Digital Systems Lab Project

EE23BTECH11224 -Sri Krishna Prabhas Yadla

EE23BTECH11212 - Manugunta Meghana Sai*

1 Explaining clock.v

In the beginning we used the variable *stand* to initialize all the variables. A *delay* variable is introduced to align the system clock with the usual second. We estimated the delay to be around 12000000 clock cycles. In the verilog code we used 6 variables of reg data type to track each digit of the clock. The variables one_sec,ten_sec,$min9$,$min5$,$hour9$,$hour2$ are used to represent the unit digit of seconds,tens digit of seconds, unit digit of minutes, tens digit of minutes, unit digit of hours and tens digit of hours.

24 hour clock

We used a series of if else statements to ensure that after 60 seconds we get a minute, and after 60 minutes we get an hour. When the variable one_sec reaches 9, there is an increment in the variable ten_sec and variable one_sec is initialized to 0. When variable ten_sec reaches 5, $min9$ variable gets incremented and variables ten_sec and one_sec are set to zero and they begin incrementing from 0. When the variable $min9$ reaches 9, there is an increment in the variable $min5$, along with variable ten_sec,$min9$ and variable one_sec is initialized to 0 and the count again begins. When variable $min5$ reaches 5, $hour9$ variable gets incremented and variables ten_sec,$min5$,$min9$ and one_sec are set to zero and they begin incrementing from 0. Finally when $hour9$ variable reaches 9 there is an increment in $hour2$ variable and variables $hour9$,ten_sec,$min5$,$min9$ and one_sec are set to zero. This goes on till we reach the time $23 : 59 : 59$ after the clock resets to zero.

12 hours clock

For generating a 12 hours clock, we noticed that only the hours need to be changed. The time till 12:59:59 pm remains same as the 24 hour clock. So, to get time in 12 hour format we wrote a condition that if the variable $hour2$ in 24 hours clock is equal to 1 and the value stored in $hour9$ is greater than 2, then $hour2$ variable is made 0 and we subtract 2 from $hour9$ variable. This ensures that there is a change in the format for all the hours ranging from 13 to 19. For the remaining we used a condition that if the hour2 variable is equal to 2 and variable $hour9$ is greater than 2, then subtract 2 from $hour9$ variable and make $hour2$ variable equal to 0. This satisfies for hours 22,23 only. For conversion of 20 and 21 hours we used another condition stating that if the $hour9$ variable is less than 2 then, make hour2 0 and add 8 to hour 9. This ensures that 20 and 21 are converted to 8 and 9 respectively. Whenever there is a change in the hours, we ensured that variable pm is set to 1. Also when hours is 12, pm=1. We used an if statement to ensure that the red led flickers when pm is equal to 1.

Timer

We use a series of if else statements to make the time initially set to zero. If the one_sec variable is zero, then ten_sec is decremented by 1 and one_sec is set to zero. Else one_sec variable is set to 9.If ten_sec variable is also 0, then we go to next variable $min9$, if it is non-zero, then we decrement it and set one_sec and ten_sec to 9 and 5 respectively. If this is also zero,we go ahead and check $min5$. If it is non-zero, then we decrement it and set one_sec,$min9$ and ten_sec to 9,9 and 5. Else we move on to the next variable $hour9$ and once again check if it is nonzero, if so decrement it by 1, else we set one_sec,$min9$,$min5$ and ten_sec to 9,9,5 and 5 respectively. When all are 0 except $hour2$, we move on to the final variable $hour2$ and decrement it if it is nonzero and set the rest of the variables to their upper limits. This process repeats multiple times until all the 6 variables are zero. When this happens the buzzer variable is made 1. The duration of the buzzer is controlled by variables delay_t and counter_alarm. This is done to ensure that there is no continuous buzzer sound.

Alarm

Here when all the 6 variables are equal to the time initially set, the buzzer is set to 1. We control the duration of the buzzer by the variables counter_alarm. The buzzer rings for 10 seconds and then is set to zero.

Timer case is different from that of alarm since time changes and when buzzer becomes 1 it cant be made again till next alarm condition but in timer it remains 0 till we are in that state. So for timer, we had to use condition at $00:00:01$ but buzzer should ring at $00:00:00$ hence we make 1s delay using delay_t.

