درس: آزمایشگاه معماری کامپیوتر(جلسه سوم)

- مروری بر VHDL(ادامه)
- آزمایش دوم

نیمسال دوم ۱۴۰۰

مقداردهی به Signal در داخل برنامه

```
begin
   --one_bit <= 'Z';     --wrong, just 0 and 1

   one_stdlogic <= '0';

   --vector_of_bits_descending <= (others => 'X'); --wrong

   vector_of_bits_ascending  <= ('1','0','1', others => '0');      --("1011", others => '0') is wrong

   vector_of_stdlogic_descending <= ('Z','U','0',OTHERS => '1');

   vector_of_stdlogic_ascending <= "Z01U";

   --two simultanious assignment is not valid

   int_signal_no_lenght <= 127;

   int_signal_pos <= 54;
```

مقداردهی به Signal درون برنامه

```
--two simultanious assignment is not valid

int_signal_no_lenght <= 127;

int_signal_pos <= 54;

int_signal_neg <= -16;

--boolean_signal <= '1';    --wrong

time_signal <= 5 hr;

string_signal <= "HelloWorld";

char_signal <= 'H';
```

نکات مقداردهی به Signal درون برنامه

- تغییر مقدار Signal یا مقداردهی به یک Signal در بدنه اصلی Code درون Architecture و بعد از begin انجام می‌شود. از علامت <= استفاده می‌شود.
- چند سیگنال یا Constant از یک نوع را می‌توان به صورت یکجا تعریف کرد
- امکان تغییر type یا تعداد بیت Signal در بدنه اصلی کد وجود ندارد و همواره ثابت می‌ماند.

STD Resolution for multiple Drive Signals

Resolution table for std_logic

resolved function

		U	X	0	1	Z	W	L	H	-
uninitialized	U	U	U	U	U	U	U	U	U	U
unknown	X	U	X	X	X	X	X	X	X	X
forcing low	0	U	X	0	X	0	0	0	0	X
forcing high	1	U	X	X	1	1	1	1	1	X
high impedance	Z	U	X	0	1	Z	W	L	H	X
weak unknown	W	U	X	0	1	W	W	W	W	X
weak low	L	U	X	0	1	L	W	L	W	X
weak high	H	U	X	0	1	H	W	W	H	X
Don't Care	-	U	X	X	X	X	X	X	X	X

```
Signal A, B, C : std_logic_vector(2 downto 0);
...
A <= B after 2 ns;
A <= C after 1 ns;
A <= B or C;
A <= ...
...
```

Slicing

- با استفاده از پرانتز به شکل زیر، می‌توان تعداد بیت مشخص یا بخشی از یک سیگنال (یا هر Variable) را انتخاب نمود.

```
Signal a : std_logic_vector(7 downto 0) := "10101100";

Signal b : std_logic_vector(3 downto 0);

...

b <= a(7 downto 4); → b="1010"

B <= a(5 downto 2); → b="1011"

...

B <= a(2 to 7); → ILLEGAL
```

Concatenation

- به معنی چسباندن دو متغیر در کنار یکدیگر است.
- این مقدار جدید از سمت راست به متغیر سیگنال (یا Variable) assign می‌شود.
- از علامت & و (... , ,) بدین منظور استفاده می‌شود.
- Type و تعداد بیت دو طرف => باید برابر باشد.

(ادامه) Concatenation

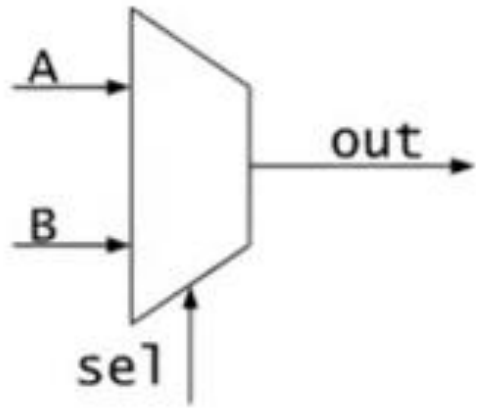
```
--signal declarations for concatenation
signal a,b,c : std_logic := '0';
signal aa : std_logic_vector(7 downto 0) := X"FF";
signal bb,d : std_logic_vector(3 downto 0) := "0001";
signal cc : std_logic_vector(15 downto 0);
```

bb <= a & b & '1' & c ; → b="0010"

d <= aa(6 downto 4) & a ; → d="1110"

