Sequential assignment Statements:

07.04.2011 ©

1- IF_THEN-ELSE statement
2- Case Statement

} in VHDL these are used within a PROCESS statement

```
PROCESS( W0, W1, S)
BEGIN
 IF S='0' THEN
  f <= W0 )
 ELSE f <= W1 ;
 ENDIF;
 END PROCESS;
```

a 2-to-1 mux

```
PROCESS(W0,W1,S)
BEGIN
f <= W0;
IF S='1' THEN
f <= W1;
ENDIF;
ENDPROCESS;
```

ex:

PRIORITY ENCODER

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

irrelevant

```
ENTITY PRIORITY IS
PORT( W: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      Y: OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
      Z: OUT STD_LOGIC);
END PRIORITY;

ARCHITECTURE Behavior OF PRIORITY IS
BEGIN
PROCESS (W)
BEGIN
```

```
IF w(3) = '1' THEN
    y <= "11";
ELSIF w(2) = '1' THEN
    y <= "10";
ELSIF w(1) = '1' THEN
    y <= "01";
ELSE
    y <= "00";
END IF;
END PROCESS;
```

```
z <= '0' WHEN w = "0000"
    ELSE '1';
END Behavior;
```

Conditional (concurrent) Statement. Cannot be taken in the PROCESS statement.

putting all the statements in the PROCESS for the previous example:

```
ARCHITECTURE Behavior OF PRIORITY IS
BEGIN
PROCESS (w)
BEGIN
  y <= "00";  -> A default assignment

IF w(1)='1' THEN y<="01"; END IF;
IF w(2)='1' THEN y<="10"; END IF;
IF w(3)='1' THEN y<="11"; END IF;
  z <= '1';
IF w="0000" THEN z<='0'; END IF;
END PROCESS;
```

ex: Comparator (1-bit comparator ex ⇒ if A=1
B=0
Then A>B

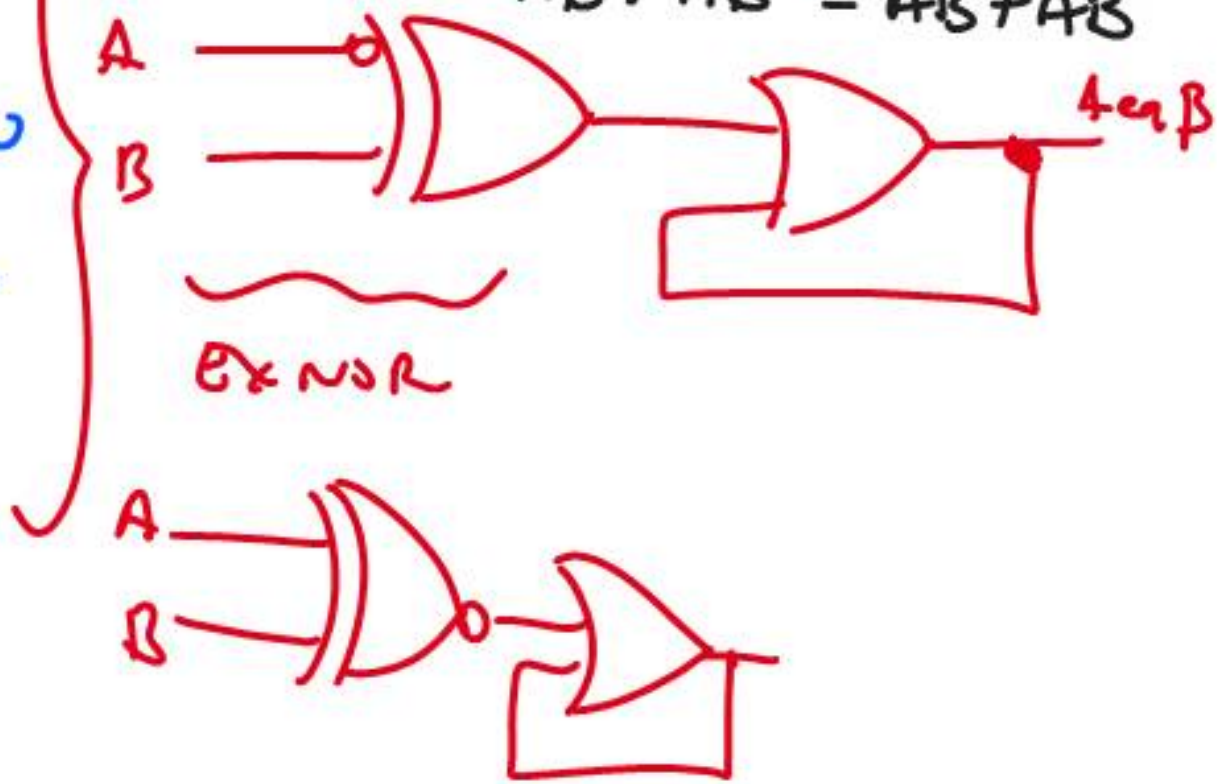
```
ENTITY COMPARATOR_1S
PORT( A, B : IN STD_LOGIC;
      AeqB : OUT STD_LOGIC);
END COMPARATOR_1S;
ARCHITECTURE Behavior OF COMPARATOR_1S
BEGIN
PROCESS(A.B)
BEGIN
AeqB <= '0';        -- Default
IF A=B THEN
AeqB <= '1';
END IF;
END PROCESS;
END BEHAVIOR;
```

