(1)Construct a 30-second down counter (timer) with pause function. When the counter goes to 0, all the LEDs will be lighted up. You can use one push button for reset and one other for pause/start function.

1.1 Implement a periodic 30-second down counter and demo with the FPGA board.

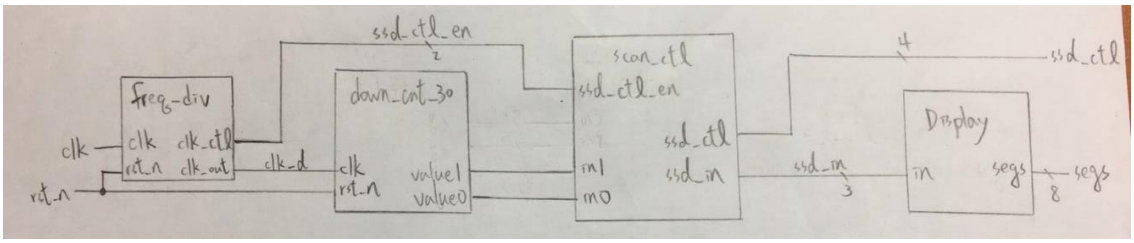
IO:

Input: clk, rst_n

Output: [3:0]ssd_ctl, [7:0]segs

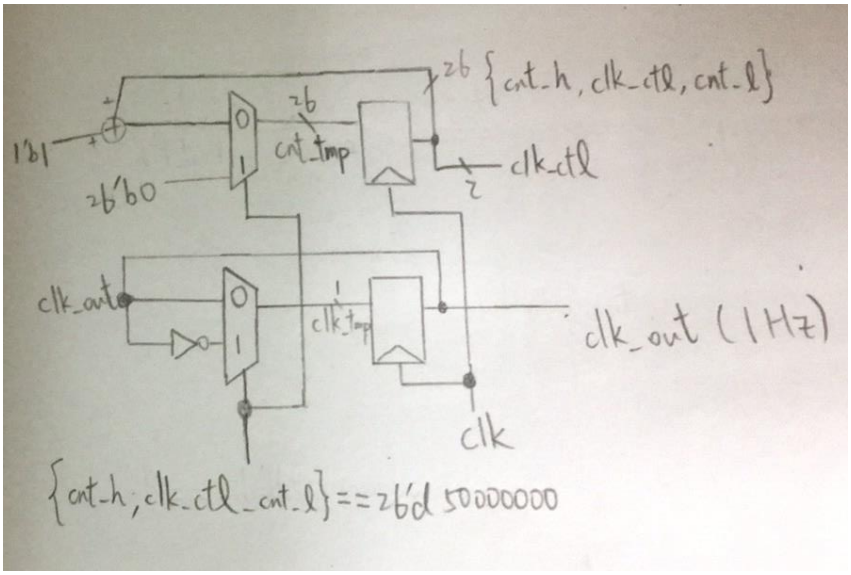
Block diagram:

需要 4 個 module 組成，分別為 frequency divider、2-digit BCD down counter、scan control 以及 7-segment display decoder。



Logic diagram:

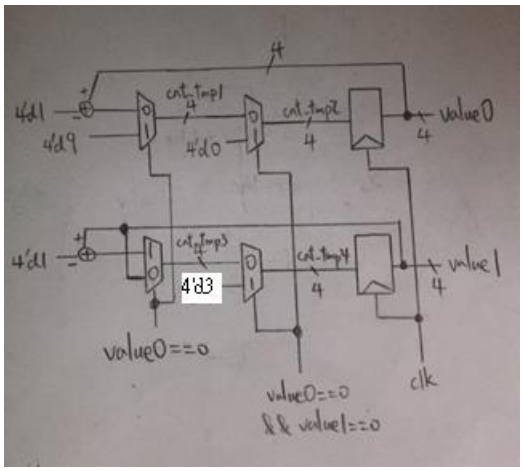
<u>100MHz to 1Hz frequency divider with 2-bit clock control</u>:



由於 FPGA 板上的頻率為 100MHz，建構理念是設計一個上限為 50M 的 binary up counter，在達到 50M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 50M 時，便將輸出值(clk_out) toggle 一次，也將 counter 歸零，如此下來，輸出值(clk_out)在等於 0 和等於 1 的時間各自都是 50M 次的 clk，也就是說，clk_out 的值 0.5 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(50M*2) = 1Hz。

另外，將上限為 50M 的 counter 中的第 16、17bit 抓出來當作 scan control 的控制項，如此可以藉由視覺暫留達成 4 個七段顯示器分別顯示各自數字的效果。

2-digit BCD down counter:



利用多工器選擇 cnt_tmp1、cnt_tmp3 的值，當個位數(value0)≠0 時，代表個位數不需要重置成 9→cnt_tmp1 = 個位數(value0)減一；個位數不需要借位，十位數(value1)維持原本的值→cnt_tmp3 = value1；當個位數 = 0 時，代表個位數下一次從 9 開始→cnt_tmp1 = 9；個位數需要借位，十位數減一→cnt_tmp3 = value1 – 1。
第二個多工器判斷倒數是否結束(value1 = value0 = 0)，若未結束→cnt_tmp2 = cnt_tmp1、cnt_tmp4 = cnt_tmp3；若已結束→cnt_tmp2 = 0、cnt_tmp4 = 3。
在每個 1Hz 的 clock 的 positive edge 來時，輸出 value0、value1 的值分別會等於 cnt_tmp2、cnt_tmp4。

Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111
ssd_ctl_en = 01 → ssd_ctl = 1011
ssd_ctl_en = 10 → ssd_ctl = 1101
ssd_ctl_en = 11 → ssd_ctl = 1110

四者輪流顯示，而且要注意的地方是 ssd_ctl 為 low active，因此 ssd_ctl_en = 00 時→顯示第一個七段顯示器(全暗)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(全暗)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(十位數秒數 value1)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(個位數秒數 value0)。

7-segment display decoder:

in = 0 → segs = 8'b0000_0011
in = 1 → segs = 8'b1001_1111
in = 2 → segs = 8'b0010_0101

in = 3 → segs = 8'b0000_1101
in = 4 → segs = 8'b1001_1001
in = 5 → segs = 8'b0100_1001
in = 6 → segs = 8'b0100_0001
in = 7 → segs = 8'b0001_1111
in = 8 → segs = 8'b0000_0001
in = 9 → segs = 8'b0000_1001
default 為 segs = 8'b1111_1111(全暗)

結合以上功能，可以完成一個週期性的 30 秒倒數計時器。

I/O pin assignment:

segs[7]	segs[6]	segs[5]	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]
W7	W6	U8	V8	U5	V5	U7	V7

ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	clk	rst_n
W4	V4	U4	U2	W5	R2

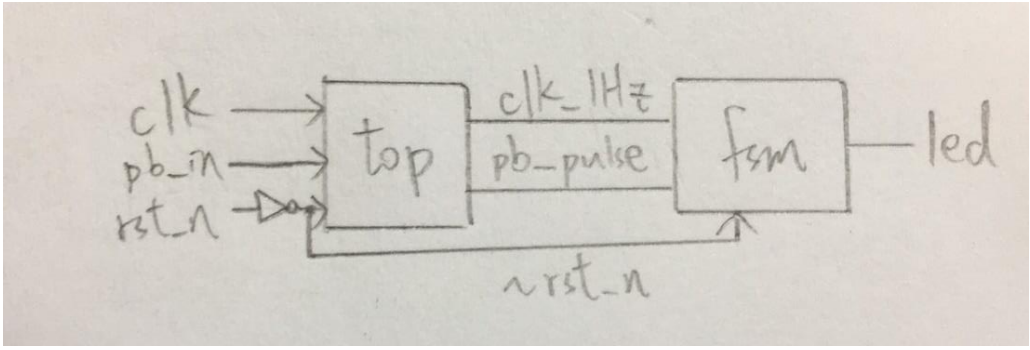
1.2 Implement Prelab 1.3 and demo with the FPGA board.