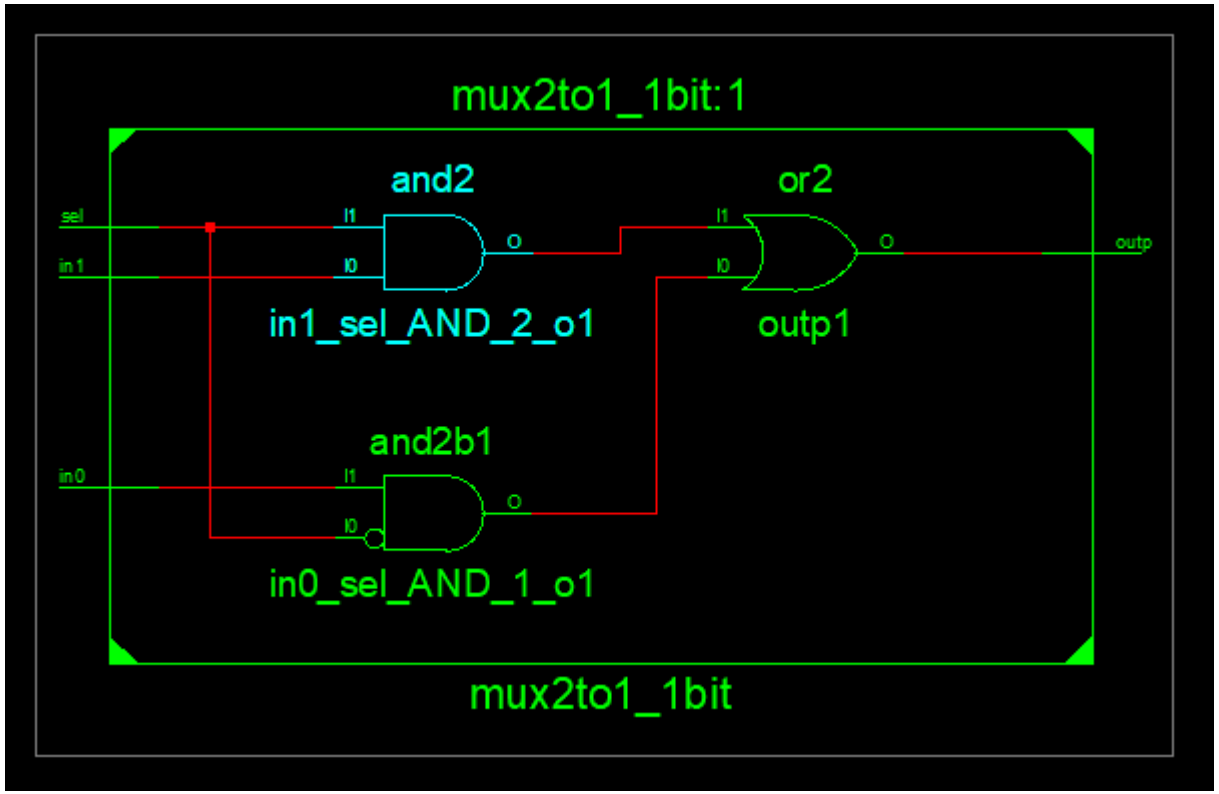
cc <= bb & bb & d & d ; → cc="0010001000010001

