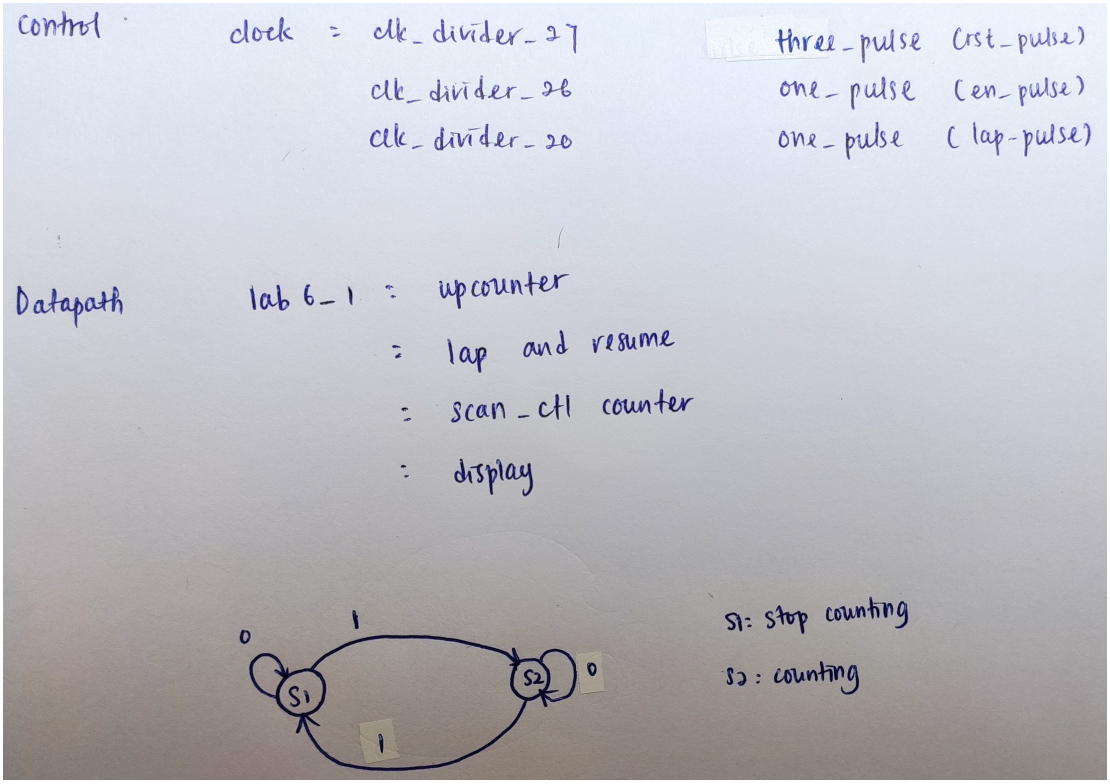
Lab 6 Report

Design Specification

Lab6_1:

clk_divider_27 as a 1Hz clock divider.
clk_divider_26 as a 2Hz clock divider.
clk_divider_20 as a 100Hz clock divider.
three_pulse generate a 2Hz reset function pulse.
one_pulse generate a 100Hz start or stop counting function pulse and a 100Hz lap function pulse.
upcounter with 1Hz clock divider will count up every second.
The seven segment display will shown current digit when lap function or stop function is triggered.
scan_ctl counter is a 20 bit counter, and generate a scan_ctl which is the 19th bit and 18th bit of the counter as a control signal for the seven segment decoder.
scan_ctl countrol which segment of the seven segment decoder to be used.
display use to display the digit of the counter.

