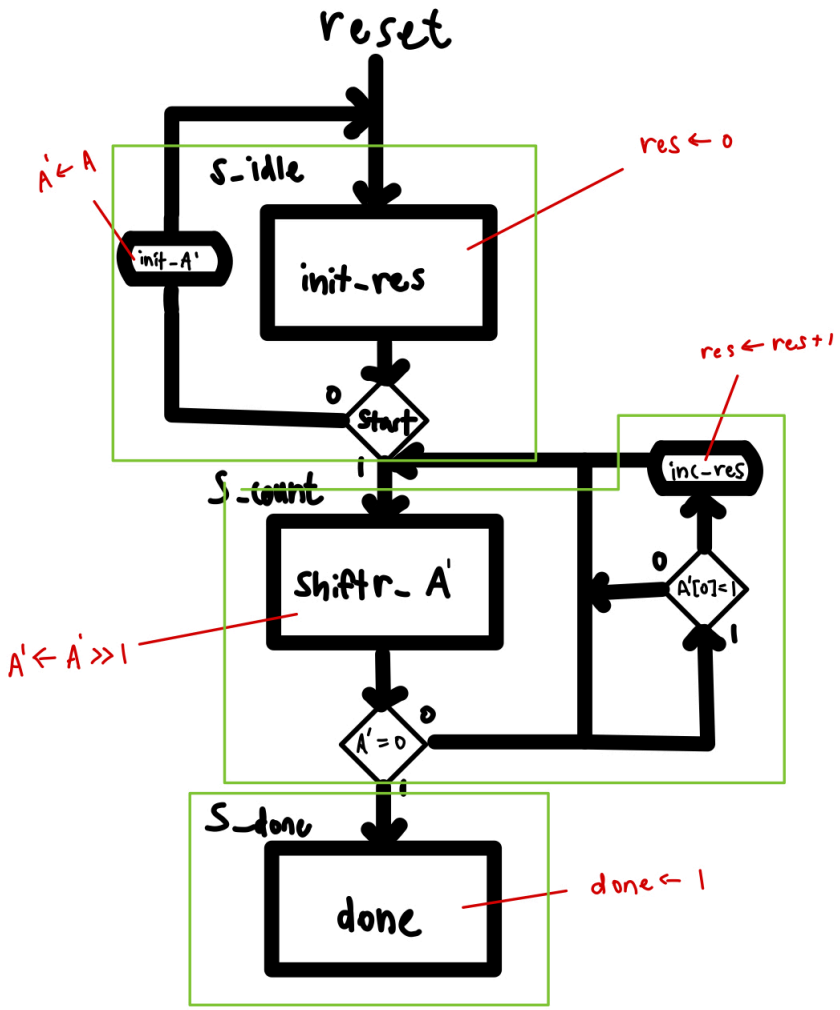
CSE 371 Lab 4 Report

Elijah Melton, 2164822
Ankith Tunuguntla, 2234509

Design Procedure ASMD

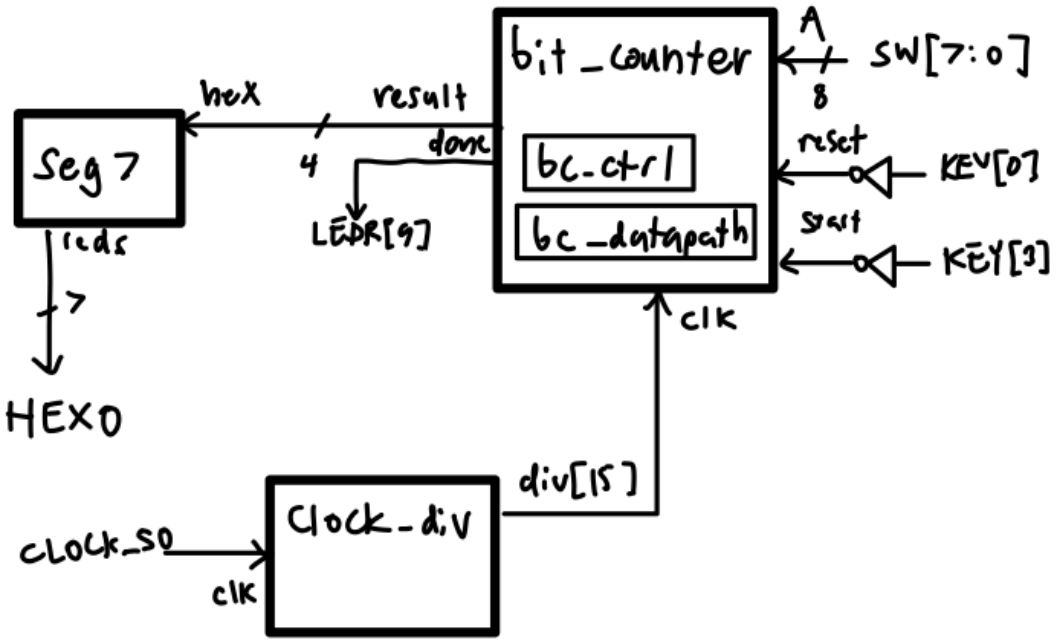
`bit_counter` ASMD

Our ASMD for bit counting. Green is the ASM blocks, RTL is written in red.