Multiplexer

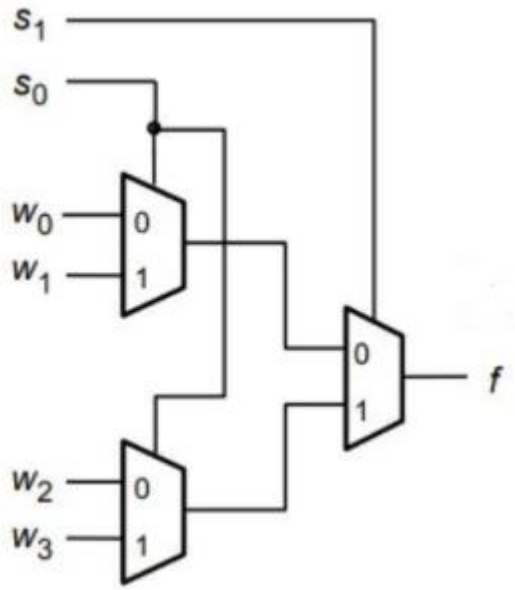


Scl	In0	In1	outp
0	0	-	0
0	1	-	1
1	-	0	0
1	-	1	1

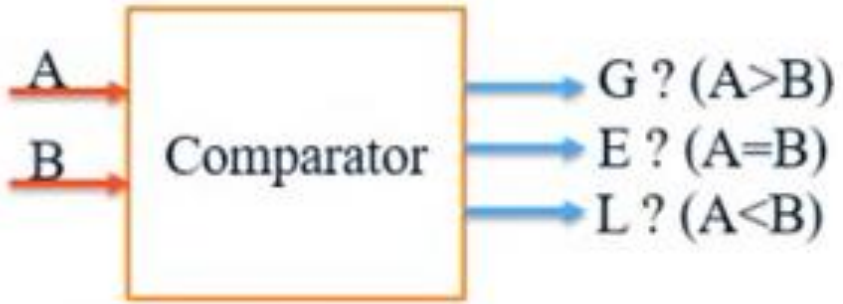
```
--------    Library and Packages    ----------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--------    Interface or Entity    ----------
entity mux2to1_1bit is
   --port (
   -- in0,in1,sel : in std_logic;
   -- outp : out std_logic
   --    );
end mux2to1_1bit;
--------    Architecture or Body    ----------
architecture Behavioral of mux2to1_1bit is
   signal in0, in1, sel, outp : std_logic;
begin
   outp <= (in0 AND NOT sel) OR (in1 AND sel);
   --outp <= (in0 AND NOT sel) OR (in1 AND sel) after 10 ns;
end Behavioral;
```



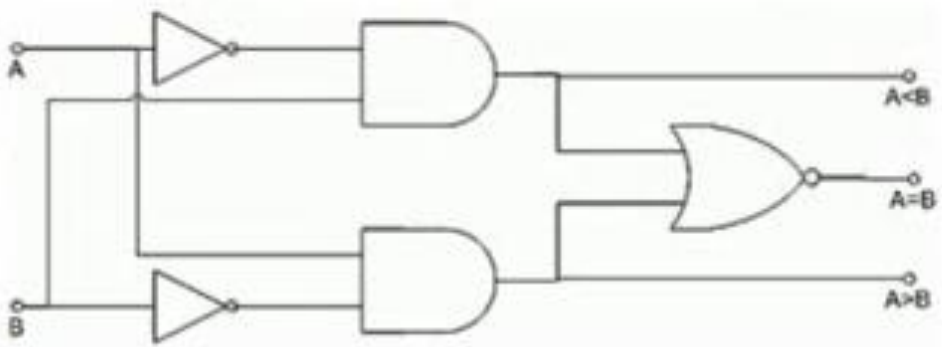
- نوشتن ساختاری مالتی پلکسر ۲ به ۱ با نمونه سازی المان ها
- نوشتن ساختازی mux4-1 با استفاده از mux2-1



Comparator



A	B	G	L	E
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

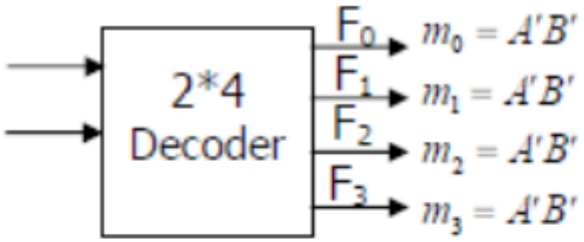



```
--------    Library and Packages    ------------
library IEEE;
--use IEEE.STD_LOGIC_1164.ALL;
--------    Interface or Entity    ------------
entity comparator is
    port(
        A, B : in bit;
        L, G, E : out bit
        );
end comparator;
--------    Architecture or Body    ------------
architecture Behavioral of comparator is
    signal A_not, B_not : bit;
begin
    G <= A and (not B);
    L <= (not A) and B;
    E <= A xnor B;

-- A_not <= NOT A;
-- B_not <= NOT B;
-- G <= A AND B_not;
-- L <= A_not AND B;
-- E <= A xnor B;

end Behavioral;
```

