جامعة نينوى
كلية هندسة الإلكترونيات

Circuit Design with VHDL 7

Textbook: Volnei A. Pedroni

Submitted By: Hussein Aideen

VHDL> Sequential Code

- PROCESSES, FUNCTIONS, and PROCEDURES are executed sequentially.
- any of these blocks is still concurrent with any other statements placed outside it.
- with it we can build sequential circuits as well as combinational circuits.

VHDL> Sequential Code

- IF, WAIT, CASE, and LOOP.
- VARIABLES restricted to be used in sequential code.

VHDL> PROCESS

- A PROCESS must be installed in the main code.

- executed every time a signal in the sensitivity list changes.

```
[label:] PROCESS (sensitivity list)
  [VARIABLE name type [range] [:= initial_value;]]
BEGIN
  (sequential code)
END PROCESS [label];
```

VHDL> PROCESS

- initial value is not synthesizable.

- monitoring a signal (clock, for example) is necessary. A common way of detecting a signal change is by means of the EVENT attribute.
- For instance, if clk is a signal to be monitored, then clk'EVENT returns TRUE when a change on clk occurs (rising or falling edge).

VHDL> PROCESS

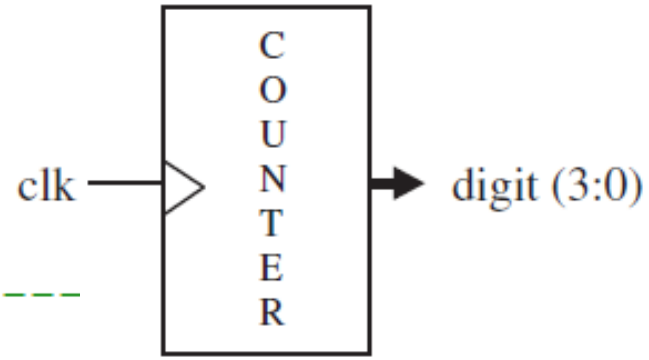
- IF statement:

```
IF conditions THEN assignments;
ELSIF conditions THEN assignments;
...
ELSE assignments;
END IF;
```

```
IF (x<y) THEN temp:="11111111";
ELSIF (x=y AND w='0') THEN temp:="11110000";
ELSE temp:=(OTHERS =>'0');
end if
```

VHDL> IF statement

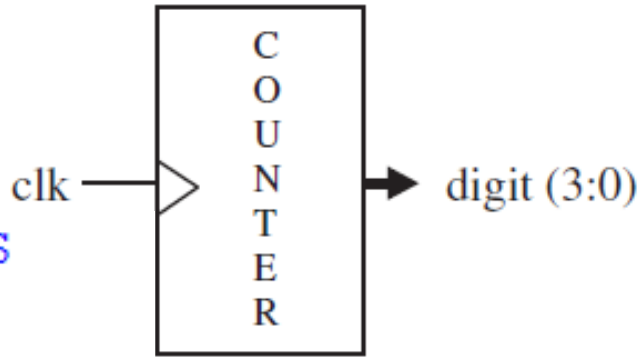
- One-digit Counter #1
  - 1-digit decimal counter (0 → 9 → 0).




```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY counter IS
6  PORT (clk : IN STD_LOGIC;
7  digit : OUT INTEGER RANGE 0 TO 9);
8  END counter;
9  ----------------------------------------
```

