Εργαστηριακή Άσκηση 6

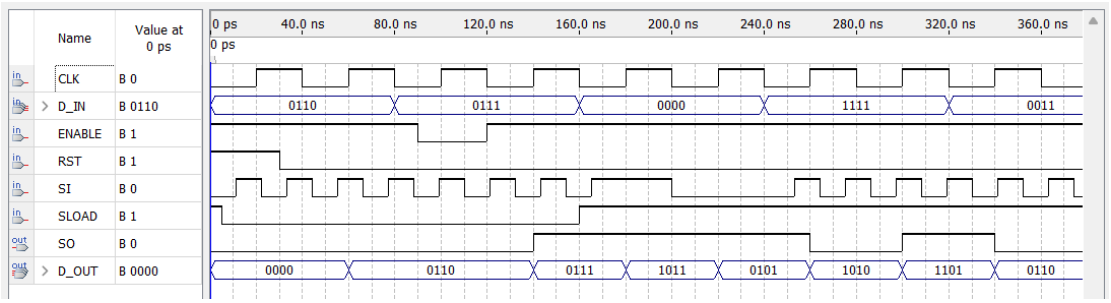
ΠΑΝΑΓΙΩΤΗΣ ΓΕΩΡΓΙΟΥ, ΑΜ: 4553
ΧΡΗΣΤΟΣ ΚΟΥΤΟΥΛΗΣ ΑΜ:5064
ΜΑΡΙΑ ΚΑΤΩΛΗ, ΑΜ:5083
ΟΜΑΔΑ 2

Καταχωρητής

VHDL:

```
1   library IEEE;
2   use IEEE.std_logic_1164.all;
3
4   entity Reg is
5      generic (n: integer:=4);
6      port (
7         D_IN: in std_logic_vector (n-1 downto 0);
8         SI, CLK, RST, SLOAD, ENABLE: in std_logic;
9         SO: out std_logic;
10        D_OUT: out std_logic_vector (n-1 downto 0));
11  end Reg;
12
13  architecture RTL of Reg is
14  signal F: std_logic_vector (n-1 downto 0);
15  begin
16  p0: process (RST, CLK)
17  begin
18     if (RST='1') then F<=(n-1 downto 0 => '0');
19     elsif (CLK'event and CLK='1') then
20        if (ENABLE='1') then
21           if (SLOAD='0') then F<=D_IN;
22           else F<=SI & F(n-1 downto 1);
23           end if;
24        end if;
25     end if;
26  end process;
27  D_OUT<=F;
28  SO<=F(0);
29  end RTL;
```

Εξομοίωση (Reg)