IO:

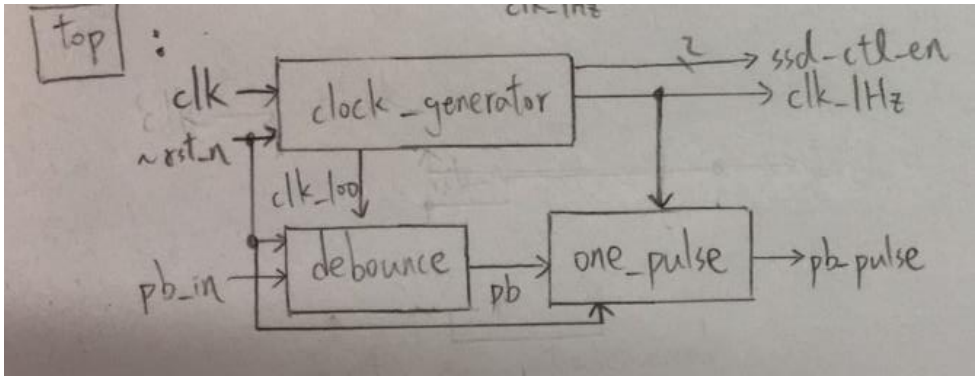
Input: clk, rst_n, pb_in
Output: led

Block diagram:

本實驗需要兩個 module 組成，分別是 top 以及 fsm。

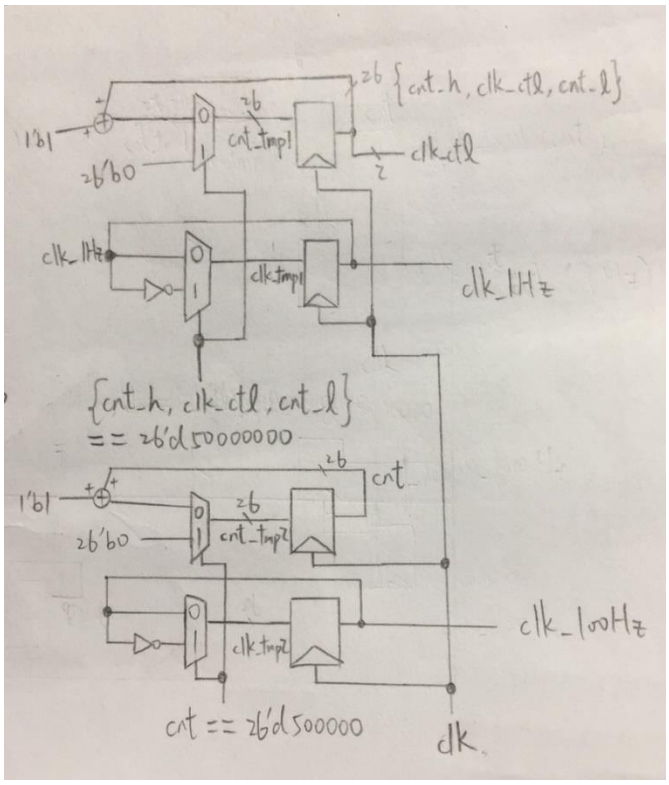


其中 top 又分為三個子 module(clock generator + debounce circuits + one pulse)



Logic diagram:

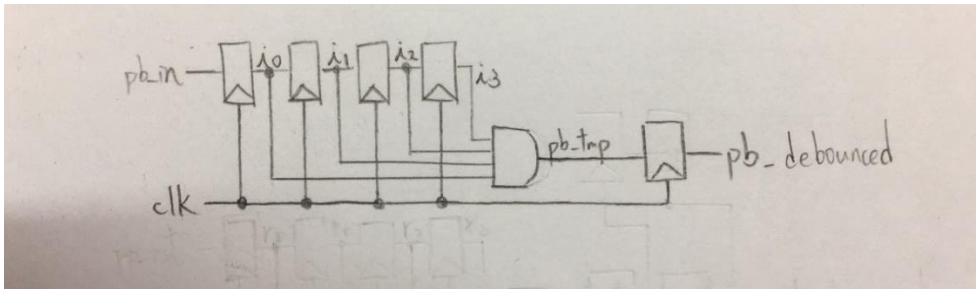
Clock generator:



本次除頻器需要製造兩個不同頻率的 clock，分別為 1Hz 和 100Hz，由於 FPGA 板上的頻率為 100MHz，建構理念是分別設計一個上限為 50M 和 500K 的 binary up counter，前者在達到 50M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 50M 時，便將輸出值(clk_1Hz) toggle 一次，也將 counter 歸零，後者則是在達到 500K 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 500K 時，便將輸出值(clk_100Hz) toggle 一次，也將 counter 歸零，如此下來，輸出值(clk_1Hz)在等於 0 和等於 1 的時間各自都是 50M 次的 clk，也就是說，clk_1Hz 的值 0.5 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(50M*2) = 1Hz；輸出值 clk_100Hz 在等於 0 和等於 1 的時間各自都是 500K 次的 clk，也就

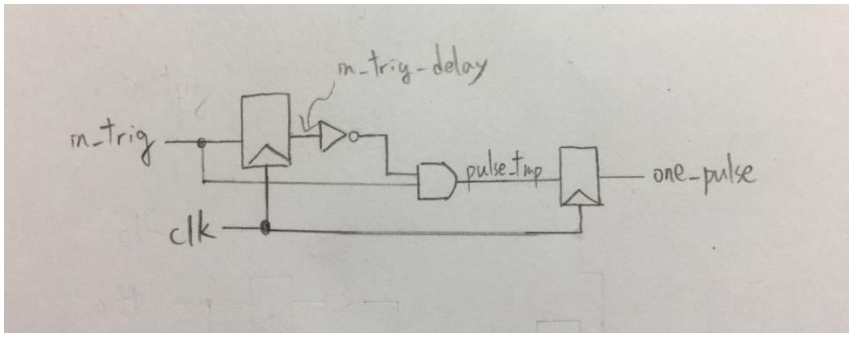
是說，clk_100Hz 的值 0.005 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(500K*2) = 100Hz。

Debounce circuits:



由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

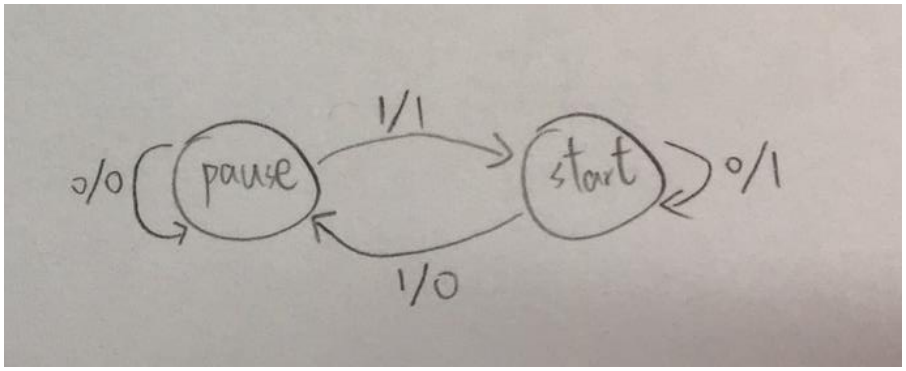
One pulse:



由於每次按下 push bottom 的時間都不一樣，要確保只在一個 clock 裡有輸入訊號，必須要經過上圖的邏輯來產生。將輸入(in_trig)延遲一個 clock，得到 in_trig_delay，再將 in_trig_delay 與接下來的輸入(in_trig)做 and，可以得到只有在一個 clock 裡有 positive signal 的輸出(one_pulse)。

Finite state machine:

若要觀察 state 的變化，可由 FSM 中的輸出 count enable 訊號接上 led 燈來看，當 state 為 start 時，count enable 為 1；當 state 為 pause 時，count enable 為 0。



