ECSE 323 — Group 47 Lab 4 Report Permutation

Jun Young Shin id. 260499663
Timothee Flichy id. 260557686

April 1, 2016

1 On-Chip Testing using the SignalTap II Logic Analyzer

During our on-chip testing of the enigma, when everything was well setup, the test output of the test bed when the reset key was pressed, we where only getting one portion of the entire cycle of the counter. The TA could not explain what was going on even though the VHDL code was working perfectly on the Altera board when we uploaded the code to it.

2 Permutation

The permutation is used to both encode and decode the input letter. To do so, we were given a table which depicts the character encryption list given in the figure 1. We were told to implement the first 4 rotor types. To make the permutation circuit, we need 2 input and 2 outputs. The inputs will receive a 2 bit rotor_type and the second input is a 5-bit input_code (letter) that must be encrypted. The outputs are a 5 bit output_code which outputs the encrypted version of the input bits determined by the rotor position and the second input is the inv_output_code giving the inverted or decrypted 5 bit code. This is given by the figure 2. We tested out vhdl code on the ModelSim

INPUT	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Rotor I	E	K	M	F	L	G	D	Q	V	Z	N	T	O	W	Y	H	X	U	S	P	A	I	B	R	C	J
Rotor II	A	J	D	K	S	I	R	U	X	B	L	H	W	T	M	C	Q	G	Z	N	P	Y	F	V	O	E
Rotor III	B	D	F	H	J	L	C	P	R	T	X	V	Z	N	Y	E	I	W	G	A	K	M	U	S	Q	O
Rotor IV	E	S	O	V	P	Z	J	A	Y	Q	U	I	R	H	X	L	N	F	T	G	K	D	C	M	W	B

Figure 1: Permutation graph.

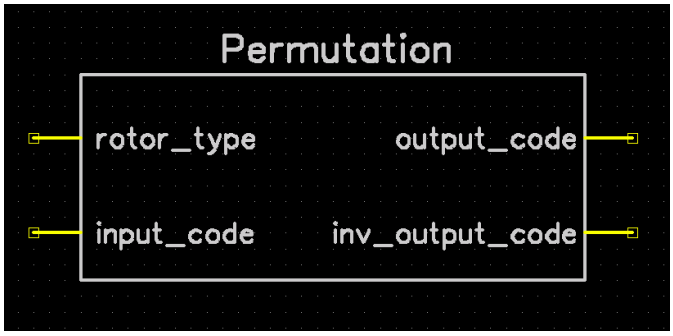

Figure 2: Permutation symbol.

and got what we were expected to get from the permutation graph as you can see from the figures 3, 4, 5, 6, and 7.

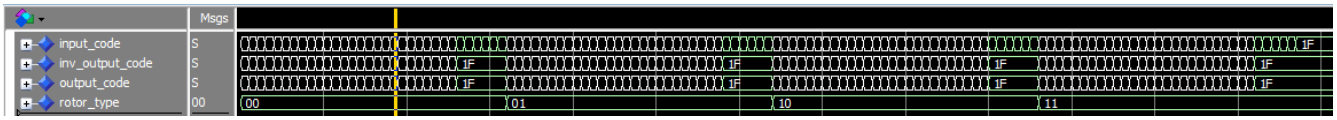

Figure 3: Tested circuit on ModelSim

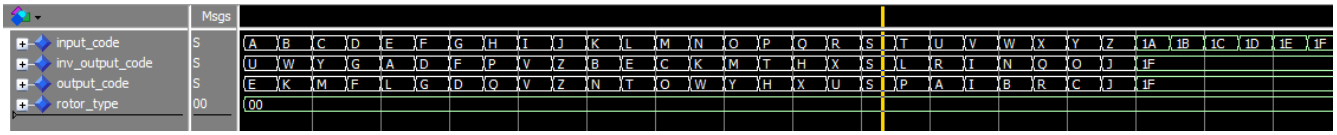

Figure 4: With rotor I.

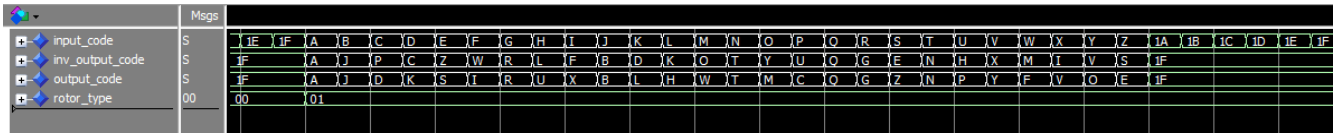

Figure 5: With rotor II.

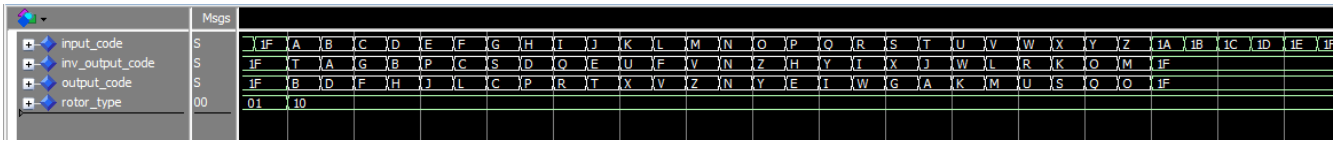

Figure 6: With rotor III.