Switches/Buttons

fsm_1_1, fsm_2_1, fsm_3_1 are variables which are used to detect sequence 0,1,1,1,...,1,0. This is overlapping sequence. This is used to know if there the switches are turned on and off(or buttons pressed). fsm_1_1 corresponds to switch_1, fsm_2_1 to switch_2, fsm_3_1 to switch_3. If the required sequence is detected we get the corresponding variable to be in state 2'b11.

When fsm_1_1 = 2'b11, move the cursor to left. The cursor starts from rightmost(at one_sec).

When fsm_2_1 = 2'b11, then we increase the value where cursor is present by 1. Once maximum value of that place is achieved, it goes to 0 when we further increase. We also make sure we don't have undesired hour values such as 24,25,etc.

When fsm_3_1 = 2'b11, we change the modes. 3'b000 is 24hr clock, 3'b001 is 12hr clock, 3'b010 is timer, 3'b011 is alarm.

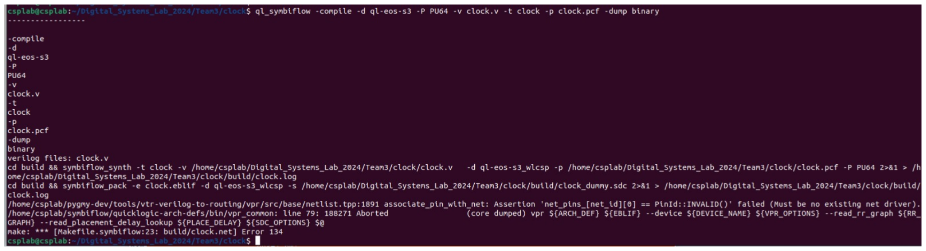
Display

We have used the 6 variables $ten\_sec\_, min5\_, hour2\_, one\_sec\_, min9\_, hour9\_$ to store the ASCII hexadecimal values of the 6 variables one_sec,min9,hour9, ten_sec,min5,hour2. These values are then fed to the FSM nibs. The variable i is used to navigate through the states and variable count is used for delay purposes.

Challenges

We were unable to give input to the Vaman board. In order to change the modes and values of the digits of the clock,we thought of taking inputs from switches, but had problem in tracing the movement of switches. We wrote two different approaches for it. One was directly using posedge. Other approach was to write an FSM to detect the sequence 0,1,1,...,1,0 and then subsequently change the modes.

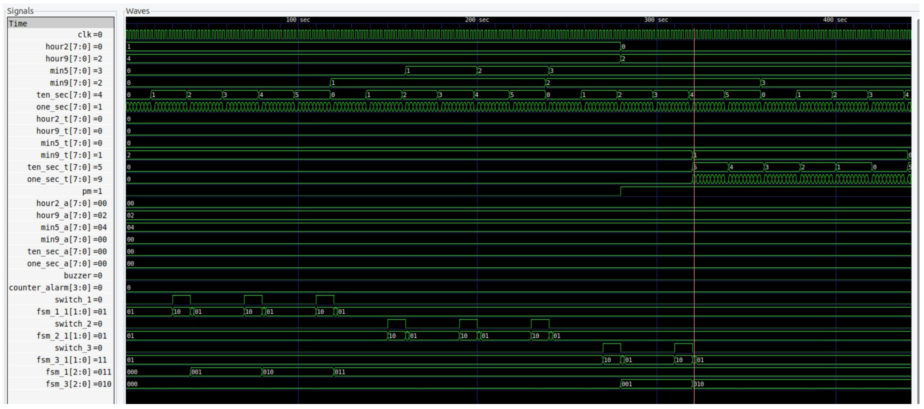
We are getting errors when we try to give inputs, it shows errors such as invalid pins, core dumped.