重置時的 state 會在 pause 的狀態，當不是 reset 時只要輸入 in 為 1，則 state 會從原本的變成另外一個 state(pause→start,輸出 count enable 為 1 / start→pause, 輸出 count enable 為 0)；而當輸入 in 為 0，state 會維持在原本的 state(pause→ pause,輸出 count enable 為 0 / start→start,輸出 count enable 為 1)。

連接以上 module，可以藉由 LED 燈觀察 state 的變化。

I/O pin assignment:

I/O	clk	rst_n	pb_in	led
pin	W5	U18	W19	U16

1.3 Combine 1.2 and 1.3 to finish the experiment.

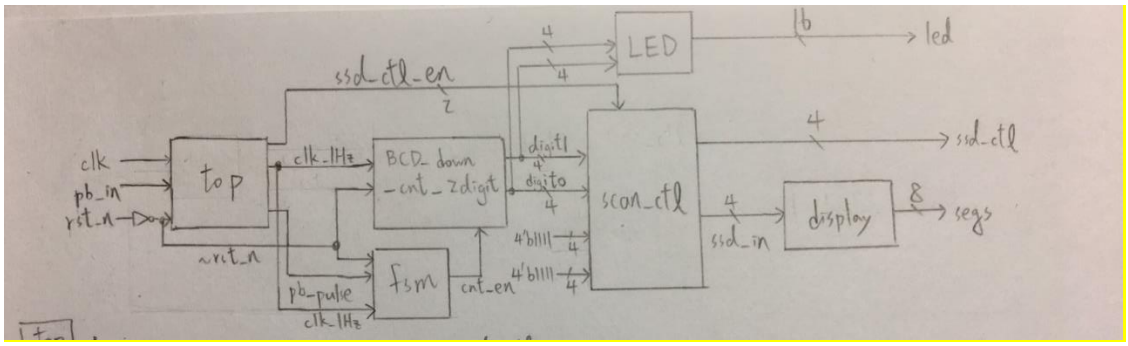
IO:

Input: clk, rst_n, pb_in

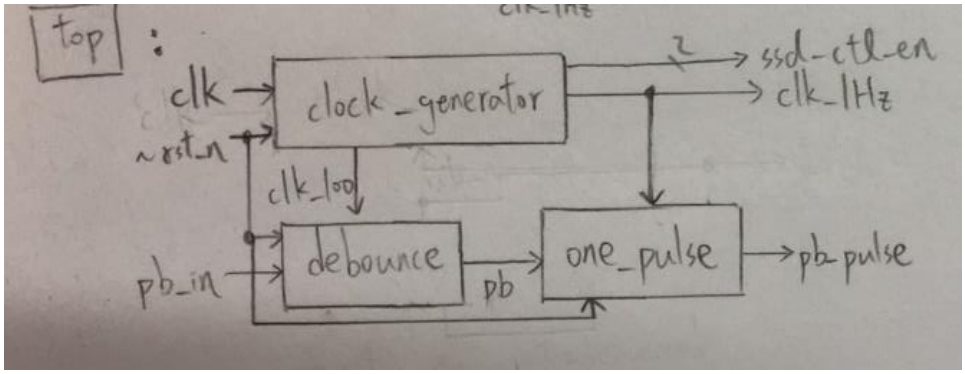
Output: [15:0]led, [3:0]ssd_ctl, [7:0]segs

Block diagram:

本實驗需要用到 6 個子 module:分別是 top、2-digit BCD down counter、finite state machine、LED、scan control 以及 7-segment display decoder。
其中由於控制 reset 的輸入為 push bottom，按下去時(訊號為 1)要 reset(0)，因此先將 push bottom 的輸入值通過 invert，就可以達到正確的效果。



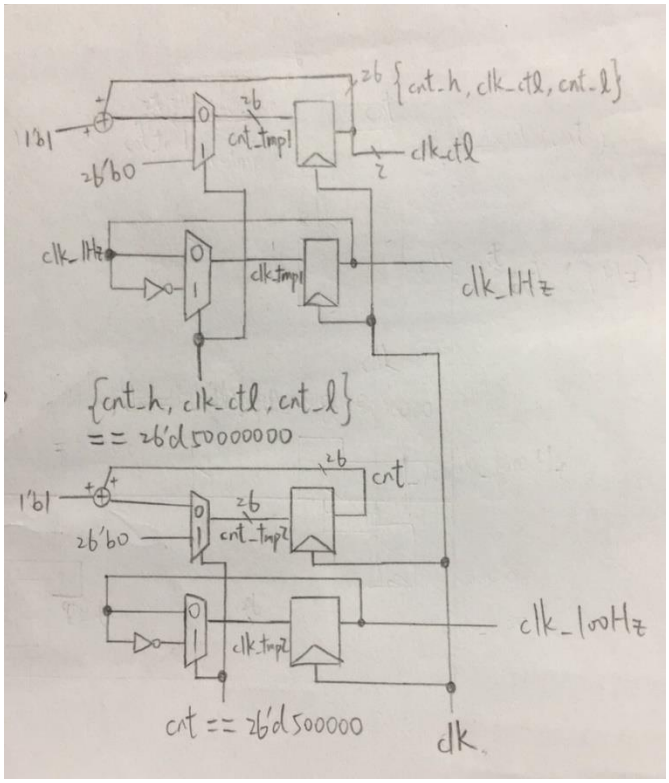
其中 top 又分為三個子 module(clock generator + debounce circuits + one pulse)



設計結果為重置後會回到 30 秒並在 pause 狀態下的 30 秒倒數計時器。
(two push bottom，one for reset and the other for pause/start)

Logic diagram:

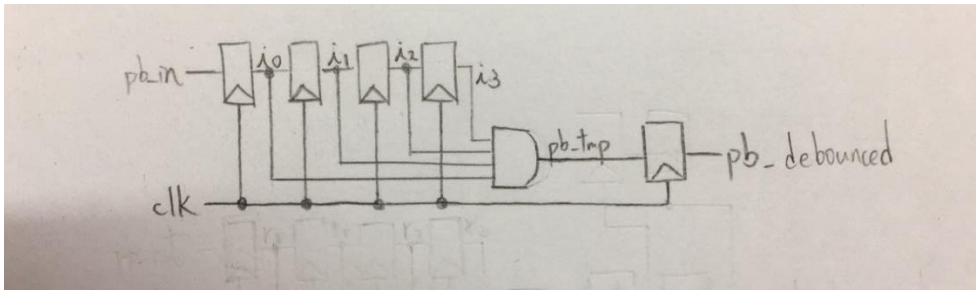
Clock generator:



本次除頻器需要製造兩個不同頻率的 clock，分別為 1Hz 和 100Hz，由於 FPGA 板上的頻率為 100MHz，建構理念是分別設計一個上限為 50M 和 500K 的 binary up counter，前者在達到 50M 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 50M 時，便將輸出值(clk_1Hz) toggle 一次，也將 counter 歸零，後者則是在達到 500K 之前，每個 clk 都讓 counter 加上 1，而當 counter 數到 500K 時，便將輸出值(clk_100Hz) toggle 一次，也將 counter 歸零，如此下來，輸出值

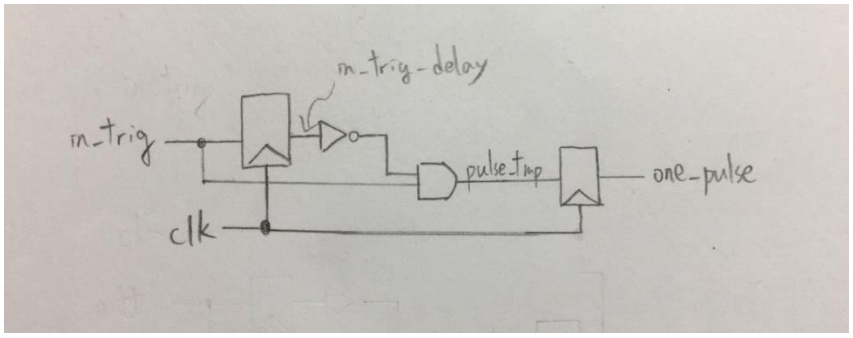
(clk_1Hz)在等於 0 和等於 1 的時間各自都是 50M 次的 clk，也就是說，clk_1Hz 的值 0.5 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(50M*2) = 1Hz；輸出值 clk_100Hz 在等於 0 和等於 1 的時間各自都是 500K 次的 clk，也就是說，clk_100Hz 的值 0.005 秒會 toggle 一次，因此輸出值的頻率將會等於 100MHz/(500K*2) = 100Hz。

