Computer-Aided Design for VLSI Design

Homework2（Student ID: ntust_B10832019｜Name: 林琛琛）

1. Provide a simple explanation of your code.

```
A <= reg_0;
B <= reg_1;
C <= reg_2;
```

這段程式將三個 shift register 的值存到 A, B, C 三個 output signal 中。雖然程式的其他部分不會用到 A, B, C，但這麼做可以方便在 waveform diagram 時察看數值變化。

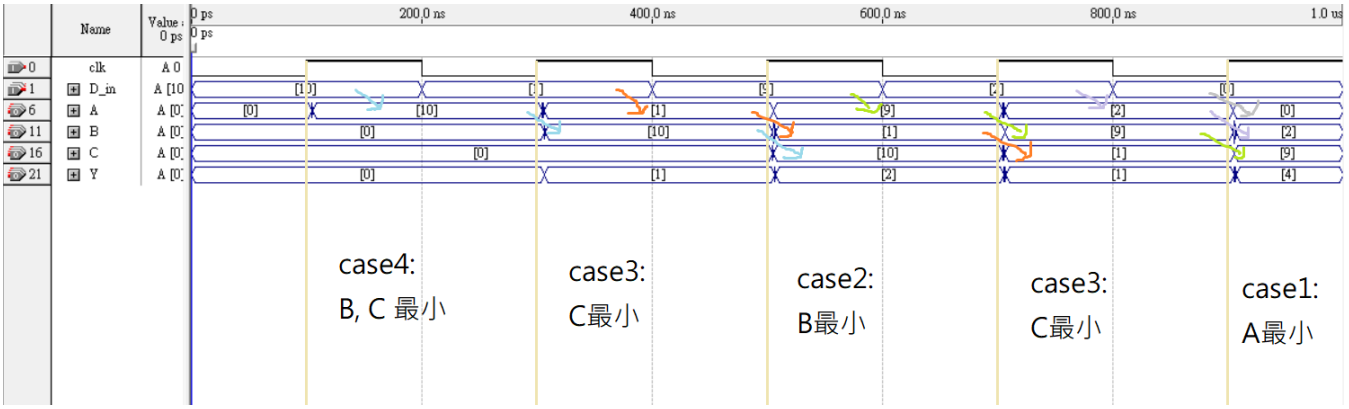
```
process(clk)
begin
    if(clk'EVENT and clk = '1') then
        reg_0 <= D_in;
        reg_1 <= reg_0;
        reg_2 <= reg_1;
    end if;
end process;
```

當 rising edge 的時候，要把 D_in 以及三個 register 的數值做 shift。

```
process(reg_0, reg_1, reg_2)
begin
    if( reg_0 < reg_1 and reg_0 < reg_2) then
        Y <= "100";
    elsif( reg_1 < reg_0 and reg_1 < reg_2) then
        Y <= "010";
    elsif( reg_2 < reg_0 and reg_2 < reg_1) then
        Y <= "001";
    else
        Y <= "000";
    end if;
end process;
```

每當三個 register 的數值有變化，代表 Y 值可能也隨之改變，因此重新比較三者的大小關係，根據情況給予 Y 相對應的數值。

2. Waveform diagram here (Simulation Results)



首先是 shift register 的部分，每當 rising edge 的時候 A 的值從 D_in 的值 shift 過去，並且 B, C 也有變化。圖上也有用不同顏色的箭頭標明數值 shift 的情況。

