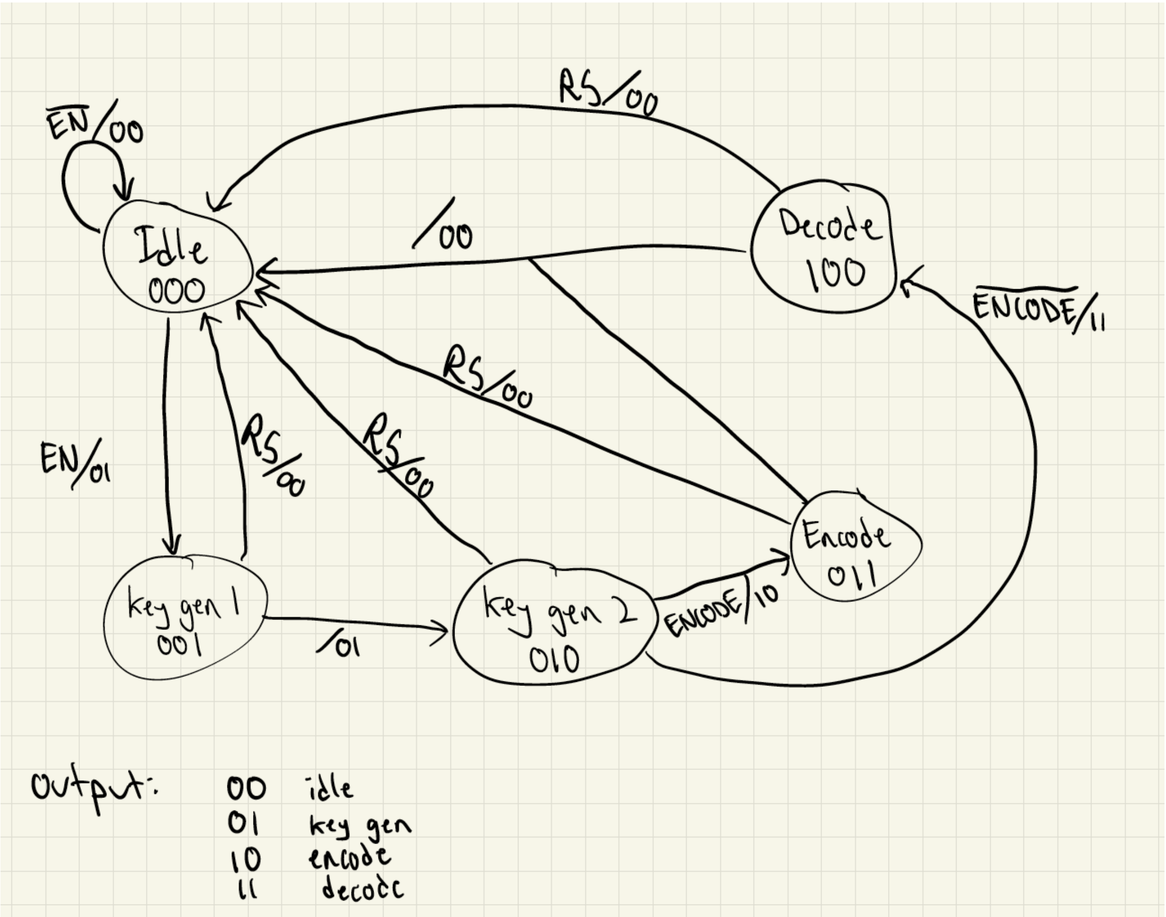


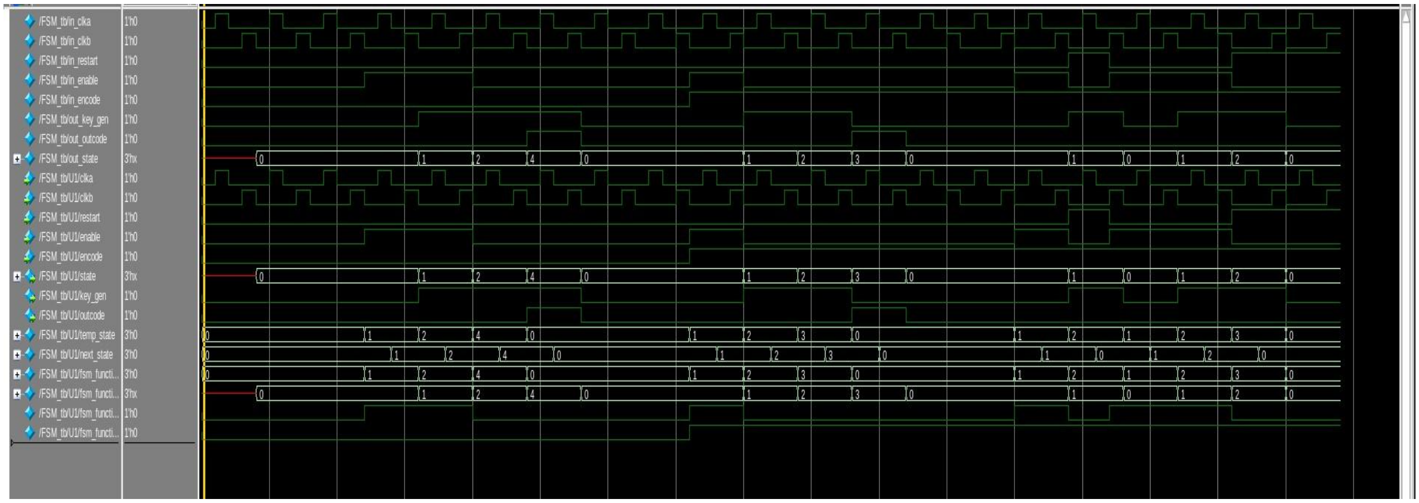
ELEC 422 FSM project
By: John Reko, Sean Hamilton, and Peter Humphreys

Our FSM will have 3 inputs: ENABLE, RESTART, and ENCODE. RESTART will act as a normal reset input, and bring the state back to idle from wherever it was. ENABLE will determine whether the state stays at idle or moves to the first round of key generation. From key gen 1, the state will move to key gen 2 regardless of the input, and once the second round of key generation is over, the ENCODE input will determine whether it moves to the encoder logic or the decoder logic. A high ENCODE signal will move to encoding, and low will move to decoding. From both of these states, the state moves back to idle. Each of the different phases of encryption will result in a different 2-bit output. At idle, the output will be 00, during key generation the output will be 01, during encoding the output will be 10, and during decoding the output will be 11. These outputs will tell the user which step in the process that the machine is in.

