CS 288: Digital Stop Watch

Dheerendra Singh Rathor
120050033
dheeru.rathor14@gmail.com

March 11, 2014

Aim:

Using Xilinx ISE design and simulate a digital stop watch module to count in milliseconds.

- Use 1 KHz as primary clock in design.
- Module will have three inputs Start, Stop and Reset to control the stop watch.
- Use an 8-bit counter for milliseconds count.

Procedure:

For designing a stop watch - I used the following pointers:

- defined 3 input variable viz start, stop and reset and a clock variable clock and an 8 bit output variable named output.
- defined clock period as 1 ms
- defined a state machine with 3 states viz s0, s1 and s2
- initialized PS and NS as s0
- defined a clock process for the clock variable with at every rising edge PS is changing to NS and other state changes are simulated with the different input conditions.

States are defined as follows:

- s0: s0 is the initial state where counter is "00000000" and counter will not increase
- s1: s1 is the counting state. When the state is s1, counter will increase by 1 after every 1 ms
- s2: s2 is the stopped state. At this state counter will poses a value but will not increase until start is triggered

Present State	NS at start	NS at stop	NS at reset
s0	s1	s0	s0
s1	s1	s2	s0
s2	s1	s2	s0

Timing Diagrams

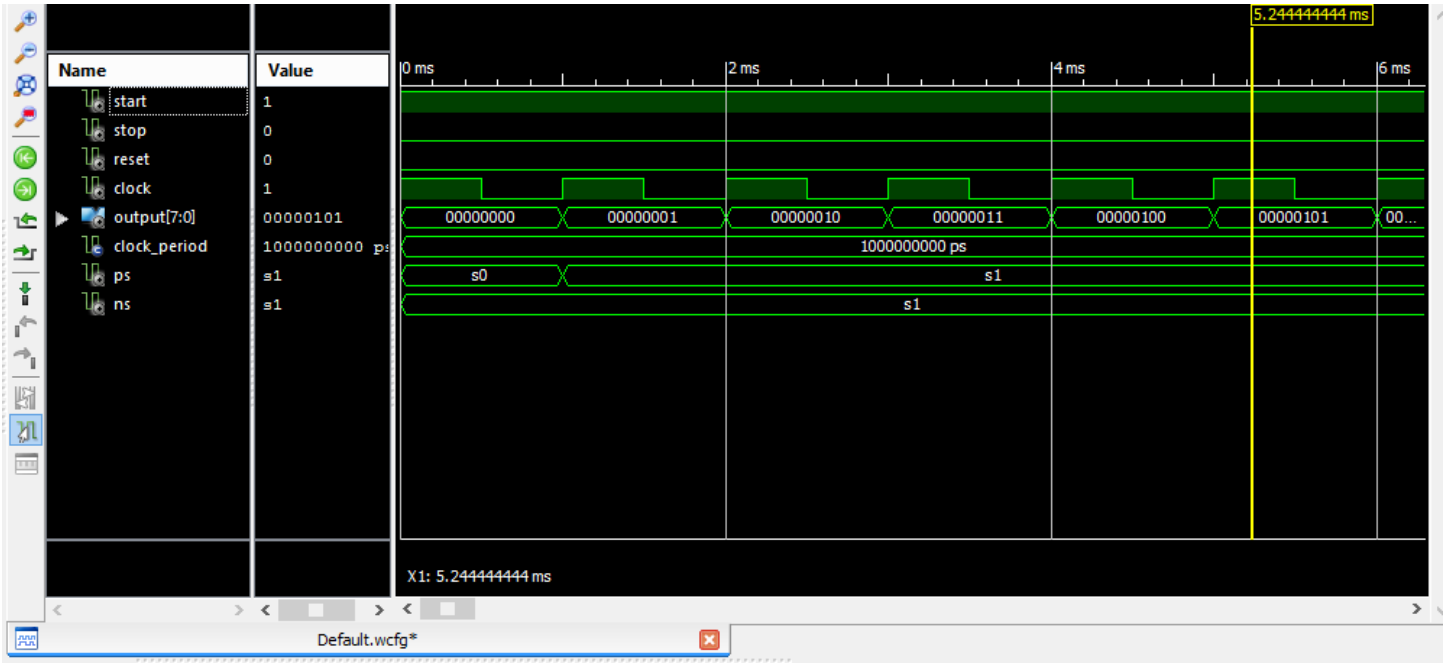

Figure 1: Timing diagram showing the increment in counter when the state of the machine is s1

