Connor Aksama
EE 371
February 12, 2023
Lab 4 Report

Procedure

This lab is comprised of two tasks. The first task involved implementing a bit counting algorithm in hardware using a given ASMD chart. The second task involved designing a binary search algorithm using an ASMD chart, then implementing the algorithm in hardware.

Task 1

I approached this task by first planning out the division between functionality in the controller and the datapath of the hardware implementation. I determined what signals would need to be output from the controller to the datapath, how those signals would affect the data flow, and what feedback would be sent back to the controller. The controller would handle FSM states and transitions, sending signals to the datapath module indicating which state it was currently in. The datapath module would then modify its internally stored data while outputting intermediate results and send feedback to the controller whether the process was complete or not.

After deciding what would be contained within each component of the system, these modules were implemented in Verilog, then simulated and verified using ModelSim.

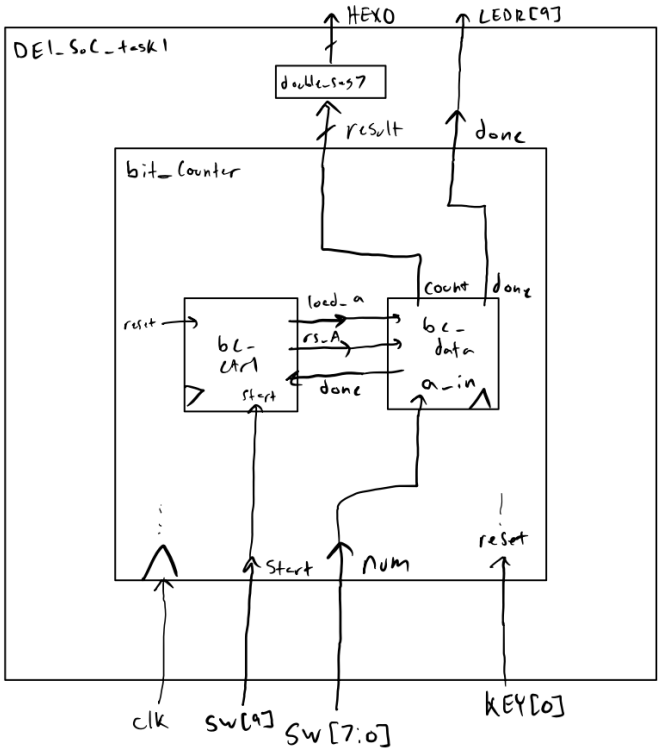


Figure 1. Top-Level Block Diagram for Lab 4 Task 1.

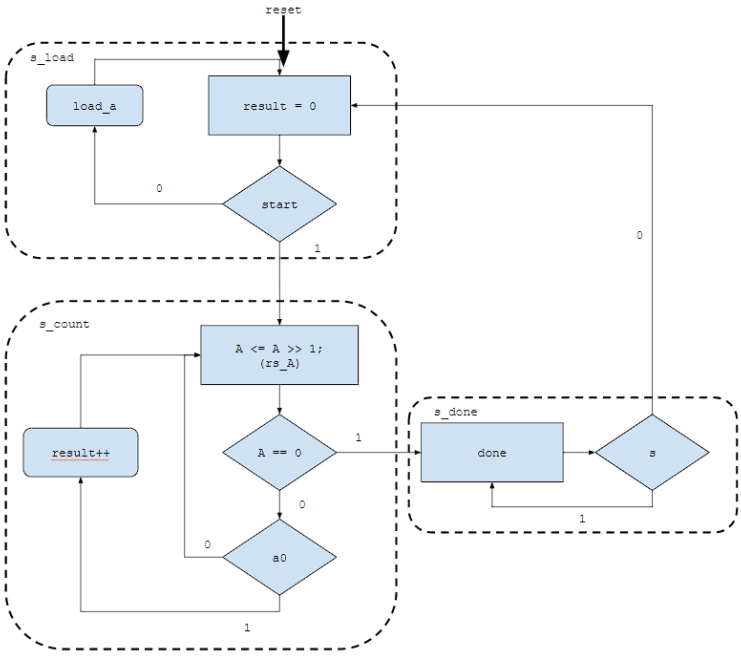


Figure 2. ASMD chart showing the bit counting algorithm.

Task 2

I approached this task by first planning out the high-level algorithm for the binary search. I then drafted an ASM chart to detail each state and the decisions to make at each one. Next, I converted this chart to an ASMD chart by deciding how values would be stored in registers and thinking about when each register would hold a valid value at each state to determine FSM states and timing for transitions between states.

After this ASMD chart was created, similarly to Task 1, distinctions were made between what would belong in a controller module and a datapath module.

