

Lab 4 README.pdf

Lab 4 Partners

Mark Lee, 36183168

Seyed-Ebrahim Mir-Mohammadsadeghi, 84477686

Break Down

1. Directory path of SOF and solution:

rtl/output_files/rc4.sof

rtl/output_files/Chain1.cdf

2. Status

Everything in the lab is working, and according to the provided instructions. There are some caveats, however, as we will discuss below.

Task 1:

- Write 00 to FF in a RAM block – Check

Task 2:

- Write a single decryption core – Check
- Decrypt messages based on switches for key – Check

Note: the code looks for an MIF file under “rtl/message.mif”, and not “rtl/secret_messages/...”

Task 3:

- Cycle from key 24'h000_000 to 24'hFFF_FFF or until code is cracked – Check
- Display key on hex display – Check
- Success/Failure LEDS – Check
- Check for valid characters – Check
 - The code immediately exits the loop upon an invalid character to save time
 - This is a superior method time-wise compared to completing the decryption and comparing each index at the end.

Task 3 With Bonus:

- Use 4-cores simultaneously – Check
- Custom LEDs that show which core completed the decryption – Extra

- Has a testbench “task3_bonus_TB.sv”

3. Annotated simulation screenshots as required by the lab

The screenshots are organized from top to bottom and in chronological order. These can be found lower in the document.

4. Information on how to run the simulations

The simulations use imitation RAM and ROM blocks written in Verilog, since the original RAM and ROM do not compile in ModelSim. The code for these imitation blocks were provided with permission by Justin Chang. This code is not used anywhere in the actual project.

For testbenching we used one massive testbench called task3_TB.sv.

5. Any additional information

Modules were created, and each has their own respective file. These files were called upon by the main “ksa.sv” file.

- “task1_fsm.sv” for task 1, the code is recycled in later stages, but not used directly
- “task2a_fsm_ebi_ver.sv” for tasks 2, 3, and 3 with bonus
- “task2b_fsm_crack.sv” for tasks 2, 3, and 3 with bonus
- “task2_fsm” which handles the task2a and task2b
- “decrypted_msg.v” for the ROM containing the output information
- “encrypted_msg.v” for the RAM containing the input information
- “task3_fsm.sv” which handles the task2 with an fsm of its own.
- “task3_bonus_fsm.sv” which handles the task2 with an fsm of its own.

There are rigorous comments in each code module that may be helpful for understanding the “what” & “why”s of the code.

Please note that comments containing “//&?&” is just a shorthand that I use to CTRL-F to areas in the professor’s code that requires adjustment.

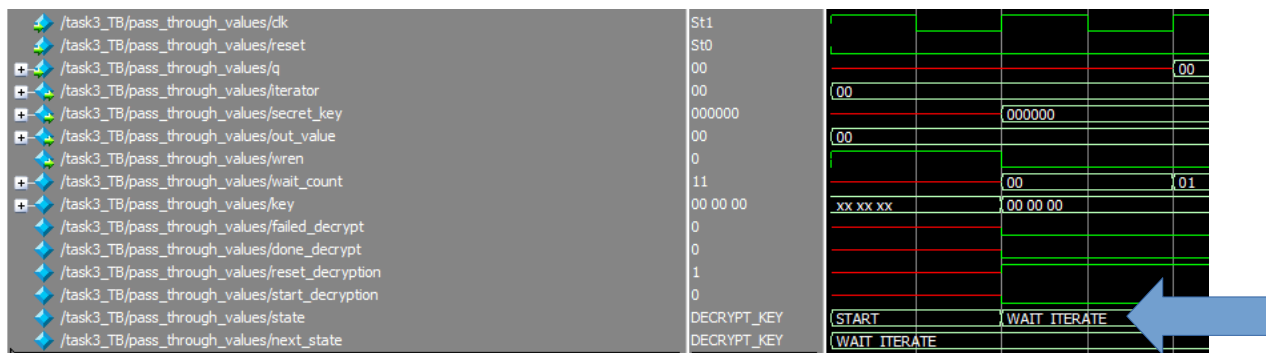
Annotated Screenshots

Each FSM here is shown.

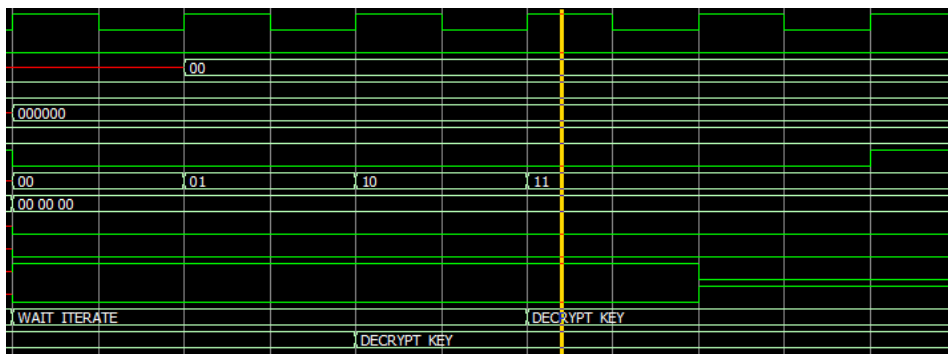
Most of these images are from the simulation of brute force on message 1, it has a small key, which makes simulating it being cracked on ModelSim far more tolerable on my computer.

task3_fsm

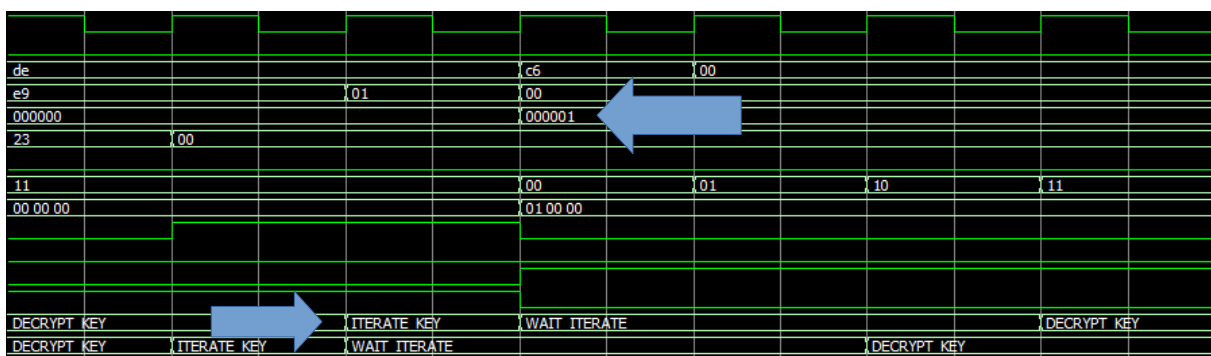
This FSM's order is "ITERATE_KEY", then "WAIT_ITERATE", "DECRYPT_KEY", and loop around until the process has been completed. However the first "ITERATE_KEY" state is skipped, so that the all zeros key is not ignored as a possibility.



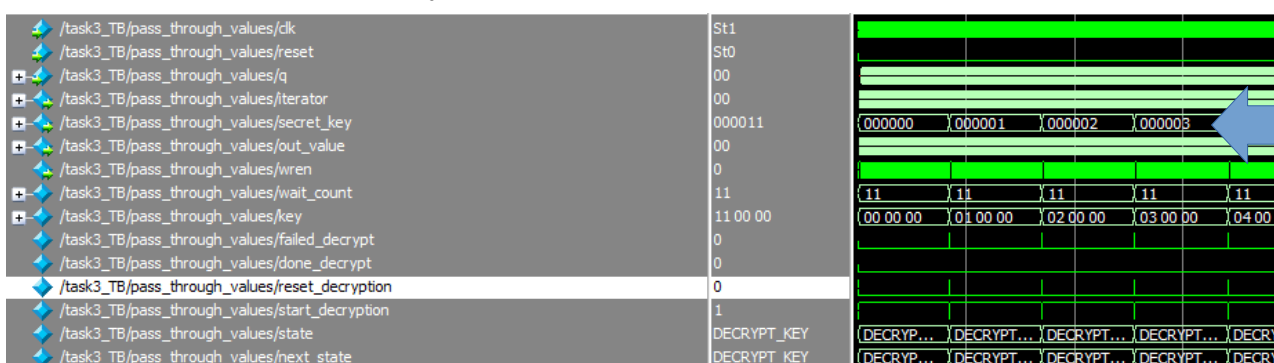
After "WAIT_ITERATE" comes the "DECRYPT_KEY" state, where all the subsequent FSMs get called. Naturally, this is the state that this FSM stays in the longest.