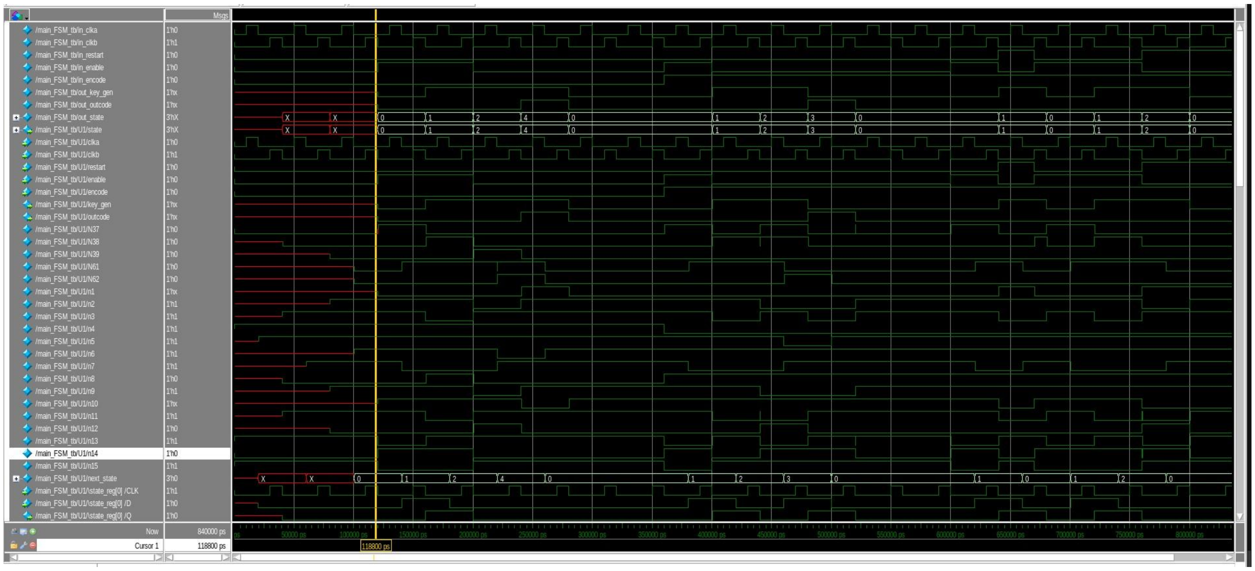
One other aspect of the design that we discussed was whether we will use a 16-bit input or a series of two 8-bit inputs. The current FSM will work for a 16-bit input, and there would be a couple slight changes to the beginning if we used two 8-bit inputs instead. We would use registers to store each of the 8-bit inputs, and only start the key generation process once both inputs have been inputted. In this design, storing data would only happen once the state is at key gen 1, and the state would loop back to itself until both registers have been filled. To keep track of what data has been stored, we would create a counter that increments each time we store data, and once the counter hits 2 then the actual key generation logic would begin.

We coded the FSM controller and here are the Modelsim results (the results match what we expected and images are attached in the assignment):

Pre DC:



Post DC:


Msgs
/main_FSM_tb/in_clka 1'h0
/main_FSM_tb/in_clkb 1'h1
/main_FSM_tb/in_restart 1'h0
/main_FSM_tb/in_enable 1'h0
/main_FSM_tb/in_encode 1'h0
/main_FSM_tb/out_key_gen 1'hx
/main_FSM_tb/out_outcode 1'hx
/main_FSM_tb/out_state 3'hX
/main_FSM_tb/U1/state 3'hX
/main_FSM_tb/U1/clka 1'h0
/main_FSM_tb/U1/clkb 1'h1
/main_FSM_tb/U1/restart 1'h0
/main_FSM_tb/U1/enable 1'h0
/main_FSM_tb/U1/encode 1'h0
/main_FSM_tb/U1/key_gen 1'hx
/main_FSM_tb/U1/outcode 1'hx
/main_FSM_tb/U1/N37 1'h0
/main_FSM_tb/U1/N38 1'h0
/main_FSM_tb/U1/N39 1'h0
/main_FSM_tb/U1/N61 1'h0
/main_FSM_tb/U1/N62 1'h0
/main_FSM_tb/U1/n1 1'hx
/main_FSM_tb/U1/n2 1'h1
/main_FSM_tb/U1/n3 1'h1
/main_FSM_tb/U1/n4 1'h1
/main_FSM_tb/U1/n5 1'h1
/main_FSM_tb/U1/n6 1'h1
/main_FSM_tb/U1/n7 1'h1
/main_FSM_tb/U1/n8 1'h0
/main_FSM_tb/U1/n9 1'h1
/main_FSM_tb/U1/n10 1'hx
/main_FSM_tb/U1/n11 1'h1
/main_FSM_tb/U1/n12 1'h0
/main_FSM_tb/U1/n13 1'h1
/main_FSM_tb/U1/n14 1'h0
/main_FSM_tb/U1/n15 1'h1
/main_FSM_tb/U1/next_state 3'h0
/main_FSM_tb/U1/\state_reg[0] /CLK 1'h1
/main_FSM_tb/U1/\state_reg[0] /D 1'h0
/main_FSM_tb/U1/\state_reg[0] /Q 1'h0
Now 840000 ps
Cursor 1 118800 ps
118800 ps