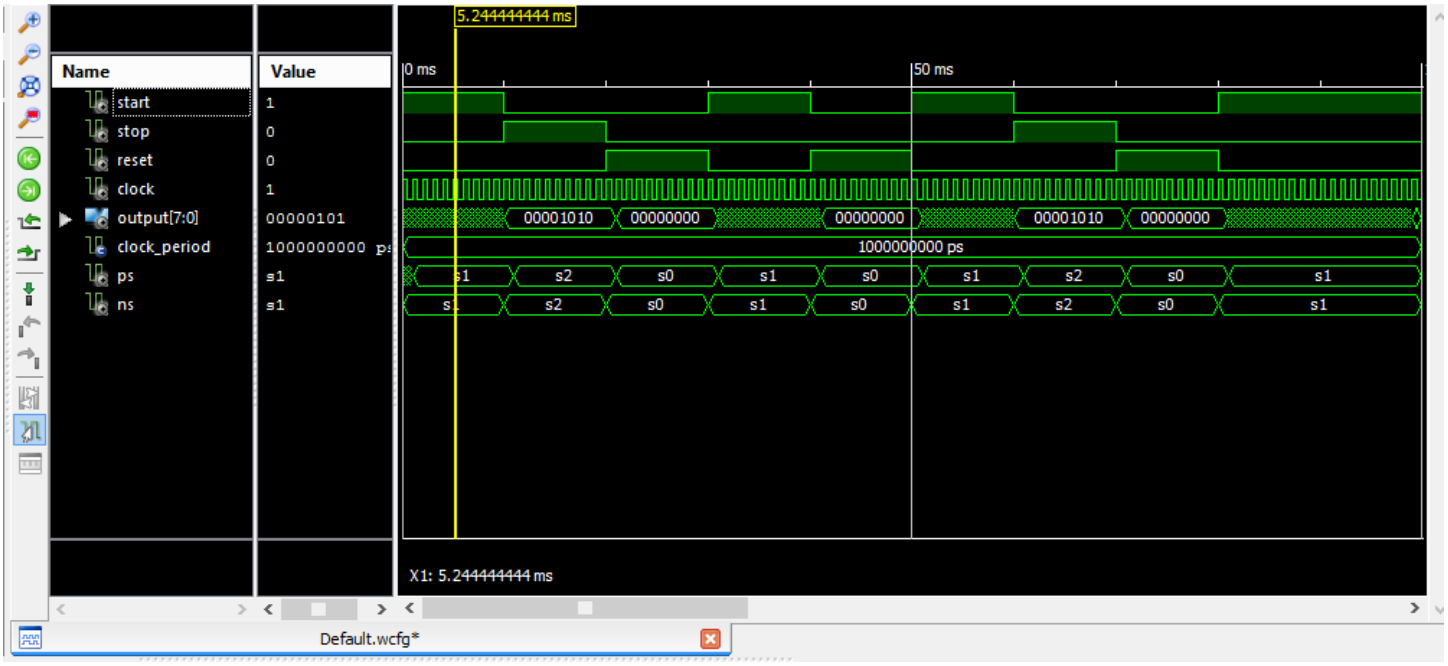

Figure 2: Timing diagram showing the changes in states with change in input

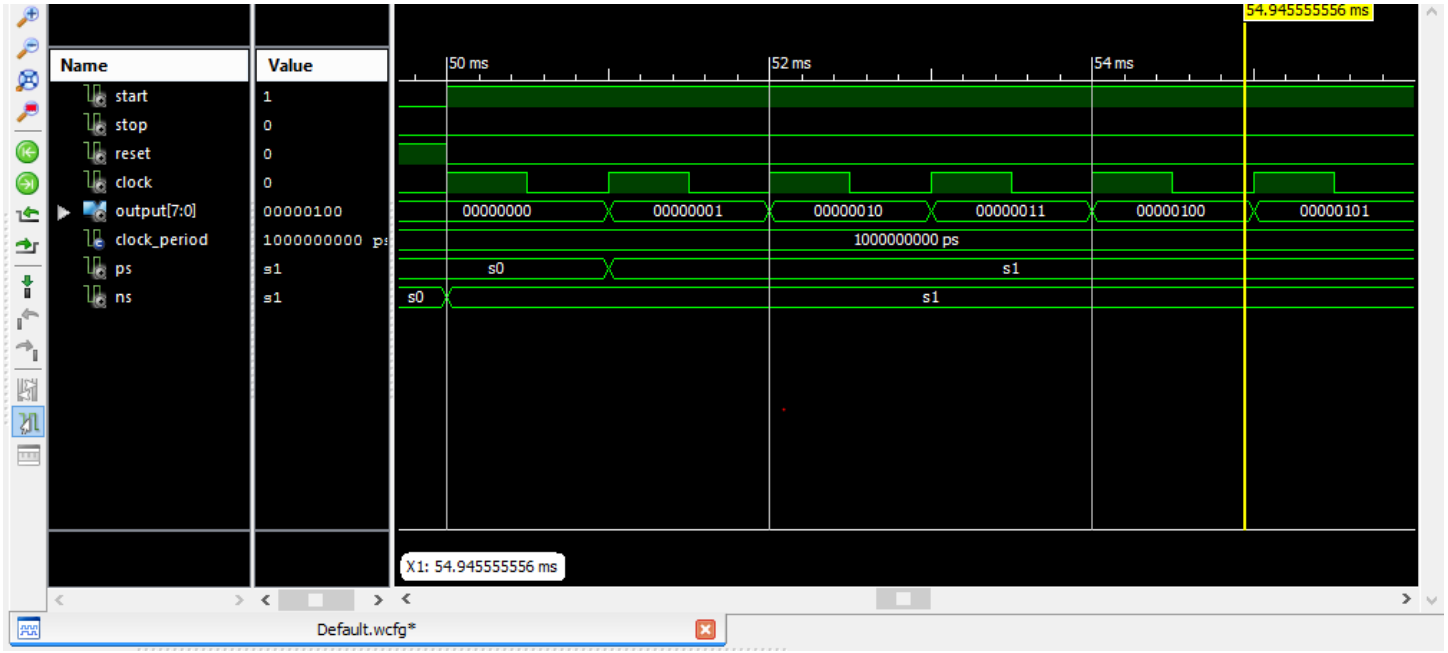

Figure 3: Timing diagram showing the change in output and change in states with the change in input

Codes

Here I have include working code for Arithmetic Logic Unit and 4-bit Adder.

1 8 bit milliseconds counter

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    15:02:32 03/11/2014
6  -- Design Name:
7  -- Module Name:    main - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use ieee.std_logic_unsigned.all;
23
24 -- Uncomment the following library declaration if using
25 -- arithmetic functions with Signed or Unsigned values
26 --use IEEE.NUMERIC_STD.ALL;
27
28 -- Uncomment the following library declaration if instantiating
29 -- any Xilinx primitives in this code.
30 --library UNISIM;
31 --use UNISIM.VComponents.all;
32
33 entity main is
34     Port ( start : in  STD_LOGIC;
35            stop : in  STD_LOGIC;
36            reset : in  STD_LOGIC;
37            clock : in  STD_LOGIC;
38            output : out  STD_LOGIC_VECTOR (7 downto 0));
39 end main;
```

```
40
41 architecture Behavioral of main is
42 constant clock_period : time := 1 ms;
43 --variable cnt : INTEGER := '0';
44 type state_type is (s0,s1,s2);
45 signal ps,ns : state_type:=s0;
46 signal temp : std_logic_vector (7 downto 0) := "00000000";
47 --begin process (start,stop,reset)
48 begin
49
50
51 SEQ:process(clock)
52 begin
53         if (rising_edge(clock)) then
54                 ps <= ns;
55                 case ps is
56                         when s0 =>
57                                 if (start = '1') then
58                                         temp <= temp + "00000001";
59                                         output <= temp;
60                                         ns <= s1;
61                                 end if;
62                                 if (stop = '1') then
63                                         output <= temp;
64                                         ns <= ps;
65                                 end if;
66                                 if (reset = '1') then
67                                         output <= temp;