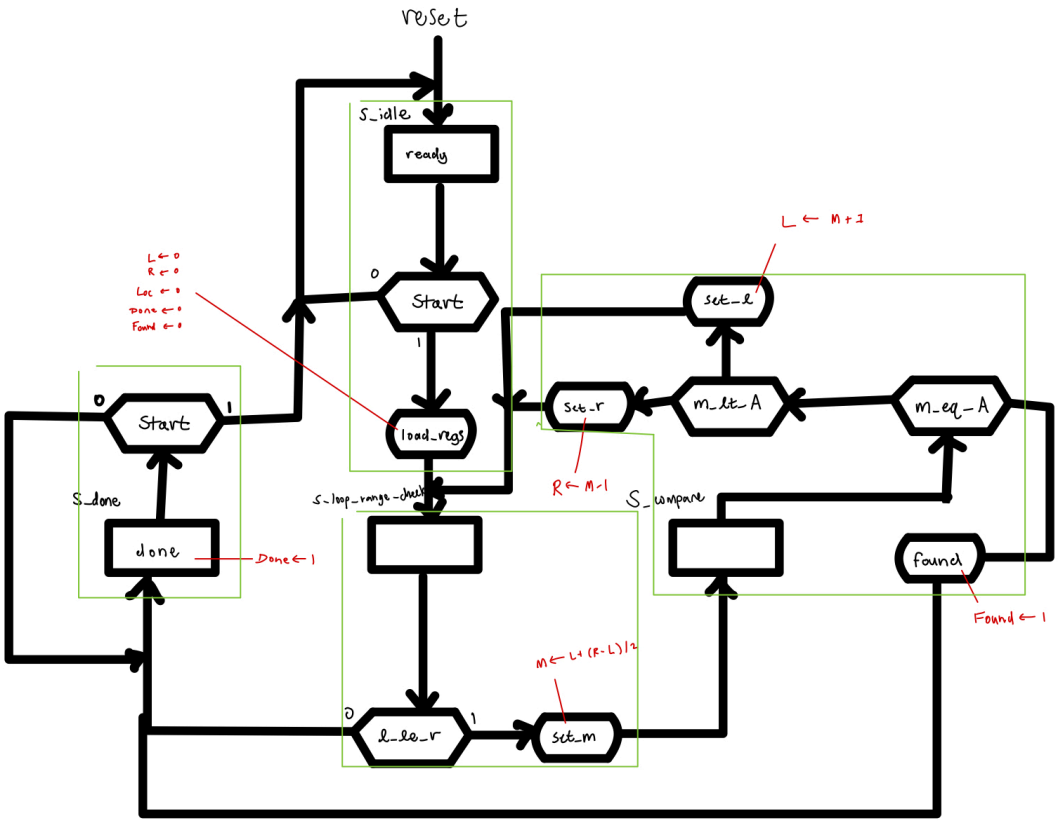
part1 Block Diagram

Block diagram for just bit counting top level module with SW 0-7 for input, KEY 0 for reset, KEY 3 for start, and HEX 0 for the result. Also includes LEDR 9 to show done