Debounce circuits:



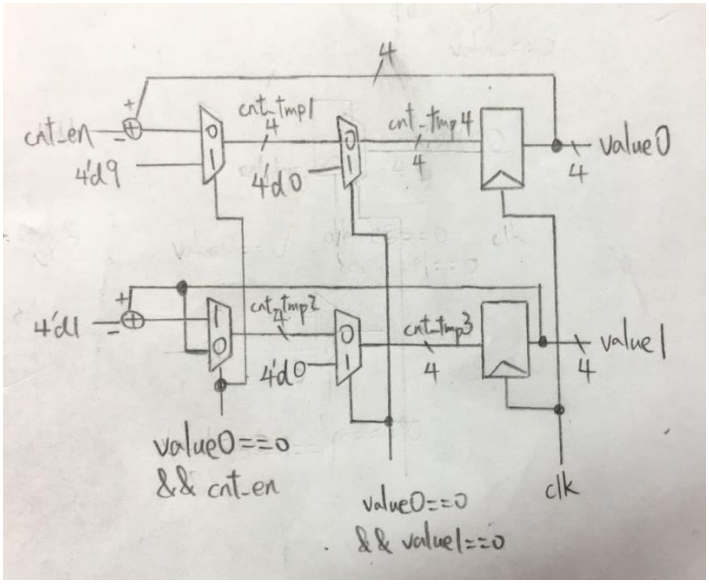
由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

One pulse:



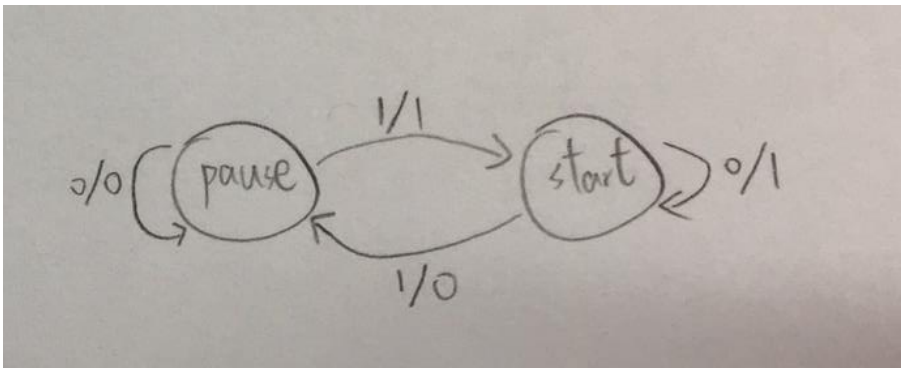
由於每次按下 push bottom 的時間都不一樣，要確保只在一個 clock 裡有輸入訊號，必須要經過上圖的邏輯來產生。將輸入(in_trig)延遲一個 clock，得到 in_trig_delay，再將 in_trig_delay 與接下來的輸入(in_trig)做 and，可以得到只有在一個 clock 裡有 positive signal 的輸出(one_pulse)。

BCD down counter(2digit):



需要製作一個 30→0 的倒數計時器。
利用多工器選擇 cnt_tmp1、cnt_tmp2 的值，當個位數(value0)≠0 時，代表個位數不需要重置成 9→cnt_tmp1 = 個位數(value0)減一；個位數不需要借位，十位數(value1)維持原本的值→cnt_tmp2 = value1；當個位數 = 0 且需要下數(cnt_en = 1)時，代表個位數下一次從 9 開始→cnt_tmp1 = 9；個位數需要借位，十位數(value1)減一→cnt_tmp2 = value1 – 1。
第二組多工器判斷倒數是否結束(value1 = value0 = 0)，若未結束→cnt_tmp4 = cnt_tmp1、cnt_tmp3 = cnt_tmp2；若已結束→cnt_tmp4 = 0、cnt_tmp3 = 0。
在每個 1Hz 的 clock 的 positive edge 來時，輸出 value0、value1 的值分別會等於 cnt_tmp4、cnt_tmp3。

Finite state machine:



重置時的 state 會在 pause 的狀態，當不是 reset 時只要輸入 in 為 1，則 state 會從原本的變成另外一個 state(pause→start,輸出 count enable 為 1 / start→pause,輸出 count enable 為 0)；而當輸入 in 為 0，state 會維持在原本的 state(pause→pause,輸出 count enable 為 0 / start→start,輸出 count enable 為 1)。

LED:

這個 module 判斷倒數計時器是否數到 00，若是的話→輸出值 led = 16’b1111111111111111→LED 燈全亮；若不是→輸出值 led = 16’b0000000000000000→LED 燈全暗。

Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111
ssd_ctl_en = 01 → ssd_ctl = 1011
ssd_ctl_en = 10 → ssd_ctl = 1101
ssd_ctl_en = 11 → ssd_ctl = 1110

四者輪流顯示，而且要注意的地方是 ssd_ctl 為 low active，因此 ssd_ctl_en = 00 時→顯示第一個七段顯示器(全暗)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(全暗)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(十位數秒數 value1)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(個位數秒數 value0)。

7-segment display decoder:

in = 0 → segs = 8'b0000_0011
in = 1 → segs = 8'b1001_1111
in = 2 → segs = 8'b0010_0101
in = 3 → segs = 8'b0000_1101
in = 4 → segs = 8'b1001_1001
in = 5 → segs = 8'b0100_1001
in = 6 → segs = 8'b0100_0001
in = 7 → segs = 8'b0001_1111
in = 8 → segs = 8'b0000_0001
in = 9 → segs = 8'b0000_1001
default 為 segs = 8’b1111_1111(全暗)

結合所有的 module 功能，可以完成一個具有 pause/start 功能的 30 秒倒數計時器。

I/O pin assignment:

led[15]	led[14]	led[13]	led[12]	led[11]	led[10]	led[9]	led[8]
L1	P1	N3	P3	U3	W3	V3	V13

led[7]	led[6]	led[5]	led[4]	led[3]	led[2]	led[1]	led[0]
V14	U14	U15	W18	V19	U19	E19	U16

segs[7]	segs[6]	segs[5]	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]
W7	W6	U8	V8	U5	V5	U7	V7

ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	clk	pb_in	rst_n
W4	V4	U4	U2	W5	W19	U18

(2)The same function as Exp. 1, instead of using two push buttons for reset/pause/start, try to use just one push button to finish the design. (Hint: You can press the push button longer to represent the reset)