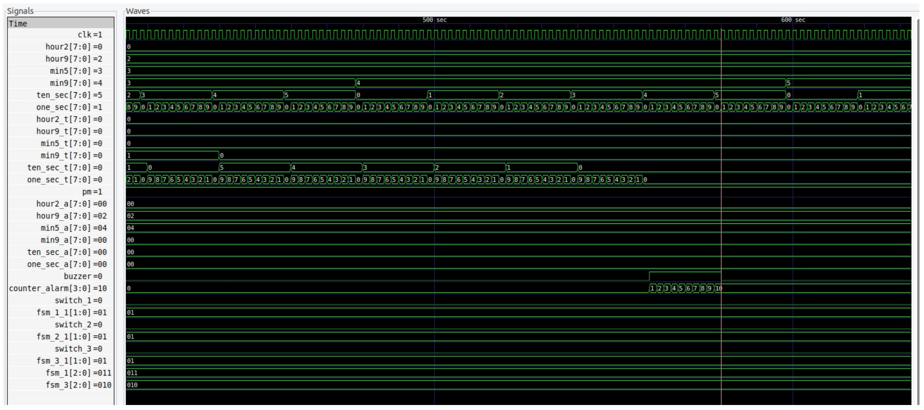


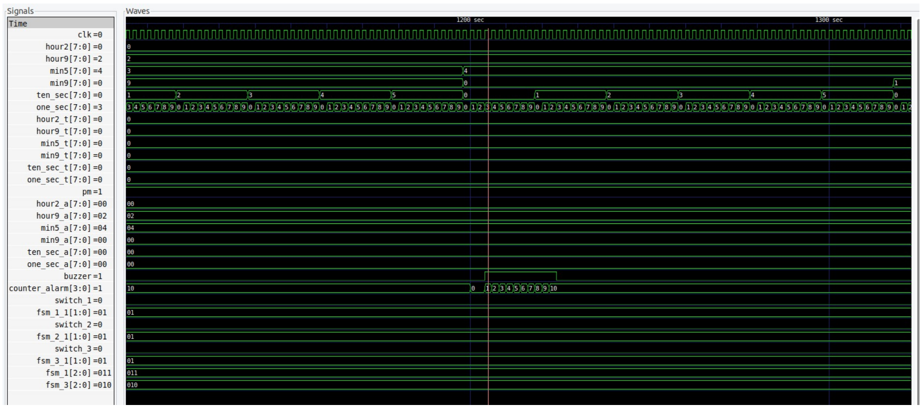
```
csplab@csplab:~/Digital_Systems_Lab_2024/Team3/clock$ ql_symbiflow -compile -d ql-eos-s3 -P PU64 -v clock.v -t clock -p clock.pcf -dump binary
-----------------

-compile
-d
ql-eos-s3
-P
PU64
-v
clock.v
-t
clock
-p
clock.pcf
-dump
binary
verilog files: clock.v
cd build && symbiflow_synth -t clock -v /home/csplab/Digital_Systems_Lab_2024/Team3/clock/clock.v   -d ql-eos-s3_wlcsp -p /home/csplab/Digital_Systems_Lab_2024/Team3/clock/clock.pcf -P PU64 2>&1 > /h
ome/csplab/Digital_Systems_Lab_2024/Team3/clock/build/clock.log
cd build && symbiflow_pack -e clock.eblif -d ql-eos-s3_wlcsp -s /home/csplab/Digital_Systems_Lab_2024/Team3/clock/build/clock_dummy.sdc 2>&1 > /home/csplab/Digital_Systems_Lab_2024/Team3/clock/build/
clock.log
/home/csplab/pygmy-dev/tools/vtr-verilog-to-routing/vpr/src/base/netlist.tpp:1891 associate_pin_with_net: Assertion 'net_pins_[net_id][0] == PinId::INVALID()' failed (Must be no existing net driver).
/home/csplab/symbiflow/quicklogic-arch-defs/bin/vpr_common: line 79: 188271 Aborted                 (core dumped) vpr ${ARCH_DEF} ${EBLIF} --device ${DEVICE_NAME} ${VPR_OPTIONS} --read_rr_graph ${RR_
GRAPH} --read_placement_delay_lookup ${PLACE_DELAY} ${SDC_OPTIONS} $@
make: *** [Makefile.symbiflow:23: build/clock.net] Error 134
csplab@csplab:~/Digital_Systems_Lab_2024/Team3/clock$
```