VHDL> IF statement

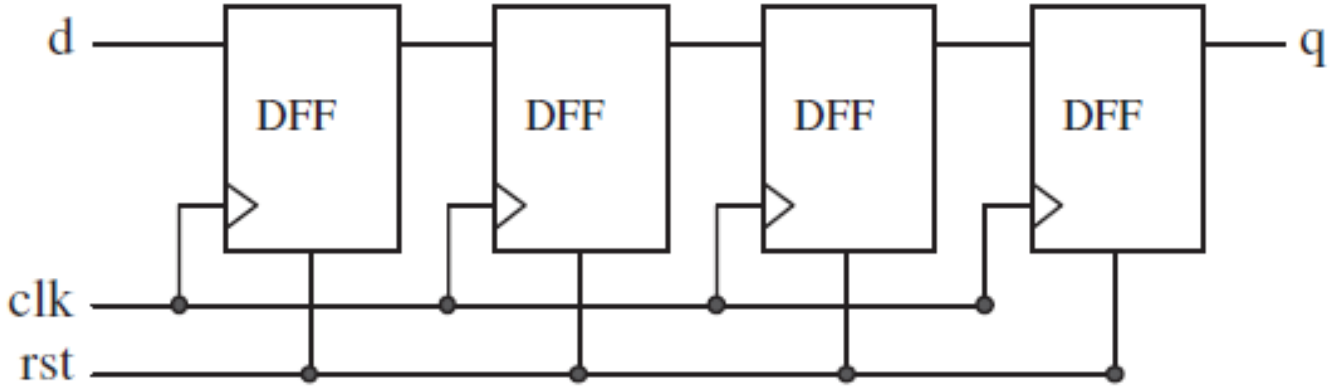
- One-digit Counter #1
  - 1-digit decimal counter (0 → 9 → 0).




```
10  ARCHITECTURE counter OF counter IS
11  BEGIN
12  count: PROCESS(clk)
13  VARIABLE temp : INTEGER RANGE 0 TO 10;
14  BEGIN
15  IF (clk'EVENT AND clk='1') THEN
16  temp := temp + 1;
17  IF (temp=10) THEN temp := 0;
18  END IF;
19  END IF;
20  digit <= temp;
21  END PROCESS count;
22  END counter;
23  ------------------
```

VHDL> IF statement

- 4 bit Shift Register:




```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY shiftreg IS
6  GENERIC (n: INTEGER := 4); -- # of stages
7  PORT (d, clk, rst: IN STD_LOGIC;
8  q: OUT STD_LOGIC);
9  END shiftreg;
```

VHDL> IF statement

- 4 bit Shift Register:

```
11  ARCHITECTURE behavior OF shiftreg IS
12  SIGNAL internal: STD_LOGIC_VECTOR (n-1 DOWNTO 0);
13  BEGIN
14  PROCESS (clk, rst)
15  BEGIN
16  IF (rst='1') THEN
17  internal <= (OTHERS => '0');
18  ELSIF (clk'EVENT AND clk='1') THEN
19  internal <= d & internal(internal'LEFT DOWNTO 1);
20  END IF;
21  END PROCESS;
22  q <= internal(0);
23  END behavior;
24  ------------------------------------------------
```

VHDL> WAIT statement

- WAIT statement: (inside Process)
- the PROCESS cannot have a sensitivity list when WAIT is employed.

```
WAIT UNTIL signal_condition;
```

```
WAIT ON signal1 [, signal2, ... ];
```

```
WAIT FOR time;
```

Simulation only

VHDL> WAIT statement

- WAIT statement:
- the PROCESS cannot have a sensitivity list when WAIT is employed.

```
Example: 8-bit_register _with synchronous reset.
PROCESS -- no sensitivity list
BEGIN
 WAIT UNTIL (clk'EVENT AND clk='1');
 IF (rst='1') THEN
   output <= "00000000";
 ELSIF (clk'EVENT AND clk='1') THEN
   output <= input;
 END IF;
END PROCESS;
```

VHDL> WAIT statement

- WAIT ON:

```
Example: 8-bit_register _with asynchronous reset.
PROCESS
BEGIN
 WAIT ON clk, rst;
 IF (rst='1') THEN
   output <= "00000000";
 ELSIF (clk'EVENT AND clk='1') THEN
   output <= input;
 END IF;
END PROCESS;
```

- WAIT FOR is intended for simulation only (waveform generation for test-benches). Example: WAIT FOR 5ns;

VHDL> WAIT statement

- Home Works:
- DFF with Asynchronous Reset #2, Page 99.
- One-digit Counter #2, Page 99-100.

