Experiment 4
Debouncing a switch

Amit Kumar — 16D070034

March 27, 2018

Overview

In this experiment, we implemented a Finite state machine to be used as a debouncing circuit to clean up the output of the push-button/switch as they take time to settle (typically a few milli-seconds) and during this settling period, the output may touch 0 and 1 several times which we wanted to prevent .
So, we implemented it in VHDL which was compiled on Quartus Prime, and simulated using ModelSim whichwas then uploaded to theKrypton v1.15M1270ZT144C5N CPLD-based board via svf file and urjtag and tested using a simple switch and led with oscilloscope.

1 Setup

First , We implemeted a counter to generate a clock of 100 Hz from 50MHz and then ysed that to implement debouncing FSM . All subparts are explained below :

1.1 Counter

Since CPLD has an inbuilt clock port of frequency 50MHz .So in order to use that clock for our deboucing circuit, I have made a counter to generate a lower frequency 100M Hz clock to check at intervals of 10ms. Following the code of counter to genarate $f/2$ if input is f :

```
-------------------------------------D Flip-flop for counter--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity D_FF is
  port (D,CLK,Reset: in std_logic; q: out std_logic);
end entity D_FF;

architecture WhatDoYouCare of D_FF is
begin

   process (CLK,Reset)
   begin
           if (Reset = '1') then
               q <= '0' ;
        elsif CLK'event and (CLK = '1') then
```

```
                q <= D;
          end if;
    end process;

end WhatDoYouCare;
---------------------------------------------Counter f/2-----------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter is
port (CLK, reset: in std_logic; Qbar: out std_logic);
end entity counter;

architecture struct of counter is

component INVERTER is
        port (a: in std_logic; b : out std_logic);
end component;

component D_FF is
  port (D, CLK,Reset: in std_logic; q: out std_logic);
end component D_FF;
signal Q,Q_bar : std_logic ;

begin
         dff1 : D_FF port map (D => Q_bar,CLK => CLK,Reset => reset,q => Q) ;
        inv1 : INVERTER port map (Q,Q_bar) ;
        Qbar <= Q_bar ;

end architecture struct;
```

Further, I have cascades 19 $f/2$ counters to genarate clock of frequency $f/2^19$ for our purpose :

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

entity counter2 is
port (CLK, reset: in std_logic; Q: out std_logic);
end entity counter2;

architecture struct of counter2 is

component counter is
port (CLK, reset: in std_logic; Qbar: out std_logic);
end component counter;

signal f : std_logic_vector(18 downto 0) ;

begin
        C1: counter port map (CLK => CLK , reset => reset ,Qbar => f(0));
l1 : for i in 0 to 17 generate
```

```
begin
        C: counter port map (CLK => f(i) , reset => reset ,Qbar => f(i+1));
end generate l1;
    C2: counter port map (CLK => f(18) , reset => reset ,Qbar => Q);
end architecture struct;
```

1.2 Debouncing FSM

Using this 100Hz clock, I have designed a debouncing FSM which has a single input from the switch/push-button (and also a reset), and a single output which produces clean switching between 0 and 1 using two input gates, inverters and D flip-flops.

FSM Logic :

The switch output is checked every 10ms, and the output of the finite-state machine is 1 (respectively, 0) if the last two values that were checked are 1 (respectively, 0).

Using the above state representation, the encoding has been done as shown below.

State	q_0	q_1
S_0	0	0
S_1	0	1
S_2	1	0

Below is the transition table with input ’x’ and putput ’y’

x	q_0	q_1	nq_0	nq_1	**y**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	1

VHDL code of debouncing FSM :

```
library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

package EE224_Components is
   component INVERTER is
        port (a: in std_logic; b : out std_logic);
   end component;
   component AND_2 is
        port (a, b: in std_logic; c : out std_logic);
   end component;
   component OR_2 is
        port (a, b: in std_logic; c : out std_logic);
   end component;
   component NAND_2 is
        port (a, b: in std_logic; c : out std_logic);
   end component;
```

```vhdl
   component XOR_2 is
        port (a, b: in std_logic; c : out std_logic);
   end component;
end EE224_Components;
---------------------D flip-flip used in mealy FSM of debouncing---------------------------
library ieee;
use ieee.std_logic_1164.all;

entity myDFF is
  port (D, CLK: in std_logic; Q: out std_logic);
end entity myDFF;

architecture WhatDoYouCare of myDFF is
begin

   process (CLK)
   begin
        if CLK'event and (CLK = '1') then
             Q <= D;
        end if;
   end process;

end WhatDoYouCare;
-------------------------2 input Logic Gates----------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
entity INVERTER is
  port (a: in std_logic;
         b: out std_logic);
end entity INVERTER;
architecture Behave of INVERTER is
begin
  b <= not a;
end Behave;

