David Elliott

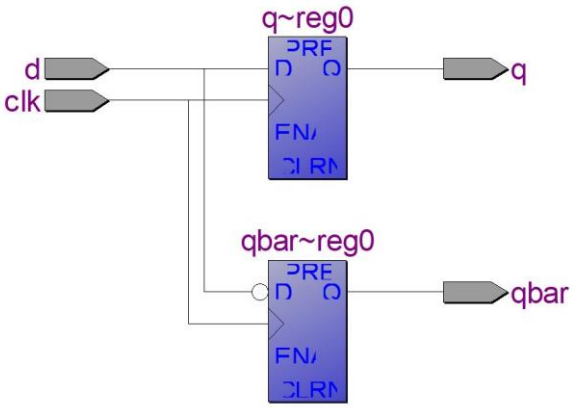
Evan Yeh

EE125 Homework #3

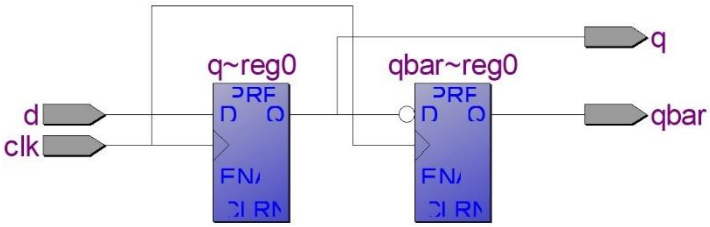
Exercise 1:

b.) arch1, arch2, arch3 are compiled and RTL views are shown below:

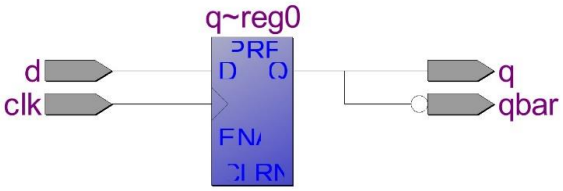
arch1:



arch2:



arch3:

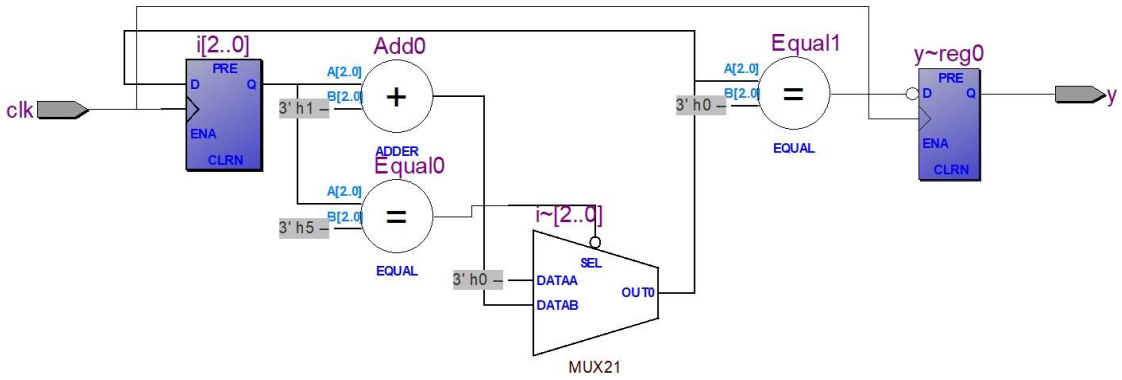


All circuits match with our predicted ones.

Exercise 2:

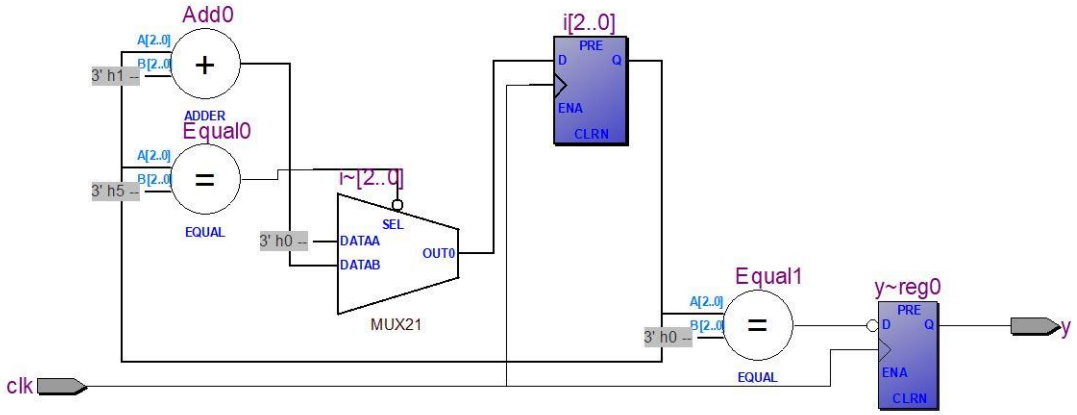
d.) RTL View:

Code1:



There are a total of 4 flip-flops in the generated view, same as our prediction.