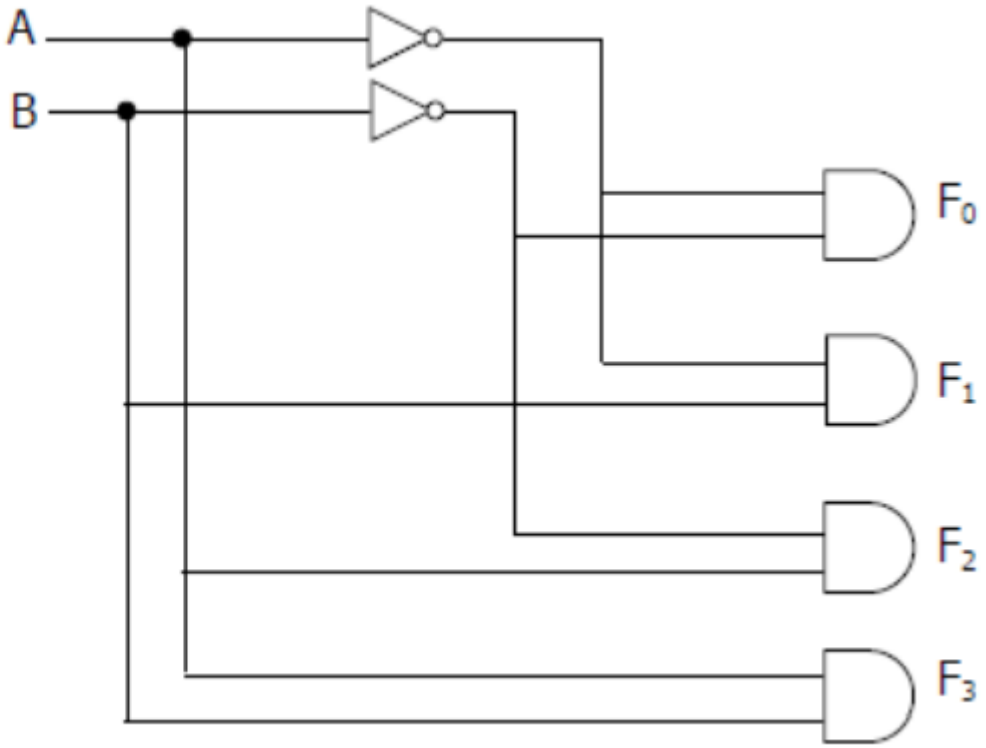
استفاده از when-else و with-select

```
Signal q: std_logic_vector(3 downto 0);
Signal t: std_logic_vector(3 downto 0);

Q<= “0000” when t=”1010” else
     “1011” when t=”0010” else
     “1111”;

with t select
q<= “0000” when “1010”,
    “1011” when “0010”,
    “1111” when others;
```

مدار دیکدر



A	B	F_0	F_1	F_2	F_3
۰	۰	۱	۰	۰	۰
۰	۱	۰	۱	۰	۰
۱	۰	۰	۰	۱	۰
۱	۱	۰	۰	۰	۱



مدار دیکدر کد (vhdl)

انکدر درست برعکس این دیکدر

```
library ieee;
use ieee.std_logic_1164.all;
entity Decoder_vec is
 port(
input: in std_logic_vector(2 downto 0);
 output: out std_logic_vector(7 downto 0)
 );
end entity Decoder_vec;
architecture behav of Decoder_vec is
 begin
output(0) <= '1' when input="000" else '0';
output(1) <= '1' when input="001" else '0';
output(2) <= '1' when input="010" else '0';
output(3) <= '1' when input="011" else '0';
output(4) <= '1' when input="100" else '0';
output(5) <= '1' when input="101" else '0';
output(6) <= '1' when input="110" else '0';
output(7) <= '1' when input="111" else '0';
--output<="00000001" when input="000" else
--"00000010" when input="001" else
--"00000100" when input="010" else
--"00001000" when input="011" else
--"00010000" when input="100" else
--"00100000" when input="101" else
--"01000000" when input="110" else
--"10000000" when input="111";
end behav;
```

مبدل کد BCD به 7-SEGMENT

A	B	C	D		a	b	c	d	e	f	g
•	•	•	•	0	\	\	\	\	\	\	•
•	•	•	\	1	•	\	\	•	•	•	•
•	•	\	•	2	\	\	•	\	\	•	\
•	•	\	\	3	\	\	\	\	•	•	\
•	\	•	•	4	•	\	\	•	•	\	\
•	\	•	\	5	\	•	\	\	•	\	\
•	\	\	•	6	•	•	\	\	\	\	\
•	\	\	\	7	\	\	\	•	•	•	•
\	•	•	•	8	\	\	\	\	\	\	\
\	•	•	\	9	\	\	\	\	•	\	\