These modules were then implemented in Verilog, then tested and simulated in ModelSim before being uploaded and tested on hardware.

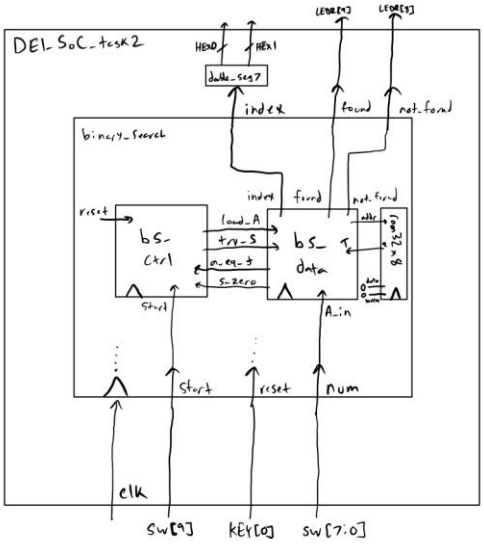


Figure 3. Top-Level Block Diagram for Lab 4 Task 2.

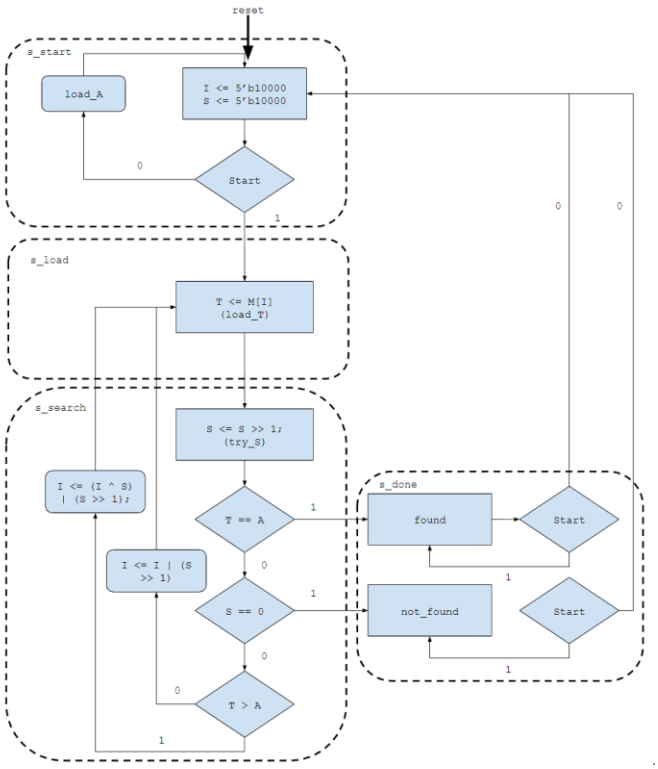


Figure 4. ASMD chart for the Lab 4 Task 2 Design.

Results

Task 1

The following are screenshots from ModelSim simulations for each module used in the bit counter design.

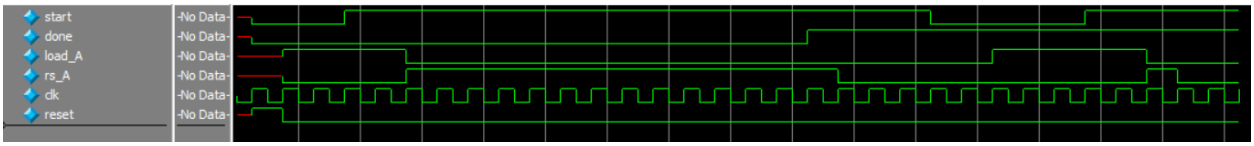


Figure 5. ModelSim Waveform for the Task 1 controller testbench.

This testbench shows the transitions between each of the states in the ASMD chart. After resetting, the system stays in the load state until the start signal is asserted. Following this, the load signal is raised until the done signal is asserted. After this, the controller remains idle until the start signal is lowered, at which point the controller returns to the start state.

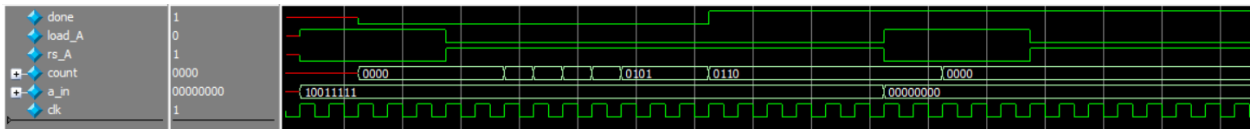


Figure 6. ModelSim Waveform for the Task 1 datapath testbench.