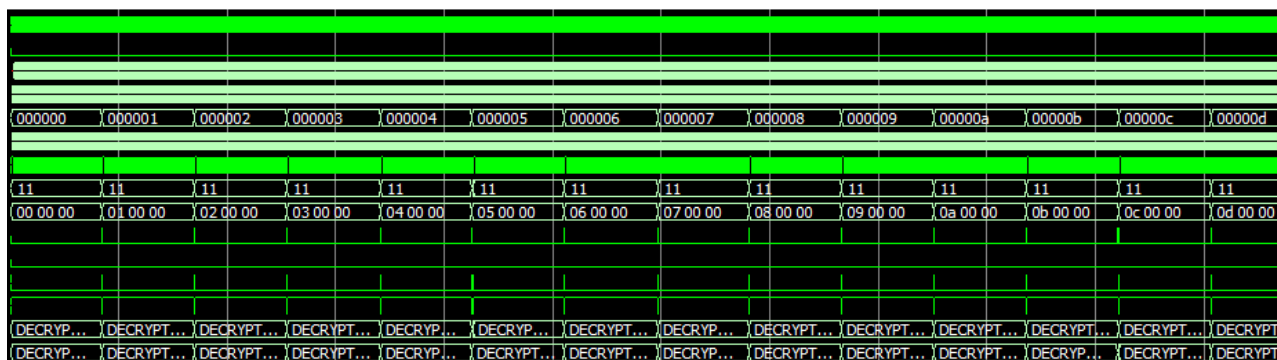
If the decryption fails for that particular key, we then iterate the key.



This overall process is continually repeated.

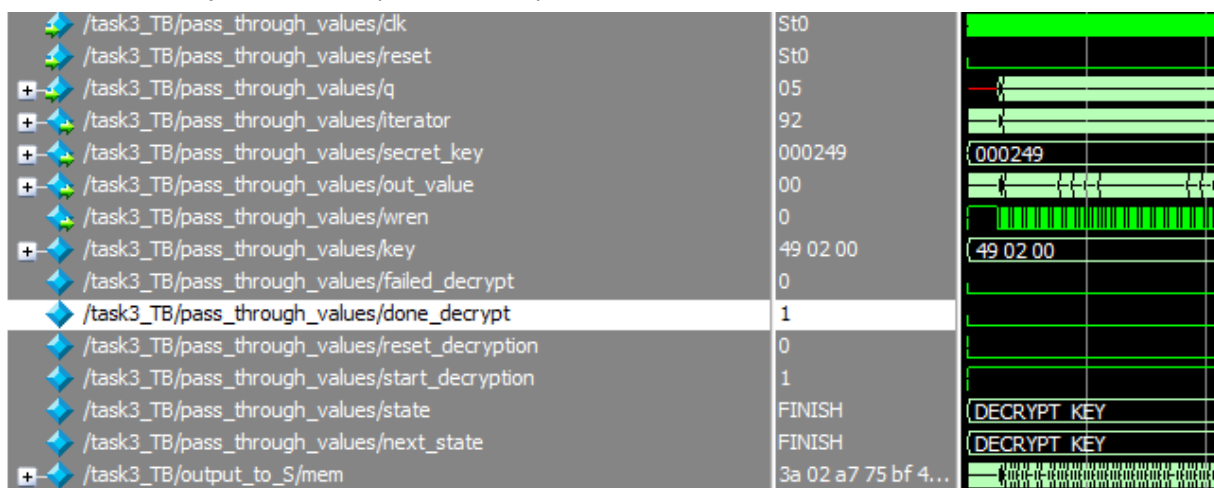


Over and over again.

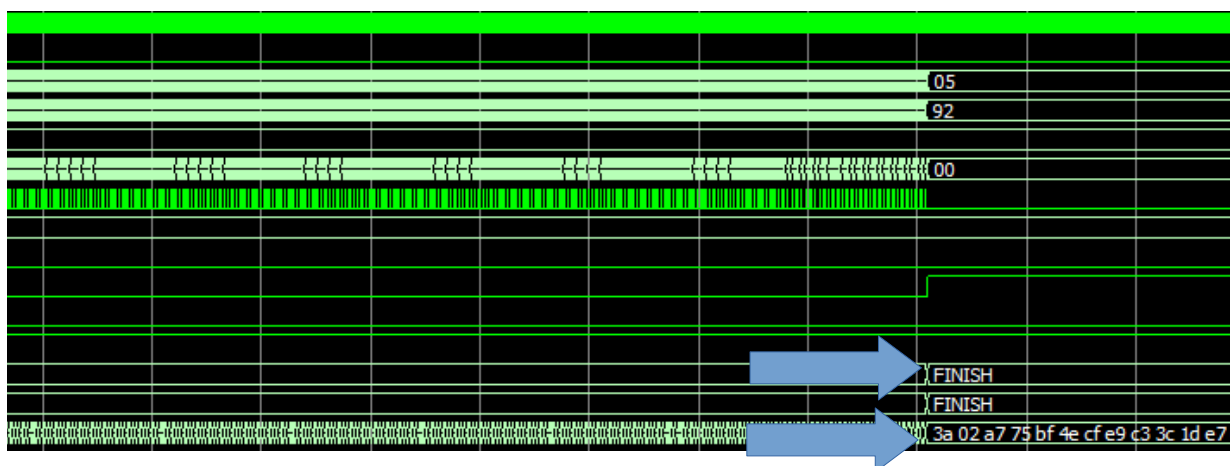


Let's now go back to the fact that this FSM simply manages others.

This FSM is a very simple one, immediately after starting, it calls the decryption core and makes it start with the key it provides (and iterates).



Once the decryption module (task2_fsm) completes its work, it sends a “done_decrypt” signal, and the task3_fsm enters its “FINISH” state, where RAM is suspended and unchanged.