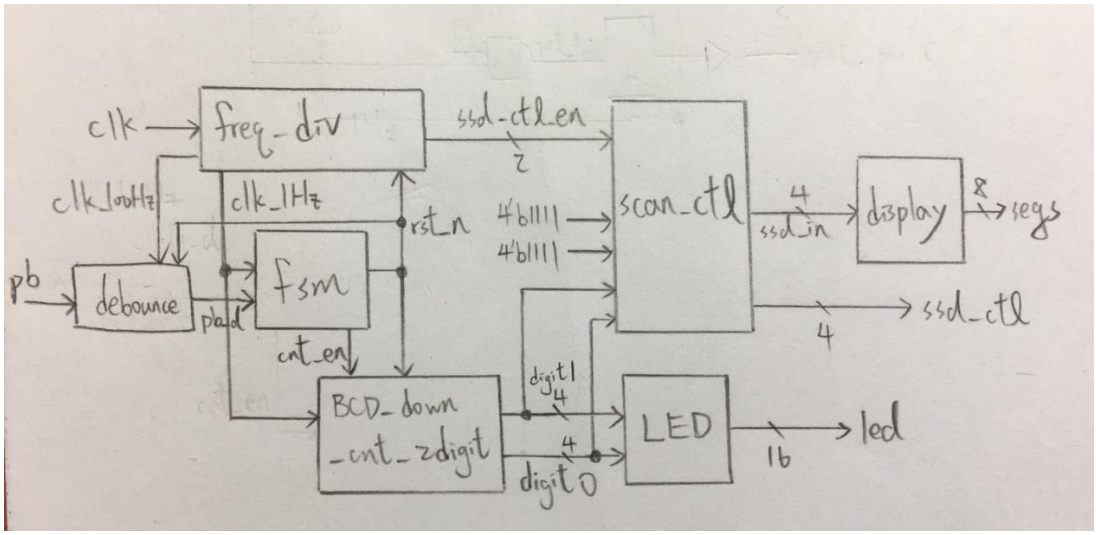
IO:

Input: clk, pb

Output: [15:0]led, [3:0]ssd_ctl, [7:0]segs

Block diagram:

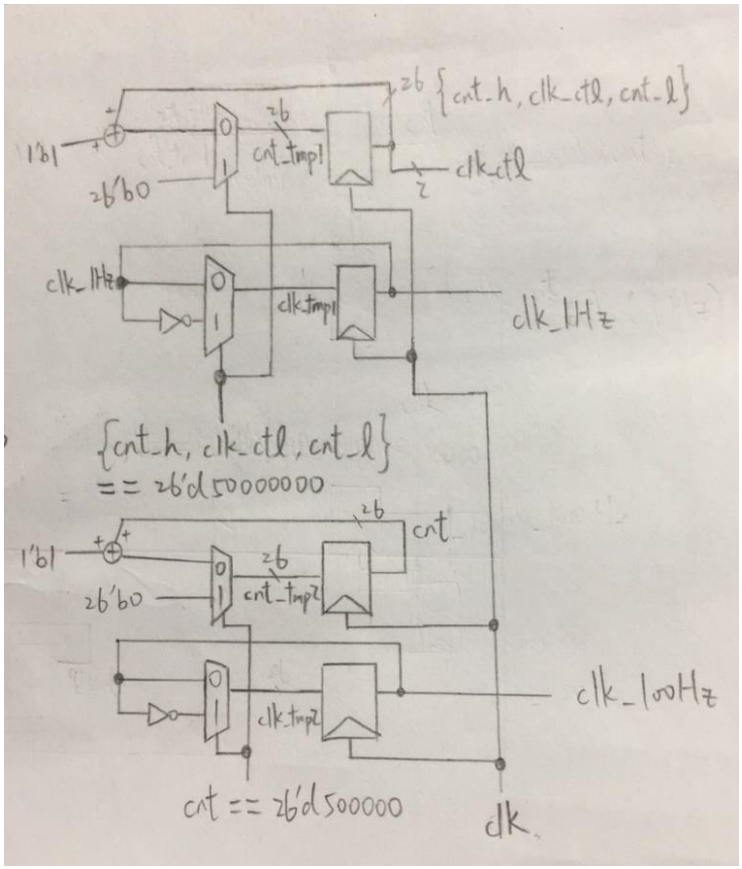
本實驗需要用到以下 7 個 module，分別為 frequency divider、debounce circuits、finite state machine、2digit BCD down counter(30second)、LED display、scan control 以及 7-segment display decoder。



設計結果為重置後會回到 30 秒並在 pause 狀態的 30 秒倒數計時器。
(只有一個 push bottom 輸入訊號，控制 start/pause/reset)

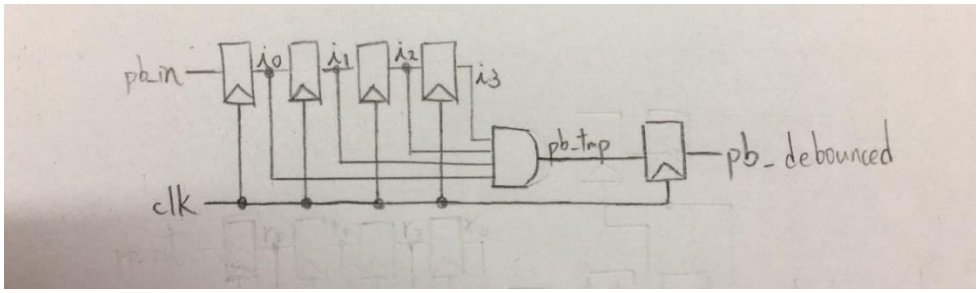
Logic diagram:

Frequency divider:



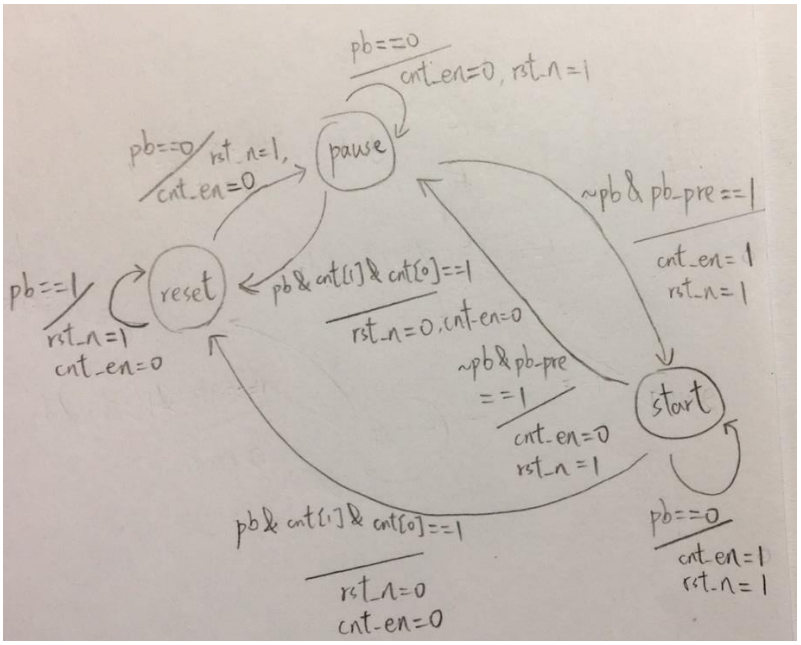
本 frequency divider 與第一小題的 clock generator 是相同的，功能為輸入 100MHz 的 clock 頻率並產生分別為 1Hz 與 100Hz 的 output 頻率。

Debounce circuits:

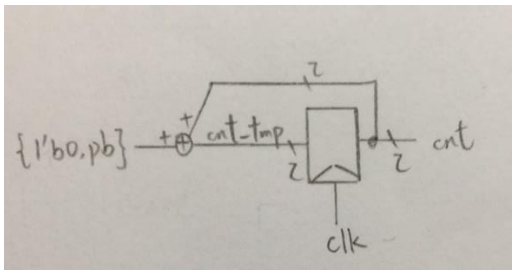


由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

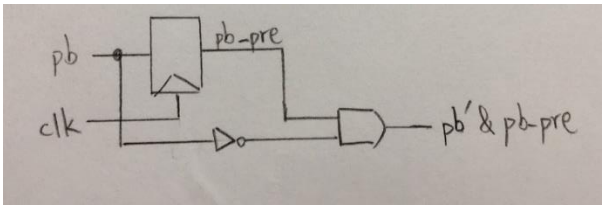
Finite state machine:



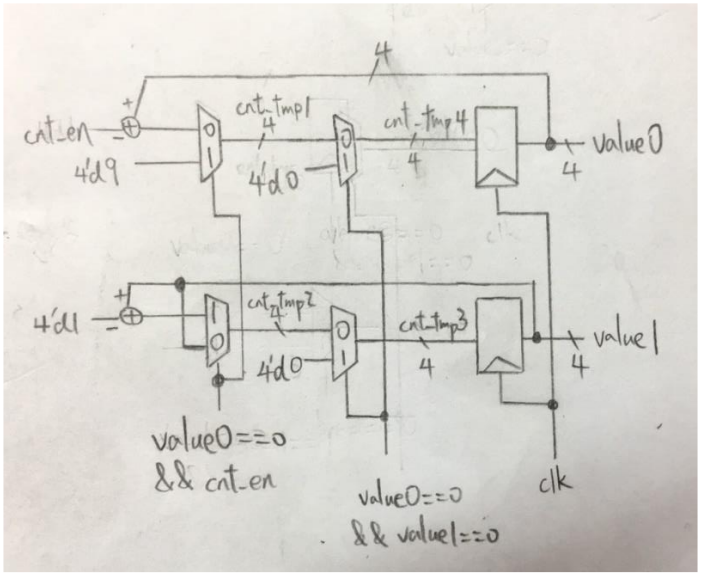
這個 FSM 的功能為在按下短時間的 push bottom 後，state 會從 start 轉成 pause 或從 pause 轉成 start，若是長時間按著 push bottom，則 state 會從 start 或 pause 轉成 reset，而放開按鈕後，state 會回到 pause。
以下是 reset 的設計過程，設計理念為製作一個 2bit 的 up counter:



每次 clock 會讓 cnt 加上 push bottom 的輸入值(pb)，並在輸入值(pb)的 falling edge 將 cnt 歸零，因此若是長按 push bottom，使得 cnt 數到 2'b11，則 pb = cnt[1] = cnt[0] = 1，FSM 在下一個 clock 後將會轉換成 reset 的狀態。
至於短時間的按下 push bottom，使得 state 在 pause 與 start 間轉換，做法為利用一個 flip flop，將上一個 clock 的 push bottom 訊號暫存下來(pb_pre)，再將 pb_pre 與當下的 push bottom 訊號(pb)的 invert 通過 and gate，就能夠判斷輸入訊號是否在上一個 clock 為 1，而現在是 0，也就是在放開 push bottom 的瞬間 pb' & pb_pre 會等於 1，使得 state 會從 pause 轉為 start 或從 start 轉成 pause。



BCD down counter(2digit):



30 秒的倒數計時器，與第一小題的相同。

Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111

ssd_ctl_en = 01 → ssd_ctl = 1011

ssd_ctl_en = 10 → ssd_ctl = 1101

ssd_ctl_en = 11 → ssd_ctl = 1110

四者輪流顯示，ssd_ctl_en = 00 時→顯示第一個七段顯示器(全暗)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(全暗)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(十位數秒數 value1)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(個位數秒數 value0)。

7-segment display decoder:

in = 0 → segs = 8'b0000_0011

in = 1 → segs = 8'b1001_1111

in = 2 → segs = 8'b0010_0101

in = 3 → segs = 8'b0000_1101

in = 4 → segs = 8'b1001_1001

in = 5 → segs = 8'b0100_1001

in = 6 → segs = 8'b0100_0001

in = 7 → segs = 8'b0001_1111

in = 8 → segs = 8'b0000_0001

in = 9 → segs = 8'b0000_1001

default 為 segs = 8’b1111_1111(全暗)

LED:

這個 module 判斷倒數計時器是否數到 00，若是的話→輸出值 led = 16’b1111111111111111→LED 燈全亮；若不是→輸出值 led = 16’b0000000000000000→LED 燈全暗。

最後，結合以上所有 module 功能，可以得到只用一個 push bottom 控制具有 reset/pause/start 功能的 30 秒倒數計時器。

I/O pin assignment:

led[15]	led[14]	led[13]	led[12]	led[11]	led[10]	led[9]	led[8]
L1	P1	N3	P3	U3	W3	V3	V13

led[7]	led[6]	led[5]	led[4]	led[3]	led[2]	led[1]	led[0]
V14	U14	U15	W18	V19	U19	E19	U16

segs[7]	segs[6]	segs[5]	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]
W7	W6	U8	V8	U5	V5	U7	V7

ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	clk	pb
W4	V4	U4	U2	W5	W19

(3)(Bonus) Use two push buttons to control a multi-function timer (mode selection, reset, start, stop). The stop timer has two modes: 30-second/1-minute countdown. When being reset, the seven-segment display shows the digits 30/1:00. When the timer counts to 0, it will stop.

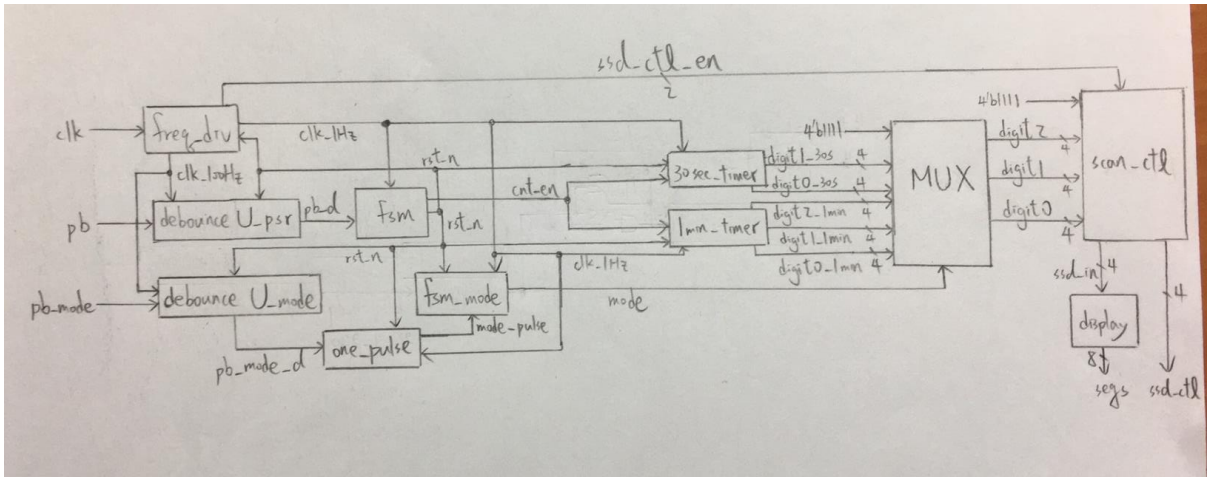
IO:

Input: clk, pb, pb_mode

Output: [3:0]ssd_ctl, [7:0]segs

Block diagram:

本實驗需要用到以下 module 功能，分別是 frequncy divider、兩個 debounce circuits、one pulse generator、兩個 finite state machine(1) for start/pause/reset(2) for mode selecting、兩個 down counter(1) 30 秒(2) 1 分鐘、選擇要顯示哪一個模式的 MUX、scan control 以及 7-segment display decoder。

