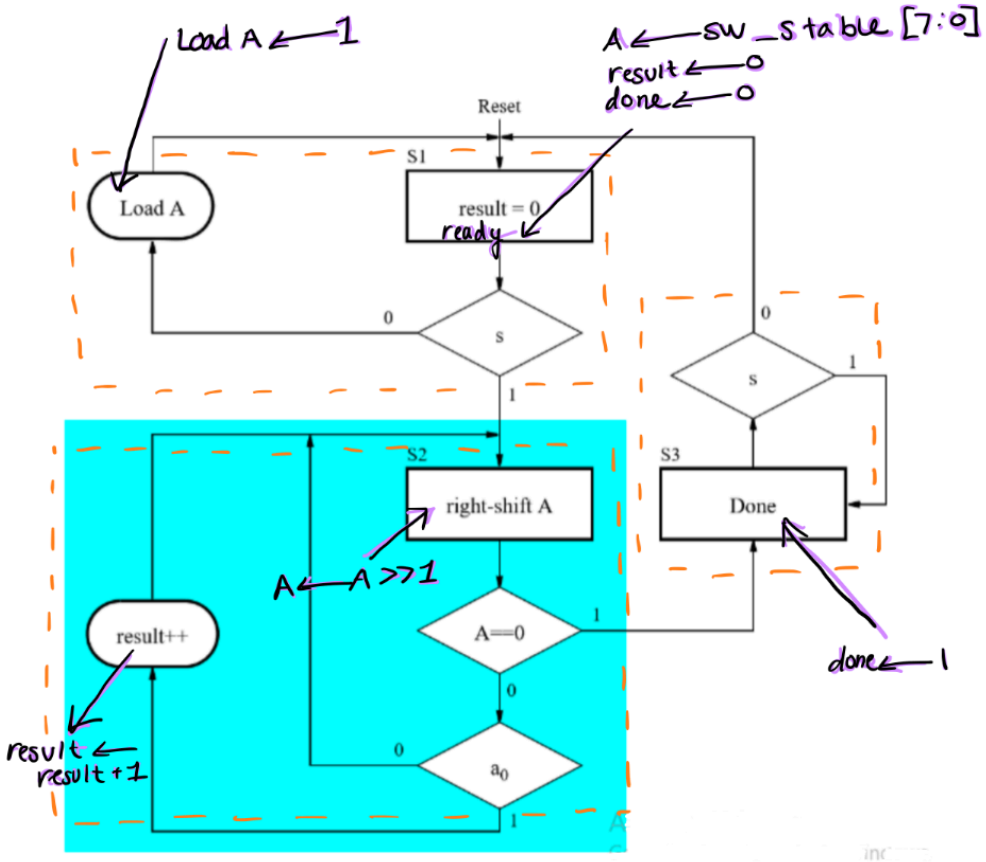
EE/CSE 371
Nov 1, 2025
Lab 4 Report

Lab 4

Design Procedure

This lab implements two algorithms via hardware on an FPGA using ASMD and datapath design with the goal to understand how hardware can execute algorithms directly with inputs and interact with memory blocks. Task 1 implements a bit counting algorithm that counts the number of 1s in an 8 bit input A which is given via switches on the FPGA board. The design displays the result on a 7-segment display while signaling successful completion by signaling an onboard LED. Task 2 implements a binary search algorithm performed on a sorted 32-element memory which is initialized with a Memory Initialization File. The control path manages low, high, and mid pointers to locate an input value of A. If the value is located, the memory location of that value is displayed on HEX's 1 and 0 on the board, alongside a found signal. Both designs handle metastability by synchronizing asynchronous inputs of switches and keys to the 50 MHz clock through two flip flops. Additionally, a top level module allows for switching between the two tasks via a physical switch on the FPGA, SW9.

