Indian Institute of Technology Delhi

COL 215 - Digital Logic and System Design

Assignment 2

Stopwatch with four input commands

Aryan Sharma, Garv Nagori
2021CS10553, 2021CS10549

Contents

1 Introduction

The objective of this assignment is to display 4 digits, in the format (M:SS:T): minutes (M) on one LED display, seconds (SS) on two LED displays and tenth of second (T) on one LED display, the same is shown below as an example.

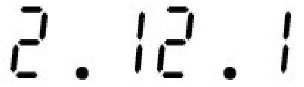


Figure 1: Display Format

2 Strategy

We are going to make a counter variable to account for the time passing with the help of the clock signal, **clk**. We will be using the program of Assignment 1 to display digits on the seven-segment display simultaneously We have used the following components, **Main, Switches, Counter, Clock, MUX, Decoder**. The last three were the same as Assignment 1 and are used solely to display all the 4 numbers simultaneously. The **new components** here are **Switches, Counter** which are described below.

2.1 Switches

This component takes as input the **clk signal , start, pause, continue, reset** from the ports in the BASYS3 board and the outputs of this component are two signals **enable_watch and enable_reset**, enable_watch and enable_reset are assigned values according to the below mentioned scheme:

(a) **start**: When start switch moves from 0 → 1, then enable_watch is set to '1'.
(b) **pause**: When pause switch moves from 0 → 1, then enable_watch is set to '0'.

(c) **continue**: When continue switch moves from 0 → 1, then enable_watch is set to '1' again.
(d) **reset**: On reset from 0 →1, reset_watch is set to '1' and the stopwatch counter is reset to 0.00.0 .

We actually only need 2 signals to have all the required functionality of our stopwatch, but as the requirement of the assignment is to have four input signals and make changes only on positive edges of these input signals. The above mentioned mapping maps these four signals to 2 signals(i.e. enable_watch and enable_reset).
To implement the above scheme in our program, we have maintained 4 previous variables one for each of the inputs, these are **prev_pause, prev_start, prev_continue, prev_reset**. At every rising edge of the signal clk(internal clock signal) we check whether any of the four signals differ from their previous value, if they do, we check whether this results in a positive edge or a negative edge, and if it is a **positive edge** then we change the values of the signals enable_watch and reset_watch, and then we also change the prev_signal to the current value of the signal.

We have used the following scheme to implement the above requirements:
(a) **Reset positive edge** : enable_watch is set to **'0'** and enable_reset to **'1'**.
(b)**Continue positive edge** : enable_watch is set to **'1'** and enable_reset to **'0'**.
(c)**Pause positive edge** : enable_watch is set to **'0'** and enable_reset to **'0'**.
(d)**Start positive edge** : enable_watch is set to **'1'** and enable_reset to **'0'**.

As a part of our design we never make signals of the type enable_watch = '1' and simultaneously reset_watch = '1'(its significance be noted in implementation of Counter), this ensures that when we turn the reset input from 0 →1, the stopwatch goes to 0.00.0 and stops(as in this case enable_watch is '0' and enable_reset is '1').
Another point to be noted is that continue and start behave exactly the same way, this is also because continue is always followed by pause(as specified in the assignments specification).

2.2 Counter

The inputs for this component are clk, enable, reset, and the outputs of this component are four std_logic_vector(3 downto 0), an0, an1, an2, an3 which represent each of the digits that are to be displayed on the seven-segment display Here enable and reset represent enable_watch and enable_reset, this leads to the following three cases, as this component will get inputs from the component Switches, :
(a) **enable = '1'** (this ensures that reset will be '0') : We continue our counting process and output the four vectors.
(b) **enable = '0' and reset = '1'** : We keep the counting process at hold and just output four vectors, where each of them is "0000".
(c) **enable = '0' and reset = '0'** : We stop the counting process and output the vectors in accordance with the current value of our counter.

We keep adding 1 to our counter variable, and use modulo operations to calculate the value of each digit. We have used the function **to_unsigned** to convert these into 4 bit vectors which are then assigned to the outputs.