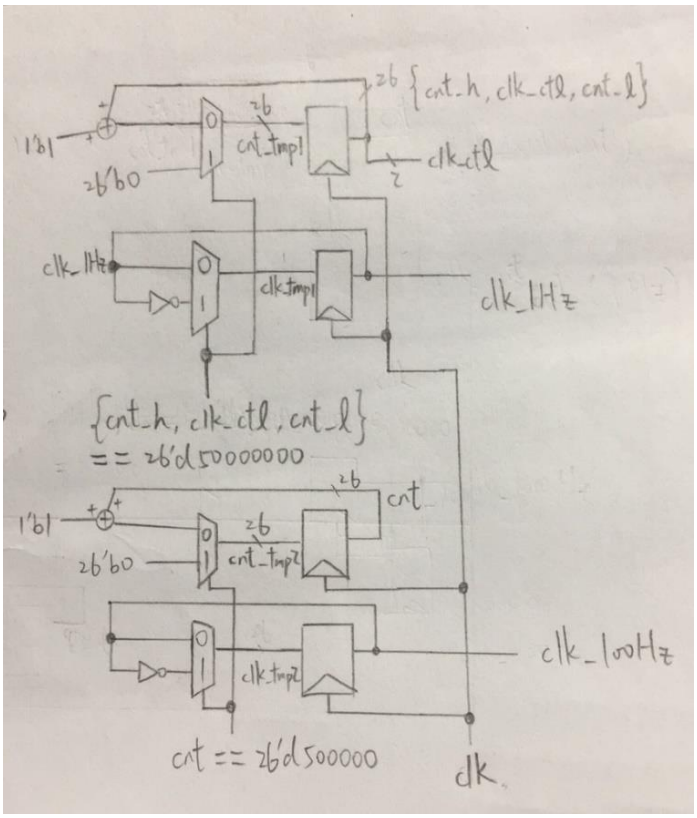


設計結果為 reset 後在 30 秒的模式並在暫停狀態，可以透過切換模式變為 1 分鐘模式並在暫停狀態下的倒數計時器。

(two push bottom，one for mode select and the other for start/pause/reset)

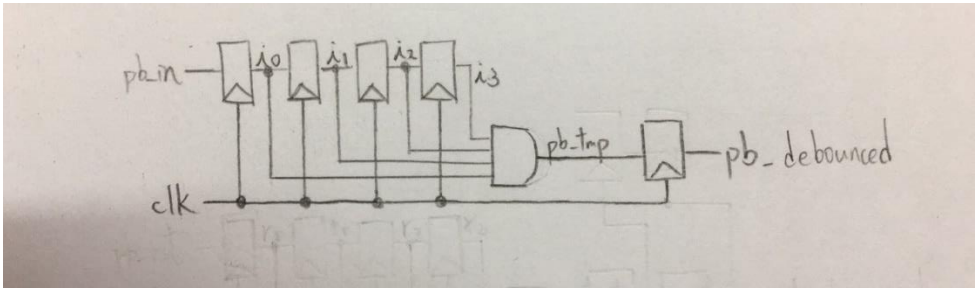
Logic diagram:

Frequency divider:



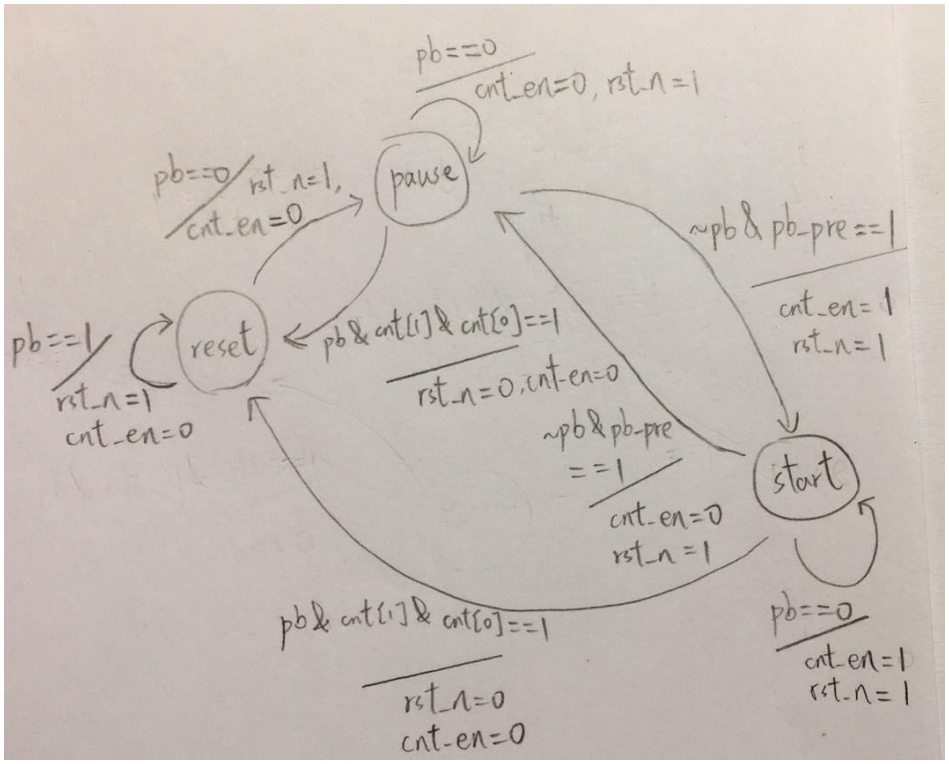
本 frequency divider 與前兩小題的 clock generator 是相同的，功能為輸入 100MHz 的 clock 頻率並產生分別為 1Hz 與 100Hz 的 output 頻率。

Debounce circuits:



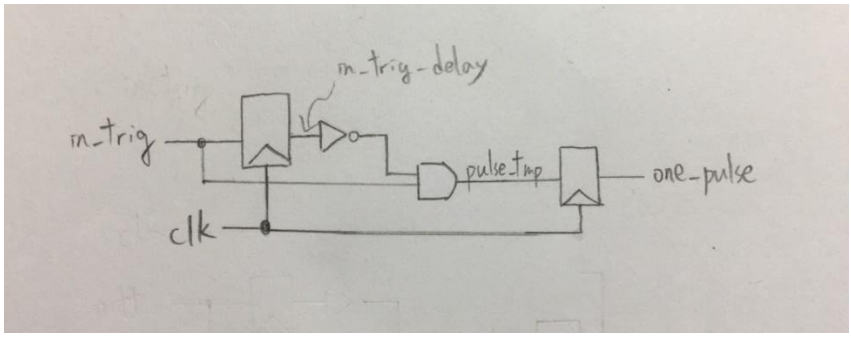
由於 push bottom 會產生 bouncing 的現象，因此將 push bottom 的輸入通過 4 個 flip flop 後，再將每個 flip flop 的輸出(i0～i3)通過 and gate，可以得到穩定的訊號值(pb_debounced)。

Finite state machine(pause/start/reset):



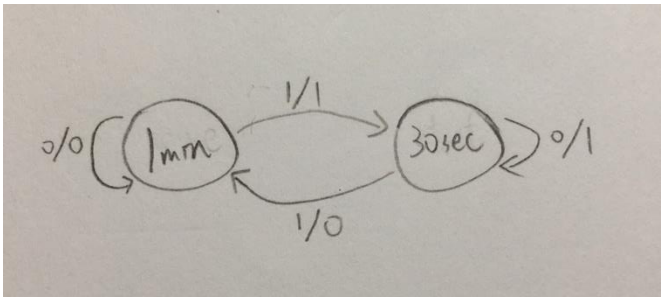
這個 FSM 控制 pause、start、reset，由其中一個 push bottom 來輸入訊號控制，其原理和設計皆與第二小題相同。

One pulse: For mode selecting



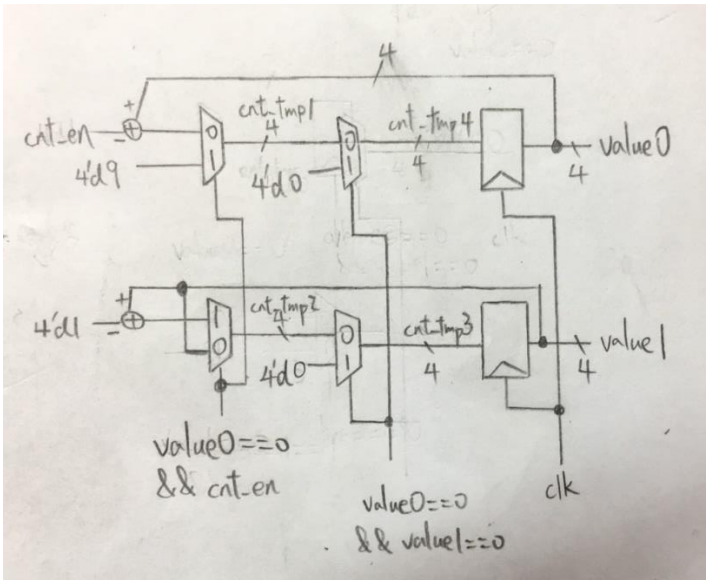
由於每次按下 push bottom 的時間都不一樣，要確保只在一個 clock 裡有輸入訊號，必須要經過上圖的邏輯來產生。將輸入(in_trig)延遲一個 clock，得到 in_trig_delay，再將 in_trig_delay 與接下來的輸入(in_trig)做 and，可以得到只有在一個 clock 裡有 positive signal 的輸出(one_pulse)。

