EDA234 Lab 1: Lab Tutorial

Weihan Gao weihanga@chalmers.se

November 8, 2022

1 First step

We implemented the second counter in three modules – Cntsecond_forBolt.vhdl, digital_show.vhdl and top.vhdl. Cntsecond module counts the 1Hz clk_1s and generates the 2 numbers – cnt_L and cnt_H, ranging from 0 to 9 and 0 to 5. They will be shown on two digital segments through the digital_show module at nearly 200Hz. Top module contains these two modules and has a process to divide the system clk frequency(100MHz) into the two clks we want.

The partner in the lab is Yuxiang Cao.

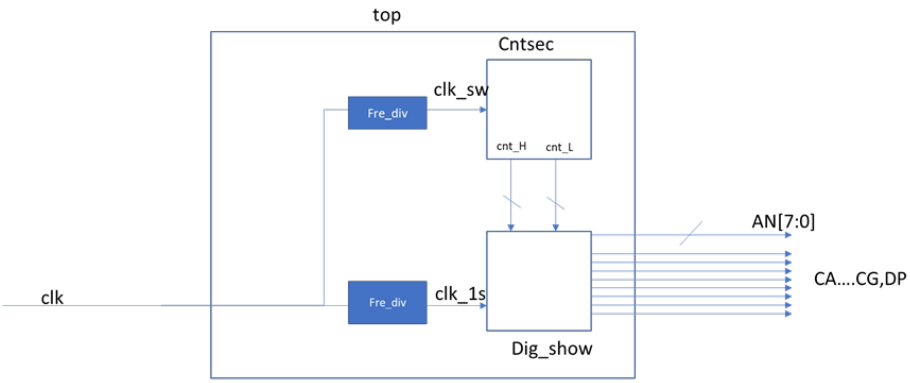

Figure 1: Simple block diagram.

2 Second step

The registers we used are about 70 after initial implementation. And we first thought to reduce the use of sequential logic. So we put the case(for encoding) out of the rising_edge in digital_show.vhdl and the number of regs is down to 63.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	61	0	63400	0.10
LUT as Logic	61	0	63400	0.10
LUT as Memory	0	0	19000	0.00
Slice Registers	70	0	126800	0.06
Register as Flip Flop	70	0	126800	0.06
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Figure 2: Initial number of regs.

```
elsif rising_edge(clk_sw) then
 sel<='1';
  if(sel='1') then
    AN_TURN_ON<="11111101";
    show_number <= '0'&cnt_H;
  elsif(sel='0') then
    AN_TURN_ON<="11111110";
    show_number <= cnt_L;
  end if;
  sel <= not(sel);
  case show_number is
    when "0000" =>
      digital <= "1000000";
    when "0001" =>
      digital <= "1111001";
    when "0010" =>
      digital <= "0100100";
    when "0011" =>
      digital <= "0110000";
    when "0100" =>
      digital <= "0011001";
    when "0101" =>
      digital <= "0010010";
    when "0110" =>
      digital <= "0000010";
    when "0111" =>
      digital <= "1111000";
    when "1000" =>
      digital <= "0000000";
    when "1001" =>
      digital <= "0010000";

    when others =>
      digital <="1111111";
  end case;
end if;
```

```
elsif rising_edge(clk_sw) then
 sel<='1';
  if(sel='1') then
    AN_TURN_ON<="11111101";
    show_number <= '0'&cnt_H;
  elsif(sel='0') then
    AN_TURN_ON<="11111110";
    show_number <= cnt_L;
  end if;
  sel <= not(sel);
end if;
case show_number is
    when "0000" =>
      digital <= "1000000";
    when "0001" =>
      digital <= "1111001";
    when "0010" =>
      digital <= "0100100";
    when "0011" =>
      digital <= "0110000";
    when "0100" =>
      digital <= "0011001";
    when "0101" =>
      digital <= "0010010";
    when "0110" =>
      digital <= "0000010";
    when "0111" =>
      digital <= "1111000";
    when "1000" =>
      digital <= "0000000";
    when "1001" =>
      digital <= "0010000";

    when others =>
    digital <="1111111";
  end case;
end process L;
```

Figure 3: Put the case out of rising_edge(clk_sw).

```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |   61 |     0 |     63400 |  0.10 |
|   LUT as Logic          |   61 |     0 |     63400 |  0.10 |
|   LUT as Memory         |    0 |     0 |     19000 |  0.00 |
| Slice Registers         |   63 |     0 |    126800 |  0.05 |
|   Register as Flip Flop |   63 |     0 |    126800 |  0.05 |
|   Register as Latch     |    0 |     0 |    126800 |  0.00 |
| F7 Muxes                |    0 |     0 |     31700 |  0.00 |
| F8 Muxes                |    0 |     0 |     15850 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

Figure 4: Number of regs after first modification.

Then we found the biggest part of regs is those in LUT, which means two counters in fre_div at top are over the limit. We could use only one counter to reach the requirement by containing the smaller counter in a bigger one. Finally, we used 43 regs and no latch.

```
signal count_1s : std_logic_vector(27 downto 0);
signal count_sw : std_logic_vector(19 downto 0);
```

Figure 5: Two counters for freq_div.

```
proc_clk: process (clk,rstn)
begin  -- process proc_clk
  if rstn='0' then
    clk_1s <= '0';
    count_1s <= x"0000000";
    --clk_sw <= '0';
  elsif rising_edge(clk) then
    if count_1s=x"2faf07f" then
    --if count_1s=x"00000100" then
      count_1s <= x"0000000";
      clk_1s <= not(clk_1s);
    else
      count_1s <= conv_std_logic_vector((unsigned(count_1s)+1),28);

    end if;

    -- if ("000000000000000000" = count_1s(17 downto 0)) then
    -- --if ("000" = count_1s(2 downto 0)) then
    --   clk_sw <= not(clk_sw);
    -- else
    --   --clk_sw <= clk_sw;
    -- end if;

  end if;
end process proc_clk;

proc_clk_sw: process (clk,rstn)
 begin  -- process proc_clk
   if rstn='0' then
     count_sw <= x"00000";
     clk_sw <= '0';
   elsif rising_edge(clk) then
     if count_sw=x"7a120" then
     --if count_1s=x"00000010" then
       count_sw <= x"00000";
       clk_sw <= not(clk_sw);
     else
       count_sw <= conv_std_logic_vector((unsigned(count_sw)+1),20);
     end if;

   end if;
 end process proc_clk_sw;
```

Figure 6: Delete the clk_sw proc and put them in the clk_1s proc. When the number is 2$\hat{1}$8, the clk_sw flips.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	52	0	63400	0.08
LUT as Logic	52	0	63400	0.08
LUT as Memory	0	0	19000	0.00
Slice Registers	43	0	126800	0.03
Register as Flip Flop	43	0	126800	0.03
Register as Latch	0	0	126800	0.00
F7 Muxes	0	0	31700	0.00
F8 Muxes	0	0	15850	0.00

Figure 7: Final regs we used.