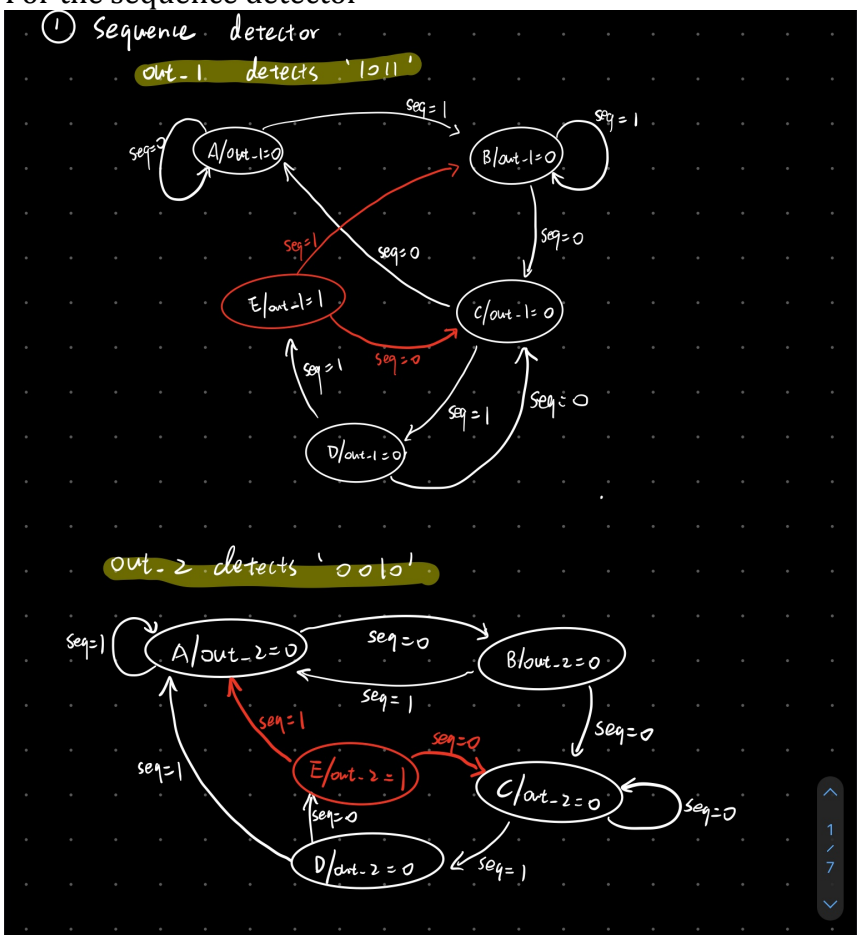
ECSE 222 VHDL Assignment 6

Yutong Wang 260839723

Hongtao Xu 260773785

1. Why is it better to use two FSMs, rather than one, in the implementation of the sequence detector from Section 3?
   - Because when we use two FSMs for sequence detector out_1 and out_2, the two FSMs can work at the same time so the result of out_1 and out_2 can update at approximately the same time. If we use one single FSM, then the circuit will be more complicate and there may be delay between the two outputs out_1 and out_2.
2. Briefly explain your VHDL code implementation of all circuits
   - For the sequence detector



The above are the two states maps for the two FSMs. The first one is for the '1011' sequence detector, the second one is the '0010' sequence detector.
We can see both of the two states maps have 5 states, and the output is '0' for state A, B, C, D. Output is '1' for state E.

So in the VHDL code, we first define 5 different states for both '1011' detector and '0010' detector. Then we define two signals (Current_state and next_state) and use them to represent the relationship between states.

Then use a process block, since reset is active low, so if reset = 0 then output = 0, and use another if statement to let signal Current_State <= next_state when enable = 1 and rising_edge(clk)

Then use another process block to describe the states diagram. Last we connect the output out_1 and out_2 to the signal current_state. When Current_state is at state E, then output is 1, when current_state is at other states, output = 0

- For the sequence counter

② Sequence Counter

The sequence counter is implemented using the sequence and two 3-bits counter, one counts for out_1, the other counts of out_2.