Finite state machine(mode selecting):



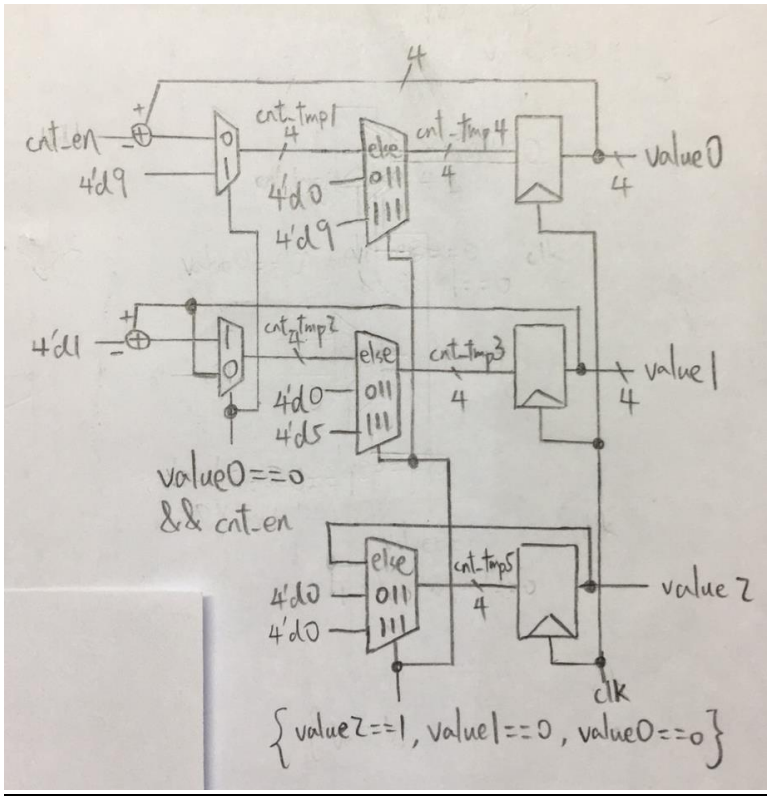
由於這個 FSM 的輸入已經通過 debounce circuits 和 one pulse generator，因此可以確保按下一次 push bottom 後只有一個 clock 內有輸入訊號，並從 1minute 的 mode 轉換成 30second 的 mode 或從 30second 的 mode 轉換成 1minute 的 mode，而當沒有按下 push bottom 時，mode 會維持在原本的 mode。
(重置時的 mode 為 30 秒模式)

30 second timer:



30 秒的倒數計時器，與前兩小題的相同。

1 min timer:



一分鐘的倒數計時器。(value2→分鐘/value1→十位數秒數/value0→個位數秒數)
設計理念為當 reset 時，將初始值設為 1:00，利用多工器選擇 cnt_tmp1、cnt_tmp2 的值，當個位數(value0)≠0 時，代表個位數不需要重置成 9→cnt_tmp1 = 個位數(value0)減一；個位數不需要借位，十位數(value1)維持原本的值→cnt_tmp2 = value1；當個位數 = 0 且需要下數(cnt_en = 1)時，代表個位數下一次從 9 開始→cnt_tmp1 = 9；個位數需要借位，十位數(value1)減一→cnt_tmp2 = value1 – 1。

第二組多工器給予計時器在 1:00 變成 0:59 的值，當 value2 = 1 , value1 = value0 = 0，代表下一秒要顯示 0:59→cnt_tmp5 = 0 , cnt_tmp3 = 5 , cnt_tmp4 = 9，和判斷倒數是否結束(value2 = value1 = value0 = 0)，若未結束→cnt_tmp5 = value2, cnt_tmp4 = cnt_tmp1, cnt_tmp3 = cnt_tmp2；若已結束→cnt_tmp5 = 0, cnt_tmp4 = 0, cnt_tmp3 = 0。

在每個 1Hz 的 clock 的 positive edge 來時，輸出 value0、value1、value2 的值分別會等於 cnt_tmp4、cnt_tmp3、cnt_tmp5。

MUX:

透過當前的 mode 選擇要顯示在 FPGA 板上的數值:

(1)當 mode 為 1→當前為 30 second timer mode→顯示 30 秒倒數計時器在 FPGA 板上。

(2)當 mode 為 0→當前為 1 min timer mode→顯示 1 分鐘倒數計時器在 FPGA 板上。

Scan control:

ssd_ctl_en = 00 → ssd_ctl = 0111

ssd_ctl_en = 01 → ssd_ctl = 1011

ssd_ctl_en = 10 → ssd_ctl = 1101

ssd_ctl_en = 11 → ssd_ctl = 1110

(1) mode 為 1→顯示 30 秒倒數計時器:

四個七段顯示器輪流顯示， ssd_ctl_en = 00 時→顯示第一個七段顯示器(全暗)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(全暗)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(十位數秒數 value1)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(個位數秒數 value0)。

(2) mode 為 0→顯示 1 分鐘倒數計時器:

四個七段顯示器輪流顯示，ssd_ctl_en = 00 時→顯示第一個七段顯示器(全暗)；ssd_ctl_en = 01 時→顯示第二個七段顯示器(分鐘 value2)；ssd_ctl_en = 10 時→顯示第三個七段顯示器(十位數秒數 value1)；ssd_ctl_en = 11 時→顯示第四個七段顯示器(個位數秒數 value0)。

Display:

in = 0 → segs = 8'b0000_0011

in = 1 → segs = 8'b1001_1111

in = 2 → segs = 8'b0010_0101

in = 3 → segs = 8'b0000_1101

in = 4 → segs = 8'b1001_1001

in = 5 → segs = 8'b0100_1001

in = 6 → segs = 8'b0100_0001

in = 7 → segs = 8'b0001_1111

in = 8 → segs = 8'b0000_0001

in = 9 → segs = 8'b0000_1001

default 為 segs = 8’b1111_1111(全暗)

結合以上 module，可以完成一個可以切換 30 秒或 1 分鐘模式並具有 start/pause/reset 功能的倒數計時器。

I/O pin assignment:

segs[7]	segs[6]	segs[5]	segs[4]	segs[3]	segs[2]	segs[1]	segs[0]
W7	W6	U8	V8	U5	V5	U7	V7

ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]	clk	pb	pb_mode
W4	V4	U4	U2	W5	W19	U18

Discussion:

這次實驗的每一小題都蠻複雜的，雖然好像每個小題都只差一點功能，但真的做起來其實沒有想像中的那麼順利，在這次的實驗裡，我發現 FSM 是所有功能中最重要的，尤其是第二題，我一開始把 reset 狀態 push bottom 為 1 時的 rst_n 設為 0，結果就導致其他 module 像是 frequency divider 被重置，進而輸出都是 0 的 divided clock，造成所有功能都停擺，包括 FSM 本身因為沒有 clock 訊號也沒辦法進入下一個 state，陷入無限的循環，那時為了找出錯誤，我甚至把每個 module 都單獨抓出來做 testbench，然後慢慢地一一連上，找了很久才發現原來是這個問題，因此即使我照著已經思考許久畫出來的 block diagram 和 logic diagram 的邏輯來寫 verilog，還是很有機會會在小地方出錯，也花了我蠻久的時間，透過這樣的反思，希望我以後能夠想出更縝密的邏輯，讓我做實驗時能夠更加的順利。