M2 Master E3A Integration Circuits Systems

TD3: Realization of Sequential Operator

1. **IFTAKHER Mohammed Akib – No. 22211204**

2. **KIBIWOTT Albert Kiplagat - No. 22211612**

Instructor: Professor Hervé MATHIAS

Realization of Sequential Operators

Part 1: Coding the clock divider

Code:

```
1
2   -- open the standard libraries
3   library ieee;
4   use ieee.std_logic_1164.all ; -- définit le types std_logic std_logic_vector
5
6   -- the TOP level entity
7   entity clk_div is
8
9   -- defines the generic parameters (project constants)
10  generic (factor: integer := 25000000);  -- facteur de division d'horloge. Ici on
    obtiendra 1 hz à partir de 50 Mhz
11
12  -- Input/Outputs definition
13  port (  RESET   :   in std_logic;      -- 50 Mhz Clock
14          CLK_IN  :   in std_logic;      -- 50 Mhz Clock
15          CLK_OUT :  out std_logic    -- 1 Hz clock
16          ) ;
17
18  end entity;
19
20
21  architecture a1 of clk_div is
22  signal clkcnt : integer := factor;
23  signal clkout : std_logic := '0';
24
25  begin
26     process(RESET, CLK_IN)
27        begin
28           if (RESET = '0') then
29              clkcnt <= factor;
30              clkout <= '0';
31
32           elsif (rising_edge(CLK_IN)) then
33              if (clkcnt = 0) then
34                 clkcnt <=  factor - 1;
35                 clkout <= not clkout;
36              else
37              clkcnt <= clkcnt - 1;
38              end if;
39           end if;
40     CLK_OUT <= clkout;
41     end process;
42
43  end architecture a1;
```

Comment: With the input clock being 50 MHz, to generate a 1 Hz, we need 25,000,000 counts as calculated below:

$Scaling\ factor = \frac{f_{in}}{f_{out}} = \frac{50\ X\ 10^6}{1} = 50,000,000$; and since a clock signal is a square wave with a 50% duty cycle; we will have 25,000,000 active cycles (rising edges).

Code Analysis: The logic behind "clkcnt = factor - 1" is that the countdown counter goes from the superior limit which is (25,000,000 -1 = 24,999,999) to 0, making a total of 25,000,000 active counts. The logic behind "clkcnt = clkcnt - 1" is that this line is used to decrement the counter by one from the initialized value (factor) until it reaches 0, so that the output clkout is inverted.

Part 2: Clock divider whose factor is programmable

```
1
2   -- open the standard libraries
3   library ieee;
4   use ieee.std_logic_1164.all ; -- définit le types std_logic std_logic_vector
5   use ieee.numeric_std.all;
6
7   -- the TOP level entity
8   entity clk_div2 is
9
10
11  -- Input/Outputs definition
12  port (  RESET   :   in std_logic;       -- 50 Mhz Clock
13          CLK_IN  :   in std_logic;       -- 50 Mhz Clock
14          CLK_OUT :  out std_logic;       -- 1 Hz clock
15          factor : in std_logic_vector(3 downto 0)
16          ) ;
17
18  end
    entity;
19
20
21  architecture a1 of clk_div2 is
22  signal clkcnt : unsigned(3 downto 0):= unsigned(factor);
23  signal clkout : std_logic := '0';
24
25  begin
26     process(RESET, CLK_IN, FACTOR)
27        begin
28           if (RESET = '0') then
29              clkcnt <= unsigned(factor);
30
31           elsif (rising_edge(CLK_IN)) then
32              if (clkcnt = 0) then
33                 clkcnt <=  unsigned(factor);
34                 clkout <= '1';
35              elsif clkcnt = (unsigned(factor)/2) + 1 then
36                 clkout <= '0' ;
37                 clkcnt <= clkcnt - "0001";
38              else
39                 clkcnt <= clkcnt - "0001";
40
41              end if;
42           end if;
43     CLK_OUT <= clkout;
44     end process;
45
46  end architecture a1;
```

Comment: The above code does not work when the factor is equal to zero, as such, the code has been modified to make the clock divider work for this value as described below:

Part 3: When the factor is equal to zero

Code:

```
1   -- open the standard libraries
2   library ieee;
3   use ieee.std_logic_1164.all ; -- définit le types std_logic std_logic_vector
4   use ieee.numeric_std.all;
5
6   -- the TOP level entity
7   entity clk_div2 is
8
9   -- Input/Outputs definition
10  port (  RESET   :   in std_logic;       -- 50 Mhz Clock
11          CLK_IN  :   in std_logic;       -- 50 Mhz Clock
12          CLK_OUT :  out std_logic;       -- 1 Hz clock
13          factor : in std_logic_vector(3 downto 0)
14          ) ;
15
16  end
    entity;



17
18
19  architecture a1 of clk_div2 is
20  signal clkcnt : unsigned(3 downto 0):= unsigned(factor);
21  signal clkout : std_logic := '0';
22
23  begin
24     process(RESET, CLK_IN, FACTOR)
25        begin
26           if (RESET = '0') then
27              clkcnt <= unsigned(factor);
28
29           elsif ( unsigned(factor) = 0) then
30              clkout <= CLK_IN;
31
32           else
33              if (rising_edge(CLK_IN)) then
34
35                 if (clkcnt = 0) then
36                    clkcnt <=  unsigned(factor);
37                    clkout <= '1';
38                 elsif clkcnt = (unsigned(factor)/2) + 1 then
39                    clkout <= '0' ;
40                    clkcnt <= clkcnt - "0001";
41                 else
42                    clkcnt <= clkcnt - "0001";
43
44                 end if;
45
46              end if;
47           end if;
48     CLK_OUT <= clkout;
49     end process;
50
51  end architecture a1;
```

Comment: As seen from the code, the clkout will be equal to CLK_IN (same frequency) should the factor be equal to 0.

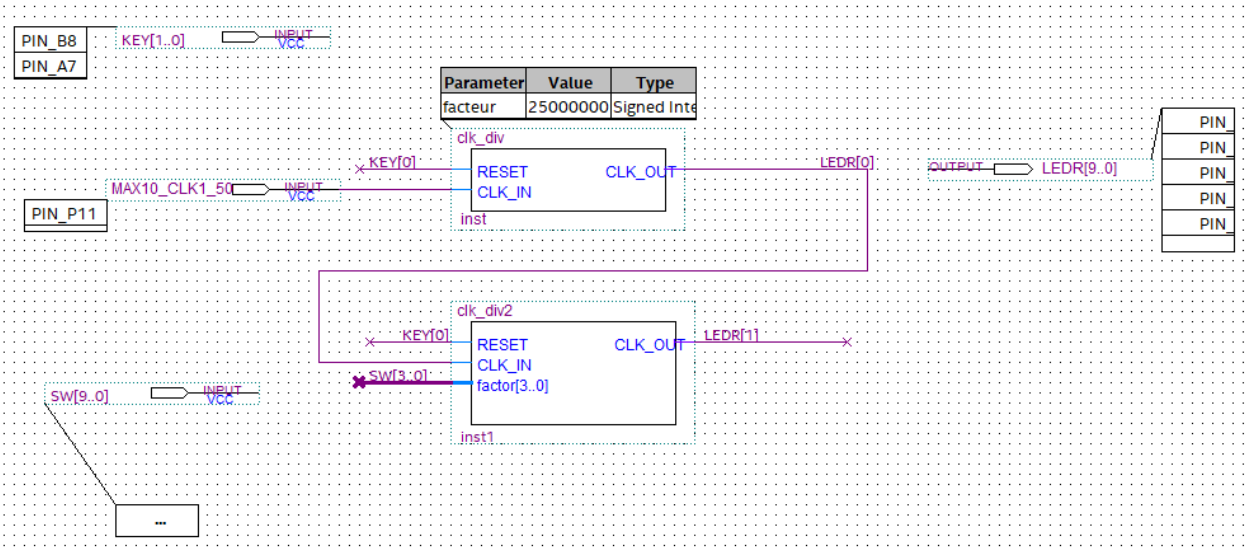

Figure: Circuit Diagram

Conclusion

In this practical session, we realized the functionality of the clock divider in two different ways, that is, with a generic parameter and a programmable parameter(4-bit input factor). Additionally, we implemented the code for the case when the factor is “0000”.