El-Shorouk Academy	The Higher Institute of Engineering
Communication and Computer Department	
First Semester 2020/2021 Third Year	
Course Name : Electronic circuits design using computer	Course Code : CCE 317
Dr. Fatma Elfouly	

Solution of Sheet No.1

1- VHDL code for NAND gate

```
entity NAND_gate is
port (a, b: in bit;
      x: out bit);
end entity NAND_gate;
architecture rtl of NAND_gate is
begin
x <= a NAND b;
end architecture rtl;
```

2- VHDL code for OR gate

```
entity OR_gate is
port (a, b: in bit;
      x: out bit);
end entity OR_gate;
architecture rtl of OR_gate is
begin
x <= a OR b;
end architecture rtl;
```

Solution of Sheet No. 2

1- VHDL code for 4 to 1 multiplixer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux4_1 is
port (sel: in std_logic_vector (1 downto 0);
a, b, c, d: in std_logic;
y: out std_logic);
end entity mux4_1;
architecture rtl of mux4_1 is
begin
with sel select
y <= a when "00",
b when "01",
c when "10",
d when "11",
a when others;
end architecture rtl;
```

2- VHDL code for 8 to 1 multiplixer

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity mux8_1 is
port (sel: in std_logic_vector (2 downto 0);
a: in std_logic_vector(7 downto 0);
y: out std_logic);
end entity mux8_1;
architecture rtl of mux8_1 is
begin
process (sel, a)
begin
if (sel = "000") then
y <= a(0);
elsif (sel = "001") then
y <= a(1);
elsif (sel = "010") then
y <= a(2);
elsif (sel = "011") then
y <= a(3);
elsif (sel = "100") then
y <= a(4);
elsif (sel = "101") then
y <= a(5);
elsif (sel = "110") then
y <= a(6);
else
y <= a(7);
end if;
end process;
end architecture rtl;
```

Solution of Sheet No. 3

1- VHDL code for 2 to 4 decoder

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity decoder2_4 is
port (a: in std_logic_vector(1 downto 0);
y: out std_logic_vector(3 downto 0));
end entity decoder2_4;
architecture rtl of decoder2_4 is
begin
y <= "0001" when a = "00" else
"0010" when a = "01" else
```

```
"0100" when a = "010" else
"1000";
end architecture rtl;
```

2- VHDL code for 3 to 8 decoder

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity decoder3_8 is
port (a: in std_logic_vector(2 downto 0);
y: out std_logic_vector(7 downto 0));
end entity decoder3_8;
architecture rtl of decoder3_8 is
begin
y <= "00000001" when a = "000" else
"00000010" when a = "001" else

"00000100" when a = "010" else
"00001000" when a = "011" else
"00010000" when a = "100" else
"00100000" when a = "101" else
"01000000" when a = "110" else
"10000000" when a = "111" else
"00000001";
end architecture rtl;
```

Solution of Sheet No. 4

1- VHDL code for Latch

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity d_latch is
port (d, en: in std_logic;
q: out std_logic);
end entity d_latch;
architecture rtl of d_latch is
begin
process (d, en)
begin
if (en = '1') then
q <= d;
end if;
end process;
end architecture rtl;
```

2- VHDL code for D Flip-Flop

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity d_flip_flop is
port (d, clk, rst: in std_logic;
q: out std_logic);
end entity d_flip_flop;
architecture rtl of d_flip_flop is
begin
process (rst, clk)
begin
if (rst = '1') then
q <= '0';
elsif (rising_edge(clk)) then
q <= d;
end if;
end process;
end architecture rtl;
```

3- VHDL code for 8 bit parallel register

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity reg_par is
port (clk, rst: in std_logic;
reg_in: in std_logic_vector(7 downto 0);
reg_out: out std_logic_vector(7 downto 0));
end entity reg_par;
architecture rtl of reg_par is
begin
process (rst, clk)
begin
if (rst = '1') then
reg_out <= "00000000";
elsif (rising_edge(clk)) then
reg_out <= reg_in;
end if;
end process;
end architecture rtl;
```

Solution of Sheet No. 5

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity reg_par_shift is
port (clk, rst: in std_logic;
shift_right, shift_left: in std_logic;
reg_in: in std_logic_vector(7 downto 0);
reg_out: out std_logic_vector(7 downto 0));
end entity reg_par_shift;
architecture rtl of reg_par_shift is
signal shift_control: std_logic_vector(1 downto 0);
signal reg_temp: std_logic_vector(7 downto 0);
begin
shift_control <= shift_left & shift_right;
process (rst, clk)
begin
if (rst = '1') then
reg_temp <= (others => '0');
elsif (rising_edge(clk)) then
case shift_control is
when "00" => reg_temp <= reg_in;
when "01" => reg_temp <= '0' & reg_temp(7 downto
1);
when "10" => reg_temp <= reg_temp(6 downto 0) &
'0';
when others => reg_temp <= reg_temp; --can be
omitted
end case;
end if;
end process;
reg_out <= reg_temp;
end architecture rtl;
```

Solution of Sheet No. 6

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity binary_count is
port (clk, rst: in std_logic;
count_out: out std_logic_vector(3 downto 0));
end entity binary_count;
architecture rtl of binary_count is
signal count_temp: unsigned(3 downto 0);
begin
process (rst, clk)
begin
```