Overall System Descriptions, ASMD Charts and Block Diagrams



Figures 1: Algorithmic State Machine with Datapath (ASMD) chart for Task 1 bit-counting circuit

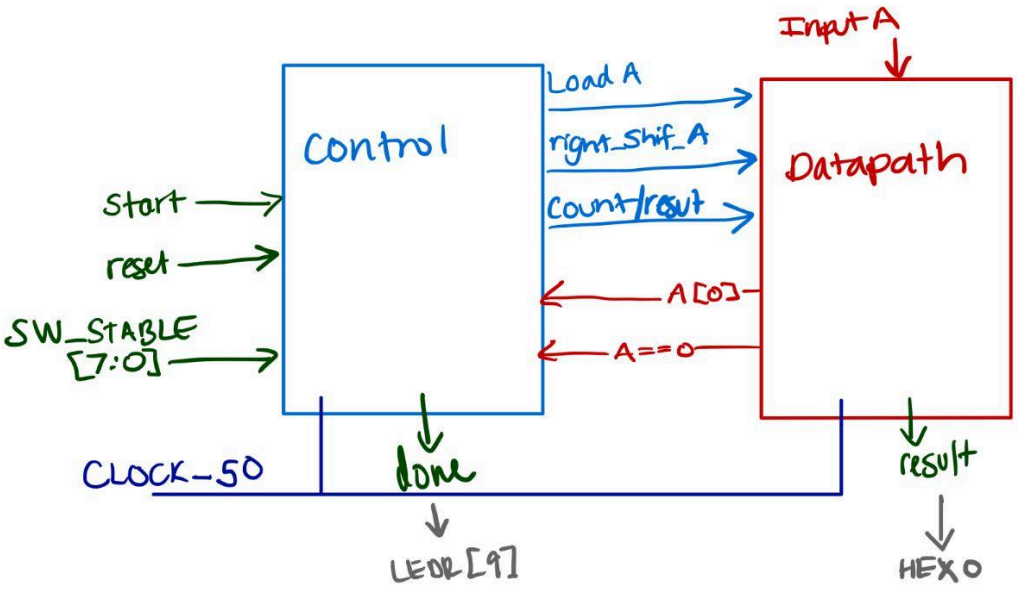


Figure 2: Block Diagram for Task 1 bit-counting circuit

Task 1

The bit_counting_algorithm for Task 1consists of an 8 bit counter which counts the number of 1s in an 8 bit input which is controlled by the user through switch input (SW[7:0]). The number of 1's is displayed on a 7-segment display in HEX0. An LED LEDR[9] turns on to indicate when the algorithm is finished counting. A 50 MHz clock is used for this task, additionally a synchronous reset via KEY[0], and start signal via KEY[3] are used as inputs from the user. Metastability is handled from inputs by synchronizing KEY and SW through two flip-flops utilizing intermediary variables Sync and Stable which are used as input logic. The FSM logic consists of three states. S1 loads the switch's value into an 8 bit register A and then resets the result counter to zero. In S1 LoadA is asserted and a new value of A is stored from the switches, done is set to zero. Once the start signal from the user is true, S2 is entered. S2 shifts the A register right one bit at a time per clock cycle while incrementing a counting counter, our result, if the least significant bit of the value is 1, hence adding to our running total of 1s in the number. Once the full value of A is 0, i.e. all 8 bits are zero, A==0 is asserted and S3 is entered from S2. S3 is the done state, asserts a done=1 flag and toggles LEDR[9], and then waits for the start signal in order to head back to S1 else it stays in S3. The datapath consists of the A register holding the value from the input switches, another 8 bit register holding the shifting value and the result register storing the current count of 1s. Using the same variable A as both input and output into the shifting logic caused unpredictable behavior because the original value was sometimes lost before it was used, thus we used an intermediate register like A_next and A_shifted. The current value is always preserved while the output is computed separately unaffecting current value. The main register A is updated only when the new value is ready, ensuring data integrity.

The module instantiates a separate right_shift_A submodule for the right shift operation and calls a seg7 module to drive the 7-segment display on HEX0. This design separates control and datapath logic, from submodules ensuring clear synchronized counting behavior that is easy to debug. The right_shift_A module is an 8 bit right shift register that loads a new value from InputToLoad when LoadNewVal is high and shifts its current contents right by one bit when LoadNewVal is low. In order to make up for the bits benign shifted out, a 0 is inserted into the most significant bit, moving all bits toward the least significant bit allowing for the counting of the number of 1s in A[0] place.