library ieee;
use ieee.std_logic_1164.all;
entity AND_2 is
  port (a, b: in std_logic;
         c: out std_logic);
end entity AND_2;
architecture Behave of AND_2 is
begin
  c <= a and b;
end Behave;

library ieee;
use ieee.std_logic_1164.all;
entity OR_2 is
  port (a, b: in std_logic;
          c: out std_logic);
end entity OR_2;
architecture Behave of OR_2 is
begin
  c <= a or b;
end Behave;

library ieee;
use ieee.std_logic_1164.all;
entity NAND_2 is
  port (a, b: in std_logic;
          c: out std_logic);
end entity NAND_2;
architecture Behave of NAND_2 is
begin
  c <= not (a and b);
end Behave;

library ieee;
use ieee.std_logic_1164.all;
entity XOR_2 is
  port (a, b: in std_logic;
          c: out std_logic);
end entity XOR_2;
architecture Behave of XOR_2 is
begin
  c <= (a xor b);
end Behave;

--------------------------Debouncing FSM-----------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity debounce is
port (x: in std_logic ; W: out std_logic;
CLK,reset: in std_logic);
end entity debounce;

architecture struct of debounce is

component INVERTER is
        port (a: in std_logic; b : out std_logic);
end component;

component AND_2 is
        port (a, b: in std_logic; c : out std_logic);
end component;

component OR_2 is
port (a, b: in std_logic; c : out std_logic);
```

```vhdl
end component;

component myDFF is
  port (D, CLK: in std_logic; Q: out std_logic);
end component myDFF;

signal nq0,nq1,q0,q1 ,x_bar , q0_bar, q1_bar ,r_bar ,s0,s1,s2 ,nq0a,nq0b,rnq0 ,nq1a,nq1b,rn

begin
      inv1 : INVERTER port map (q0,q0_bar) ;
      inv2 : INVERTER port map (q1,q1_bar) ;
      inv3 : INVERTER port map (x,x_bar) ;
      inv4 : INVERTER port map (reset,r_bar);

      s_0 : AND_2 port map (q0_bar,q1_bar,s0); ----------State Encoding
      s_1 : AND_2 port map (q0_bar,q1,s1);
      s_2 : AND_2 port map (q0,q1_bar,s2);
    -----------------------------------------------------------------------------
      nq0_a : AND_2 port map (s1,x,nq0a) ;
      nq0_b : AND_2 port map (s2,x,nq0b) ;

      r_nq0 : OR_2 port map (nq0a,nq0b,rnq0) ;

      nq_0 : AND_2 port map (rnq0,r_bar,nq0) ;
---------------------------------------------------------------------------------
      nq1_a : AND_2 port map (s0,x,nq1a) ;
      nq1_b : AND_2 port map (s2,x_bar,nq1b) ;

      r_nq1 : OR_2 port map (nq1a,nq1b,rnq1) ;

      nq_1  : AND_2 port map (rnq1,r_bar,nq1) ;
-------------------------------------------------------------------------------

     W_r : OR_2 port map (nq0a,s2,Wr) ;
      W1 : AND_2 port map (Wr,r_bar,W) ;

   dff2 : myDFF port map (nq1,CLK,q1);

   dff1 : myDFF port map (nq0,CLK,q0);