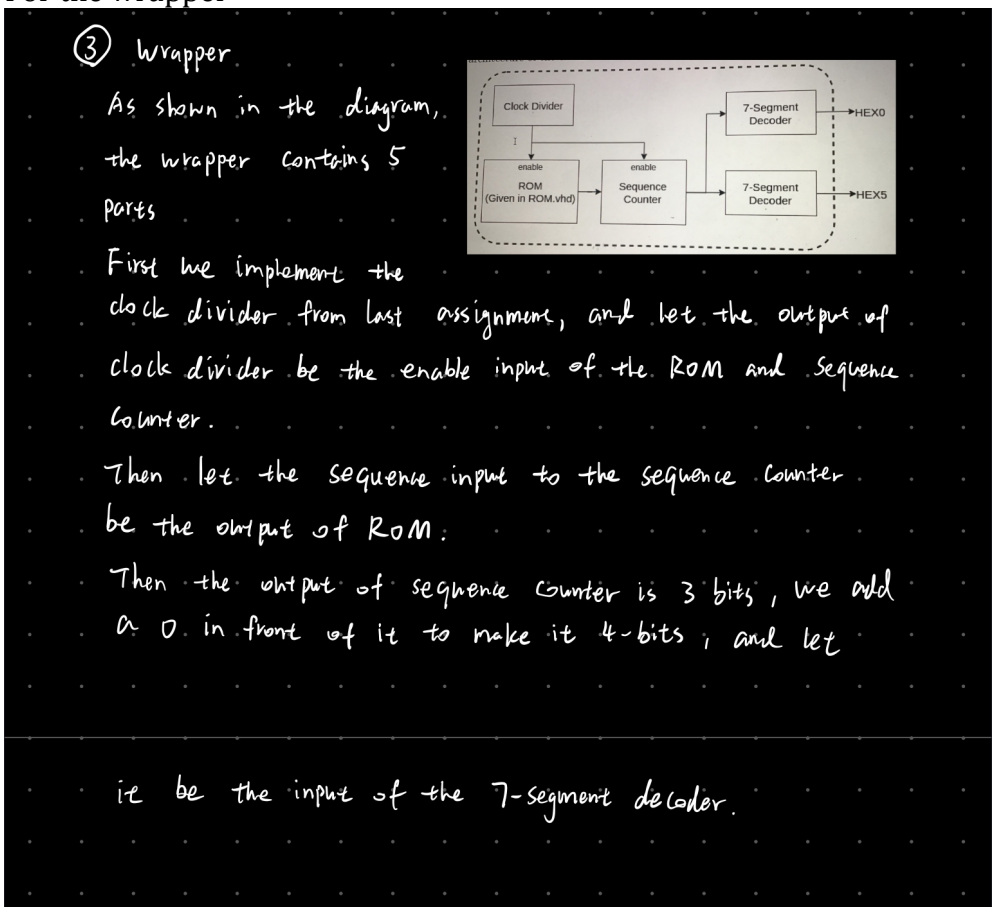
In addition to the VHDL code of the sequence detector, we define two more signals, count_signal_one and count_signal_two they are both integer range from 0 to 7.

Since the output increase by 1 everytime the current_state reaches state E, so we use the when statement and write: when st_one_E => count_signal_one <= count_signal_one + 1

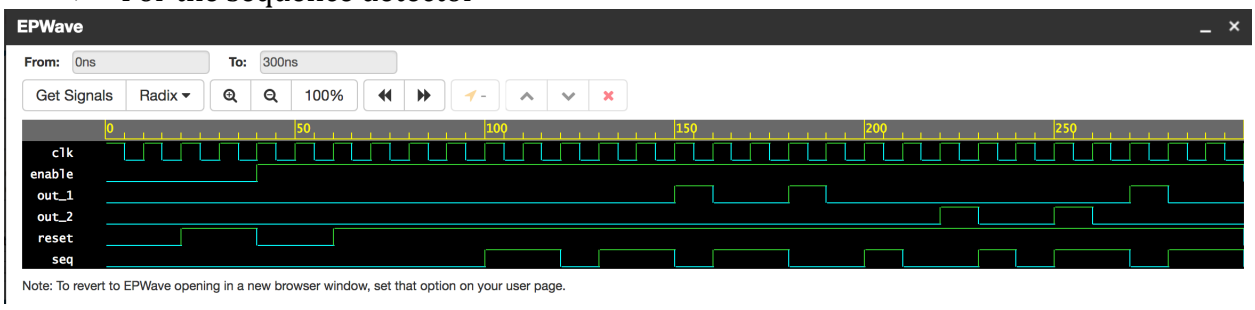
Also when the current_state is at states A, B, C, D, the output remain it's previous value.
So we write: when others => count_signal_one <= count_signal_one

Last we relate the output cnt_1 and cnt_2 to the signals count_signal_one and count_signal_two by using to_unsigned.