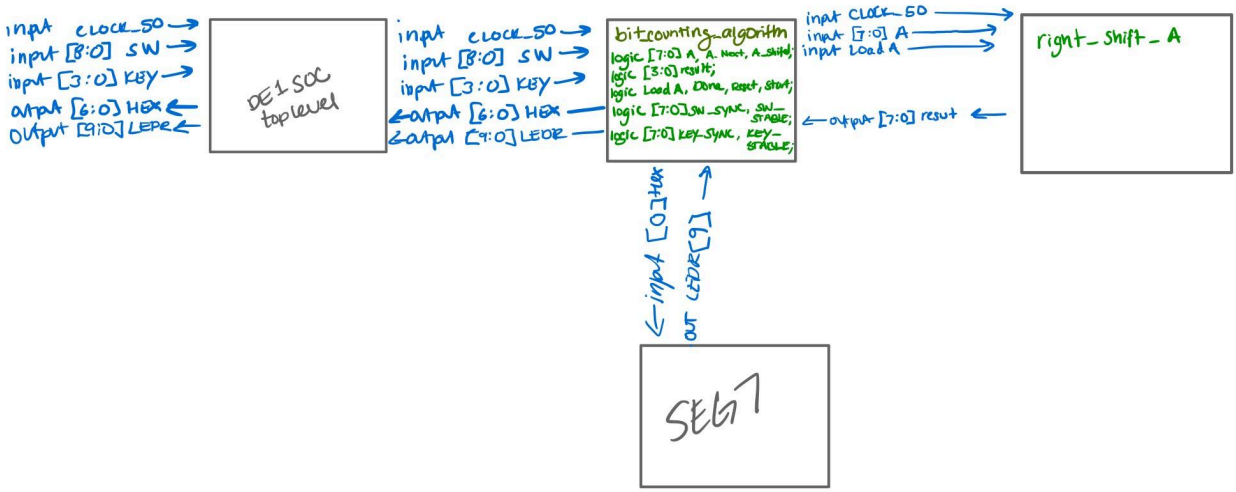


Figure 3: Diagram of Task 1 module layout

Task 2

The binary search algorithm for task 2 is a binary search implementation that takes an 8 bit unsigned integer as an input through the switches SW[7:0] and searches for it in a sorted 32 element array stored in memory. The address where the target value (A) is found is displayed on 2 HEX displays (HEX0 and HEX1) on the board, with HEX1 showing the tens digit and HEX0 showing the ones digit. LEDR9 and LEDR 0 show status, with LEDR9 showing if the search is completed and LEDR0 showing if the given number was found in the array. Additional inputs include a clock (we use the 50hz system clock) and synchronous KEY0 reset and KEY3 start inputs from the user.

Metastability is handled by synchronizing the KEY signals through 2 flip flops using the temp variables start_sync1 and start_sync2, with edge detection producing a single cycle clock pulse.

This module has 7 states:

- idle (S0) waits for the start signal and clears the found flag.
- Init (S1) initializes the target values A from the switches into a register and initializes the search bounds to low = 0 and high = 31.
- Mid_calculate (s2) calculates the midpoint address in search range (low+ high) /2 and registers it as mem_addr to send to memory.

- Mem_wait1 (S3) and mem_wait2 (S4) are 2 wait states that account for the 2 cycle latency of the memory, as the memory has both registered inputs (address, data, and write enable) and a registered output.
- Compare (S5) performs the comparison between target value A and the data retrieved from memory (mem_data), and it updates the search bounds based on the results.
    - If A = mem_data, the value is found and location is saved and it goes to doneState (S6).
    - If A is less than mem_data, the upper_bound is decreased and high = mid-1 to search the lower half.
    - If A is greater than mem_data, the lower bound is increased and low = mid + 1 to search the upper half.
    - If the search space is exhausted (i.e.: low > high), the algorithm goes to doneState irrespective of whether a match is found.
- S6 toggles LEDR9 and goes to idle.