end architecture struct;
```

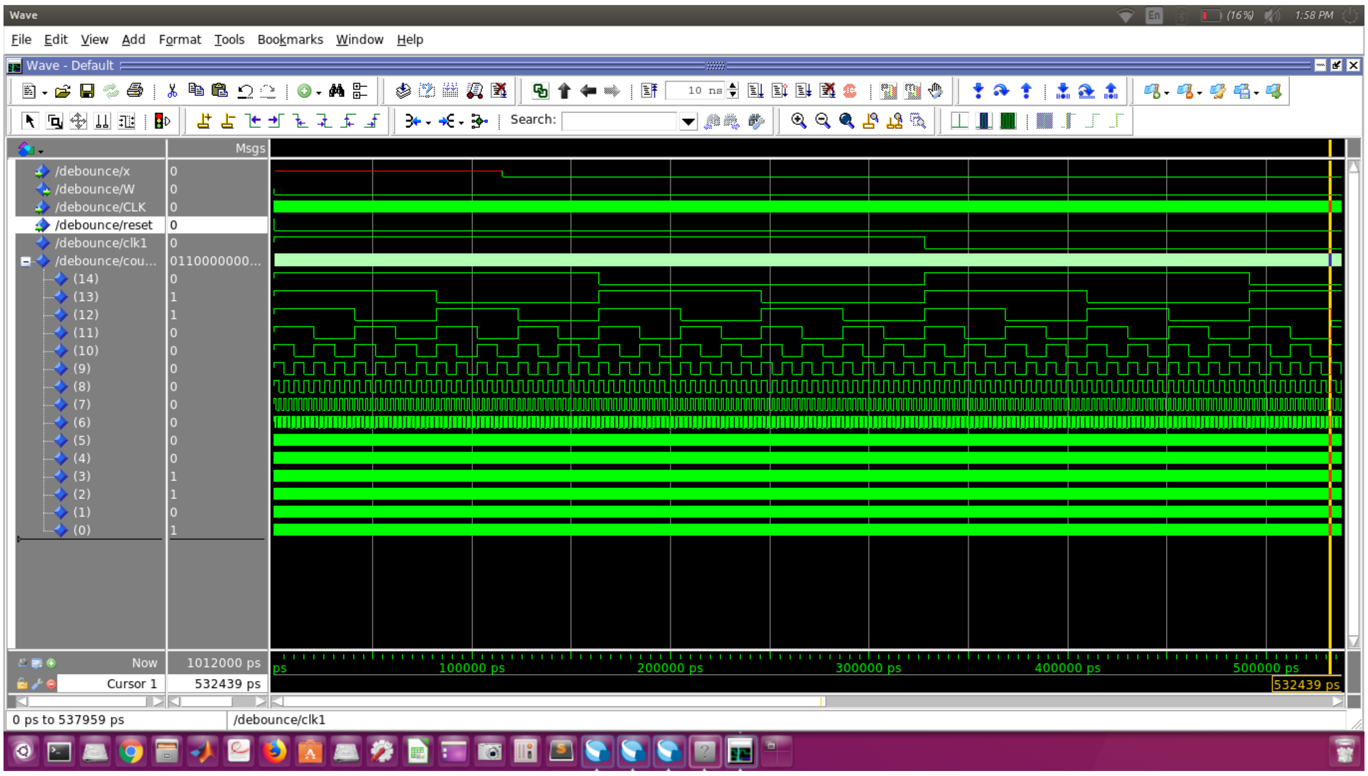
Final Integration of our counter and Debouncing FSM :

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

entity DUT is
port (x: in std_logic ; W: out std_logic;
CLK,reset: in std_logic);
end entity DUT;
```

```

architecture struct of DUT is

component counter2 is
port (CLK, reset: in std_logic; Q: out std_logic);
end component counter2;

component debounce is
port (x: in std_logic ; W: out std_logic;
CLK,reset: in std_logic);
end component debounce;

signal Q: std_logic ;

begin

count: counter2 port map (CLK => CLK ,reset => reset , Q => Q  );

debou: debounce port map (x,W,Q,reset);

end architecture struct;
```