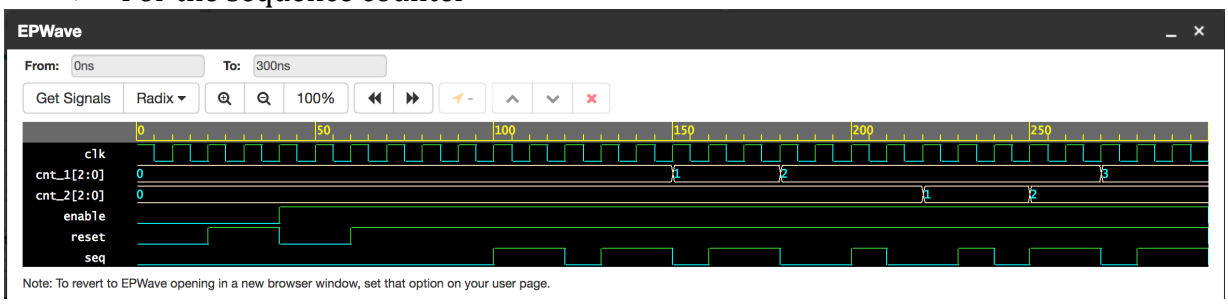
- For the wrapper



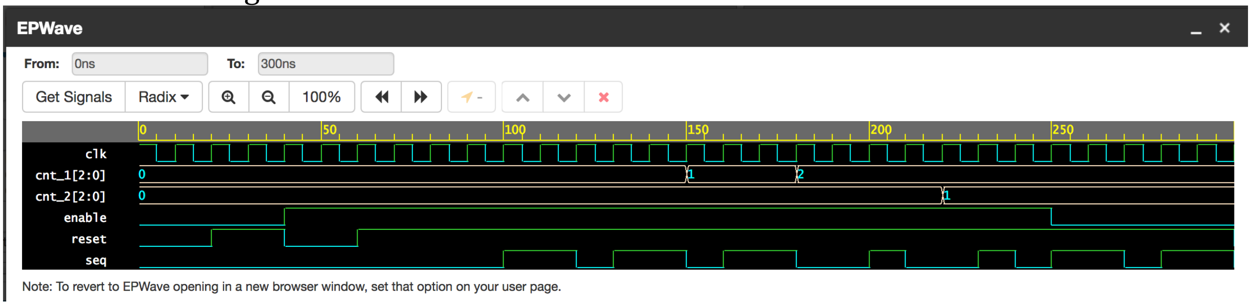
3. Provide waveforms for each of the individual circuits (each section) and for the wrapper.
   - For the sequence detector



