
1	000	00000001
1	001	00000010
1	010	00000100
1	011	00001000
1	100	00010000
1	101	00100000
1	110	01000000
1	111	10000000
0	XXX	00000000



نماد گرافیکی دیکودر 3×8



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x



Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

X = dont care

تکالیف: تاریخ تحویل یکشنبه 28 مهر

- تمرین های 1، 2 و 3 فصل دوم کتاب Pedroni
- شبیه سازی در نرم افزار ISE
- کد های شبیه سازی و توضیحات ارسال به ایمیل