This testbench shows how the internal data of the datapath module is manipulated given external signals from a controller module. While the load signal is asserted, the module samples the input number and holds the count at 0. Once the load signal is low and the right shift signal is high, the process of right shifting and counting bits begins and stops and outputs done once the internal number reaches 0 where it stays until the load signal is asserted again.

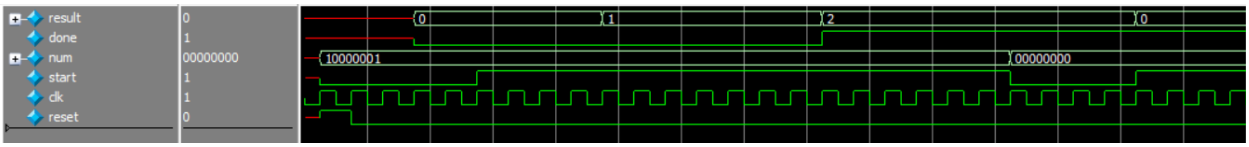


Figure 7. ModelSim Waveform for the Bit Counting module.

This testbench shows the behavior of the design with the controller and datapath modules connected. A number is loaded into the module with the start signal lowered with the result remaining at 0. Once the start signal is asserted, the result begins to increment until the final answer is reached. The system then stays at this state until start is lowered. At this point the next number is loaded in, the result reset, and the process begins again.

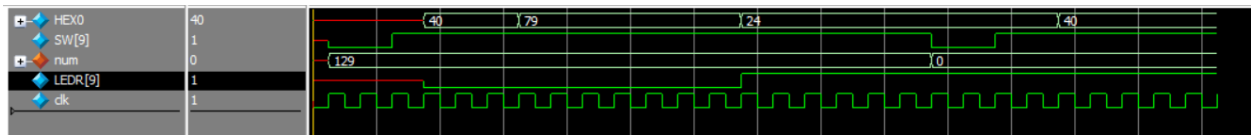


Figure 8. ModelSim Waveform for the Task 1 top-level design module

This testbench shows the correct functionality for the system when connected to the external ports of the board. The testing process is identical to the one shown in Figure 7, except with input and output signals coming from or going to the board's peripherals.

See Figure 13 for the testbench of the double seven-segment display module.

Task 2

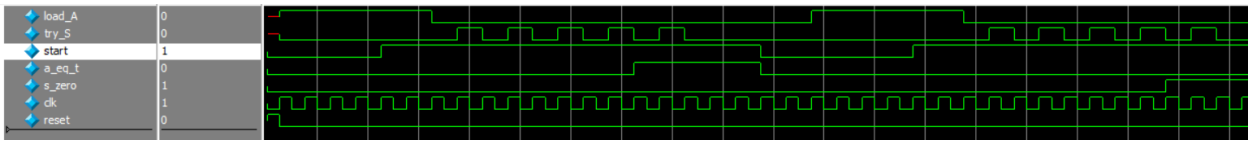


Figure 9. ModelSim Waveform for the Task 2 controller module.

This testbench illustrates the state transitions of the controller module in response to sample feedback from a connected datapath module. On reset, the module outputs the load signal. When the start signal is raised, the controller begins oscillating between a signal to try the current bit index (S) and loading in a value from RAM. Once the equality signal or the zero signal is raised, the controller goes idle until the start signal is lowered.

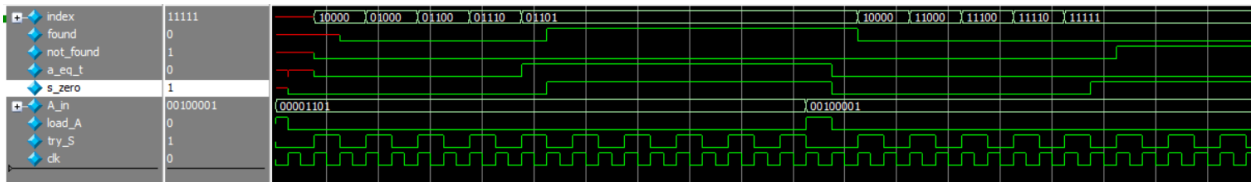


Figure 10. ModelSim Waveform for the Task 2 datapath module.

This testbench shows how data is manipulated in response to signals from a sample connected controller module. On a load signal, the input number is loaded. Then as input signals oscillate between signals to load a value from the connected RAM and to try the current bit index, the output result is shown being updated as the target value is being approached. If the value is found, the module outputs a found signal, and a not found signal otherwise.