2.3 Assignment 1 program

We have used programs from Assignment 1 in our implementation, which are used to display the the digits on the seven-segment display. As mentioned before these are the components Decoder, MUX, Clock. We have also changed the MUX component slightly to include the two dots after the minute and the second numbers.

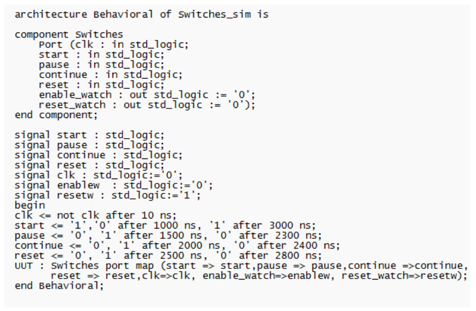
3 Main Component

We use the above three discussed components together to finally implement the timing circuit, In this we take as input 4 std_logic_vector(3 downto 0) namely an0,an1,an2,an3 and the clk signal(from the internal clock of the BASYS3 board). Two temporary signals are created to store the value of output of the component clock, these valiues are then passed into the MUX along with the vectors an0,an1,an2,an3, another temporary antemp and temp are also created to map the output signals of the component MUX. The signal temp is then mapped to the input of Decoder, the corresponding output is the 7 segment segjents being assigned corresponding signals, the anodes are assigned the values of antmep. This completes our timing circuit implementation. As a good representative measure we have also mapped the fours leds corresponding to the start, pause, continue and reset switches.

4 Simulations and Test Runs

4.1 'Switches_sim'

Simulation of the 'switches' component. Inputs given are start, pause, continue and reset.

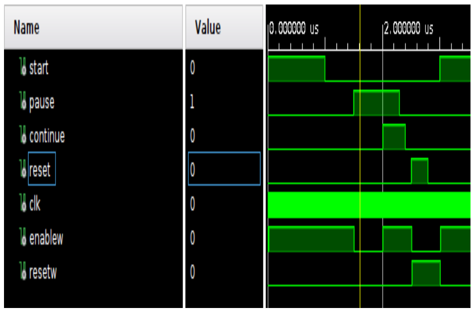


```
architecture Behavioral of Switches_sim is

component Switches
    Port (clk : in std_logic;
    start : in std_logic;
    pause : in std_logic;
    continue : in std_logic;
    reset : in std_logic;
    enable_watch : out std_logic := '0';
    reset_watch : out std_logic := '0');
end component;

signal start : std_logic;
signal pause : std_logic;
signal continue : std_logic;
signal reset : std_logic;
signal clk : std_logic:='0';
signal enablew  : std_logic:='0';
signal resetw : std_logic:='1';
begin
clk <= not clk after 10 ns;
start <= '1','0' after 1000 ns, '1' after 3000 ns;
pause <= '0', '1' after 1500 ns, '0' after 2300 ns;
continue <= '0', '1' after 2000 ns, '0' after 2400 ns;
reset <= '0', '1' after 2500 ns, '0' after 2800 ns;
UUT : Switches port map (start => start,pause => pause,continue =>continue,
      reset => reset,clk=>clk, enable_watch=>enablew, reset_watch=>resetw);
end Behavioral;
```

(a) Switches$_sim.vhd$

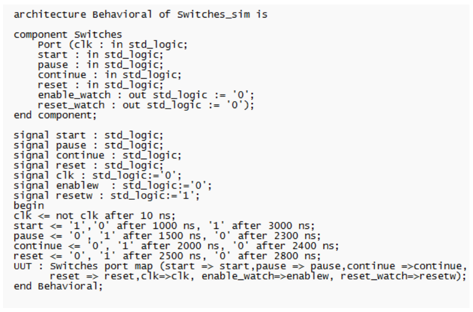


(b) Output Waveforms

Figure 2: Simulation of the Switches.vhd file to check it

4.2 'Countersim'

Simulation of the 'switches' component. Inputs given are enable_watch and reset_watch (which are controlled by Switches.vhd in the program).