task2_fsm

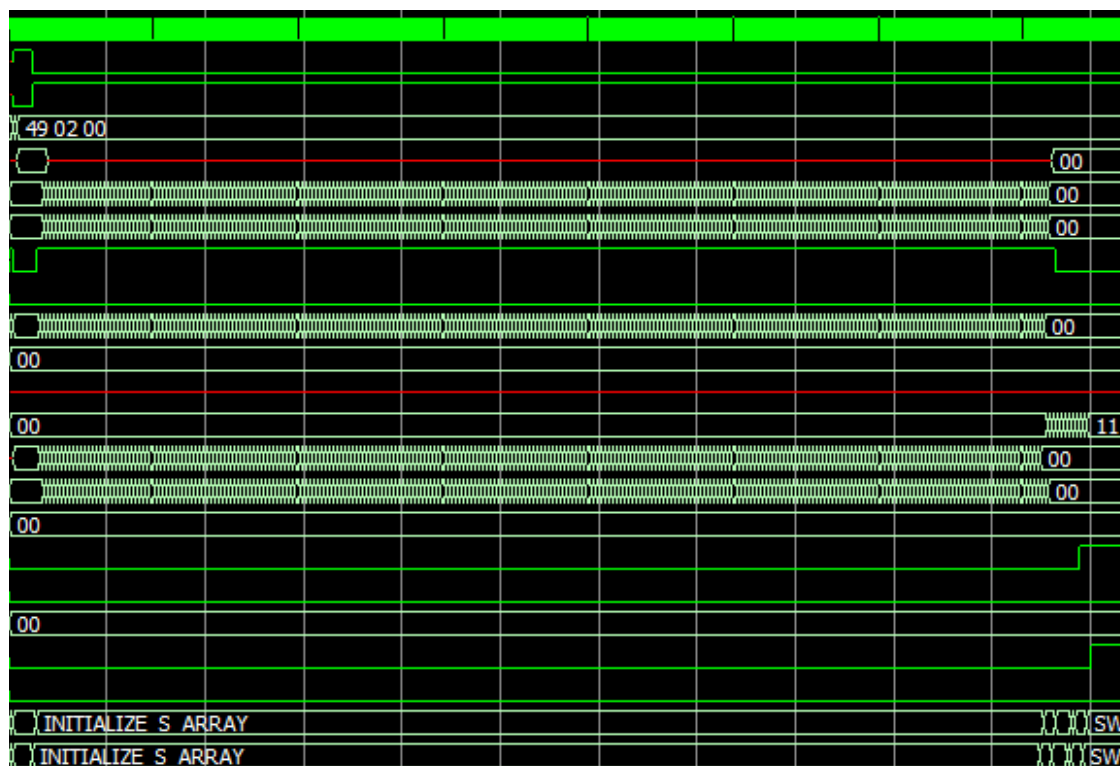
This is nominally labelled “fsm” but it is simply a collection of wires, regs, and muxes to handle the combination of task2a_fsm and the task2b_fsm. And does not have any states.

task2a_fsm

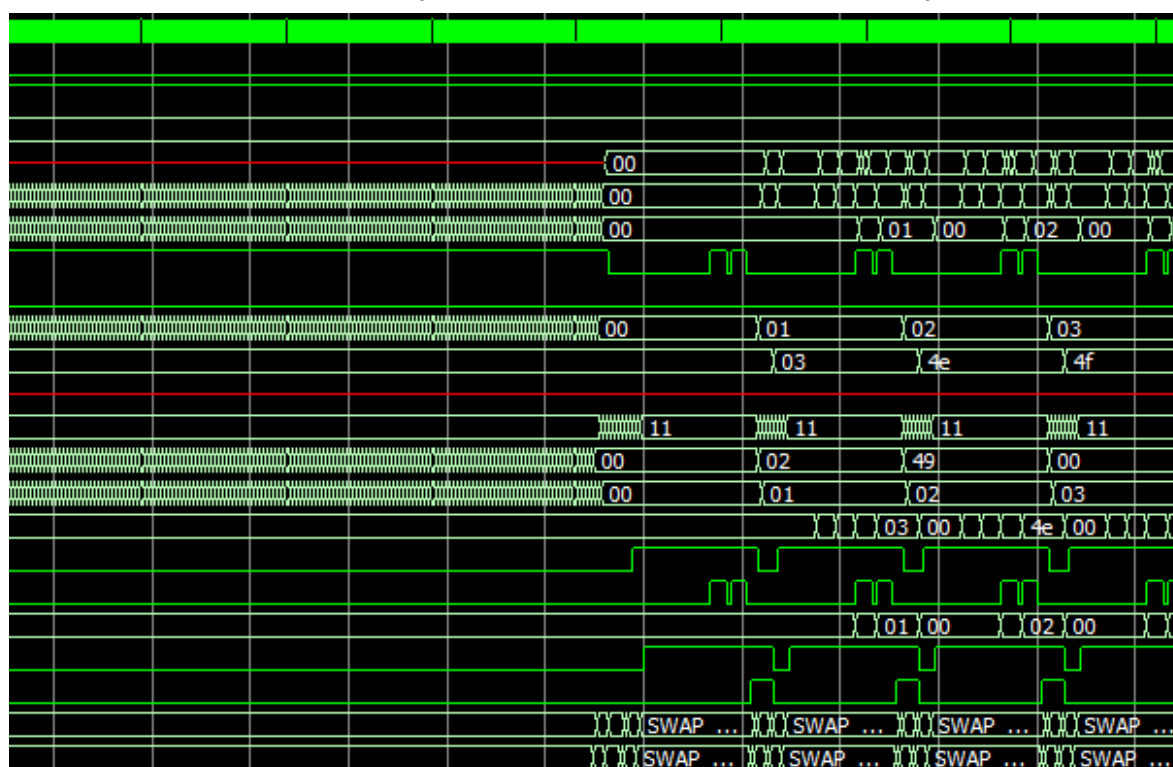
The task2a_fsm does not use iterator_k, the wire is from a slightly previous version of the code, which has been removed. The code has been tweaked for extra wires such as these.

/task3_TB/pass_through_values/decryption_module/FSM_1/clk	St0	
/task3_TB/pass_through_values/decryption_module/FSM_1/reset	St0	
/task3_TB/pass_through_values/decryption_module/FSM_1/start_FSM_1	St1	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/secret_key	49 02 00	49 02 00
+ /task3_TB/pass_through_values/decryption_module/FSM_1/q	00	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/iterator	00	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/out_value	00	
/task3_TB/pass_through_values/decryption_module/FSM_1/wren	0	
/task3_TB/pass_through_values/decryption_module/FSM_1/finish_FSM_1	0	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/iterator_i	04	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/iterator_j	55	00
+ /task3_TB/pass_through_values/decryption_module/FSM_1/iterator_k	xx	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/wait_count	11	00
+ /task3_TB/pass_through_values/decryption_module/FSM_1/mods	02	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/requested_i...	04	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/requested_i...	00	00
/task3_TB/pass_through_values/decryption_module/FSM_1/select_swap...	1	
/task3_TB/pass_through_values/decryption_module/FSM_1/request_to_...	0	
+ /task3_TB/pass_through_values/decryption_module/FSM_1/requested_...	00	00
/task3_TB/pass_through_values/decryption_module/FSM_1/start_swap	1	
/task3_TB/pass_through_values/decryption_module/FSM_1/finished_loo...	0	
/task3_TB/pass_through_values/decryption_module/FSM_1/state	SWAP_IJ_LOOP_2	INITIALIZE S
/task3_TB/pass_through_values/decryption_module/FSM_1/next_state	SWAP_IJ_LOOP_2	INITIALIZE S

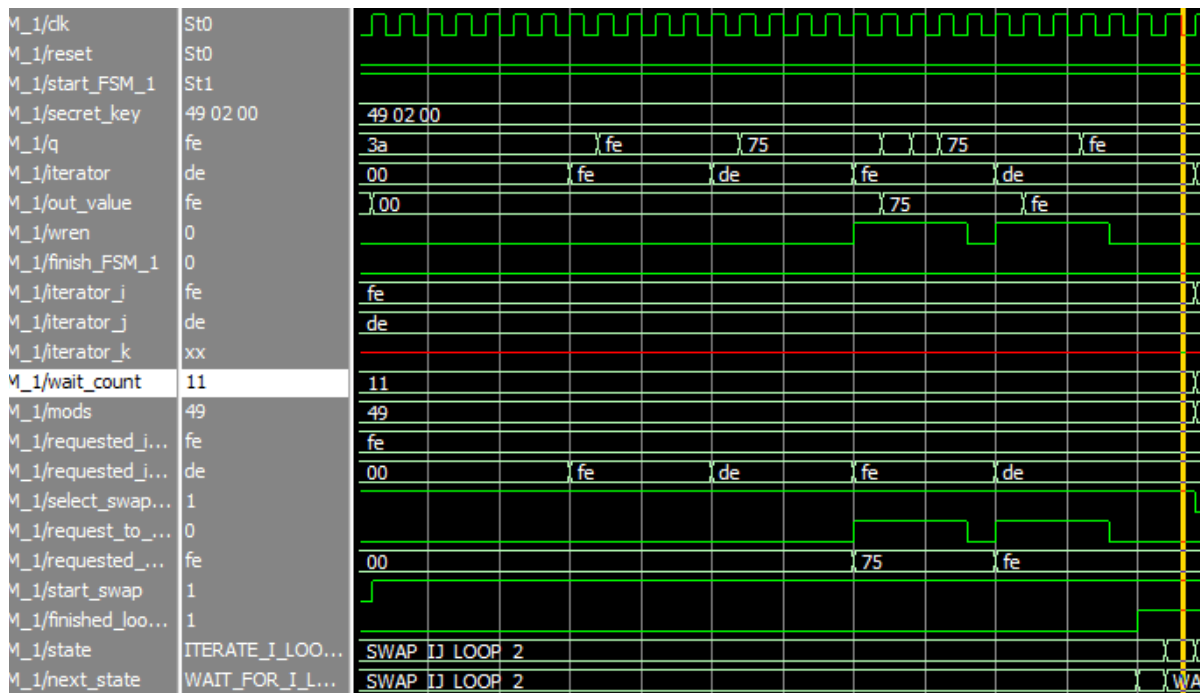
The task2a_fsm starts by doing the first loop, and by placing the values into S_memory. During this process, the value of “q” is irrelevant to us so it is seen as don’t cares.