Input: clk(100MHz clock), lap(push button for lap and reset function), en(push button to start and stop function)
Output: [3:0] DIGIT(seven segment decoder), [7:0] DISPLAY(segment to display on the seven segment decoder) , [15:0] led(16bit led)



Lab6_2:

fre_divider1 as a 1Hz clock divider.
fre_divider2 as a 10Hz clock divider.
fre_divider3 as a 100Hz clock divider.
fre_divider4 as a 2Hz clock divider.
debounce with 100Hz clock divider used to generate start_debounced, resume_debounced, min_debounced, sec_debounced.
one_pulse with 10Hz clock divider used to generate start_one_pulse, resume_one_pulse, min_one_pulse, sec_one_pulse.
three_pulse generate a 2Hz reset pulse.
fsm_start is the state of start and stop function.
fsm_resume is the state of resume and pause function.
bcd_upcounter with a 10Hz clock divider used to set the timer count down time.
Bcd_downcounter with a 1Hz clock divider used to counting down.
scan_ctl counter is a 16 bit counter, and generate a scan_ctl which is the 15^{th} bit and 14^{th} bit of the counter as a control signal for the seven segment display.
ssd_ctl which segment of the seven segment display to be used.
display use to display the digit of the counters.

Input: clk(100MHz clock), set_min(push button for setting minutes for timer) , set_sec(push button for setting seconds for timer), start(push button for start or stop function), resume(push button for resume or pause and reset function), timer_ctl(switch for timer setting)
Output: [3:0] ssd_ctl(seven segment decoder), [7:0] seg(segment to display on the seven segment decoder), [15:0] led(16 bit led)

control

clk : fre_divider1 (1Hz)
fre_divider2 (10Hz)
fre_divider3 (100Hz)
fre_divider4 (2Hz)

fsm : fsm_start
fsm_resume

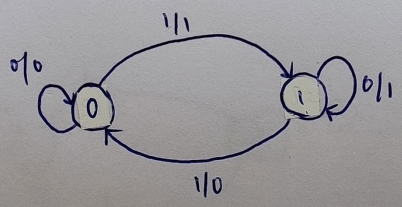
debounce (start_debounced, resume_debounced, min_debounced, sec_debounced)

one_pulse (start_one_pulse, resume_one_pulse, min_one_pulse, sec_one_pulse)

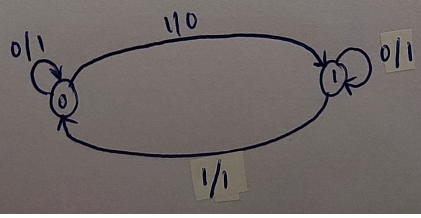
three_pulse (rst)

datapath

lab6_2 : bcd_upcounter
: bcd_downcounter
: scan_ctl counter
: Scan_ctl
: display



0 : stop counting
1 : counting



0: stop
1: start

Design Implementation

Lab6_1:

The clk_divider_27 will generate a 1Hz clock divider, called clk_div_27.
The clk_divider_26 will generate a 2Hz clock divider, called clk_div_26.
The clk_divider_20 will generate a 100Hz clock divider, called clk_div_20.