68                                         ns <= ps;
69                                 end if;
70                         when s1 =>
71                                 if (start = '1') then
72                                         temp <= temp + "00000001";
73                                         output <= temp;
74                                         ns <= ps;
75                                 end if;
76                                 if (stop = '1') then
77                                         output <= temp;
78                                         ns <= s2;
79                                 end if;
80                                 if (reset = '1') then
81                                         temp <= "00000000";
82                                         output <= temp;
83                                         ns <= s0;
84                                 end if;
```

```
85                          when s2 =>
86                                  if (start = '1') then
87                                          temp <= temp + "00000001";
88                                          output <= temp;
89                                          ns <= s1;
90                                  end if;
91                                  if (stop = '1') then
92                                          output <= temp;
93                                          ns <= ps;
94                                  end if;
95                                  if (reset = '1') then
96                                          temp <= "00000000";
97                                          output <= temp;
98                                          ns <= s0;
99                                  end if;
100                         when others =>
101                                 null;
102                 end case;
103         end if;
104 end process;
105
106 end Behavioral;
```

Inference

In this assignment I inferred the following:

1. Working of the clock with different frequencies in VHDL
2. Learned about using and simulating state machines in VHDL
3. Learned using the library ieee.std_logic_unsigned.all
4. Representing State Diagrams in VHDL