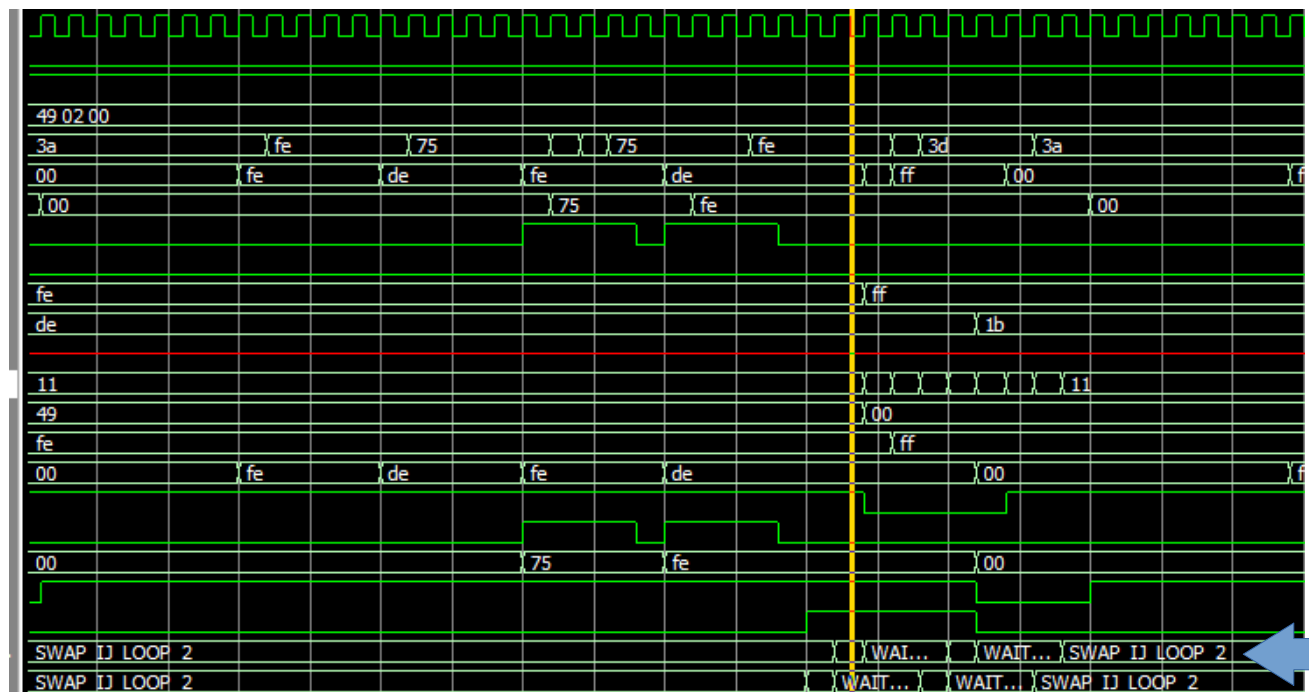
Once the S_memory has been filled from 00 to FF, it continues onto the second loop. Here the values of iterator_i and iterator_j are incremented, and then the swap_ij_fsm is called.



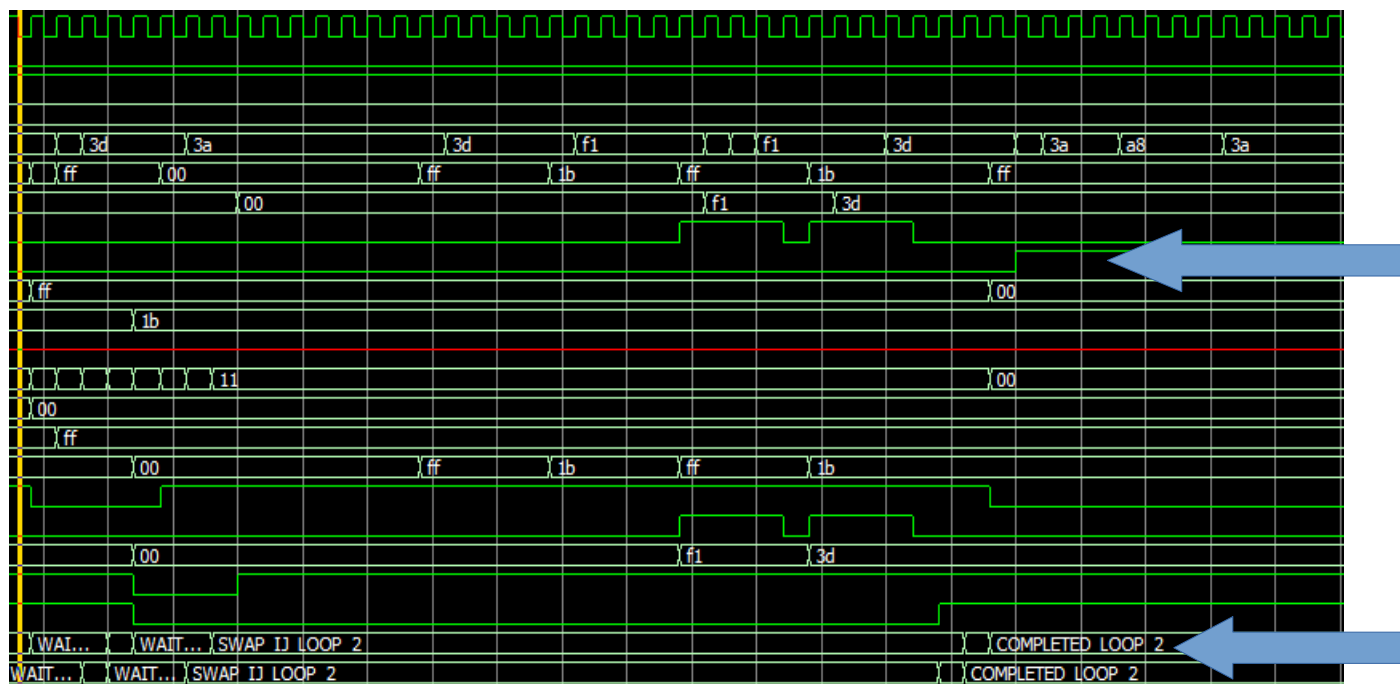
From a more zoomed in perspective we can see that while this FSM is in the SWAP_IJ_LOOP_2 state, the swap_fsm is changing values.



After the swap_ij_fsm finishes, the task2a_fsm iterates iterator_i, waits, iterates iterator_j, waits again, and then calls the swap_ij_fsm once more.