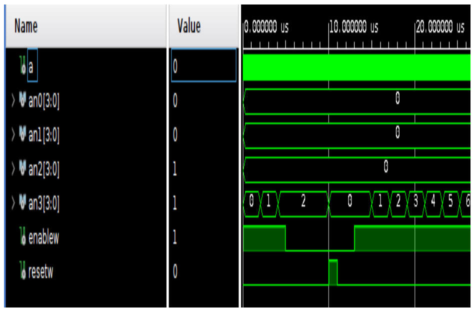


```
architecture Behavioral of Switches_sim is

component Switches
    Port (clk : in std_logic;
    start : in std_logic;
    pause : in std_logic;
    continue : in std_logic;
    reset : in std_logic;
    enable_watch : out std_logic := '0';
    reset_watch : out std_logic := '0');
end component;

signal start : std_logic;
signal pause : std_logic;
signal continue : std_logic;
signal reset : std_logic;
signal clk : std_logic:='0';
signal enablew  : std_logic:='0';
signal resetw : std_logic:='1';
begin
clk <= not clk after 10 ns;
start <= '1','0' after 1000 ns, '1' after 3000 ns;
pause <= '0', '1' after 1500 ns, '0' after 2300 ns;
continue <= '0', '1' after 2000 ns, '0' after 2400 ns;
reset <= '0', '1' after 2500 ns, '0' after 2800 ns;
UUT : Switches port map (start => start,pause => pause,continue =>continue,
      reset => reset,clk=>clk, enable_watch=>enablew, reset_watch=>resetw);
end Behavioral;
```

(a) countersim.vhd

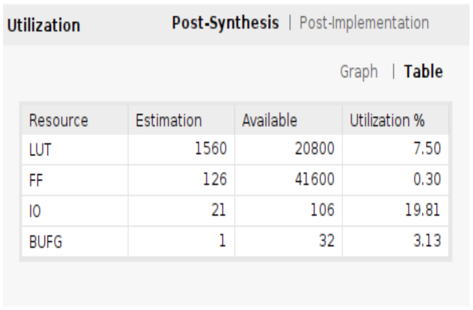


(b) Output Waveforms

Figure 3: Simulation of the Counter.vhd file to check it

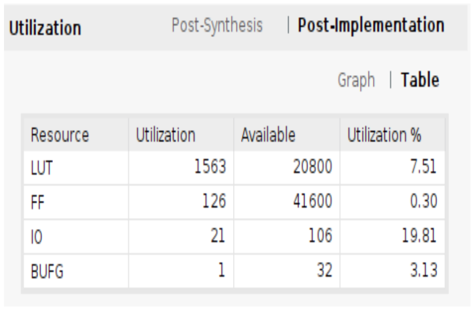
4.3 Synthesis and Implementation

After reviewing our two new components we ran synthesis and implementation of our code to verify it and then generated bitstream to upload on the FPGA BASYS3 board.


Utilization
Post-Synthesis | Post-Implementation
Graph | Table

Resource	Estimation	Available	Utilization %
LUT	1560	20800	7.50
FF	126	41600	0.30
IO	21	106	19.81
BUFG	1	32	3.13

(a) Post Synthesis Table


Utilization
Post-Synthesis | Post-Implementation
Graph | Table

Resource	Utilization	Available	Utilization %
LUT	1563	20800	7.51
FF	126	41600	0.30
IO	21	106	19.81
BUFG	1	32	3.13

(b) Post Implementation Table

Figure 4: Post Synthesis and post Implementation Utilization Tables

5 Conclusion

In this Assignment we used the previous Assignment to make a stopwatch with least count 0.1 seconds. The stopwatch can be user-controlled using four switches - start, pause, continue and reset. We had to write a program such that it only detects when a switch is turned from 'on' to 'off' and do nothing if it is turned 'off' again.

Google Drive link containing Block Diagram and a Video Demonstration of the Stopwatch on FPGA BASYS3 Board.