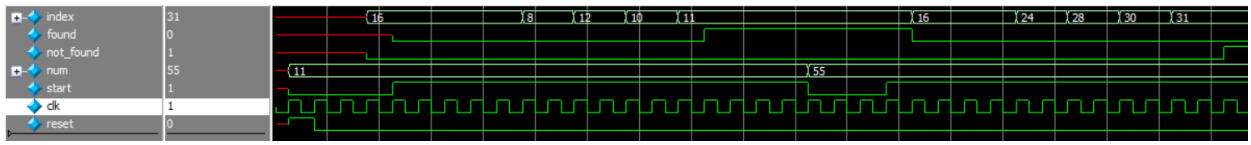


Figure 11. ModelSim Waveform for the binary search module.

This testbench shows the behavior of the design with the controller and datapath modules connected. A number is loaded into the module with the start signal lowered with the result remaining at its start value. Once the start signal is asserted, the result begins to change until the search terminates. The

system then stays at this state until start is lowered. At this point the next number is loaded in, the result reset, and the process begins again.

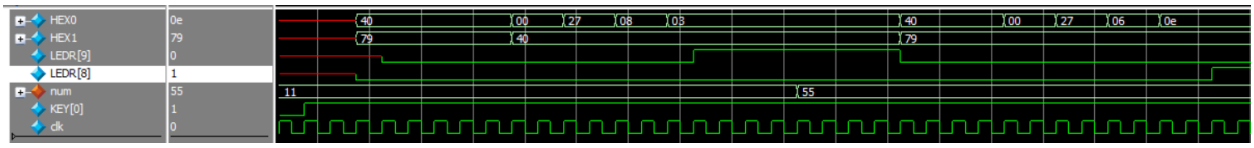


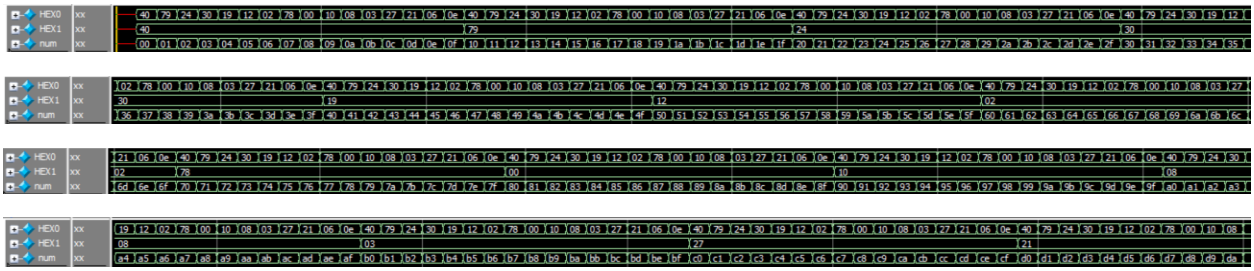
Figure 12. ModelSim Waveform for the Task 2 top-level design module

This testbench shows the correct functionality for the system when connected to the external ports of the board. The testing process is identical to the one shown in Figure 11, except with input and output signals coming from or going to the board's peripherals.

For conciseness, HEX displays are encoded in hexadecimal. Use the following table to convert from hexadecimal code to the corresponding seven segment display:

ModelSim Hexadecimal Code	Seven Segment Display Character
40	0
79	1
24	2
30	3
19	4
12	5
02	6
78	7
00	8
10	9
08	A
03	b
27	c
21	d
06	E
0e	F

Table 1. Conversion between hexadecimal values shown in ModelSim and the character shown on a seven-segment display.



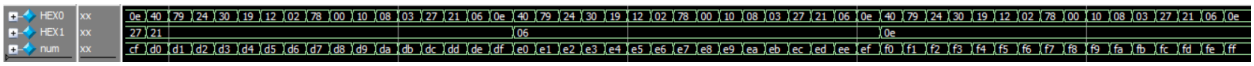


Figure 13. ModelSim Waveform for the Double Seven Segment Display module testbench.

This waveform demonstrates the two digit, hexadecimal display given any number from 0x00 to 0xff. Refer to Table 1 to decode the hexadecimal values to their corresponding characters.

Final Product

The overarching goal of this lab was to gain experience designing, interpreting, and implementing ASMD charts. The main result of Task 1 was a module that implemented a bit counting algorithm using a given ASMD chart. This system would load in an 8-bit number inputted by a user via an array of switches, and when a start signal was triggered with another switch, the number of '1' bits in the number would be shown on the HEX display along with an LED signal indicating whether the process was complete or not.

In Task 2, a binary search algorithm was created using an ASMD chart, then implemented in hardware. This system would take in a target value inputted by a user via an array of switches, and when a start signal was triggered with another switch, the index of the target value in a preprogrammed RAM module would be shown on the HEX display along with LED signals indicating whether the value was successfully found in the RAM or not.

Appendix: SystemVerilog Code

(See next page)