This process is repeated until loop_2 is completed. Where we can see the finish_FSM_1 wire is set high.



task2_swap_ij_fsm

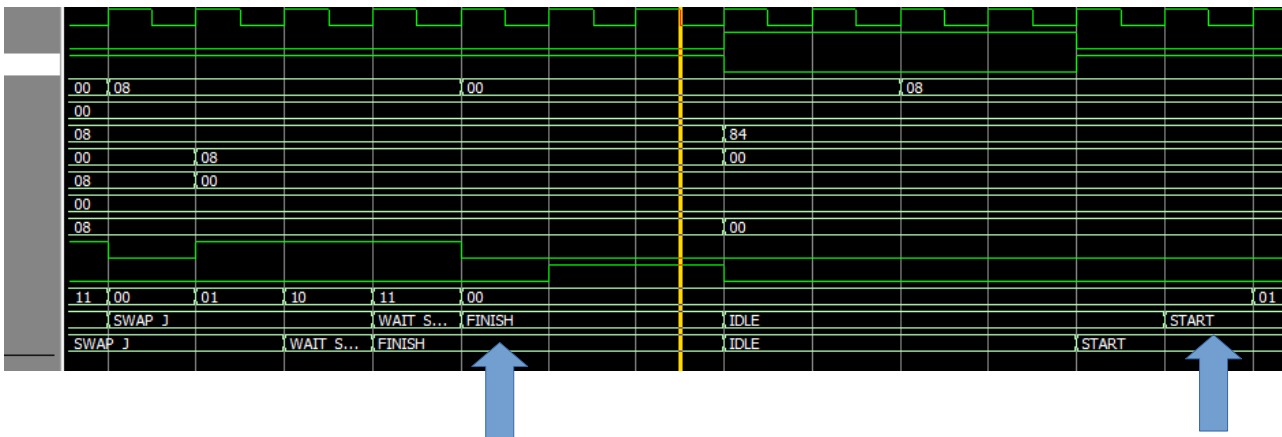
As mentioned earlier this fsm is used in both task2a and task2b. Here it is held in a suspended state until it is told to start.

/task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/dk	St0	
/task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/reset	St0	
/task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/fsm_start	St1	
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/q	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/iterator_i	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/iterator_j	08	00 08
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/iterator	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/out_value	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/saved_value_i	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/saved_value_j	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/wren	0	
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/fsm_finished	0	
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/wait_count	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/state	COPY_I	IDLE
+ /task3_TB/pass_through_values/decryption_module/FSM_2a/swap_fsm2/next_state	COPY_I	IDLE

Here we can see that “fsm_reset” was disabled, and “fsm_start” was enabled. Immediately we exit the “IDLE” state and give the S_memory time to gather the necessary information while in the “START” and “WAIT_START” states. “IDLE” initializes values, “START” increments a counter for number of cycles to wait, and “WAIT_START” restarts that counter, so that the next state can use the same reg for its own counter.

Timing diagram for a 10-bit shift register. The diagram shows a clock signal (green) and a data input (yellow). The data input is a sequence of bits: 00, 01, 10, 11, 00, 01, 10, 11, 00, 01. The output of the shift register is shown in a row of boxes below the data input. The output sequence is: COPY I, WAIT C..., COPY J, WAIT C..., SWAP I. Blue arrows point to the output of the shift register at each clock edge.

The below image is spliced down the middle (at the yellow bar) for easier viewing.



An FSM diagram has been included at the end of the document for the `swap_ij_fsm`.

task2b_fsm

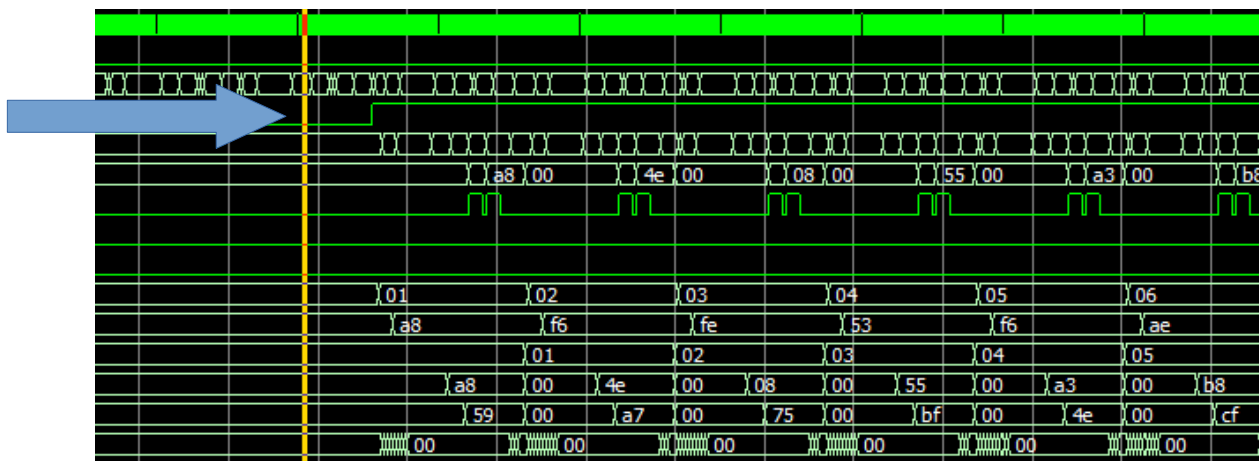
We start in the "START" state,

+ /task3_TB/pass_through_values/decryption_module/FSM_2/requested_i...	00000000	00000000	
+ /task3_TB/pass_through_values/decryption_module/FSM_2/requested_j...	00000000	00000000	
/task3_TB/pass_through_values/decryption_module/FSM_2/select_swap...	0		
/task3_TB/pass_through_values/decryption_module/FSM_2/start_swap	0		
/task3_TB/pass_through_values/decryption_module/FSM_2/finished_swap	0		
/task3_TB/pass_through_values/decryption_module/FSM_2/state	START	START	
/task3_TB/pass_through_values/decryption_module/FSM_2/next_state	START	START	

Here all our values are initialized to be zeros.

/task3_TB/pass_through_values/decryption_module/FSM_2/dk	St1	
/task3_TB/pass_through_values/decryption_module/FSM_2/reset	St0	
+ /task3_TB/pass_through_values/decryption_module/FSM_2/q	3d	
/task3_TB/pass_through_values/decryption_module/FSM_2/finish_FSM_1	St0	
+ /task3_TB/pass_through_values/decryption_module/FSM_2/iterator	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/out_value	00	00
/task3_TB/pass_through_values/decryption_module/FSM_2/wren	0	
/task3_TB/pass_through_values/decryption_module/FSM_2/failed_decrypt	0	
/task3_TB/pass_through_values/decryption_module/FSM_2/done_decrypt	0	
+ /task3_TB/pass_through_values/decryption_module/FSM_2/iterator_i	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/iterator_j	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/iterator_k	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/saved_value_i	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/saved_value_j	00	00
+ /task3_TB/pass_through_values/decryption_module/FSM_2/wait_count	00	00

momentarily the finish_FSM_1 signal comes in, and the FSM starts iterating through loop 3.



Here the FSM goes into ITERATE_I, WAIT_FOR_I, ITERATE_J, WAIT_FOR_J,

START		ITERATE I	WAIT FOR I		ITERATE J	WAIT FOR J
START	ITERATE I	WAIT FOR I		ITERATE J	WAIT FOR J	

and then SWAP_IJ.

WAIT FOR J			SWAP IJ			
J			SWAP IJ			

We then retrieve the information from the encrypted RAM in the “RETRIEVE_K” state.

21						00		
SWAP IJ		RETRIEVE K		OUTPUT K	ITERATE K	ITERATE I	WAIT F...	START
SWAP IJ	RETRIEVE K		OUTPUT K	ITERATE K	ITERATE I	WAIT FOR I		START

From there the value for k is used to access a location in memory in both the RAM and ROM. The moment where the encrypted memory is first called and q_m becomes valid is apparent.

