Lab 4 – Iterative Divider

Name: Len Quach

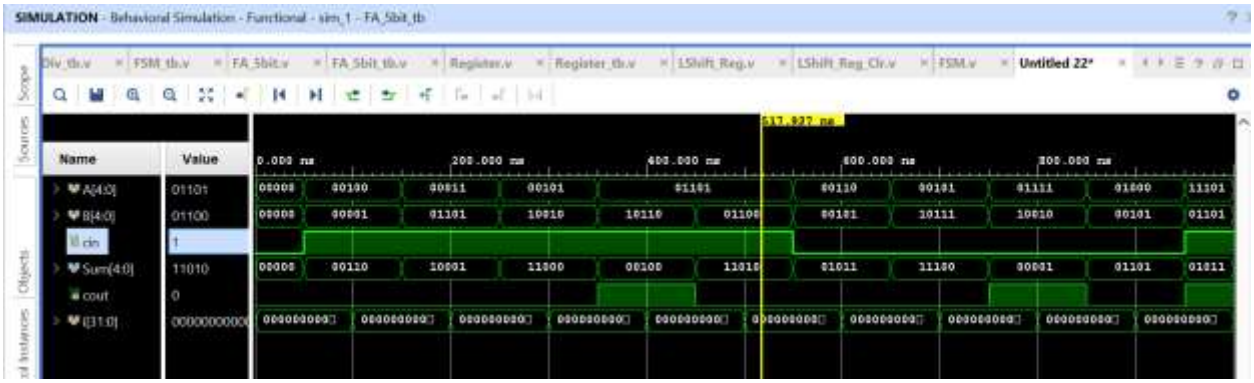
ID: 026440113

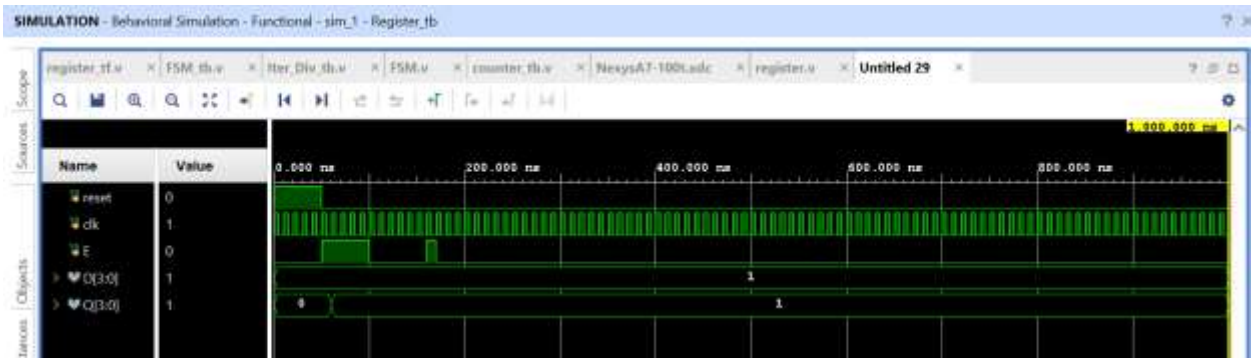
We are using a Finite State to model sequential logic that transits among a finite number of internal states. FSM designs with the least possible amount of states to perform the function. The transitions through states occur as functions of the present state of the machine and the current inputs, and the outputs at any state are functions of the present state or functions of the present state and present inputs.