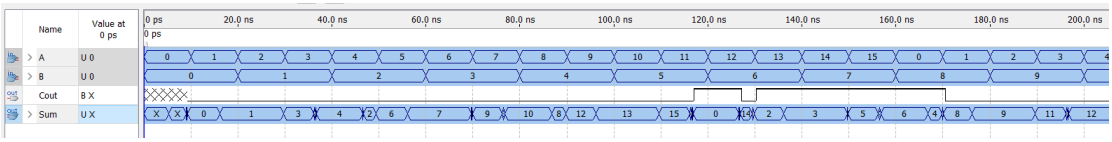


Αθροιστής

VHDL:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity Adder is
7      generic ( n: integer := 4 );
8      port ( A, B: in std_logic_vector (n-1 downto 0);
9          Sum: out std_logic_vector (n-1 downto 0);
10         Cout: out std_logic );
11 end Adder;
12
13 architecture DataFlow of Adder is
14     signal F: std_logic_vector (n downto 0);
15 begin
16     F <= ('0' & A) + ('0' & B);
17     Cout <= F(n);
18     Sum <= F(n-1 downto 0);
19 end DataFlow;
```

Εξομοίωση



Control Logic

VHDL:

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity CtrlLogic is
   generic (n: integer:=8);
   port( Rst,CLK: in std_logic;
         SL_A,SL_B,SL_H,SL_L,SL_C: out std_logic;
         EN_A,EN_B,EN_H,EN_L,EN_C: out std_logic;
         FLAG: out std_logic);
end CtrlLogic;

architecture RTL of CtrlLogic is
type state_type is (LOAD, ADD, SHIFT, FINISH);
signal state: state_type;
signal count: std_logic_vector (n downto 0);
begin
p0: process(Rst,CLK)
begin
   if (Rst='1') then
      count <= (n downto 0 => '0');
   elsif (CLK'event and CLK='1') then
      count <= count + '1';
   end if;
end process;
p1: process(Rst,CLK)
begin
   if (Rst='1') then state <= LOAD;
   elsif (CLK'event and CLK='1') then
      case state is
         when LOAD=> state <= ADD;
         when ADD=> state<=SHIFT;
         when SHIFT=> if (conv_integer(count)=2*n) then state <= FINISH; else state <= ADD; end if;
         when FINISH=> null;
      end case;
   end if;
end process;
EN_A <= '1' when (state=LOAD) else '0';
SL_A <= '0';
EN_B <= '1' when (state=LOAD or state=SHIFT) else '0';
SL_B <= '1' when (state=SHIFT) else '0';
EN_H <= '1' when (state=ADD or state=SHIFT) else '0';
SL_H <= '1' when (state=SHIFT) else '0';
EN_L <= '1' when (state=SHIFT) else '0';
SL_L <= '1' when (state=SHIFT) else '0';
EN_C <= '1' when (state=ADD) else '0';
SL_C <= '0';
FLAG <= '1' when (state=FINISH) else '0';
end RTL;
```

Test bench:

```vhdl
LIBRARY ieee  ;
LIBRARY std  ;
USE ieee.std_logic_1164.all  ;
USE ieee.std_logic_textio.all  ;
USE ieee.STD_LOGIC_UNSIGNED.all  ;
USE ieee.std_logic_unsigned.all  ;
USE std.textio.all  ;
use work.Declarations.all;

ENTITY CtrlLogic_tb  IS
  GENERIC (
    n  : INTEGER   := 4 );
END ;

ARCHITECTURE CtrlLogic_tb_arch OF CtrlLogic_tb IS
  SIGNAL SL_B   :  STD_LOGIC  ;
  SIGNAL RST   :  STD_LOGIC  ;
  SIGNAL SL_C   :  STD_LOGIC  ;
  SIGNAL EN_A   :  STD_LOGIC  ;
  SIGNAL EN_B   :  STD_LOGIC  ;
  SIGNAL EN_C   :  STD_LOGIC  ;
  SIGNAL SL_H   :  STD_LOGIC  ;
  SIGNAL CLK   :  STD_LOGIC  ;
  SIGNAL SL_L   :  STD_LOGIC  ;
  SIGNAL EN_H   :  STD_LOGIC  ;
  SIGNAL EN_L   :  STD_LOGIC  ;
  SIGNAL SL_A   :  STD_LOGIC  ;
  SIGNAL monitor_count : std_logic_vector (n downto 0 );
  signal monitor_state  :  state_type;



  COMPONENT CtrlLogic
    GENERIC (
      n  : INTEGER  );
    PORT (
      SL_B  : out STD_LOGIC ;
      RST  : in STD_LOGIC ;
      SL_C  : out STD_LOGIC ;
      EN_A  : out STD_LOGIC ;
      EN_B  : out STD_LOGIC ;
      EN_C  : out STD_LOGIC ;
      SL_H  : out STD_LOGIC ;
      CLK  : in STD_LOGIC ;
      SL_L  : out STD_LOGIC ;
      EN_H  : out STD_LOGIC ;
      EN_L  : out STD_LOGIC ;
      SL_A  : out STD_LOGIC );
  END COMPONENT ;
BEGIN
  DUT  : CtrlLogic
    GENERIC MAP (
      n  => n   )
    PORT MAP (
      SL_B   => SL_B  ,
      RST   => RST  ,
      SL_C   => SL_C  ,
      EN_A   => EN_A  ,
      EN_B   => EN_B  ,
      EN_C   => EN_C  ,
      SL_H   => SL_H  ,
      CLK   => CLK  ,
      SL_L   => SL_L  ,
      EN_H   => EN_H  ,
      EN_L   => EN_L  ,
      SL_A   => SL_A   ) ;