Timing diagram for the first 16 clock cycles of the program. The diagram shows the execution of instructions: MOV R0, #0x01; MOV R1, #0x00; MOV R2, #0xf5; MOV R3, #0x00; MOV R4, #0x7d; MOV R5, #0x00; MOV R6, #0x14; MOV R7, #0x00; MOV R8, #0xf1; MOV R9, #0x00; MOV R10, #0x87; MOV R11, #0x00; MOV R12, #0x29; MOV R13, #0x00; MOV R14, #0x56; MOV R15, #0x8f; MOV R16, #0x7a; MOV R17, #0xa9; MOV R18, #0x38; MOV R19, #0x73; MOV R20, #0x5f; MOV R21, #0x87. The diagram includes signals for clock, instruction address, instruction data, and various register outputs. A blue arrow points to the first instruction address.

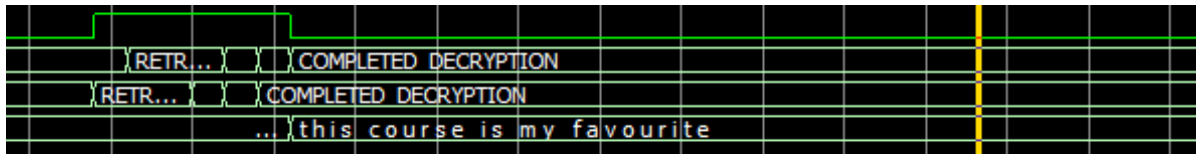
Eventually the decryption is completed.

f8	16	25	2d		
00... 00... 00... 10010010	00000000				
S...	S...	S...	COMPLETED	DECRYPTION	

and we can see the “done_decrypt” signal is set high.

The timing diagram shows a sequence of data and control signals. A blue arrow points to a signal that transitions from low to high, coinciding with the start of a new data word '1e'.

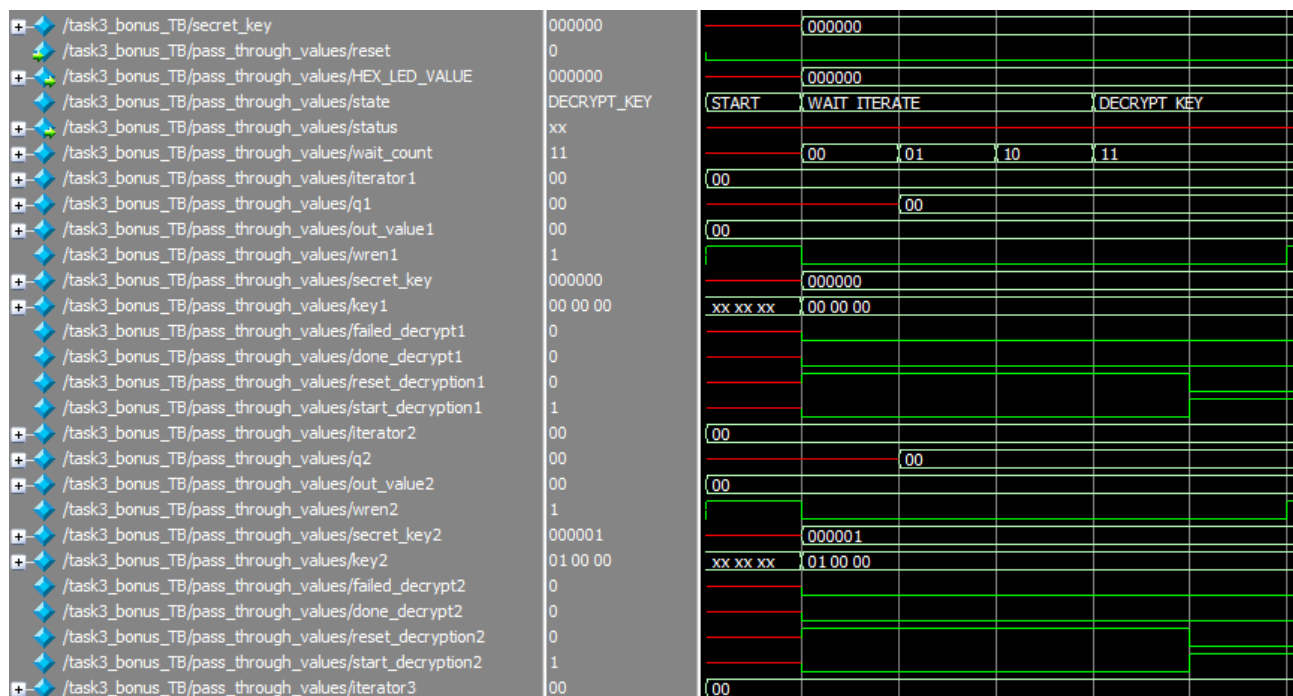
As well as the result.



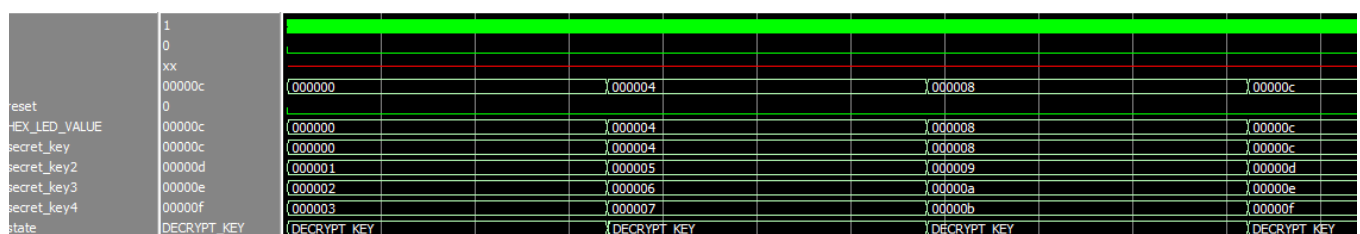
task3_bonus_fsm

This fsm calls and handles 4 different decryption cores (task2_fsm).

The states are the same as in task3_fsm. It behaves nearly identically, it simply has more cores to start up.



Another detail is that it increments the secret_key by four at a time, with the main secret_key going in the first core and delegates each of the next 3 keys to the next 3 cores.



And that is the totality of the unique behaviour.

Messages

Messages 1-3,

Key = 249: "this course is my favourite "

Key = 3FF: "congratulations on task two "

Key = 2AA: "ubc elec and comp engineering "

Message 4, Key: 087B2D

In-System Memory Content Editor - E:/Courses/CPEN311/Ebi/Labs/CPEN311_LAB_4/rtl/rc4 - rc4

File Edit View Processing Tools Window Help

Search altera.com

Instance Manager: Acquisition in progress

Index	Instance ID	Status	Width	Depth	Type
0	S	Unloading data	8	256	RAM/ROM
1	DECR	Unloading data	8	32	RAM/ROM
2	ENCR	Unloading data	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1] Setup...

Device: @2: 5CSE(BA5|MA5)/! Scan Chain

File: ...