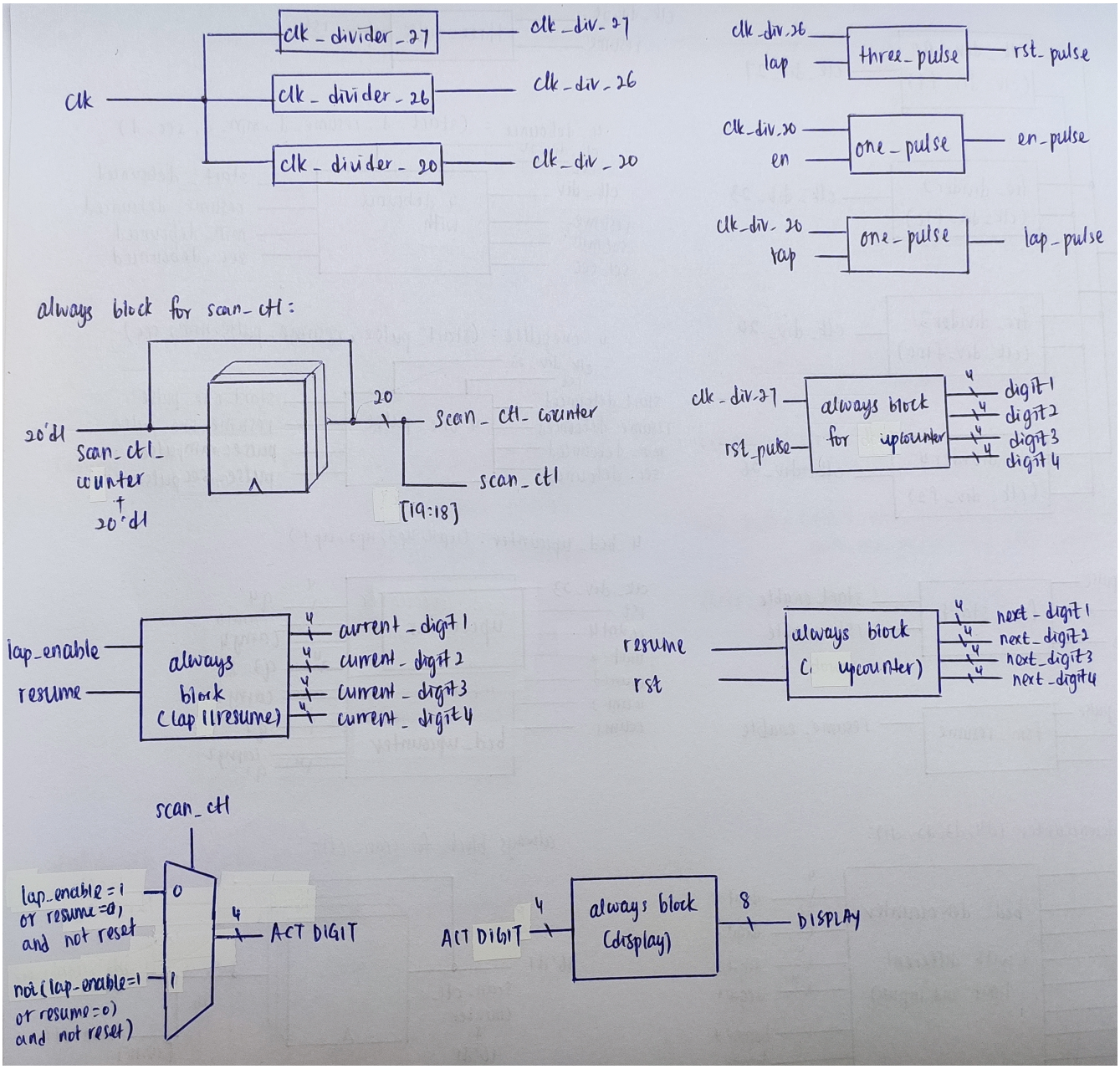
The three_pulse is connected to a 2Hz clock divider, it will generate a rst_pulse when the lap function button is pressed for 2 seconds.
The one_pulse is connected to a 100Hz clock divider, it will generate a en_pulse and a lap_pulse when the push button of lap function and resume function is pressed once.
The scan_ctl_counter always block is a 20 bit counter used for generate the scan_ctl for the seven segment decoder. The seven segment decoder will turn on according to the input from the scan_ctl counter.

The upcounter is designed when the resume(en_pulse) is 1, it will keep counting up, which is keep adding by 1. When the digit1 reached 9, digit2 will be added by 1; when digit1 reached 9 and digit2 reached 5, digit3 will be added by 1; when digit1 reached 9, digit2 reached 5, digit3 reached 9, digit4 will be added by 1. If all 4 digit1, digit2, digit3, digit4 reached 5959, the counter will start from 0000, and continue counting.

The lap function is done by a if condition, which is if lap function is triggered, the ACT_DIGIT will display current digit, else it will display digit(digit from the counter).

The led is designed as to differentiate when the lap function is triggered, the led will changed. The led is initially set as the leftmost bit will light as in the initial state and during counter mode. When the lap is pressed, the led lighted up will change to the 2nd rightmost led.