```

```
71  -- Start Time = 20 ns, End Time = 1 us, Period = 0 ns
72    Process
73     Begin
74      rst  <= '1'  ;
75     wait for 20 ns ;
76      rst  <= '0'  ;
77     wait for 980 ns ;
78   -- dumped values till 1 us
79     wait;
80    End Process;
81
82
83  -- "Clock Pattern" : dutyCycle = 50
84  -- Start Time = 0 ns, End Time = 1 us, Period = 20 ns
85    Process
86     Begin
87      clk  <= '0'  ;
88     wait for 10 ns ;
89   -- 10 ns, single loop till start period.
90     for Z in 1 to 49
91     loop
92         clk  <= '1'  ;
93        wait for 10 ns ;
94         clk  <= '0'  ;
95        wait for 10 ns ;
96   -- 990 ns, repeat pattern in loop.
97     end  loop;
98      clk  <= '1'  ;
99     wait for 10 ns ;
100  -- dumped values till 1 us
101    wait;
102   End Process;
103   monitor_count <= <<signal DUT.count : std_logic_vector(n downto 0) >>;
```

Πολλαπλασιαστής

VHDL:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity Multiplier is
    generic (n: integer := 8);
    port (Rst, CLK, SI : in std_logic;
          I_A,I_B: in std_logic_vector (n-1 downto 0);
          Low , High : out std_logic_vector (n-1 downto 0);
          A_OUT,B_OUT: out std_logic_vector (n-1 downto 0);
          C_OUT: out std_logic_vector (0 downto 0);
          FIN: out std_logic);
end Multiplier;


architecture RTL of Multiplier is
    component Reg
       generic (n: integer);
       port (D_IN: in std_logic_vector (n-1 downto 0);
             SI, CLK, RST, SLOAD, ENABLE: in std_logic;
             SO: out std_logic;
             D_OUT: out std_logic_vector (n-1 downto 0));
    end component;
    component Adder
       generic (n: integer:=8);
       port (A, B: in std_logic_vector (n-1 downto 0);
             SUM: out std_logic_vector (n-1 downto 0);
             COUT: out std_logic);
    end component;
    component CtrlLogic
       generic (n: integer:=8);
       port (Rst, CLK : in std_logic;
             SL_A, SL_B, SL_H, SL_L, SL_C : out std_logic;
             EN_A, EN_B, EN_H, EN_L, EN_C : out std_logic;
```

```
          FLAG: out std_logic);
  end component;
  signal SL_A, SL_B, SL_H, SL_L, SL_C, EN_A, EN_B, EN_H, EN_L, EN_C: std_logic;
  signal SO_A, SO_H, F: std_logic;
  signal A, B, SUM, H: std_logic_vector (n-1 downto 0);
  signal C, COUT : std_logic_vector (0 downto 0);
  begin
  R_A: Reg generic map (n)
          port map (I_A, SI, CLK, RST, SL_A, EN_A, SO_A, A);
  A_OUT <= A;
  R_B: Reg generic map (n)
          port map (I_B, SO_A, CLK, RST, SL_B, EN_B, open, B);
  B_OUT <= B;
  R_C: Reg generic map (n => 1)
          port map (COUT, '0', CLK, RST, SL_C, EN_C, open, C);
  C_OUT <= C;
  R_H: Reg generic map (n)
          port map (SUM, C(0), CLK, RST, SL_H, EN_H, SO_H, H);
  High <= H;
  R_L: Reg generic map (n)
          port map ((n-1 downto 0 => '0'), SO_H, CLK, RST, SL_L, EN_L, open, Low);
  U_Add: Adder generic map (n)
              port map (H, ((n-1 downto 0 => B(0)) and A), SUM, COUT(0));
  U_Ctl: CtrlLogic GENERIC MAP (n)
            port map (RST, CLK, SL_A, SL_B, SL_H, SL_L, SL_C, EN_A, EN_B, EN_H, EN_L, EN_C, F);
  FIN<=F;
END RTL;
```

Test bench:

```
LIBRARY ieee  ;
LIBRARY std  ;
USE ieee.std_logic_1164.all  ;
USE ieee.std_logic_textio.all  ;
USE ieee.STD_LOGIC_UNSIGNED.all  ;
USE ieee.std_logic_unsigned.all  ;
USE std.textio.all  ;
use work.Declarations.all;