Instance 0: S

```

000000 08 84 88 8F 2F FB EA BC 78 EF CB E8 58 B1 63 04 26 6E CD 95 .... /...x...X.c.&n..
000014 5F CE 0D 4B B3 27 11 9E 52 C0 D1 9B 23 93 16 AC D8 2A C2 0B _..K.'..R...#....*..
000028 6C F3 90 1E 66 2D D6 B6 C8 46 13 AD 22 03 FC 74 B2 09 ED 44 l...f-...F.."..t...D
00003c 15 54 7F 65 99 47 31 D4 4D 91 5C 20 D0 80 12 A7 19 67 4A 6D .T.e.Gl.M.\ .....gJm
000050 76 81 DB 33 E6 9A A5 6B 9D 2E 57 87 05 43 73 BD 83 A8 E2 8B v..3...k..W..Cs.....
000064 3A 8C 64 5B 1C F4 21 06 2B 02 3F B7 1D A0 69 AA 68 56 0E E7 :.d[...!.+.?...i.hV..
000078 7A EC 3B 00 F9 C3 53 AF 14 F1 EE 6F 01 3E 92 CF 48 F6 E4 F5 z.;...S....O.>..H...
00008c 35 1A BE CA 2C C7 37 BF 6A 7C 25 B5 9F 5E FF 18 30 F8 32 62 5...,.7.j|%.^..0.2b
0000a0 DD A3 E9 24 DE E5 B0 F7 40 A2 5D 71 29 97 C5 4F 0C 55 0A 34 ...$....@.]q)..O.U.4
0000b4 FA 5A 86 59 D3 B9 60 D7 C4 61 E0 C1 C9 7E 3C 72 DF 41 77 10 .Z.Y..`..a...~<r.Aw.
0000c8 D2 F0 A4 50 70 FE B8 07 0F 82 94 BA 36 E3 DA 79 AE 42 B4 E1 ...Pp.....6..y.B..
0000dc 75 7B 4E BB A1 A9 3D 51 AB 38 96 D5 17 DC 89 EB 85 39 8E 49 u{N...=Q.8.....9.I
0000f0 8A A6 1F 8D 98 D9 1B 4C 9C C6 45 CC 7D F2 28 FD .....L..E.).(.

```

Instance 1: DECR

```

000000 72 63 20 66 6F 75 72 20 69 73 20 6E 6F 74 20 76 65 72 79 20 rc four is|not very
000014 73 65 63 75 72 65 20 20 20 20 20 20 20 20 20 20 secure

```

Instance 2: ENCR

```

000000 C5 AF B0 4D 4E FD A5 58 2C 54 10 D6 4F D0 86 B8 05 57 42 10 ...MN..X,T..O....WB.
000014 1C E9 27 36 35 04 C6 E7 A7 1D C4 E8 ..'65.....

```

0% 00:00:00 Instance 1: DECR Word: 0x00000a Bit: 0x000007

Message 5, Key: 04000C

In-System Memory Content Editor - E:/Courses/CPEN311/Ebi/Labs/CPEN311_LAB_4/rtl/rc4 - rc4

File Edit View Processing Tools Window Help

Search altera.com

Instance Manager: Acquisition in progress

Index	Instance ID	Status	Width	Depth	Type
0	S	Unloading data	8	256	RAM/ROM
1	DECR	Unloading data	8	32	RAM/ROM
2	ENCR	Unloading data	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1] Setup...

Device: @2: 5CSE(BAS|MA5)/! Scan Chain

File: ...

Instance 0: S

```

000000 23 7A B0 34 33 44 36 D2 28 CD 43 72 E7 A8 D8 C2 BE 7C 6B 15 #z.43D6.(.Cr.....|k.
000014 87 4D 99 B3 82 B4 F5 92 55 0E 21 4C 75 52 D1 E5 E1 CE 2A 20 .M.....U.!LuR....*
000028 9D C8 F6 F9 09 06 AC FE 26 90 79 9A 46 DB 70 88 9B A7 57 2C .....&.y.F.p...W,
00003c 3F 86 E4 0A F4 D0 3A 12 A5 EE 22 29 47 85 4A DA 73 FA C9 8E ?.....")G.J.s...
000050 25 1D 83 B5 38 7D E3 FC 1C 18 BC 16 2E 13 77 AD 94 24 48 58 %...8}.....w..$HX
000064 89 32 C4 5F 98 B2 1B FF 8B DC DF C6 51 35 8F E8 2F 9E 50 1A .2_.....Q5../.P.
000078 D5 C5 03 BD B1 37 63 CA 59 EF C3 67 7F AB 45 A9 CF A0 95 8A .....7c.Y..g..E....
00008c 39 F7 4F 05 31 3B 7B CC D6 3C A2 0F EA 6A 96 D7 B8 CB 5D 10 9.O.1;{.<...j....}.
0000a0 71 27 DE 97 17 19 5E F1 BB 14 C0 8D 5A AE 53 00 E0 B7 54 E6 q'....^.....Z.S...T.
0000b4 6F 3D D4 30 C7 C1 02 EC 5B F8 42 93 DD 2D 76 6C 65 1E 41 0B o=.0....[.B..-vle.A.
0000c8 7E E2 84 40 F0 04 FD D3 9F 60 49 2B 0D F2 08 11 5C A1 78 4B ~..@.....`I+....\xK
0000dc 07 F3 BA A4 64 61 66 6E 0C 1F 8C 3E 74 EB 69 ED 91 81 A6 BF ....dafn...>t.i....
0000f0 E9 6D B9 4E A3 56 01 62 9C AF 80 68 FB B6 AA D9 .m.N.V.b...h....

```

Instance 1: DECR

```

000000 79 6F 75 20 6E 6F 77 20 68 61 76 65 20 73 6F 6C 69 64 20 76 you now have solid v
000014 68 64 6C 20 73 6B 69 6C 6C 73 20 20 hdl skills

```

Instance 2: ENCR

```

000000 22 03 BC 50 91 2E 28 9A 2B F0 E7 D6 54 D3 F2 17 B5 C4 FF 96 "...P..(+...T.....
000014 5C 38 18 A3 0F DC B5 62 C0 70 8E 03 \8....b.p..

```

0% 00:00:00 Instance: Word: Bit:

Message 6, Key: 000000

In-System Memory Content Editor - E:/Courses/CPEN311/Ebi/Labs/CPEN311_LAB_4/rtl/rc4 - rc4

File Edit View Processing Tools Window Help

Search altera.com

Instance Manager: Acquisition in progress

Index	Instance ID	Status	Width	Depth	Type
0	S	Unloading data	8	256	RAM/ROM
1	DECR	Unloading data	8	32	RAM/ROM
2	ENCR	Unloading data	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1] Setup...

Device: @2: SCSE(BAS|MA5)! Scan Chain

File: ...

Instance 0: S

```

000000 00 C6 0A 25 FD C0 AB B7 63 08 90 67 D0 BF 95 09 F4 F3 5E 96 ...%.c.g.....^
000014 A9 97 D2 11 DD 32 20 BA D4 7A 48 C8 EC 2A E4 23 C3 F5 03 5B .....2..zH.*.#...[
000028 CE 17 EB 1B 8A 12 8F FA D6 4C 7B D9 84 F9 CB 7F 40 3E 21 3C .....L{.....@>!<
00003c F8 55 B1 06 8E 53 6E 8C 29 87 C4 EE 9C F2 8D 43 05 B9 83 3F .U...Sn.).....C...?
000050 89 2B AC 79 46 86 ED 82 19 2C 99 A6 4E C9 0C 77 D7 07 7E 72 .+.yF.....N..w..~r
000064 61 0B 35 04 FE 2D 66 85 E6 58 C1 81 A0 7C 54 6C EF BD 98 78 a.5...-f..X...|Tl...x
000078 73 CF EA B0 56 9D A4 BB 47 01 0F 3A 1D 15 2E 91 F7 A2 5F E5 s...V...G...:.....
00008c 0D E2 9F AF 38 64 60 CA 65 B2 9A 2F CD 6A 94 68 5D 70 1A A5 ....8d`.e../.j.h)p..
0000a0 80 F6 92 DA 42 D3 41 5A FC 13 28 31 DF AE FF 33 4D E3 30 DC ....B.AZ..(1...3M.0.
0000b4 A8 76 E0 62 4B 69 7D C7 49 52 39 B5 51 AD 44 34 E8 16 02 D8 .v.bKi}.IR9.Q.D4....
0000c8 71 1E 6D A3 5C 3D 0E 24 26 E1 4F E7 AA F0 14 DB CC A1 B4 BC q.m.\=. $&.O.....
0000dc 74 BE F1 C5 B3 57 4A 93 50 36 45 10 A7 DE 88 D5 37 B6 8B 18 t....WJ.P6E.....7...
0000f0 D1 FB 3B 1C 6F 59 9E 9B C2 6B E9 22 75 B8 1F 27 ...;oY...k."u..'