If the default AeqB <= '0' statement is forgotten:

```
PROCESS(A,B)
BEGIN
  IF A=B THEN
    AeqB <= '1';
  END IF;
END PROCESS;
END BEHAVIOR;
```

This creates a memory element:

$\bar{A} \oplus B = \bar{A}B + A\bar{B} = AB + \bar{A}\bar{B}$



* CASE STATEMENT

used always in a PROCESS statement (since it is a sequential statement)

ex 2-to-1 mux

```
:
ARCHITECTURE Behavior OF ....
BEGIN
PROCESS(w0, w1, s)
BEGIN
CASE s IS
   WHEN '0' =>
      f <= w0;
   WHEN OTHERS =>
      f <= w1;
END CASE;
END PROCESS;
END BEHAVIOR;
```

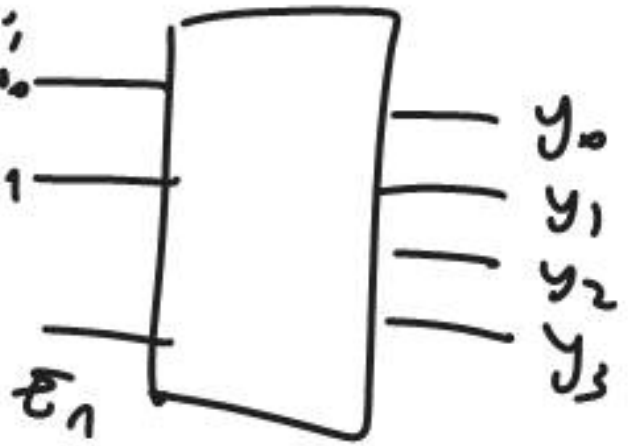
Remember

```
WITH s SELECT
f <= w0 WHEN '0';
f <= w1 WHEN OTHERS;
```