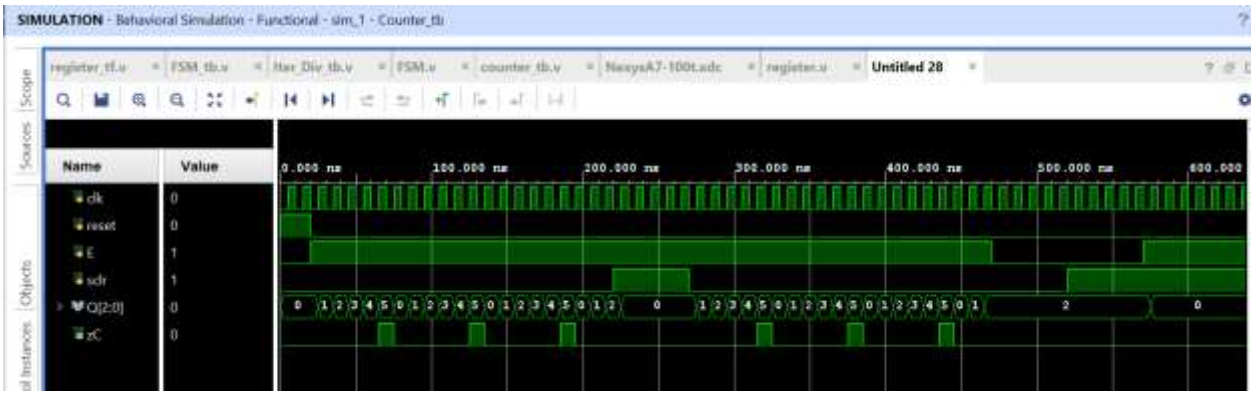
In this lab, we will implement an iterative divider circuit with inputs A and B are unsigned numbers. A (6-bit wide) is the dividend and B (4-bt wide) is the divisor. That will produce a quotient Q and remainder R. A clearable left shift register stores the value of R to hole the remainder. If E = 1, a division will start. Inputs A and B will be captured, after every clock cycle the circuit will shift in the next bit of A or shift in the next bit of A and subtract B. If the division operation is complete, the output signal done = 1, and the values of Q and R are valid.

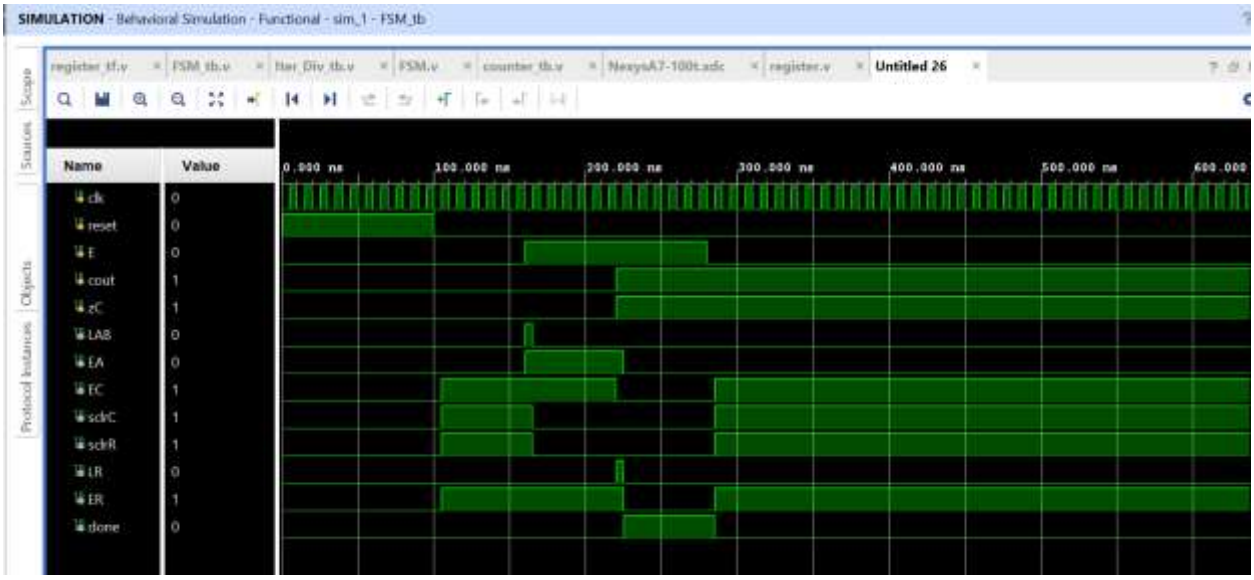
I added the two modules LShift_Reg.v and LShift_Reg_Clr.v from Beachboard to sotring the calculated values of Q and R. Then I created Register module to implement a 4-bit register with a load enable signal, then test it out with Register_tb. I also created FA_5bit to implement Full Adder circuit that adds two 5-bit inputs, and then verify the functionality by FA_5bit_tb. Next, Counter module will implement a modulo 6 counter, so the counter will count up to 5 before cycling again, then I perform the simulation to verify its functionality. FSM module will control the whole circuit, and I simulate the module to make sure that the output signals match the state