VHDL> CASE statement

- CASE statement:

```
CASE identifier IS
    WHEN value => assignments;
    WHEN value => assignments;
    ...
END CASE;
```

```
CASE control IS
   WHEN "00" => x<=a; y<=b;
   WHEN "01" => x<=b; y<=c;
   WHEN OTHERS => x<="0000"; y<="ZZZZ";
END CASE;
```

VHDL> CASE statement

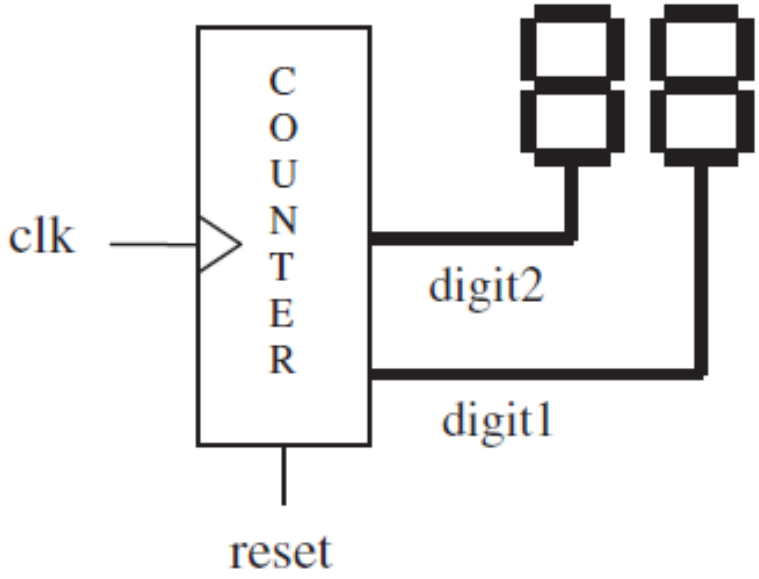
- CASE statement:

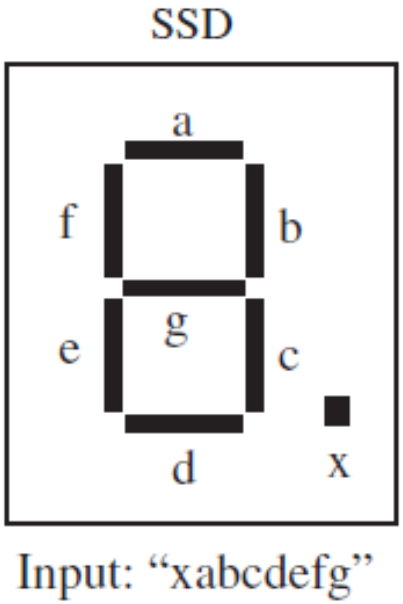
- CASE statement (sequential) is very similar to WHEN (combinational)

- keyword **OTHERS** is often helpful.
- Another important keyword is **NULL** (the counterpart of UNAFFECTED), which should be used when no action is to take place.
- CASE allows multiple assignments for each test condition.

VHDL> CASE statement

- Two-digit Counter with SSD Output:





VHDL> Two-digit Counter with SSD

```
1  ----------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------------------
5  ENTITY counter2digit IS
6  PORT (clk, reset : IN STD_LOGIC;
7  digit1, digit2 : OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
8  END counter2digit;
9  ----------------------------------------------------
10 ARCHITECTURE counter OF counter2digit IS
11 BEGIN
12 PROCESS(clk, reset)
13 VARIABLE temp1: INTEGER RANGE 0 TO 10;
14 VARIABLE temp2: INTEGER RANGE 0 TO 10;
15 BEGIN
16 ---- counter: ----------------------
17 IF (reset='1') THEN
18 temp1 := 0;
19 temp2 := 0;
20 ELSIF (clk'EVENT AND clk='1') THEN
```

VHDL> Two-digit Counter with SSD

```
20   ELSIF (clk'EVENT AND clk='1') THEN
21   temp1 := temp1 + 1;
22   IF (temp1=10) THEN
23   temp1 := 0;
24   temp2 := temp2 + 1;
25   IF (temp2=10) THEN
26   temp2 := 0;
27   END IF;
28   END IF;
29   END IF;
30   ---- BCD to SSD conversion: --------
```