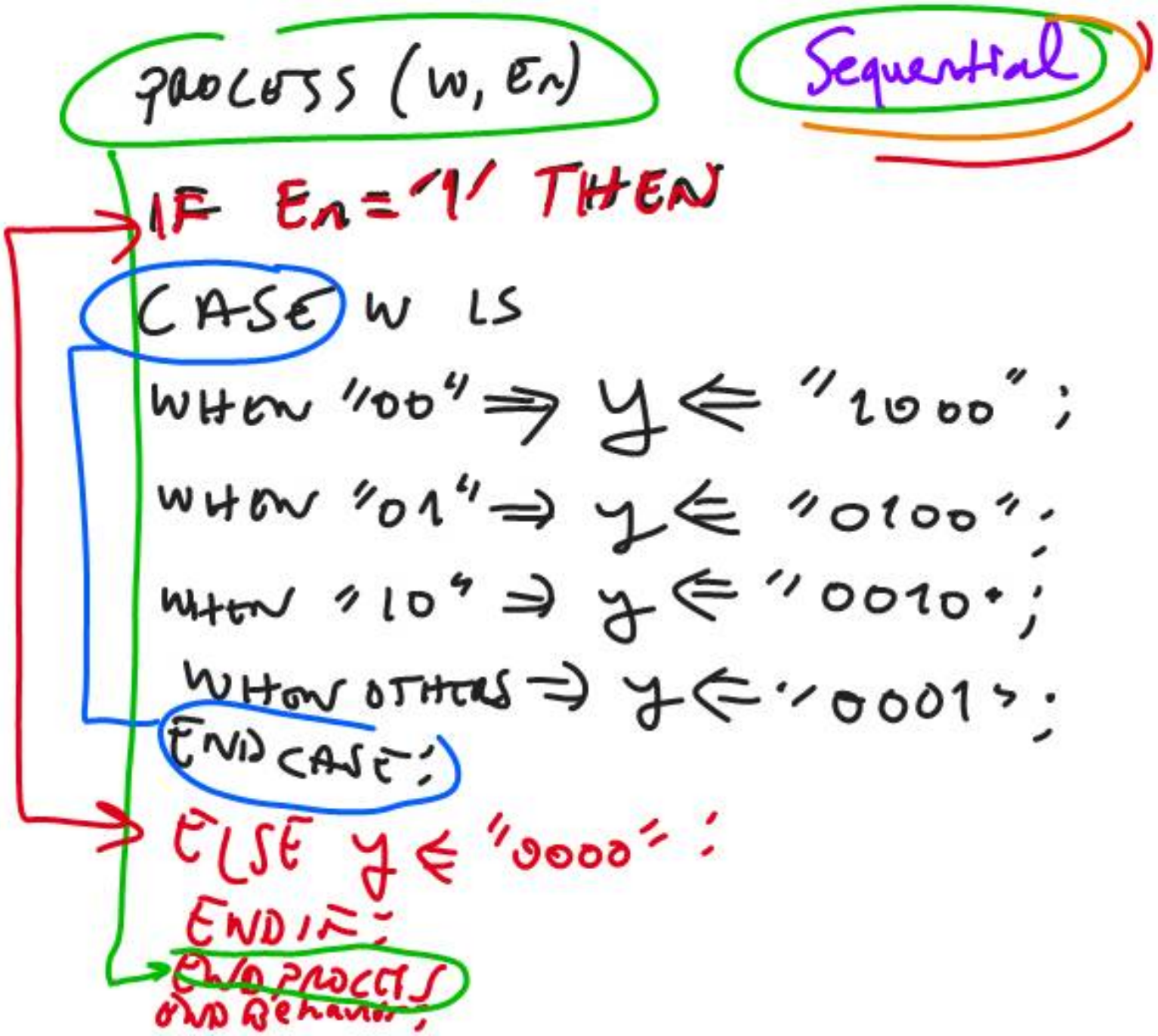
* 2-TO-4 DECODER WITH CASE STATEMENT

(IF AND CASE STATEMENTS TOGETHER)

```
:
PORT( W : IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
      En: IN  STD_LOGIC;
      Y : OUT STD_LOGIC_VECTOR(0 TO 3));
END DEC2TO4;
```




```
ARCHITECTURE BEHAVIOR OF DEC2TO4 IS
BEGIN
```


PROCESS (W, En)
Sequential
IF En='1' THEN
CASE W IS
WHEN "00" => y <= "1000";
WHEN "01" => y <= "0100";
WHEN "10" => y <= "0010";
WHEN OTHERS => y <= "0001";
END CASE;
ELSE y <= "0000";
END IF;
END PROCESS;

alternating way of doing the same thing:

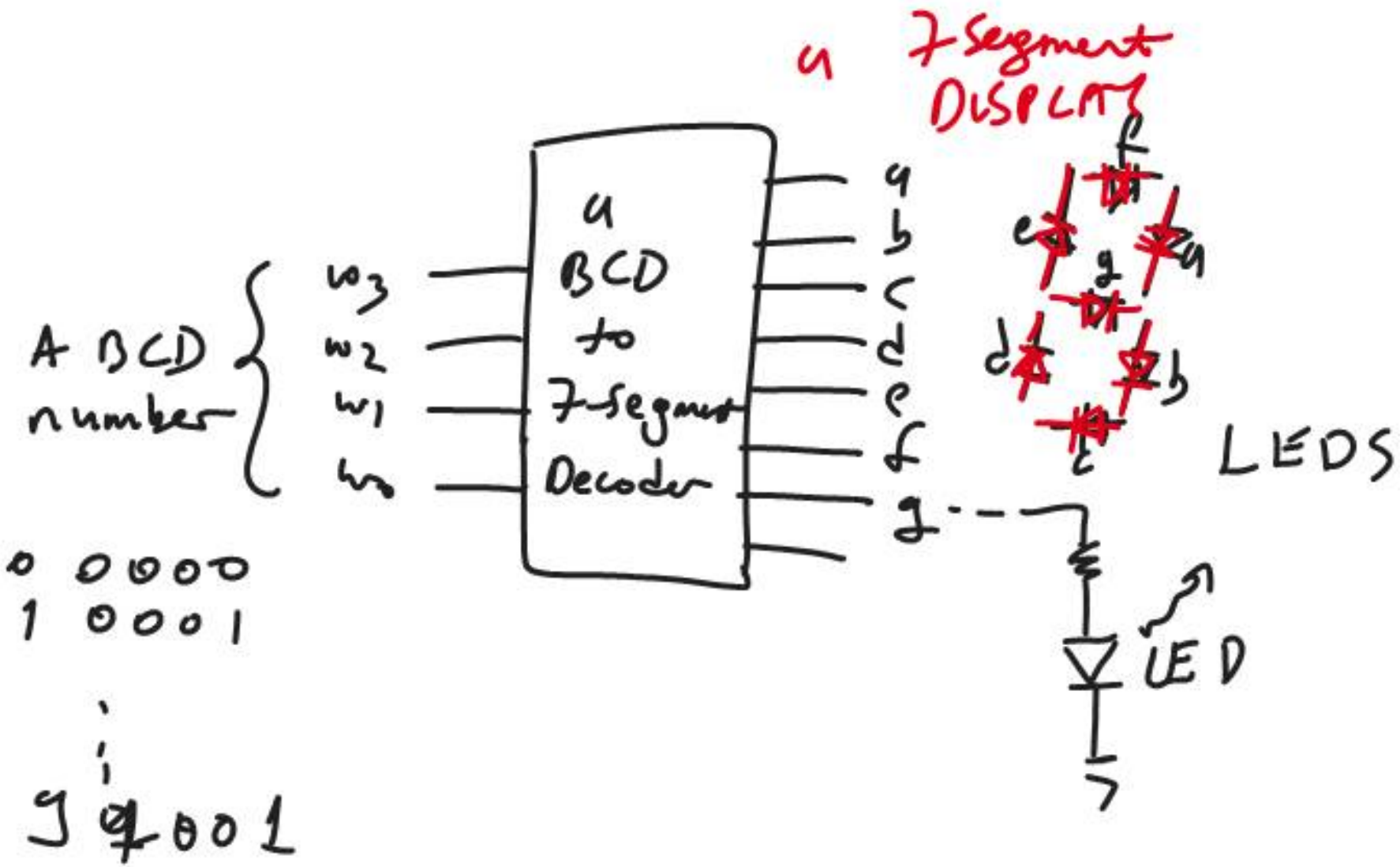
WITH CONDITIONAL STATEMENT (concurrent)

```
ARCHITECTURE Behavior OF RTL2TO4LS
SIGNAL EnW : STD_LOGIC_VECTOR (2 DOWNTO 0);
BEGIN
  EnW <= En & W;          -- Concatenation
  y <= "1000" WHEN EnW = "100" ELSE   -- EnW = "100"
       "0100" WHEN EnW = "101" ELSE
       "0010" WHEN EnW = "110" ELSE
       "0001" WHEN EnW = "111" ELSE
       "0000";
END Behavior;
```

Conditional Statement

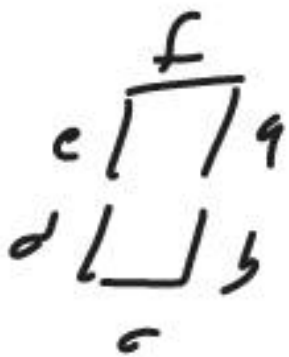
* using Selected (concurrent)

```
:
WITH ENW SELECT

y <= "1000" WHEN "200";
     "0100" WHEN "101";
     "0010" WHEN "110";
     "0001" WHEN "111";

"0000" WHEN OTHERS;
END BEHAVIOR;
```