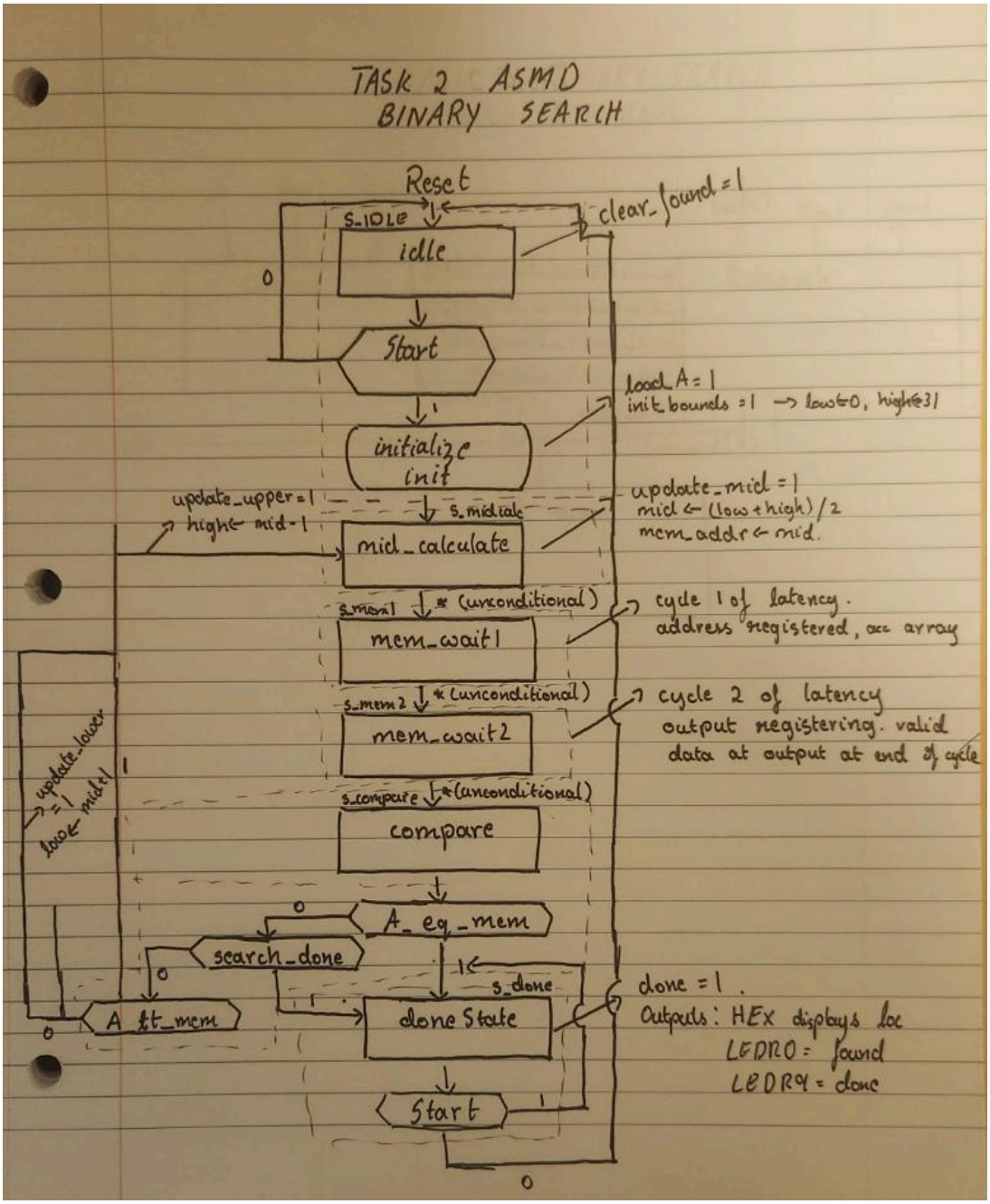


Figure 4: ASMD for the binary search algorithm

The datapath consists of registers for the target value A, search bounds (low and high) memory address (mem_addr), found location (loc) and found flag. It also includes combinational logic for calculating midpoint and performing comparisons (A_eq_mem, A_lt_mem, search_done). The module instantiates a separate ram32x8 memory module containing a sorted array initialized for a .mif file, and calls 2 seg7 module instances to drive HEX0 and HEX1.