```
if (rst = '1') then
count_temp <= (others => '0');
elsif (rising_edge(clk)) then
count_temp <= count_temp + 1;
end if;
end process;
count_out <= std_logic_vector(count_temp);
end architecture rtl;
```

Solution of Sheet No. 7

1- VHDL code for half adder

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity HA is
port (x, y: in std_logic;
      s, c: out std_logic);
end entity HA;
architecture rtl of HA is
begin
s <= x XOR y;
c <= x AND y;
end architecture rtl;
```

2- VHDL code for half subtractor

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity HS is
port (x, y: in std_logic;
      d, b: out std_logic);
end entity HS;
architecture rtl of HS is
begin
s <= x XOR y;
c <= (NOT (x)) AND y;
end architecture rtl;
```

3- VHDL code for full adder

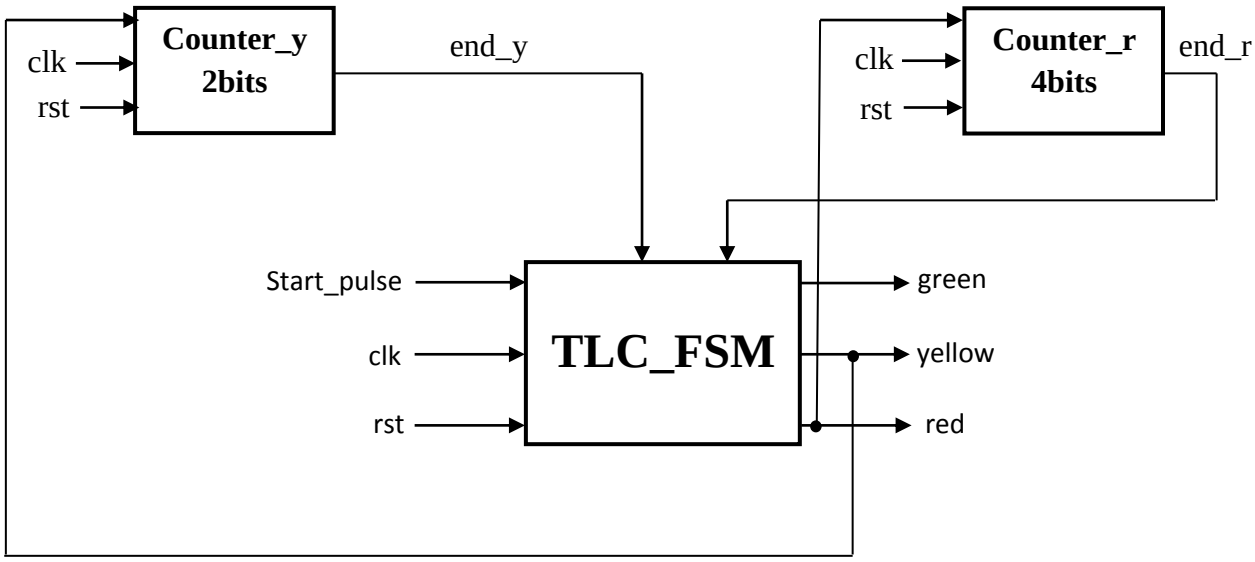
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity FA is
port (A, B, Cin: in std_logic;
      sum, cout: out std_logic);
end entity FA;
```

```
architecture rtl of FA is
begin
sum <= A XOR B XOR cin;
cout <= (A AND B) OR (B AND Cin) OR (A AND Cin);
end architecture rtl;
```

4- VHDL code for full subtractor

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
entity FS is
port (A, B, bin: in std_logic;
      d, b: out std_logic);
end entity FS;
architecture rtl of FS is
begin
d <= A XOR B XOR bin;
b <= ((NOT(A) AND B) OR (B AND bin) OR (NOT(A) AND bin);
end architecture rtl;
```

Solution of Sheet No. 8



Count_4bits code:

```
ENTITY count_4bits IS
port(clk,en,rst: in std_logic;
q: out std_logic);
END count_4bits ;
ARCHITECTURE rtl OF count_4bits IS
signal count_sig: unsigned (3 downto 0);
BEGIN
process (rst,clk)
begin
if (rst='1') then
count_sig <=(others=> '0');
elsif (rising_edge (clk)) then
if (en='1') then
count_sig <= count_sig+1;
end if;
end if;
end process;
q <= count_sig(0) and count_sig(1)and count_sig(2) and
count_sig(3);
END rtl;
```

Count_2bits code:

```
ENTITY count_2bits IS
port(clk,en,rst: in std_logic;
q: out std_logic);
END count_2bits;
ARCHITECTURE rtl OF count_2bits IS
signal count_sig: unsigned (1 downto 0);
BEGIN
process (rst,clk)
begin
if (rst='1') then
count_sig <=(others=> '0');
elsif (rising_edge (clk)) then
if (en='1') then
count_sig <= count_sig+1;
end if;
end if;
end process;
q <= count_sig(0) and count_sig(1);
END rtl;
```

TLC_FSM:
FSM is the most common style to describe control.