ENTITY Multiplier_tb  IS
  GENERIC (
    n  : INTEGER   := 4 );
END ;

ARCHITECTURE Multiplier_tb_arch OF Multiplier_tb IS
  SIGNAL A_OUT   :  STD_LOGIC_VECTOR (n - 1 downto 0)  ;
  SIGNAL SI   :  STD_LOGIC  ;
  SIGNAL B_OUT   :  STD_LOGIC_VECTOR (n - 1 downto 0)  ;
  SIGNAL RST   :  STD_LOGIC  ;
  SIGNAL CLK   :  STD_LOGIC  ;
  SIGNAL H_OUT   :  STD_LOGIC_VECTOR (n - 1 downto 0)  ;
  SIGNAL L_OUT   :  STD_LOGIC_VECTOR (n - 1 downto 0)  ;
   signal monitor_state :  state_type;

  COMPONENT Multiplier
    GENERIC (
      n  : INTEGER  );
    PORT (
      A_OUT  : out STD_LOGIC_VECTOR (n - 1 downto 0) ;
      SI  : in STD_LOGIC ;
      B_OUT  : out STD_LOGIC_VECTOR (n - 1 downto 0) ;
      RST  : in STD_LOGIC ;
      CLK  : in STD_LOGIC ;
      H_OUT  : out STD_LOGIC_VECTOR (n - 1 downto 0) ;
```

```
      L_OUT   : out STD_LOGIC_VECTOR (n - 1 downto 0) );
  END COMPONENT ;
BEGIN
  DUT   : Multiplier
    GENERIC MAP (
      n   => n    )
    PORT MAP (
      A_OUT    => A_OUT   ,
      SI    => SI   ,
      B_OUT    => B_OUT   ,
      RST    => RST   ,
      CLK    => CLK   ,
      H_OUT    => H_OUT   ,
      L_OUT    => L_OUT    ) ;


-- "Constant Pattern"
-- Start Time = 5 ns, End Time = 1 us, Period = 0 ns
  Process
   Begin
    rst   <= '1'   ;
   wait for 5 ns ;
    rst   <= '0'   ;
   wait for 995 ns ;
-- dumped values till 1 us
   wait;
 End Process;


-- "Clock Pattern" : dutyCycle = 50
-- Start Time = 0 ns, End Time = 1 us, Period = 20 ns
  Process
```

```
  Process
   Begin
    clk  <= '0'  ;
   wait for 10 ns ;
-- 10 ns, single loop till start period.
   for Z in 1 to 49
   loop
       clk  <= '1'  ;
      wait for 10 ns ;
       clk  <= '0'  ;
      wait for 10 ns ;
-- 990 ns, repeat pattern in loop.
   end  loop;
    clk  <= '1'  ;
   wait for 10 ns ;
-- dumped values till 1 us
   wait;
 End Process;
 SI   <= '1',  '0'   after 60 ns,   '1'   after 80 ns,   '0'   after 100   ns,   '1'   after 120   ns,   '0'   after 140   ns;
 monitor_state <= << signal DUT.U_Ctl.state  :  state_type  >>;