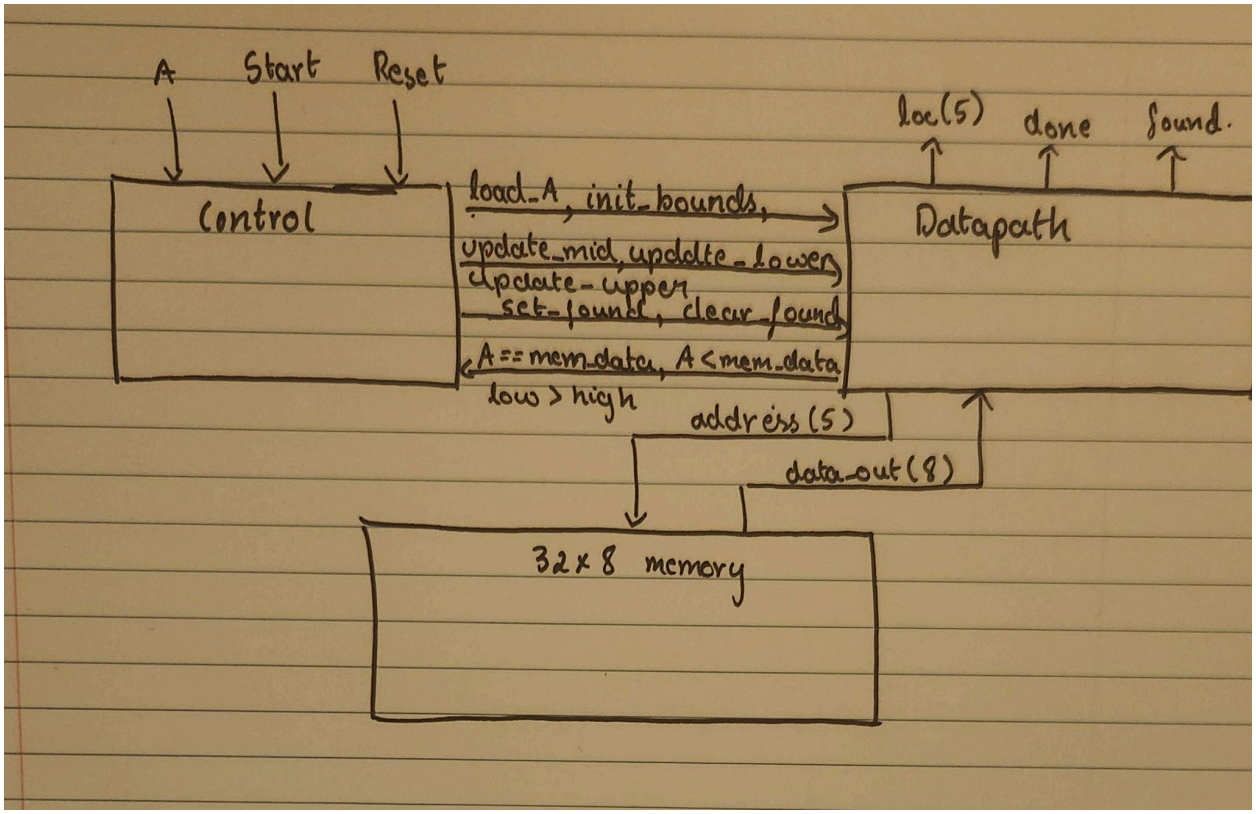


Figure 5: Block diagram for the binary search module

The ram32x8 module is a 32 word by 8 bit single port asynchronous RAM with registered inputs and registered output, created from the IP catalog. It has registers on the address, data input and write enable ports. And a register on the output port.This leads to a 2 cycle latency from when an address is given to when valid data is in the output. The memory is initialized with a sorted array of values (5,10,15,...,160) loaded from the my_array.mif file. During binary search, write is not enabled as it is only a read operation, and memory responds to address changes by outputting data after 2 cycles registered delay, accounted for by the wait states mem_wait1 and mem_wait2 in the fsm.