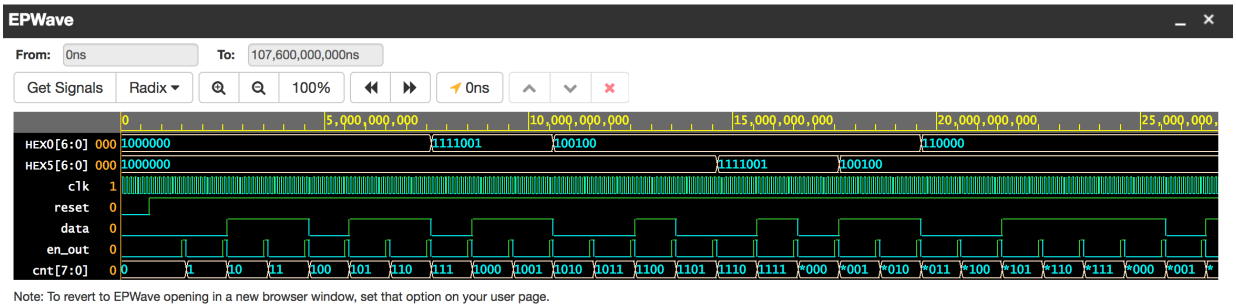
   - For the sequence counter

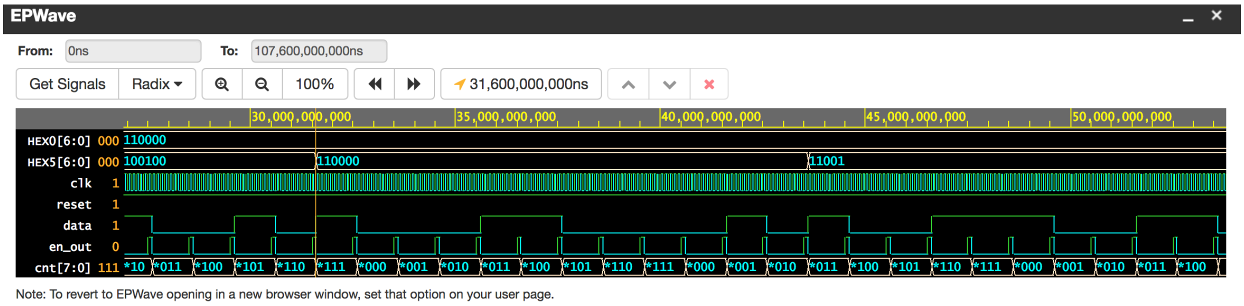


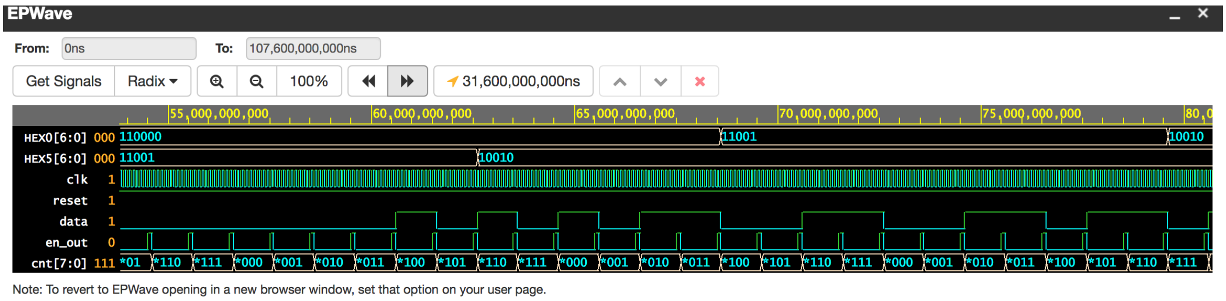
Test for enable signal:

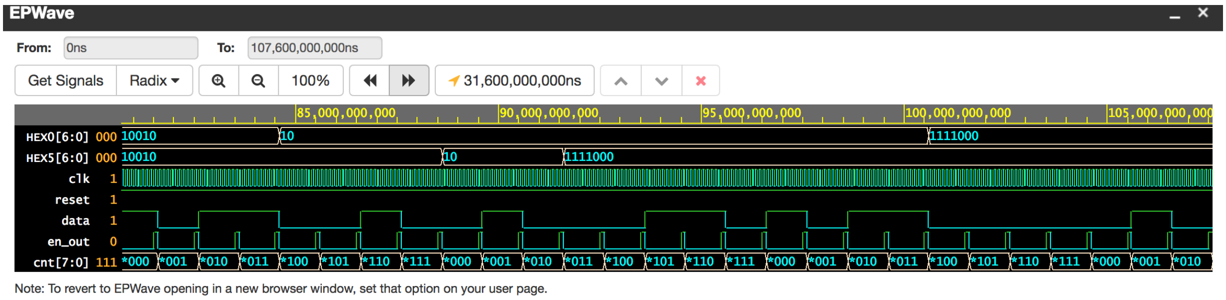


- For the wrapper









- When I increase the runtime of the wrapper code, the Epwave can only display up to 107,600,000,000 ns. So the screenshots above

don't include the full ROM output sequence, but I think they are able to show that my VHDL code for the Wrapper works properly.

4. Perform timing analysis of the wrapper and find the critical path(s) of the circuit. What is the delay of the critical path(s)?

Log | Share

```
# Info:                    CTE Path Report
# Info: Critical path #1, (path slack = 17.614):, Logic Levels = 3
# Info: SOURCE CLOCK: name: clk period: 20.000000
# Info:      Times are relative to the 1st rising edge
# Info:   DEST CLOCK: name: clk period: 20.000000
# Info:      Times are relative to the 2nd rising edge
# Info: NAME                          GATE      DELAY    ARRIVAL DIR  FANOUT LEVEL
# Info: reg_q(0)/C                 FDCE                  0.000   up
# Info: reg_q(0)/Q                 FDCE        0.456     0.456   up
# Info: cnt(0)                     (net)       0.393                 6     0
# Info: ix39222z60357/I5           LUT6                  0.849   up
# Info: ix39222z60357/O            LUT6        0.124     0.973   up
# Info: nx39222z7                  (net)       0.333                 1     1
# Info: ix39222z1571/I3            LUT6                  1.306   up
# Info: ix39222z1571/O             LUT6        0.124     1.430   dn
# Info: nx39222z3                  (net)       0.432                 8     2
# Info: ix3034z1316/I0             LUT2                  1.862   dn
# Info: ix3034z1316/O              LUT2        0.124     1.986   dn
# Info: nx3034z1                   (net)       0.333                 1     3
# Info: reg_current_state_two(0)/D FDPE                  2.319   dn
# Info:                  Initial edge separation:     20.000
# Info:                  Source clock delay:      -    1.692
# Info:                  Dest clock delay:        +    1.692
# Info:                                           -----------
# Info:                  Edge separation:             20.000
# Info:                  Setup constraint:        -    0.067
# Info:                                           -----------
# Info:                  Data required time:          19.933
# Info:                  Data arrival time:       -    2.319   ( 35.71% cell delay, 64.29% net delay )
# Info:                                           -----------
# Info:                  Slack:                       17.614
# Info: End CTE Analysis ..... CPU Time Used: 0 sec.
```

   - The critical path is from reg_q(0)/C to reg_current_state_two(0)/D
   - Delay of the path = 20 –slack = 20 – 17.614 = 2.386 s
5. Report the number of pins and logic modules used.

	Wrapper
Logic Utilization (in LUTs)	Used: 39 Avail: 63400
Total pins	Used: 16 Avail: 210