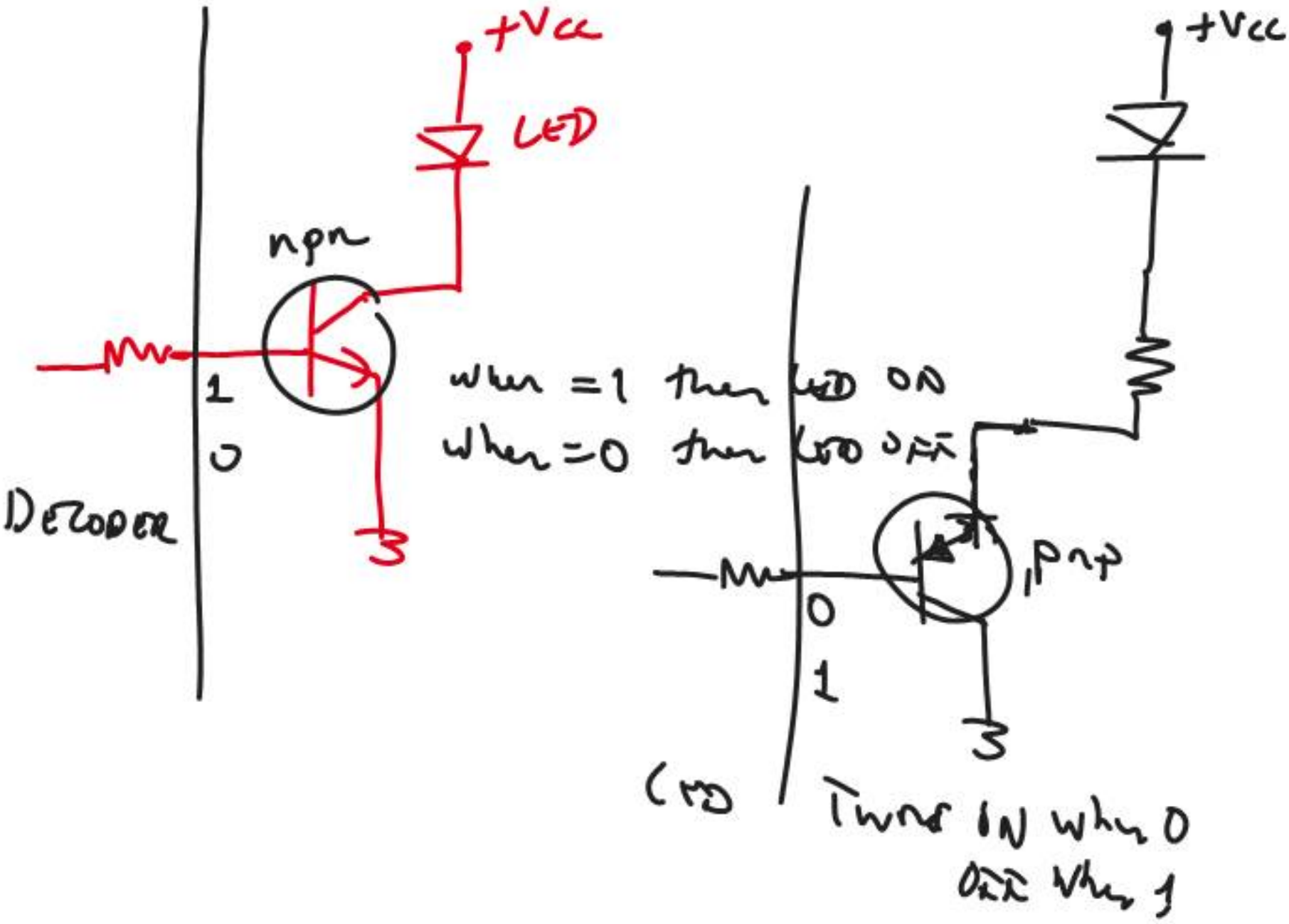

A BCD number
w3
w2
w1
w0
4 BCD to 7-segment Decoder
a
b
c
d
e
f
g
4 7 Segment DISPLAYS
LEDS
LED
0 0000
1 0001
9 1001

```
;
ENTITY SEVENSEG IS
  PORT ( bcd : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
         LEDs : OUT STD_LOGIC_VECTOR(0 TO 7)); -- wired
END SEVENSEG;

ARCHITECTURE Behavior OF SEVENSEG IS
BEGIN
PROCESS(bcd)
BEGIN
CASE bcd IS
```

```
                                    abcdefg
WHEN "0000" => LEDS <= "1111110";
WHEN "0001" => LEDS <= "0110000";
      :
      :
WHEN "1001" => LEDS <= "1110011";
WHEN OTHERS => LEDS <= "-------";
END CASE;
END PROCESS;
END BEHAVIOR;
```