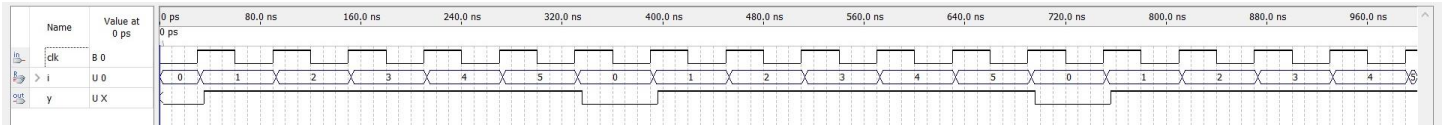
Code2:



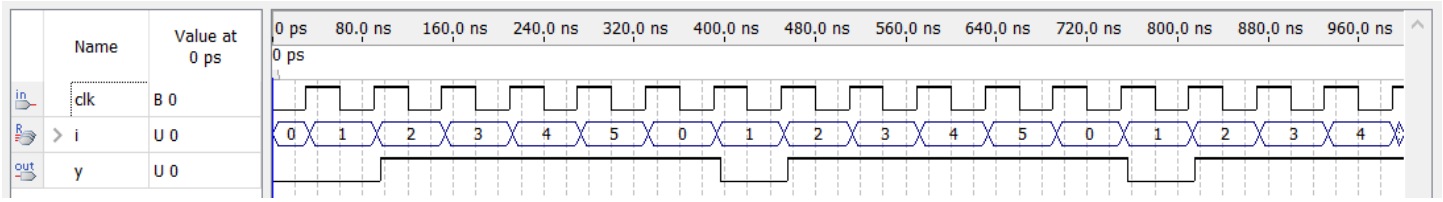
There are still 4 flip-flops in the generated view, same as our prediction.

e.) Simulation Results

Code1:



Code2:



Both waveforms align with our predictions.

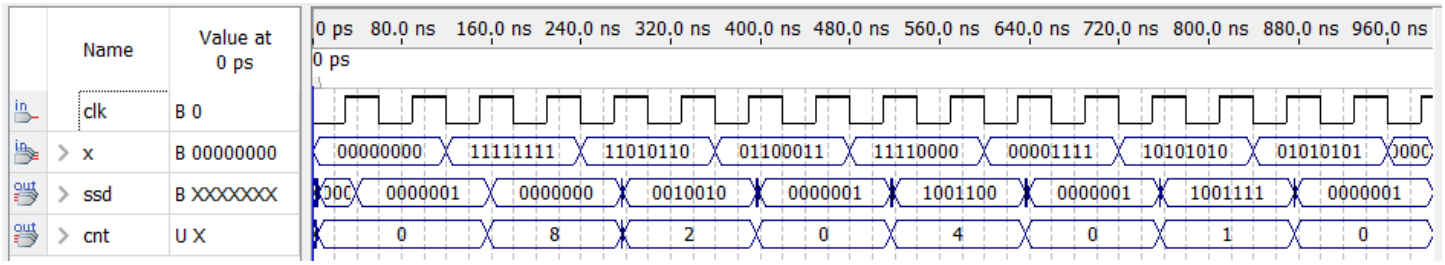
Exercise 3:

a.)

```vhdl
----------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
----------------------------------------------------------------------------
entity leading_ones_sequential is
    generic (N: integer := 8 ); --number of input bits
    port (x:   in std_logic_vector(N-1 downto 0);
          clk: in std_logic;
          ssd: out std_logic_vector(6 downto 0);
          cnt: buffer integer);
end entity;
----------------------------------------------------------------------------
architecture leading_ones_sequential of leading_ones_sequential is
begin
    process(x,clk)
        variable count: integer := 0;
        begin
            if rising_edge(clk) then
                count := 0;
                for i in N-1 downto 0 loop
                    if (x(i) = '1') then
                        count := count + 1;
                    else
                        exit;
                    end if;
                end loop;

                cnt <= count;

                --encode into ssd bits

                case count is
                    when 0 => ssd <= "0000001";
                    when 1 => ssd <= "1001111";
                    when 2 => ssd <= "0010010";
                    when 3 => ssd <= "0000110";
                    when 4 => ssd <= "1001100";
                    when 5 => ssd <= "0100100";
                    when 6 => ssd <= "0100000";
                    when 7 => ssd <= "0000001";
                    when 8 => ssd <= "0000000";
                    when 9 => ssd <= "0000100";
                    when others => ssd <= "0110000";
                end case;
            end if;
    end process;
end architecture;
----------------------------------------------------------------------------
```

b.) Sequential code is easier to write. It is more intuitive and requires fewer signals.