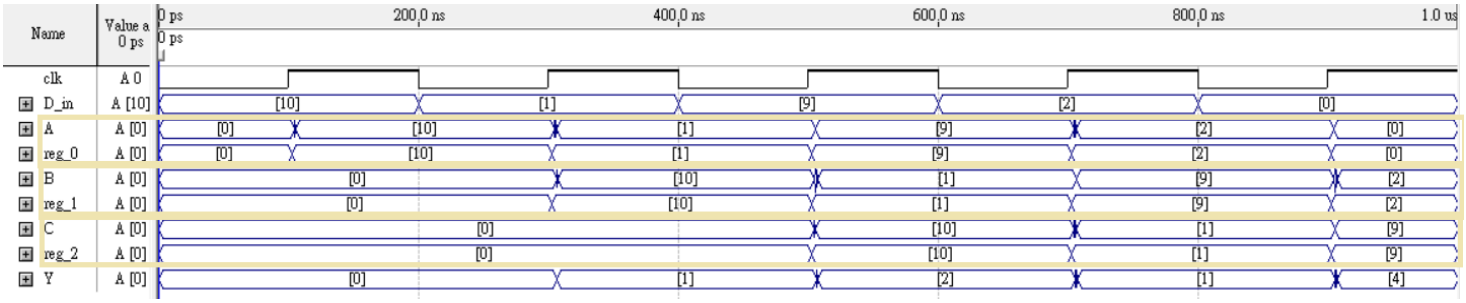
再來是 Y 判斷 A, B, C 三數值大小，分別用波型圖模擬出四種情況，也就是 A, B, C 各為唯一最小值，或者同時有至少兩個最小值。圖上同樣有將不同的情況標註。

另外從該圖也能看到 A, B, C, Y 的值，都是在 clk 的 rising edge 過後延遲一小段時間才會更新。

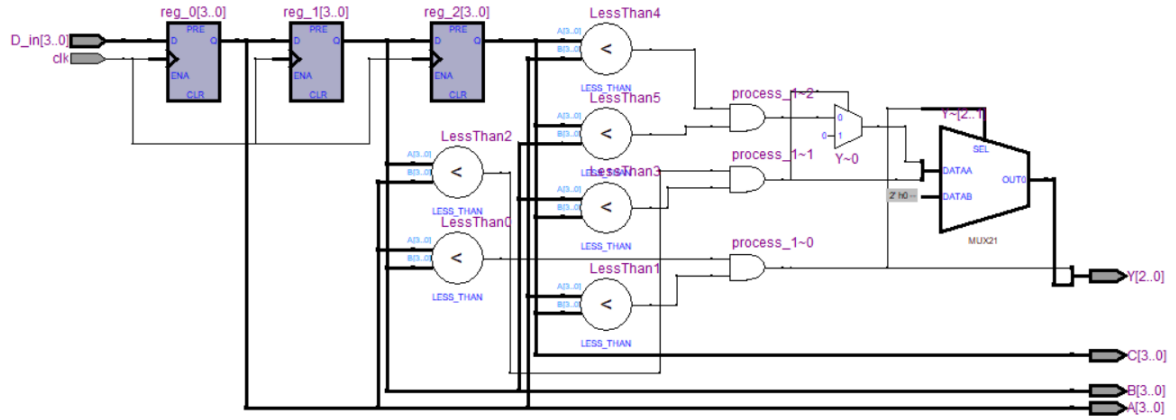
3. Reflections and discussions

這次的作業主要卡在兩個地方。第一個是要怎麼實作 shift register，後來想到把投影片中 DFF 的程式修改，再加上 reg_0, reg_1, reg_2 三個 signal。還有處理 waveform 的時候，也想了很久怎麼把 reg_0, reg_1, reg_2 的值也秀出來，因為沒在 Node_finder 找到。後來改額外加上三個 output，也就是 A, B, C 來做。
寫完作業後才發現，其實可以在 Node finder 的 filter 設定從預設的 Pins : all 選項改選為 Registers 就能找到 reg_0, reg_1, reg_2。我把 reg_0, reg_1, reg_2 跟 A, B, C 都放進 waveform simulation 比較，如下圖，發現 reg_0, reg_1, reg_2 的值會先更新，接著 A, B, C 才變，符合程式的運行邏輯。

(為了方便閱讀，繳交的作業把 reg_0, reg_1, reg_2 拿掉了)



