ECEN302 Lab 5 - IP Catalog

Joshua Benfell - 300433229

September 19, 2020

1 Introduction

The main objective of this lab was to become familiar with the IP catalog and implement a counter utilising a pre-made component. Using a pre-defined seven segment display decoder in conjunction with a clock defined by the vivado clocking wizard from the IP catalog, a 2 digit BCD counter was implemented on an FPGA. Doing this with the IP catalog demonstrates how we as FPGA designers don't have to reinvent the wheel and can speed up development time.

2 1 Hz Clock

To make the 1Hz clock the IP Catalog was first used to select a pre-made step down clock. This clock was configured such that it took a 100MHz clock as an input and outputted a 5MHz clock. This then had to be stepped down to a 1Hz clock which was done with a 2.5 million counter to provide 1 second at a 50% duty cycle.

The ports enabled on the auto-generated clock module were the clk_in and clk_out pins, and the reset and locked pins. The reset pin was used to restart the 5MHz cycle and would turn off the locked led which proves the locked ports function of identifying if the clock is running.

Additional functionality that was added to this is an enable switch. I had issues implementing this as I initially had the enable pin wrapping around the clk functionality (the conversion to a 1Hz clock.). This turned out to cause clocking issues as it was tying the clocks function to an asynchronous input. The solution to this was to make the clock the main wrapper in the process and inside that clocked function, if the enable bit was set, then have the clock out bit set to the one hz clock.

3 Two Digit Counter

The next segment of code added was a two digit counter. This was also done with the IP catalog as this greatly reduced the time to develop this program. This was initially done with each counter having the pins CLK, SCLR, THRESH0 and Q. CLK was the input 1Hz clock, SCLR was the synchronous clear, which at this point I couldn't get working as it required the clock to clear and the clock was being halted on the reset. The THRESH0 pin was configured to output high when the counter reach 9. By tying this as the clock input to the second counter, it caused the count to go 08, 19, 10, which is clearly wrong. This was solved by adding the concurrent line 'flipOver¡= not threshold ;' as the second counters clock is forced to wait a cycle of the 1 hz clock.

As this didn't work with the reset due to the synchronous clear requiring a clock and not receiving one, I attempted to implement it without IP blocks and instead within the processes. This worked as a normal counter, but I couldn't get the reset function working and the counter would go 99, A0, 00, which is also not correct, so I revisited the IP Block version.

Retrying with IP blocks, I added the Clock enable pin and set up the output of counter 1s threshold bit to the CE pin. This was set up so that the SCLR function would take priority over the CE function. From here, both counters were tied to the 1Hz clock and the compromise of not resetting the clock had to be made as I prioritised resetting the counter and could not figure out how to do both. This worked as a functional counter with the reset ability and didn't count weird. However, not everything was able to be reset.

It was also found that the keyword open could be used to indicate to vivado that no connection was on a port which saved me creating a dummy signal for the second counters threshold bit. I was also able to remove the flipover line as this was no longer necesary due to the clock enable pin allowing the next clock cycle.

4 Code

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date: 12.09.2019 13:30:19
6  -- Design Name:
7  -- Module Name: one_second_clock_behaviour - Behavioural
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
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use IEEE.numeric_std.ALL;
25 -- Uncomment the following library declaration if using
26 -- arithmetic functions with Signed or Unsigned values
27 --use IEEE.NUMERIC_STD.ALL;
28
29 -- Uncomment the following library declaration if instantiating
30 -- any Xilinx leaf cells in this code.
31 --library UNISIM;
32 --use UNISIM.VComponents.all;
33
34 entity one_second_clock_behaviour is
35     Port ( mclk : in STD_LOGIC; -- Master Clock 100MHz
36            reset : in STD_LOGIC;  -- Reset Bit BTNU
37            enable : in STD_LOGIC;  -- Enable Input SW0
```