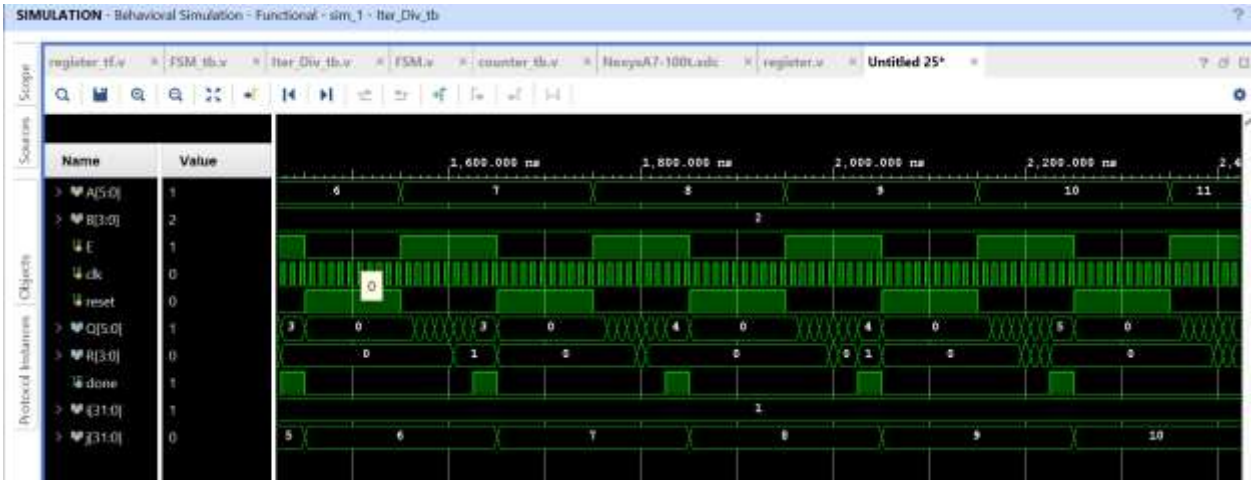
diagram at every state transition. Finally, Iter_Div will implement the given circuit by creating instances of all the design modules and making the interconnections. For this Iter_Div module, I use an extensive simulation to define the test cases and display the result of each test case.








SIMULATION - Behavioral Simulation - Functional - sim_1 - FSM_tb
register_tf.v
FSM_tb.v
Iter_Div_tb.v
FSM.v
counter_tb.v
NexysA7-100t.xdc
register.v
Untitled 26
Name
Value
clk
reset
E
cout
zC
LAB
EA
EC
sclrC
sclrR
LR
ER
done
0.000 ns
100.000 ns
200.000 ns
300.000 ns
400.000 ns
500.000 ns


SIMULATION - Behavioral Simulation - Functional - sim_1 - Iter_Div_tb
register_tf.v
FSM_tb.v
Iter_Div_tb.v
FSM.v
counter_tb.v
NexysA7-100t.xdc
register.v
Untitled 25*
Name
Value
A[5:0]
B[3:0]
E
clk
reset
Q[5:0]
R[3:0]
done
i[31:0]
j[31:0]
1,600.000 ns
1,800.000 ns
2,000.000 ns
2,200.000 ns


NEXYS A7
DIGILENT
100T