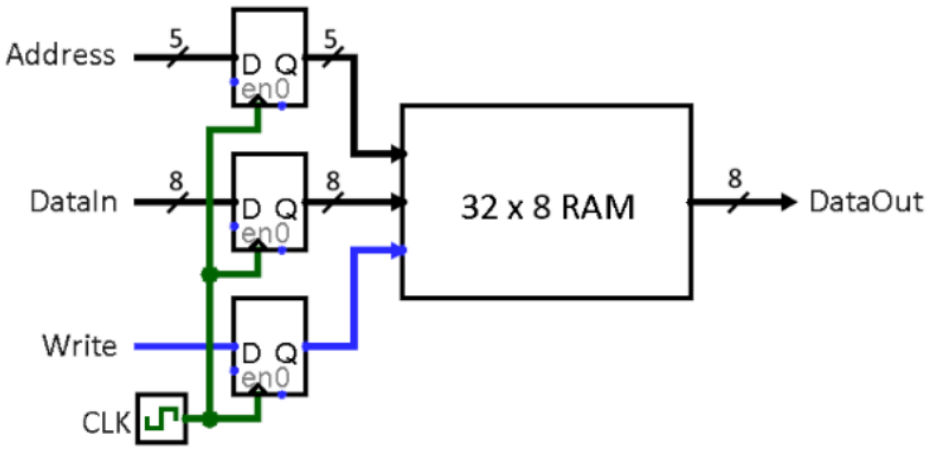


Figure 6: Diagram of 32 X 8 RAM for Task 2

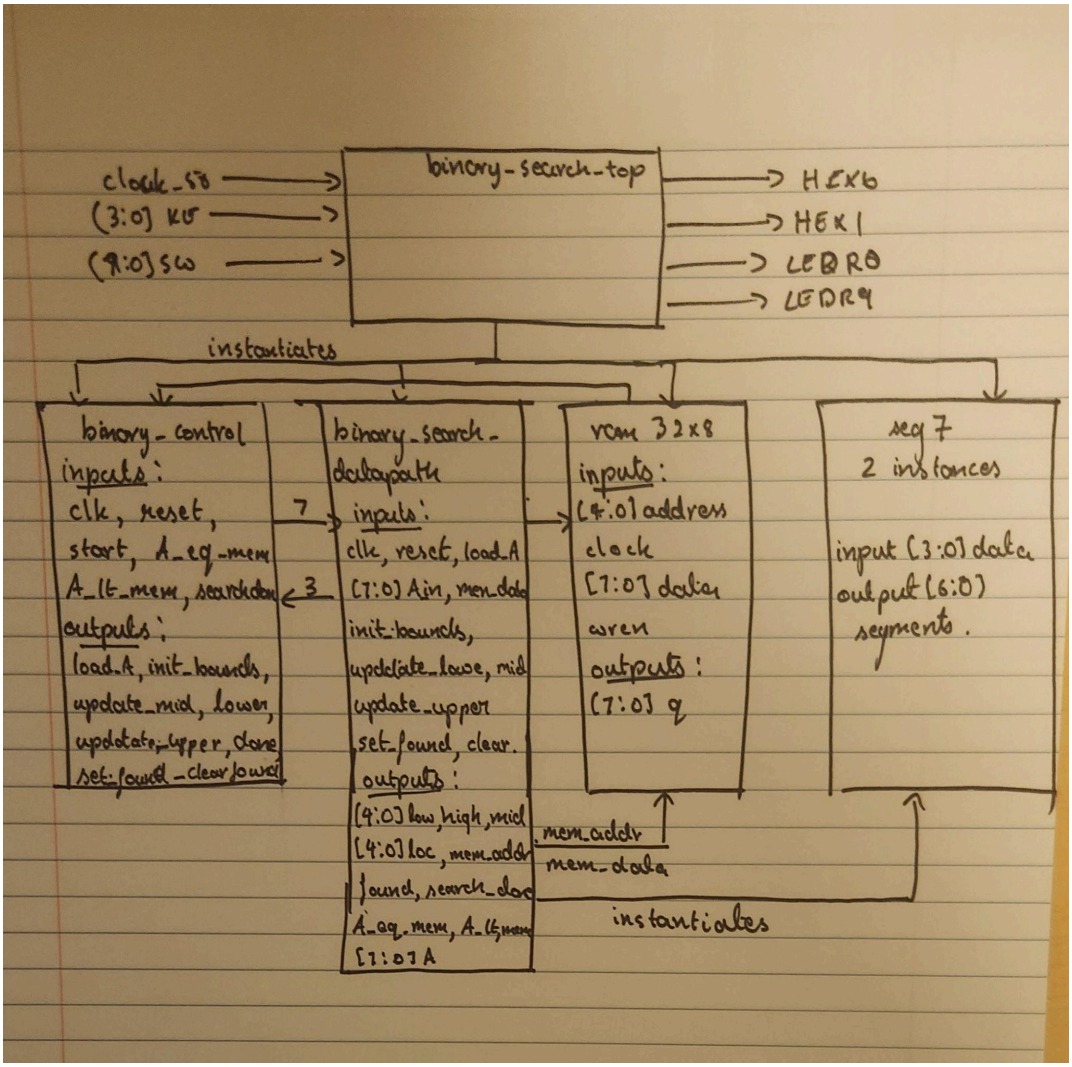


Figure 7: binary search module interactions

Lab_top

This is our top level module that we used to multiplex between tasks 1 and 2. It uses SW9 to change modes between task 1 and 2, with SW9 = 0 being task 1 and SW9 = 1 being task 2. Both tasks operate on the same 8-bit input from switches SW[7:0] and share common control inputs (KEY buttons for reset (KEY0) and start (KEY3)), but produce different outputs depending