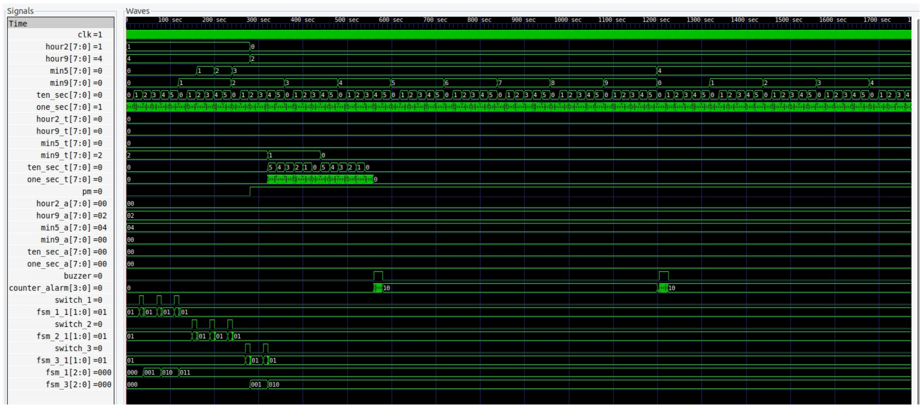
Figure 1: Error in generating bin file

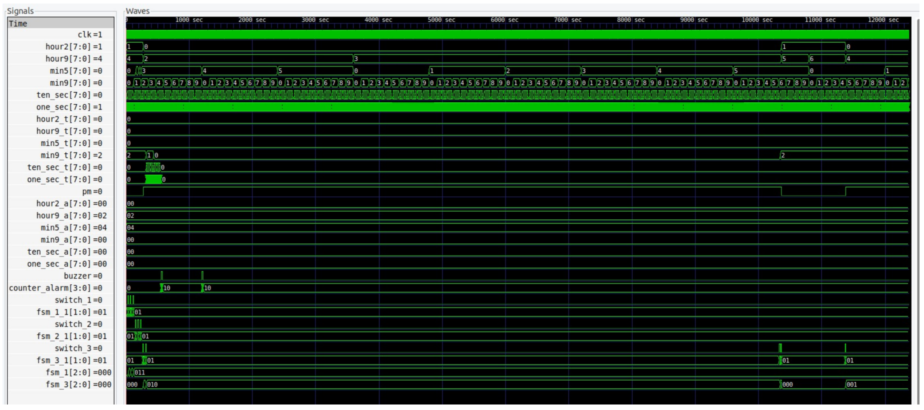
When we tried to run it in iverilog and simulate the results using testbench, we are getting desired outputs but we are unable to generate bin file for this code. For simulation we commented out the part where we give clk in macro and instead gave clk as input, also we comment out delay. The testbench code along with waveform.vcd, out.vvp is attached. The following are the waveform results, for these we have used initial block instead of variable stand as delay since initial block is not synthesizable and is just used to see how it works.


Signals
Waves
Time
100 sec
200 sec
300 sec
400 sec
clk=0
hour2[7:0]=0
hour9[7:0]=2
min5[7:0]=3
min9[7:0]=2
ten_sec[7:0]=4
one_sec[7:0]=1
hour2_t[7:0]=0
hour9_t[7:0]=0
min5_t[7:0]=0
min9_t[7:0]=1
ten_sec_t[7:0]=5
one_sec_t[7:0]=9
pm=1
hour2_a[7:0]=00
hour9_a[7:0]=02
min5_a[7:0]=04
min9_a[7:0]=00
ten_sec_a[7:0]=00
one_sec_a[7:0]=00
buzzer=0
counter_alarm[3:0]=0
switch_1=0
fsm_1_1[1:0]=01
switch_2=0
fsm_2_1[1:0]=01
switch_3=0
fsm_3_1[1:0]=11
fsm_1[2:0]=011
fsm_3[2:0]=010