As a small conclusion, I designed my stopwatch by using the if statement, which is if the resume is 1, the counter will keep counting, else if the resume is 0, it will display current digit. Then if is lap, it will display current digit, else it will display the counter digit. And if the lap is pressed for 2 seconds long, the program will be reset, and can be used to count again if the resume is 1. And there is a led to differentiate the mode of the counter and when lap function is triggered.


clk
clk_divider_27
clk_divider_26
clk_divider_20
clk_div_27
clk_div_26
clk_div_20
clk_div_26
lap
three_pulse
rst_pulse
clk_div_20
en
one_pulse
en_pulse
clk_div_20
lap
one_pulse
lap_pulse
always block for scan_ctl:
20'd1
scan_ctl_counter
20'd1
20
scan_ctl_counter
scan_ctl
[19:18]
clk_div_27
rst_pulse
always block for upcounter
digit1
digit2
digit3
digit4
lap_enable
resume
always block (lap||resume)
current_digit1
current_digit2
current_digit3
current_digit4
resume
rst
always block (upcounter)
next_digit1
next_digit2
next_digit3
next_digit4
scan_ctl
lap_enable=1 or resume=0, and not reset
not(lap_enable=1 or resume=0) and not reset)
0
1
4
ACT DIGIT
ACT DIGIT
always block (display)
8
DISPLAY

Lab6_2:

The fre_divider1 will generate a 1Hz clock divider, called clk_div_27.
The fre_divider2 will generate a 10Hz clock divider, called clk_div_23.
The fre_divider3 will generate a 100Hz clock divider, called clk_div_20.
The fre_divider4 will generate a 2Hz clock divider, called clk_div_26.

The three_pulse with a 2Hz clock divider will generate a rst pulse, in which when the resume is pressed for 2 seconds long, the program will be reset.

There are 4 different debounce used in the program. The input of the 4 different decounces are the 4 push buttons, which is start, resume, set_min, set_sec. The start button is used to start and stop the counting program, and the resume is for resume or pause the program, when they are pressed for once. The set_min and set_sec push buttons are used to set the time for counting down, when they are pressed for once, the digit increased by one.
The 4 different one_pulse are connected to these 4 different decounces to generate the 4 different pulse, which are start_one_pulse, resume_one_pulse, min_one_pulse, and sec_one_pulse.

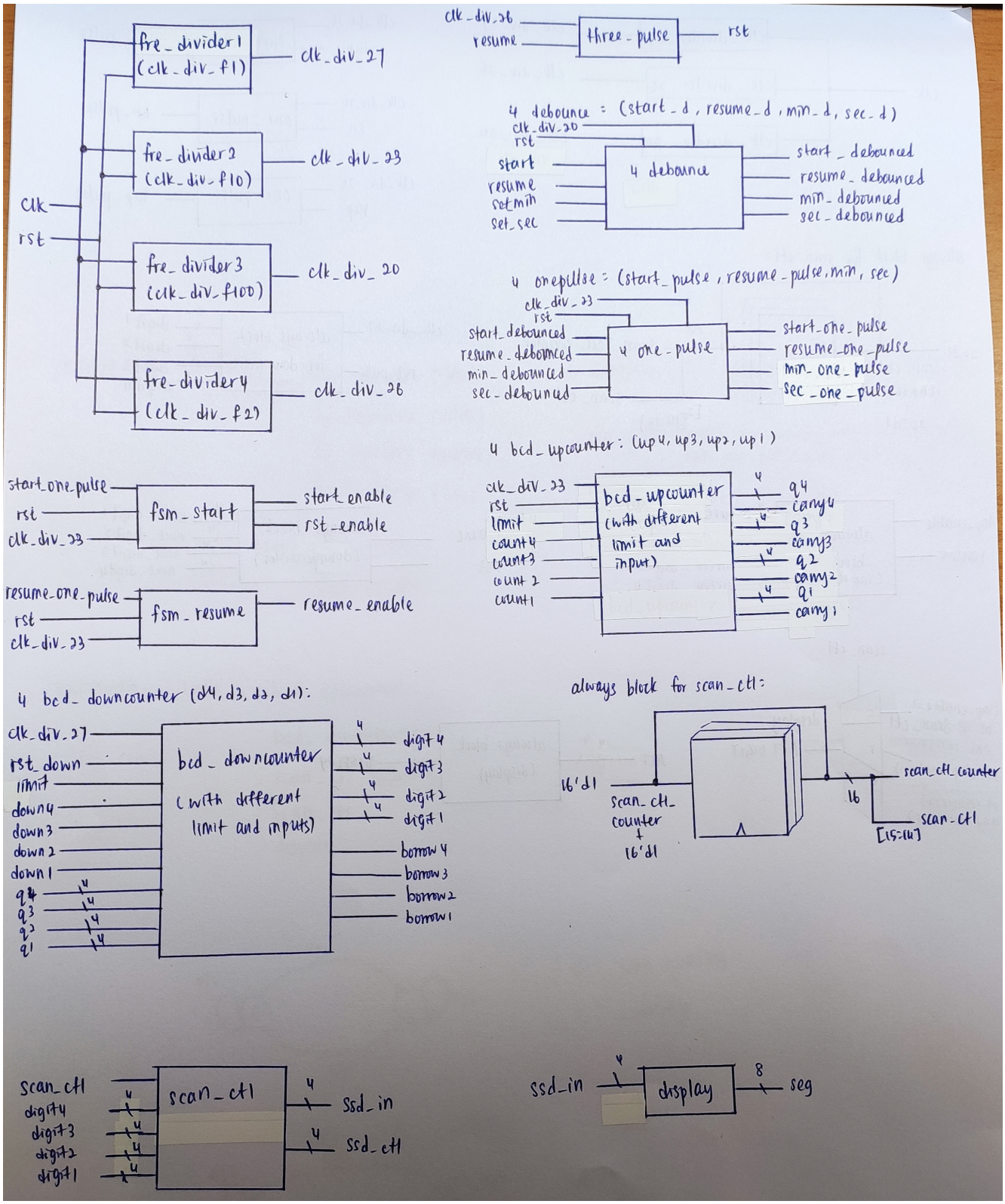
There are 2 fsm designed for this program, 1 is the fsm for the start and stop push button, and the other one is the resume and pause push button.