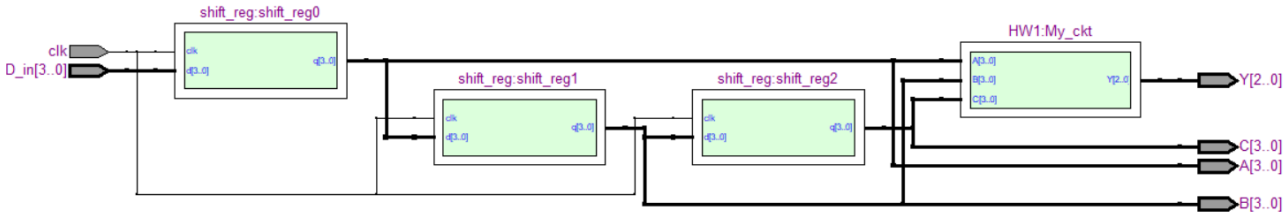
程式的部分同時用了 behavior model 跟 dataflow model，再一次感受到跟平常寫的程式語言的思考方式有很大不同。像是這次的作業，把 shift register 跟 My_ckt_1 拆開成兩個 process，想清楚彼此會怎麼觸發，才能寫得出來。



觀察合成後用 RTL viewer 的電路圖，比起 HW1 的結果，確實多了三個額外的 register。

另外，上課的時候也有講到 structural model 與 generate 語法，我認為這次的作業也能用它們來實現。不過作業二的電路較不適合用 generate，因為只有三個 shift register 的情況下，還要依照不同的 case 寫不同的 generate 指令，反倒不划算，直接用三個 component 代替即可。所以 generate 語法更適合一次需要大量相同元件的時候。

為做比較，我把 HW1 的比較跟三個 shift register 各自寫成 component 來實作，以下是電路合成圖：



當使用 component 完成此程式時，合成出來的成果也將相對應的部分包成一個元件，點擊 My_ckt 與 shift_reg 後可以看到內部和成的結果，則跟用 bahavior model 的結果相同。

component 版本的程式如下：

HW2.vhd

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

entity HW2 is
port(
        clk : in bit;
        D_in : in unsigned(3 downto 0);
        A, B, C : out unsigned(3 downto 0);
        Y : out std_logic_vector(2 downto 0));
end HW2;

architecture My_ckt_1 of HW2 is
component shift_reg
        port( d : in unsigned(3 downto 0);
        clk : in bit;
        q : out unsigned(3 downto 0));
end component;

component HW1
        port(A, B, C : in unsigned(3 downto 0);
                Y : out std_logic_vector(2 downto 0));
end component;

signal reg_0, reg_1, reg_2 : unsigned(3 downto 0);
begin
```

```
	A <= reg_0;
	B <= reg_1;
	C <= reg_2;

	shift_reg0: shift_reg port map(D_in, clk, reg_0);
	shift_reg1: shift_reg port map(reg_0, clk, reg_1);
	shift_reg2: shift_reg port map(reg_1, clk, reg_2);

	My_ckt: HW1 port map(A => reg_0, B => reg_1, C => reg_2, Y => Y);

end My_ckt_1;
```

shift_reg.vhd

```
library ieee;
use ieee.numeric_bit.all;

entity shift_reg is
port( d : in unsigned(3 downto 0);
	clk : in bit;
	q : out unsigned(3 downto 0));
end shift_reg;

architecture dataflow of shift_reg is
begin
	process(clk)
	begin
		if clk'event and clk = '1' then
			q <= d;
		end if;
	end process;
end dataflow;
```

HW1.vhd

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

entity HW1 is
port(
	A, B, C : in unsigned(3 downto 0);
	Y : out std_logic_vector(2 downto 0));
end HW1;
```

```
architecture My_ckt_1 of HW1 is
begin
	process(A, B, C)
	begin
		if (A < B and A < C) then
			Y <= "100";

		elsif B < A and B < C then
			Y <= "010";

		elsif C < A and C < B then
			Y <= "001";

		else
			Y <= "000";
		end if;
	end process;
end My_ckt_1;
```