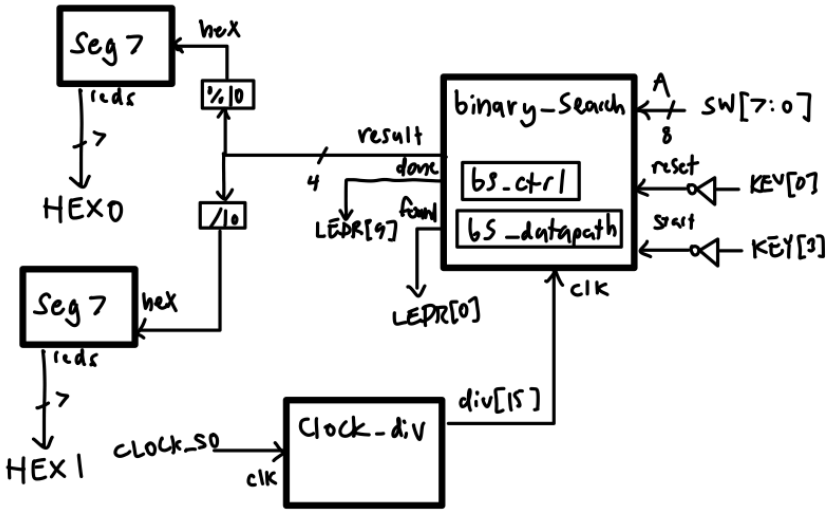
binary_search ASMD

Our ASMD for binary search. Green is the ASM blocks, RTL is written in red.