Signals
Waves
Time
500 sec
600 sec
clk=1
hour2[7:0]=0
hour9[7:0]=2
min5[7:0]=3
min9[7:0]=4
ten_sec[7:0]=5
one_sec[7:0]=1
hour2_t[7:0]=0
hour9_t[7:0]=0
min5_t[7:0]=0
min9_t[7:0]=0
ten_sec_t[7:0]=0
one_sec_t[7:0]=0
pm=1
hour2_a[7:0]=00
hour9_a[7:0]=02
min5_a[7:0]=04
min9_a[7:0]=00
ten_sec_a[7:0]=00
one_sec_a[7:0]=00
buzzer=0
counter_alarm[3:0]=10
switch_1=0
fsm_1_1[1:0]=01
switch_2=0
fsm_2_1[1:0]=01
switch_3=0
fsm_3_1[1:0]=01
fsm_1[2:0]=011
fsm_3[2:0]=010


Signals
Waves
Time
1200 sec
1300 sec
clk=0
hour2[7:0]=0
hour9[7:0]=2
min5[7:0]=4
min9[7:0]=0
ten_sec[7:0]=0
one_sec[7:0]=3
hour2_t[7:0]=0
hour9_t[7:0]=0
min5_t[7:0]=0
min9_t[7:0]=0
ten_sec_t[7:0]=0
one_sec_t[7:0]=0
pm=1
hour2_a[7:0]=00
hour9_a[7:0]=02
min5_a[7:0]=04
min9_a[7:0]=00
ten_sec_a[7:0]=00
one_sec_a[7:0]=00
buzzer=1
counter_alarm[3:0]=1
switch_1=0
fsm_1_1[1:0]=01
switch_2=0
fsm_2_1[1:0]=01
switch_3=0
fsm_3_1[1:0]=01
fsm_1[2:0]=011
fsm_3[2:0]=010


Signals
Waves
Time
clk=1
hour2[7:0]=1
hour9[7:0]=4
min5[7:0]=0
min9[7:0]=0
ten_sec[7:0]=0
one_sec[7:0]=1
hour2_t[7:0]=0
hour9_t[7:0]=0
min5_t[7:0]=0
min9_t[7:0]=2
ten_sec_t[7:0]=0
one_sec_t[7:0]=0
pm=0
hour2_a[7:0]=00
hour9_a[7:0]=02
min5_a[7:0]=04
min9_a[7:0]=00
ten_sec_a[7:0]=00
one_sec_a[7:0]=00
buzzer=0
counter_alarm[3:0]=0
switch_1=0
fsm_1_1[1:0]=01
switch_2=0
fsm_2_1[1:0]=01
switch_3=0
fsm_3_1[1:0]=01
fsm_1[2:0]=000
fsm_3[2:0]=000


Signals
Waves
Time
clk=1
hour2[7:0]=1
hour9[7:0]=4
min5[7:0]=0
min9[7:0]=0
ten_sec[7:0]=0
one_sec[7:0]=1
hour2_t[7:0]=0
hour9_t[7:0]=0
min5_t[7:0]=0
min9_t[7:0]=2
ten_sec_t[7:0]=0
one_sec_t[7:0]=0
pm=0
hour2_a[7:0]=00
hour9_a[7:0]=02
min5_a[7:0]=04
min9_a[7:0]=00
ten_sec_a[7:0]=00
one_sec_a[7:0]=00
buzzer=0
counter_alarm[3:0]=0
switch_1=0
fsm_1_1[1:0]=01
switch_2=0
fsm_2_1[1:0]=01
switch_3=0
fsm_3_1[1:0]=01
fsm_1[2:0]=000
fsm_3[2:0]=000

2 Explaining clock24.v

This is same as that of 24hr section of clock.v. But this worked since we have no input but only output. It just counts from 00 : 00 : 00. If we want we can initialize it to whatever value we want inside the stand delay part and get desired output. We have attached a video and also .pcf file.

3 Explaining timer.v

Similar to clock24 but just decrement instead of increment by using if else if conditions. Whenever boundary condition we make lower ones to max value and decrease the value of rightmost non-zero number by 1 and go on. We initialised timer to start from 00 : 01 : 30. If we want we can initialize it to whatever value we want inside the stand delay part and get desired output. Once it becomes 00 : 00 : 00, the buzzzer rings. We have attached a video and also .pcf file.

Note: We got .bin files for both timer.v and clock24.v but not for clock.v, the display also works as expected for timer and clock24.

4 Circuit