process
   begin
      wait  on monitor_state;
      if (monitor_state = FINISH)   then
         wait  on clk;
         assert (FALSE) report "Checking..." severity note;
         --assert (L_OUT="0011"  and H_OUT="0010") report "Check  Failed"  severity error;
         assert(L_OUT="0011" and H_OUT="0010") report "Check Failed" severity error;
      end   if;
end   process;
END;
```

Control Logic (Παραδοτέα)

VHDL:

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.std_logic_unsigned.all;
4
5  entity CtrlLogic_SI is
6     generic (n: integer:=4);
7     port( Rst,CLK: in std_logic;
8           SL_A,SL_B,SL_H,SL_L,SL_C: out std_logic;
9           EN_A,EN_B,EN_H,EN_L,EN_C: out std_logic);
10 end CtrlLogic_SI;
11
12 architecture RTL of CtrlLogic_SI is
13 type state_type is (LOAD, ADD, SHIFT, FINISH);
14 signal state: state_type;
15 signal count: std_logic_vector (n downto 0);
16 begin
17 p0: process(Rst,CLK)
18 begin
19    if (Rst='1') then
20       count <= (n downto 0 => '0');
21    elsif (CLK'event and CLK='1') then
22       count <= count + '1';
23    end if;
24 end process;
25 p1: process(Rst,CLK)
26 begin
27    if (Rst='1') then state <= LOAD;
28    elsif (CLK'event and CLK='1') then
29       case state is
30          when LOAD=> if (conv_integer(count)=2*n-1) then state <= ADD; end if;
31          when ADD=> state<=SHIFT;
32          when SHIFT=> if (conv_integer(count)=4*n-1) then state <= FINISH; else state <= ADD; end if;
33          when FINISH=> null;
34       end case;
```

```
35    end if;
36 end process;
37 EN_A <= '1' when (state=LOAD) else '0';
38 SL_A <= '1' when (state=LOAD) else '0';
39 EN_B <= '1' when (state=LOAD or state=SHIFT) else '0';
40 SL_B <= '1' when (state=LOAD or state=SHIFT) else '0';
41 EN_H <= '1' when (state=ADD or state=SHIFT) else '0';
42 SL_H <= '1' when (state=SHIFT) else '0';
43 EN_L <= '1' when (state=SHIFT) else '0';
44 SL_L <= '1' when (state=SHIFT) else '0';
45 EN_C <= '1' when (state=ADD) else '0';
46 SL_C <= '0';
47 end RTL;
```

Πολλαπλασιαστής (Παραδοτέα)

VHDL:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity Multiplier_SI is
6     generic (n: integer := 4);
7     port (Rst, CLK, SI : in std_logic;
8           Low , High : out std_logic_vector (n-1 downto 0);
9           A_OUT, B_OUT: out std_logic_vector (n-1 downto 0));
10 end Multiplier_SI;
11
12
13 architecture RTL of Multiplier_SI is
14    component Reg
15       generic (n: integer);
16       port (D_IN: in std_logic_vector (n-1 downto 0);
17             SI, CLK, RST, SLOAD, ENABLE: in std_logic;
18             SO: out std_logic;
19             D_OUT: out std_logic_vector (n-1 downto 0));
20    end component;
21    component Adder
22       generic (n: integer:=4);
23       port (A, B: in std_logic_vector (n-1 downto 0);
24             SUM: out std_logic_vector (n-1 downto 0);
25             COUT: out std_logic);
26    end component;
27    component CtrlLogic_SI
28       generic (n: integer:=4);
29       port (Rst, CLK : in std_logic;
30             SL_A, SL_B, SL_H, SL_L, SL_C : out std_logic;
31             EN_A, EN_B, EN_H, EN_L, EN_C : out std_logic);
32    end component;
33    signal SL_A, SL_B, SL_H, SL_L, SL_C, EN_A, EN_B, EN_H, EN_L, EN_C: std_logic;
34    signal SO_A, SO_H: std_logic;
```

```
35    signal A, B, SUM, H: std_logic_vector (n-1 downto 0);
36    signal C, COUT : std_logic_vector (0 downto 0);
37    begin
38    R_A: Reg generic map (n)
39             port map ((n-1 downto 0 => '0'), SI, CLK, RST, SL_A, EN_A, SO_A, A);
40    A_OUT <= A;
41    R_B: Reg generic map (n)
42             port map ((n-1 downto 0 => '0'), SO_A, CLK, RST, SL_B, EN_B, open, B);
43    B_OUT <= B;
44    R_C: Reg generic map (n => 1)
45             port map (COUT, '0', CLK, RST, SL_C, EN_C, open, C);
46    R_H: Reg generic map (n)
47             port map (SUM, C(0), CLK, RST, SL_H, EN_H, SO_H, H);
48    High <= H;
49    R_L: Reg generic map (n)
50             port map ((n-1 downto 0 => '0'), SO_H, CLK, RST, SL_L, EN_L, open, Low);
51    U_Add: Adder generic map (n)
52                  port map (H, ((n-1 downto 0 => B(0)) and A), SUM, COUT(0));
53    U_Ctl: CtrlLogic_SI GENERIC MAP (n)
54                 port map (RST, CLK, SL_A, SL_B, SL_H, SL_L, SL_C, EN_A, EN_B, EN_H, EN_L, EN_C);
55 END RTL;
```