VHDL> Two-digit Counter with SSD

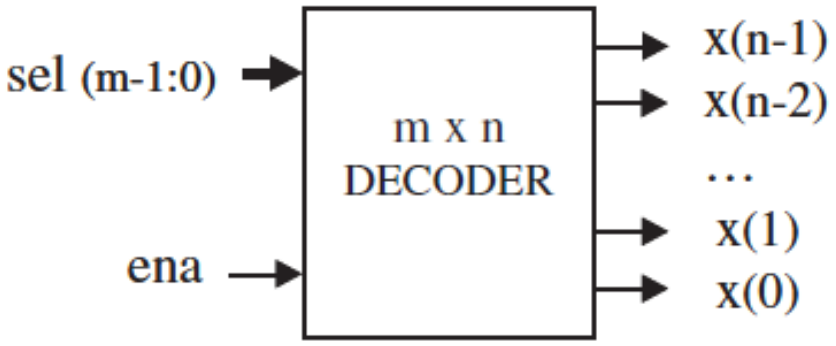
```
30   ---- BCD to SSD conversion: --------
31   CASE temp1 IS
32   WHEN 0 => digit1 <= "1111110"; --7E
33   WHEN 1 => digit1 <= "0110000"; --30
34   WHEN 2 => digit1 <= "1101101"; --6D
35   WHEN 3 => digit1 <= "1111001"; --79
36   WHEN 4 => digit1 <= "0110011"; --33
37   WHEN 5 => digit1 <= "1011011"; --5B
38   WHEN 6 => digit1 <= "1011111"; --5F
39   WHEN 7 => digit1 <= "1110000"; --70
40   WHEN 8 => digit1 <= "1111111"; --7F
41   WHEN 9 => digit1 <= "1111011"; --7B
42   WHEN OTHERS => NULL;
43   END CASE;
```

VHDL> Two-digit Counter with SSD

```
44   CASE temp2 IS
45   WHEN 0 => digit2 <= "1111110"; --7E
46   WHEN 1 => digit2 <= "0110000"; --30
47   WHEN 2 => digit2 <= "1101101"; --6D
48   WHEN 3 => digit2 <= "1111001"; --79
49   WHEN 4 => digit2 <= "0110011"; --33
50   WHEN 5 => digit2 <= "1011011"; --5B
51   WHEN 6 => digit2 <= "1011111"; --5F
52   WHEN 7 => digit2 <= "1110000"; --70
53   WHEN 8 => digit2 <= "1111111"; --7F
54   WHEN 9 => digit2 <= "1111011"; --7B
55   WHEN OTHERS => NULL;
56   END CASE;
57   END PROCESS;
58   END counter;
59   ----------------------------------
```

VHDL> Examples

- Generic **Decoder:**
  - If ena = '0', then all bits of x should be high; otherwise, the output bit selected by sel should be low.



$m = \log_2 n$

ena	sel	x
0	00	1111
1	00	1110
	01	1101
	10	1011
	11	0111

VHDL> Examples

- Generic **Decoder:**

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
--------------------------------------------
ENTITY decoder IS
PORT ( ena : IN STD_LOGIC;
sel : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
x : OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END decoder;
ARCHITECTURE generic_decoder OF decoder IS
BEGIN
PROCESS (ena, sel)
VARIABLE temp1 : STD_LOGIC_VECTOR (x'HIGH DOWNTO 0);
VARIABLE temp2 : INTEGER RANGE 0 TO x'HIGH;
BEGIN
temp1 := (OTHERS => '1');
IF (ena='1') THEN
temp2:=conv_integer(signed(sel));
temp1(temp2):='0';
END IF;
x <= temp1;
END PROCESS;
END generic_decoder;
```

VHDL> Examples

- Gray code counter:

```
entity gray_counter is
   port (
      cout   :out std_logic_vector (7 downto 0);  -- Output of t
      enable :in  std_logic;                      -- Enable counting
      clk    :in  std_logic;                      -- Input clock
      reset  :in  std_logic                       -- Input reset
   );
end entity;
```

binary	gray
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

```
architecture rtl of gray_counter is
   signal count :std_logic_vector (7 downto 0);
begin
   process (clk, reset) begin
      if (reset = '1') then
         count <= (others=>'0');
      elsif (rising_edge(clk)) then
         if (enable = '1') then
            count <= count + 1;
         end if;
      end if;
   end process;

   cout <= (count(7) &
         (count(7) xor count(6)) &
         (count(6) xor count(5)) &
         (count(5) xor count(4)) &
         (count(4) xor count(3)) &
         (count(3) xor count(2)) &
         (count(2) xor count(1)) &
         (count(1) xor count(0)) );
end architecture;
```