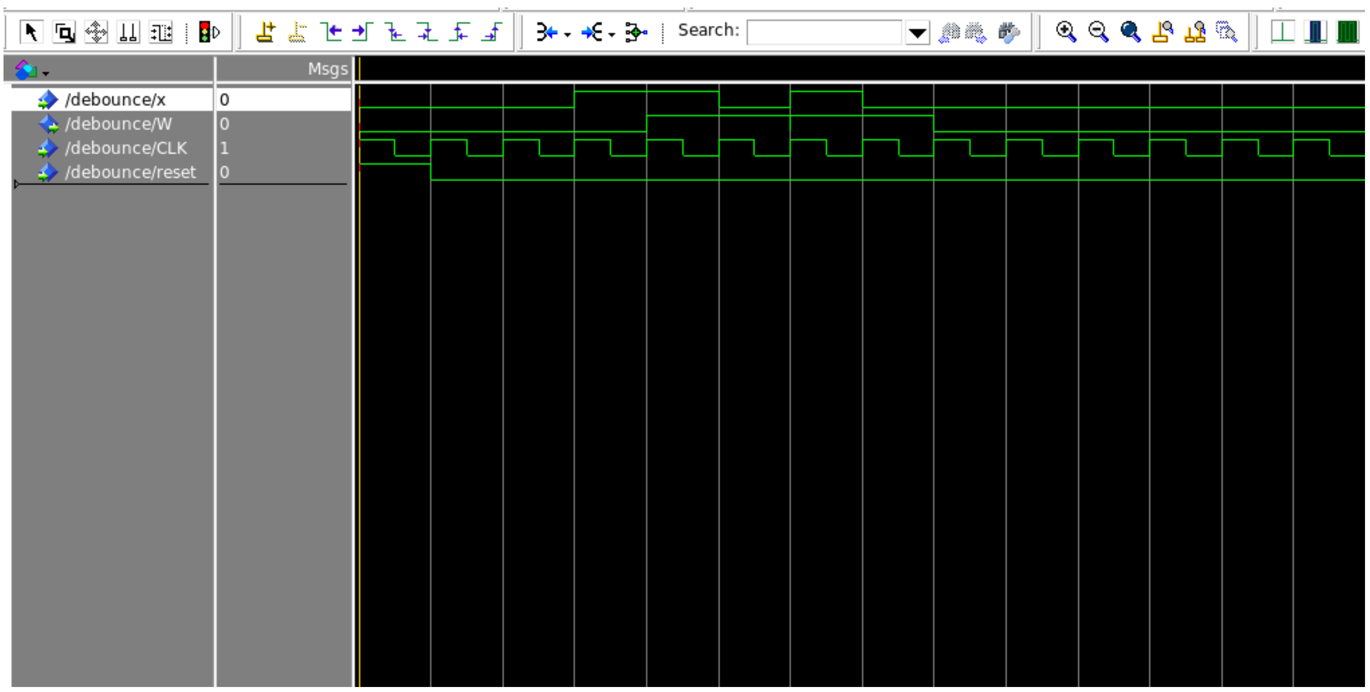
2 Observations

After implementing the design in code, the next major part is to simulate and test the code for a set of inputs. RTL simulation was performed on the machine to validate the implementation.



```
                      Msgs
/debounce/x           0
/debounce/W           0
/debounce/CLK         0
/debounce/reset       0
/debounce/clk1        0
/debounce/cou...      0110000000...
  (14)                0
  (13)                1
  (12)                1
  (11)                0
  (10)                0
  (9)                 0
  (8)                 0
  (7)                 0
  (6)                 0
  (5)                 0
  (4)                 0
  (3)                 1
  (2)                 1
  (1)                 0
  (0)                 1

Now                   1012000 ps
Cursor 1              532439 ps

0 ps to 537959 ps     /debounce/clk1
```

Figure 1: Working of counter for generating 100 Hz



	Msgs
/debounce/x	0
/debounce/W	0
/debounce/CLK	1
/debounce/reset	0

Figure 2: Simulation of debouncer for a particular test case

3 DSO Testing

We have tested the logic using the RTL simulations,then done pin planning and burned svf file of code on the Krypton board using urjtag . Next, we need to check that the code is actually running as it is expected to, on the board.Hence we connected an additional LED and switch and observed on oscilloscope.

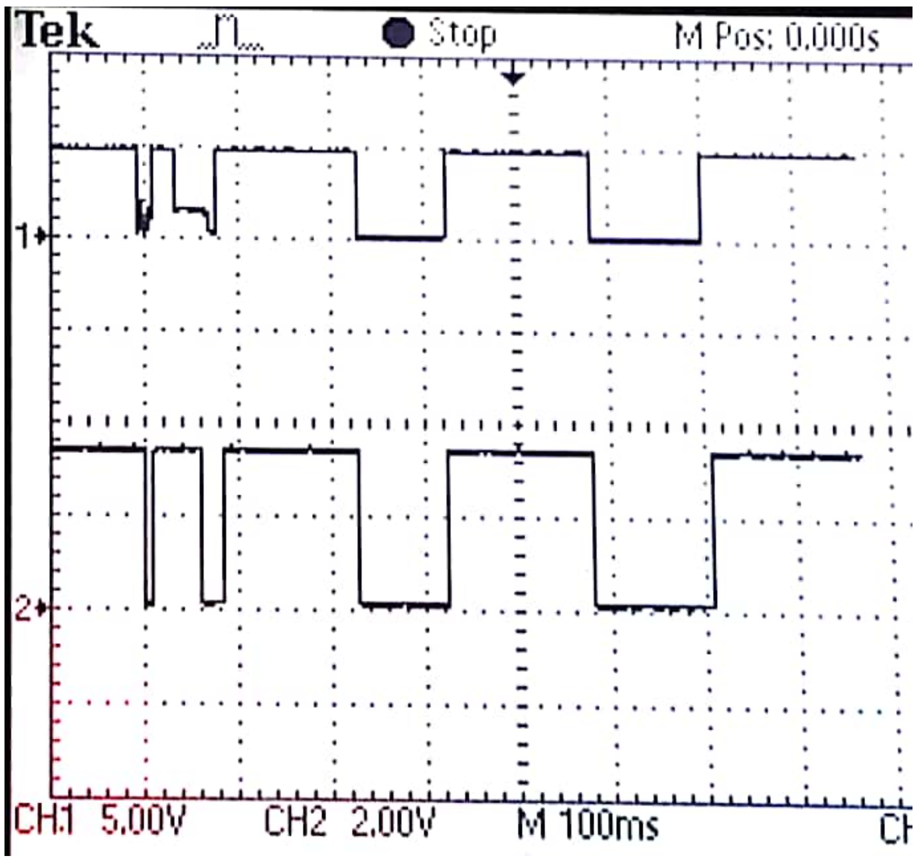


Figure 3: Debounced output

P.S: In above figure , Initial is not a bounce , it appears as time scale is 100 ms and input was 0 for past 20ms .