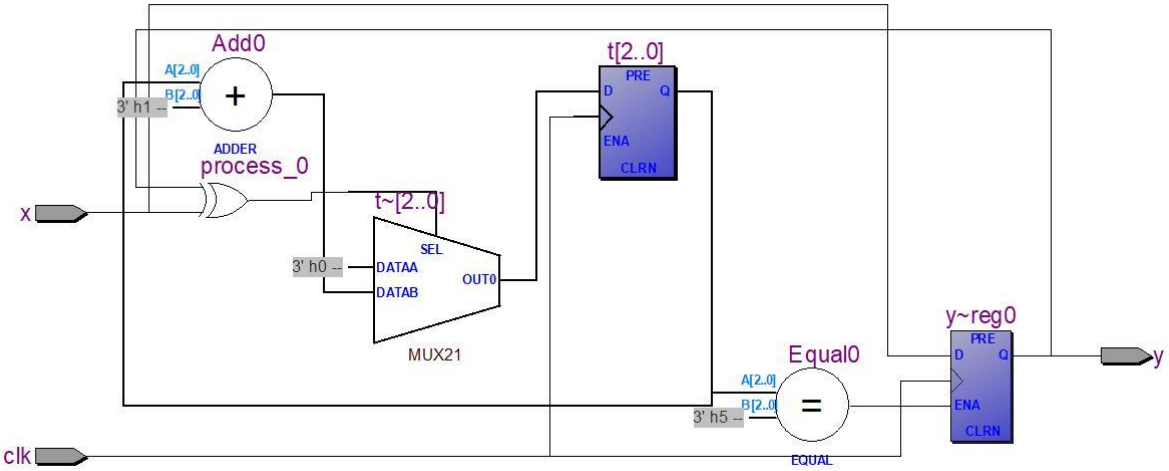
c.) Using the same test cases as last week:



Exercise 4

c.) For simulation, we counted 5 clock cycles before debouncing, so our debounce counter was 3 bits -> 3 flip flops. Then there is one more flip flop for if the button is debounced. (4 total flip flops) For synthesis, we want to count 1000000 clocks, so 20 bits would be needed for the counter and then one more for if the button is debounced (21 total flip flops).

d.)



The above RTL view agrees with our prediction of 4 flip flops (for simulation).

Code:

debouncer_demo.vhd

```
1   ----------------------------------------------------------------------------
2   library ieee;
3   use ieee.std_logic_1164.all;
4   ----------------------------------------------------------------------------
5   entity debouncer_demo is
6       generic (t_debounce: integer := 1000000);
7       PORT (x: in std_logic;
8               clk: in std_logic;
9               rst: in std_logic;
10              ssd_debounced_pins: OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
11              ssd_bounced_pins: OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
12  END ENTITY;
13  ----------------------------------------------------------------------------
14  architecture debouncer_demo of debouncer_demo is
15      signal y: std_logic;
16      signal num_debounced: integer range 0 to integer'high;
17      signal num_bounced: integer range 0 to integer'high;
18  begin
19      debouncer: entity work.switch_debouncer(switch_debouncer) port map(x, y, clk);
20      ssd_debounced : entity work.ssd(ssd) port map(num_debounced, ssd_debounced_pins);
21      ssd_bounced : entity work.ssd(ssd) port map(num_bounced, ssd_bounced_pins);
22      process(x, y, rst)
23      begin
24              if rst = '0' then
25                  num_debounced <= 0;
26                  num_bounced <= 0;
27              else
28                  if falling_edge(x) then
29                      num_bounced <= num_bounced + 1;
30                      if num_bounced = 9 then
31                          num_bounced <= 0;
```

```
                    end if;
                end if;
                if falling_edge(y) then
                    num_debounced <= num_debounced + 1;
                    if num_debounced = 9 then
                        num_debounced <= 0;
                    end if;
                end if;
            end if;
    end process;
end architecture;
-------------------------------------------------------------------------
```

switch_debouncer.vhd

```
-------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
-------------------------------------------------------------------------
entity switch_debouncer is
    generic (t_debounce: integer := 1000000);
    PORT (x: in std_logic;
            y: buffer std_logic;
            clk: in std_logic);
END ENTITY;
-------------------------------------------------------------------------
architecture switch_debouncer of switch_debouncer is
begin
    process(clk)
        variable t: integer range 0 to t_debounce;
        begin
            if rising_edge(clk) then
                if t = t_debounce then
                    y <= x;
                end if;
                if x /= y then
                    t := t + 1;
                else
                    t := 0;
                end if;
            end if;
        end process;
end architecture;
-------------------------------------------------------------------------
```

ssd.vhd

```
-------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------------------------------------------
ENTITY ssd IS
    PORT (num: in integer;
            ssd: OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END ENTITY;
-------------------------------------------------------------------------
ARCHITECTURE ssd OF ssd IS
BEGIN
    with num select
        ssd <= "0000001" when 0,
                "1001111" when 1,
```

```
                  "0010010" when 2,
                 "0000110" when 3,
                 "1001100" when 4,
                  "0100100" when 5,
                  "0100000" when 6,
                  "0001111" when 7,
                  "0000000" when 8,
                  "0000100" when 9,
                  "0110000" when others;
END ARCHITECTURE;
--------------------------------------------------------------------------------
```