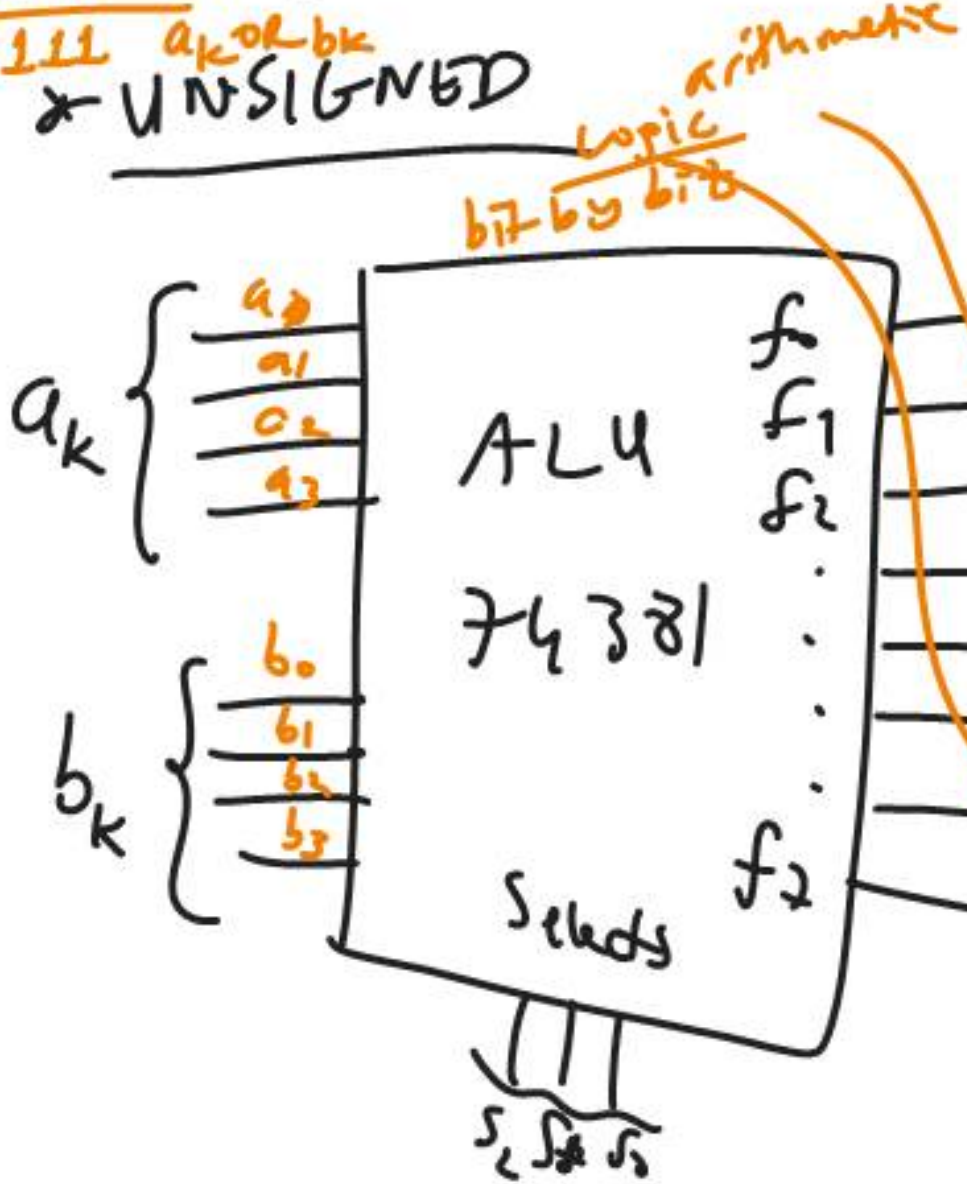
means DON'T CARE to VHDL


+Vcc
LED
npn
1
0
DECODER
3
when =1 then LED ON
when =0 then LED OFF
+Vcc
pnp
0
1
3
(TO
Turns ON when 0
OFF when 1

1011 a
0111 b
1111 a_k OR b_k

An Arithmetic logic Unit (ALU)

* UNSIGNED



OPERATION	Inputs S2 S1 S0	outputs f
Clear	0 0 0	0 0 0 0
B - A	0 0 1	B - A
A - B	0 1 0	A - B
ADD	0 1 1	A + B
XOR	1 0 0	A XOR B
OR	1 0 1	A OR B
AND	1 1 0	A AND B
preset	1 1 1	1 1 1 1