The bcd_upcounter is used when the set_min and set_sec is triggered. And these 2 set_min and set_sec can be used if the timer_ctl is 1. So, this mean, the bcd_upcounter needed the timer_ctl to trigger it for upcounting. Then timer_ctl is 1, the set_min and set_sec can be pressed, and the upcounter will start counting up according to the number of set_min and set_sec are pressed. Then, the timer_ctl will need to set 0, which is turn off the switch in order to start counting down.
The limit for the limit for q1 and q3 is the same, which is 9; and the limit for q2 and q4 is the same, which is 9. This mean, when they are pressed up to the limit, then the next digit will be added by 1(eg. Now is q1, pressed up to 9, then q2 is the next digit, it will be added by 1).

The bcd_downcounter is used when the timer_ctl is 0, and the time(set_min and set_sec) is set. Then it will receive the time and start counting down. The counter will keep decreasing by 1 until 0, and then start counting from the limit. The limit for digit1 and digit3 is the same, which is 9; the limit for digit2 and digit4 is the same, which is 5. When the digit decrease up to their limit, they will borrow a 1 from the next digit, then next digit will be decreased by 1, and the digit will count down from the limit(eg. Now digit1 is count down to 0, then digit2 is the next digit, it will be decreased by 1, digit1 will start counting down from 9 again).

The scan_ctl_counter always block is a 16 bit counter used for generate the scan_ctl control signal for the seven segment decoder.

scan_ctl will turn on the seven segment decoder according to the scan_ctl control signal. The display is used to display the segment according to the ssd get from the scan_ctl, which is the digits of the counter.



Discussion

From this lab, I had learn how to build a stopwatch with lap function and a timer which can set the time required and start count down.
The timer is quite challenged because it has more function, which is set the time, and start counting from the time set, and stop it and reset.
The lap function did went wrong at the first because of the reset function. The clock divider is not used correctly, and the lap function cannot function correctly. After fixing the reset by connect to a 2Hz clock divider and three pulse, the program was corrected.

Conclusion

Learn how to build a stopwatch and timer. Getting more familiar with concept of the up and down counter. Implementation of the debounce and one pulse and the display seven segment decoder become more clearer.