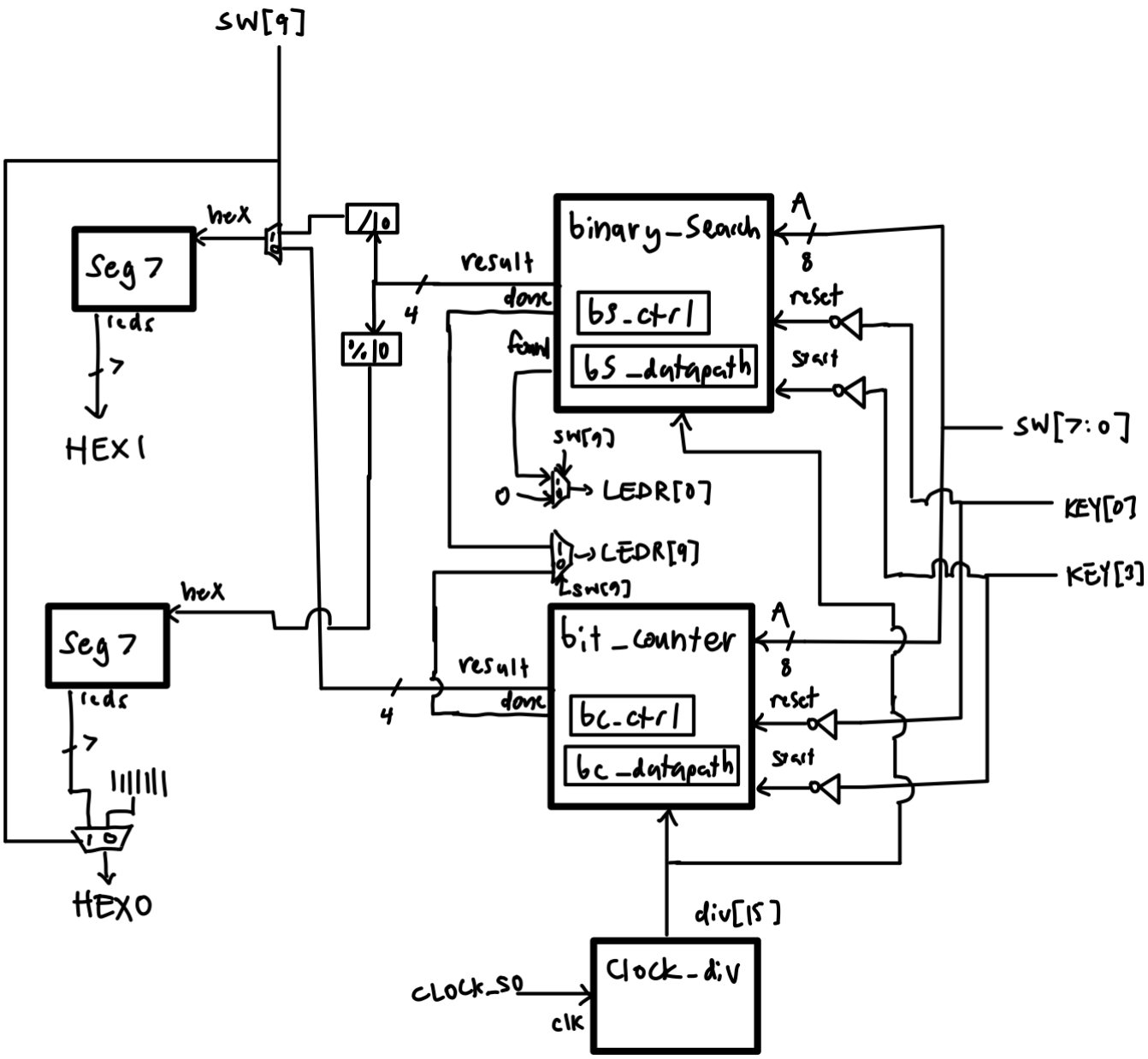
part2 Block Diagram

Block diagram for just binary search top level with SW 0-7 for input, KEY 0 for reset KEY 3 for start, and HEX 0-1 for showing the result. Also includes LEDR 9 to show done and LEDR 0 for showing found



`both` Block Diagram

Diagram for combined binary search and bit counting, muxed with `SW[9]` with all other inputs and outputsthe same



Results

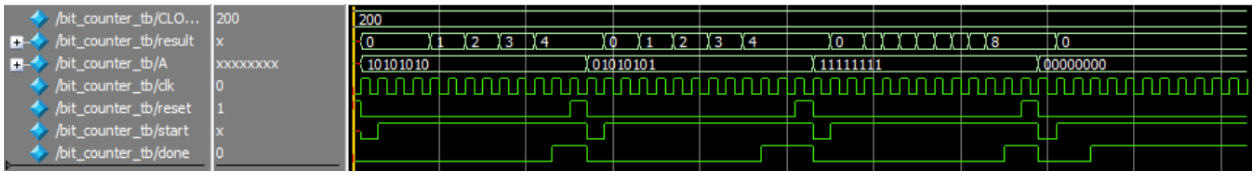
Our implementation ended up being pretty similar to our design, with mostly only minor differences (signal names), as well as a few notable changes for handling reset (perform init datapath logic on reset). We used a clock divider, and also made sure to use a start loop to handle metastability.

We separated both algorithms into 3 modules as we did in lecture: `*_ctrl`, `*_datapath`, and `*`, where `*` is the algorithm. Then, in our top level modules we simply instantiated the `*` module, along with a `clock_div` and `seg7` and all the required hardware mappings.

To multiplex between the two algorithms, we used a single `always_comb` block with default initializations, and then two cases for `SW[9]` that toggle between outputs of the given algorithm.