```
38              clk_out : out STD_LOGIC;  -- Output LED LD0
39              lock : out STD_LOGIC;  -- MMCM Lock Output SW15. High when enable
                    is low.
40              segmentOut: out STD_LOGIC_VECTOR(6 downto 0);
41              segmentEnable: out STD_LOGIC_VECTOR(7 downto 0));
42 end one_second_clock_behaviour;
43
44 architecture Behavioural of one_second_clock_behaviour is
45
46 component clk_5MHz
47 port
48  (-- Clock in ports
49   -- Clock out ports
50   clk_out1          : out    std_logic;
51   -- Status and control signals
52   reset             : in     std_logic;
53   locked            : out    std_logic;
54   clk_in1           : in     std_logic
55  );
56 end component;
57
58 COMPONENT c_counter_binary_0
59   PORT (
60     CLK : IN STD_LOGIC;
61     CE : IN STD_LOGIC;
62     SCLR : IN STD_LOGIC;
63     THRESH0 : OUT STD_LOGIC;
64     Q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
65   );
66 END COMPONENT;
67
68 component ssd_decoder
69 port
70  ( clk : in STD_LOGIC;
71    in_0 : in STD_LOGIC_VECTOR (3 downto 0);
72    in_1 : in STD_LOGIC_VECTOR(3 downto 0);
73    Seg_Out : out STD_LOGIC_VECTOR (6 downto 0);
74    Seg_enable : out STD_LOGIC_VECTOR(7 downto 0)
75  );
76 end component;
77
78
79 signal counter : integer := 0;
80 signal clk_1Hz : STD_Logic := '0';
81 signal clk5_out: STD_LOGIC;
82 signal clk500: std_logic;
83 signal threshold :std_LOGIC;
84
85 signal num0 : STD_LOGIC_VECTOR (3 downto 0) := "0000";
86 signal num1 : STD_LOGIC_VECTOR (3 downto 0) := "0000";
87
88
89 signal segOut : STD_LOGIC_VECTOR(6 downto 0);
90 signal segEn : STD_LOGIC_VECTOR(7 downto 0);
91
92 signal flipOver : std_logic := '0';
93
94
95 begin
```

```
96
97  clk5 : clk_5MHz
98     port map (
99        -- Clock out ports
100        clk_out1 => clk5_out,
101       -- Status and control signals
102        reset => '0',
103        locked => lock,
104        -- Clock in ports
105        clk_in1 => mclk
106     );
107
108  ssd : ssd_decoder
109     port map (
110          clk => clk500,
111          in_0 => num0,
112          in_1 => num1,
113          Seg_Out => segmentOut,
114          Seg_enable => segmentEnable
115     );
116
117  Counter0 : c_counter_binary_0
118    PORT MAP (
119      CLK => clk_1Hz,
120      CE => '1',
121      SCLR => reset,
122      THRESH0 => threshold,
123      Q => num0
124    );
125
126  Counter1 : c_counter_binary_0
127     PORT MAP (
128          CLK => clk_1Hz,
129          CE => threshold,
130          SCLR => reset,
131          THRESH0 => open,
132          Q => num1
133     );
134
135
136  clock_divider: process (clk5_out)
137  begin
138     if (clk5_out'event and clk5_out='1') then  -- Count to 5 million
139          counter <= counter + 1;
140          if (counter = 2500000) then  -- Once we hit 5 million toggle the 1
                Hz clock
141               if (enable = '1') then
142                    clk_1Hz <= not clk_1Hz;
143               end if;
144               counter <= 0;
145          end if;
146          if(counter mod 500 = 0) then
147               clk500 <= not clk500;
148          end if;
149
150          if (enable = '1') then
151               clk_out <= clk_1Hz;  -- Set LD0 to the state of the 1Hz Clock
152          end if;
153
```

```
154
155     end if;
156 end process;
157
158 flipOver <= not threshold;  --  Makes the switch happen on the falling edge
159
160 end Behavioural;
```

code/one_second_clock_behaviour.vhd