```

Instance 1: DECR

```

000000 74 68 69 73 20 6F 6E 65 20 69 73 20 74 72 69 63 6B 79 20 77 this one is tricky w
000014 69 74 68 20 7A 65 72 6F 20 6B 65 79 ith zero key

```

Instance 2: ENCR

```

000000 AA 70 E0 32 83 58 33 5F AA 6F 6D 47 23 1C FB 0E AC 63 5F D4 .p.2.X3_.omG#....c_
000014 99 B8 83 B7 3F 4E 3F 5D 07 FD 3A E7 ....?N?]...

```

0% 00:00:00 Instance: Word: Bit:

Message 7, Key: 02640F

In-System Memory Content Editor - E:/Courses/CPEN311/Ebi/Labs/CPEN311_LAB_4/rtl/rc4 - rc4

File Edit View Processing Tools Window Help

Search altera.com

Instance Manager: Acquisition in progress

Index	Instance ID	Status	Width	Depth	Type
0	S	Unloading data	8	256	RAM/ROM
1	DECR	Unloading data	8	32	RAM/ROM
2	ENCR	Unloading data	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1] Setup...

Device: @2: 5CSE(BAS[MAS])/! Scan Chain

File: [Download] [Browse]

Instance 0: S

```

000000 4A 2D 1C 25 B0 51 62 FF D9 E2 AA C2 3A 1B 93 16 A7 EF EA E0 J-%.Qb.....
000014 2F F3 D6 27 42 A1 57 13 DC C7 97 5A 95 69 9F D1 05 09 3B DE /..'B.W....Z.i....;
000028 07 82 CE B2 CA 46 59 AB BB 50 91 A8 FD 6C 01 54 60 7B 53 9D .....FY..P...l.T`{S.
00003c 6F 40 C9 AC AE FE CB 08 39 5C 87 E4 9E 4B A4 4E 0C 0B 85 41 o@.....9\...K.N...A
000050 AF 38 74 79 CF 28 A9 F6 BE 4D 8E C3 83 72 D4 68 5B 49 35 90 .8ty.(...M...r.h[I5.
000064 7C A5 2B 67 F5 8C B3 EC 10 0E E7 44 EB D2 48 64 C0 80 88 B1 |.+g.....D..Hd....
000078 B9 F1 7A D3 8D 9B 19 47 1A E9 34 C4 03 E6 F7 9C F8 6A 0D 58 ..z....G..4.....j.X
00008c D8 B4 C6 23 A6 F9 9A BC B7 37 14 29 1F 56 1D F2 20 A0 26 00 ...#.....7.)..V... &
0000a0 84 06 8A 70 02 A3 30 17 CD 6E 32 89 FA 2E 73 DB B8 55 D7 96 ...p..0..n2....s..U..
0000b4 22 8B 3C DD 15 99 E8 C8 B5 FB 71 E3 BD C5 ED DA 86 52 36 7E ".<.....q.....R6~
0000c8 65 2A 98 3E 33 77 E1 31 C1 12 11 FC 3F AD 6B 5D 63 4F 94 F0 e*.>3w.1....?k]cO..
0000dc 1E 6D CC 24 BF 81 2C 4C 45 75 EE 92 F4 D5 0A 5F D0 3D 8F 04 .m.$...,LEu....._.=..
0000f0 66 18 43 E5 DF 78 76 5E 21 A2 B6 7D 7F BA 61 0F f.C..xv^!...}.a.

```

Instance 1: DECR

```

000000 76 68 64 6C 20 6E 69 6E 6A 61 20 73 61 76 65 73 20 74 68 65 vhd1 ninja saves the
000014 20 64 61 79 20 20 20 20 20 20 20 20 20 20 20 20 day

```

Instance 2: ENCR

```

000000 C1 0C 35 D0 8D 27 66 4B 0D AD 4B B6 31 29 32 C2 90 0D 68 1E ..5..'fK..K.1)2...h.
000014 76 5E 02 62 DA 45 70 C3 A5 92 74 BE v^.b.Ep...t.

```

0% 00:00:00 Instance: Word: Bit:

Message 8, Key: 3FFFFFFF

In-System Memory Content Editor - E:/Courses/CPEN311/Ebi/Labs/CPEN311_LAB_4/rtl/rc4 - rc4

File Edit View Processing Tools Window Help

Search altera.com

Instance Manager: Acquisition in progress

Index	Instance ID	Status	Width	Depth	Type
0	S	Unloading data	8	256	RAM/ROM
1	DECR	Unloading data	8	32	RAM/ROM
2	ENCR	Unloading data	8	32	RAM/ROM

JTAG Chain Configuration: JTAG ready

Hardware: DE-SoC [USB-1] Setup...

Device: @2: SCSE(BA5|MA5)/! Scan Chain

File: ...

Instance 0: S

000000	00 5F 9F A1 A4 30 53 41 61 92 DC 8F B2 83 EA E1 3F FF 1F D8 0SAa ? . . .
000014	62 50 95 60 40 8C 57 FD DB C3 3A 44 FC A2 37 38 22 BF E4 F6	bP . ` @ . w . . : D . 78 " . .
000028	71 99 3E 0A 56 F8 EF D9 0C AE 03 14 1A 73 10 94 6E A5 C4 78	q . > . V s . n . x
00003c	75 F7 4D 4C A0 43 4F F9 3C BB 19 8B 13 81 D6 EB C7 5E E6 35	u . ML . CO . < ^ . 5
000050	11 79 CA A6 5C 93 F2 27 CC 6C 5A 1C 74 65 9A AB CF AA E0 96	. y . \ . ' . l z . te
000064	5B F0 23 68 1D 8A BD D5 80 63 DD D0 42 0B 2F B8 84 87 85 A3	[. # h c . B . /
000078	B3 DA 7F 8D 54 24 B1 21 6D 6F DF 31 A7 3B 49 9C 5D A9 48 BA T \$. ! mo . l . ; I .] . H .
00008c	D2 58 B0 D7 D4 01 2C 69 09 15 EC 82 0E B7 66 AC 07 45 76 CD	. X , i f . . Ev .
0000a0	55 8E E8 B6 C8 16 1B 4A 59 C0 25 46 ED E9 86 32 6A 17 C6 A8	U JY . % F . . 2 j . .
0000b4	90 EE 1E E7 33 2B 52 12 3D C9 20 91 47 9E CB FB 7B 0F BE AD 3 + R . = . . G . . { . .
0000c8	36 FA E3 77 D3 29 89 02 64 26 28 18 B5 9D DE 4E 88 D1 0D 98	6 . . w .) . . d & (. . . N . . .
0000dc	B9 67 7A B4 F4 E2 BC 4B 72 CE 06 08 FE 70 04 F3 E5 05 2D C5	. gz Kr p - .
0000f0	6B 97 F5 7D 2E 7C 34 2A C2 7E C1 F1 9B AF 51 39	k . . } . 4 * . ~ Q9

Instance 1: DECR

000000	67 6F 6F 64 20 6C 75 63 6B 20 6F 6E 20 79 6F 75 72 20 65 78	good luck on your ex
000014	61 6D 73 20 20 20 20 20 20 20 20 20 20 20 20 20	ams

Instance 2: ENCR

000000	CC 11 01 FD A2 DC 6A 16 85 41 60 7A 71 DE F0 21 6A DD 2F 26 j . . A ` z q . . ! j . / &
000014	02 E0 07 AD 0A B8 08 95 D5 3E 91 5E > . ^

0% 00:00:00 Instance: Word: Bit:

Reference

SWAP_IJ_FSM Diagram

Reset has not been drawn on all states for the sake of visual simplicity. In the actual desing, the FSM can be reset at any moment.