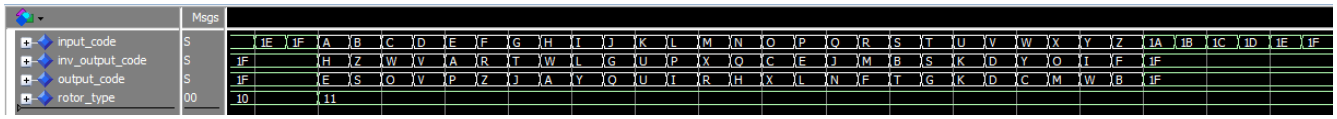

Figure 7: With rotor IV.

3 fsm

For the Finite State Machine circuit (FSM), we built a circuit that performed the requirements mentioned on the lab sheet. The states are as following shown in the figure 8. To make this work, we wrote a FSM where we have 5

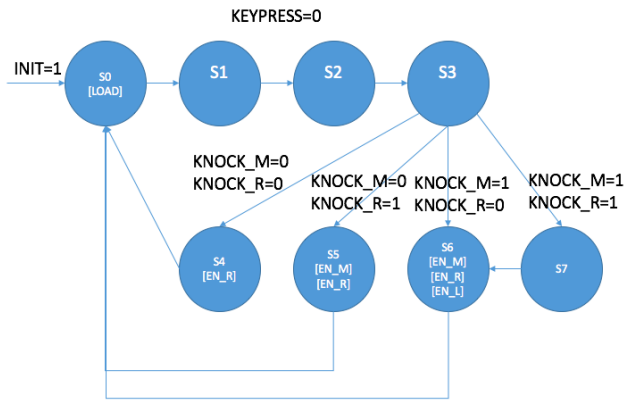

Figure 8: State Diagram.

inputs and 4 outputs as shown in the figure 9. Circuit goes as follows. The circuit checks for the INIT to go high.

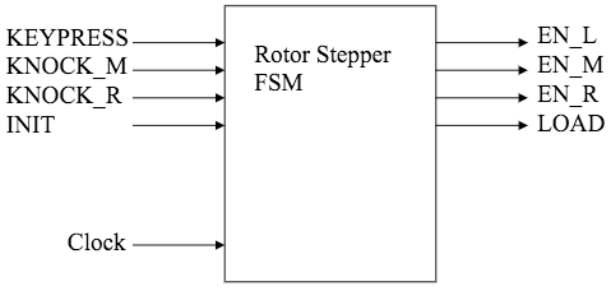

Figure 9: FSM symbol.

When INIT is high, the LOAD will go high. After this, it waits for the KEYPRESS to go low (the Altera board is active high). Next it checks the KNOCK_M and KNOCK_R. When both KNOCK values are low, then only the EN_R is set high. When only KNOCK_R is high, then the EN_R and EN_M is set high. When only KNOCK_M is high, the all EN is go high. When both KNOCK's are high, is will go to the state where all EN are high. After the states where EN are set, it will return to state 1 (where it waits) and start over again. At this point the INIT does not need to be checked. The figure 10 shows the circuit in a simulated environment. To the run the circuit,

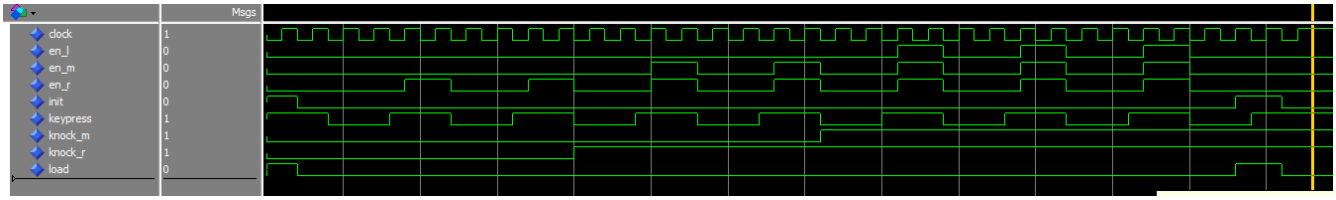

Figure 10: FSM symbol.

we have found in the slow model that Fmax=75.04MHz, Setup slack to be 3.337 ns, and hold time of 0.445 ns. In the fast model, setup time is 7.848 ns and hold time to be 0.215 ns.

4 fsm_testbed

For our testbed, using the necessary circuits, we have made the circuit to work as shown on figure 11

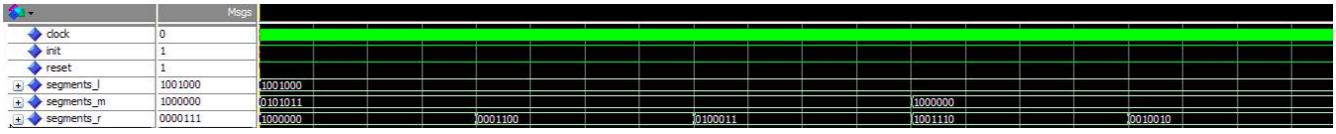

Figure 11: fsm_testbed outputs and inputs.