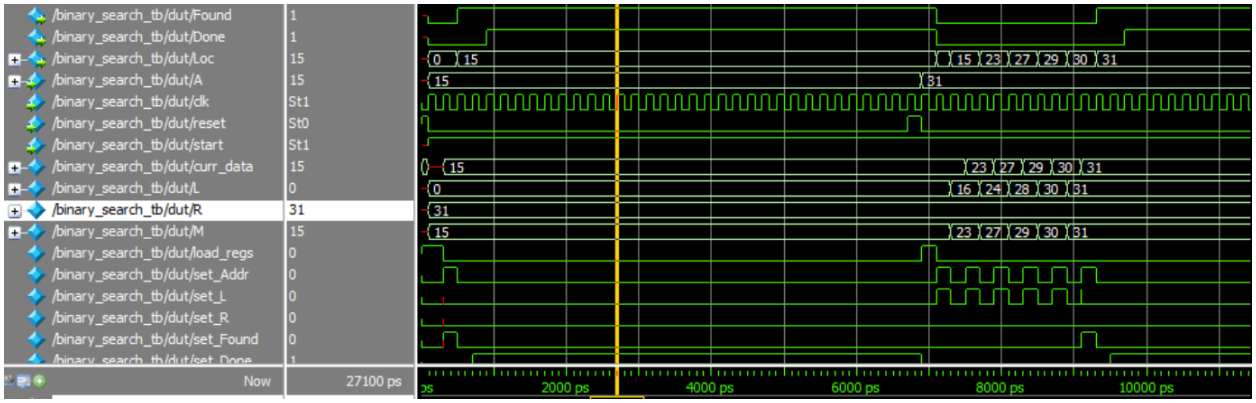
`bit_counter_tb` simulation

With the input `10101010`, it correctly outputs 4.
With the input `01010101`, it correctly outputs 4.
With the input `11111111`, it correctly outputs 8.
With the input `00000000`, it correctly outputs 0.



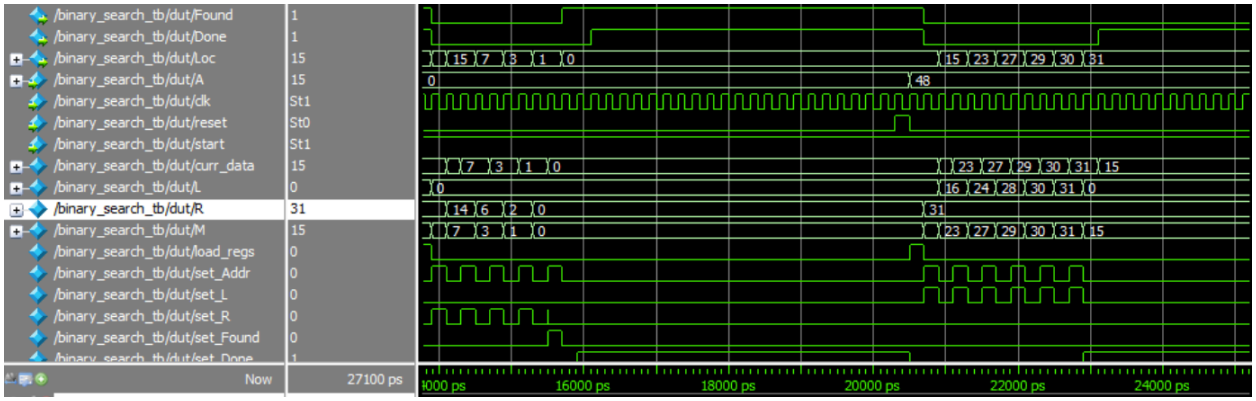
`binary_search` simulation

- First I search for `15`, and find it in one loop.
- Then I search for `31` and find it in 5



- Then I search for `0` and find it in 5

- Finally, I search for `48`, which doesn't exist and tests an overflow edge case. It correctly ends with `Done` but not `Found` asserted



Experience Report

Feedback

This lab was fairly easy compared to the previous ones, but still challenging since I am fairly new to implementing ASMD

Significant Issues

None to speak of

Tips/Tricks

- Write pseudo code and the signals out before making a diagram
- Test all edge cases of the algorithms
- Avoid overflow when finding midpoint with `a + (b - a)/2` trick

Time Spent

~11 hours