on the selected mode. It uses a slow clock of 3hz created by using the previously provided clock_divider module, and it instantiates both task modules simultaneously to use this clock. It uses combinational logic to route the outputs to appropriate parts of the board. For task 1, it routes to HEX0 and LEDR9, for task 2, it routes to HEX0, HEX1 LEDR9 and LEDR0. It then turns off remaining HEX's because they aren't used by either task.

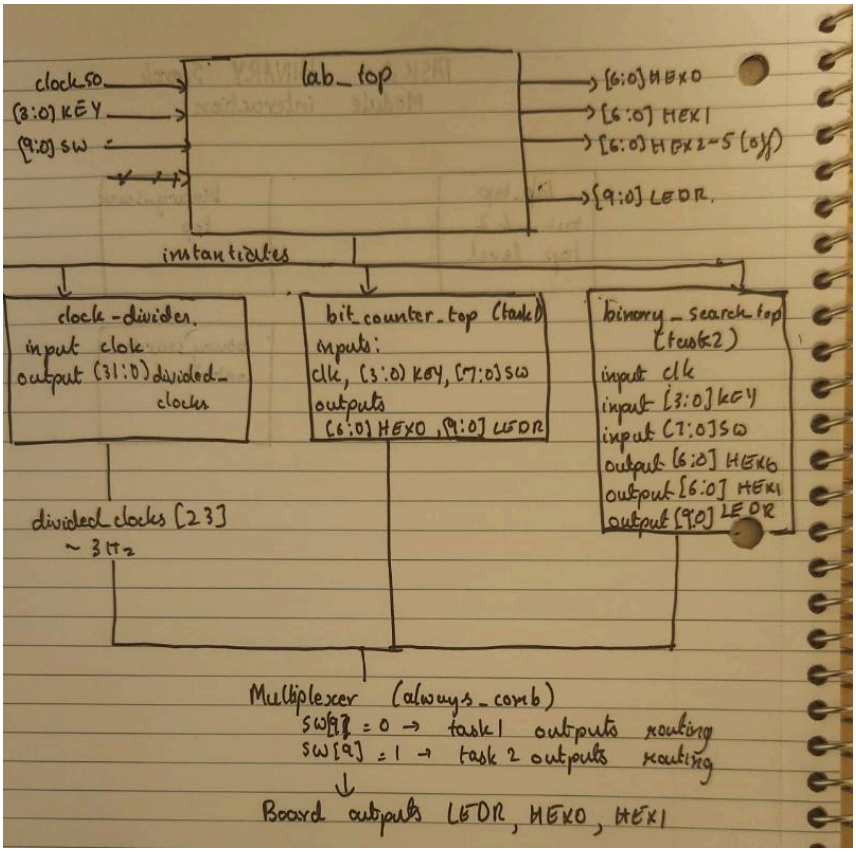


Figure 8: Lab_top module interactions

Results and Testing

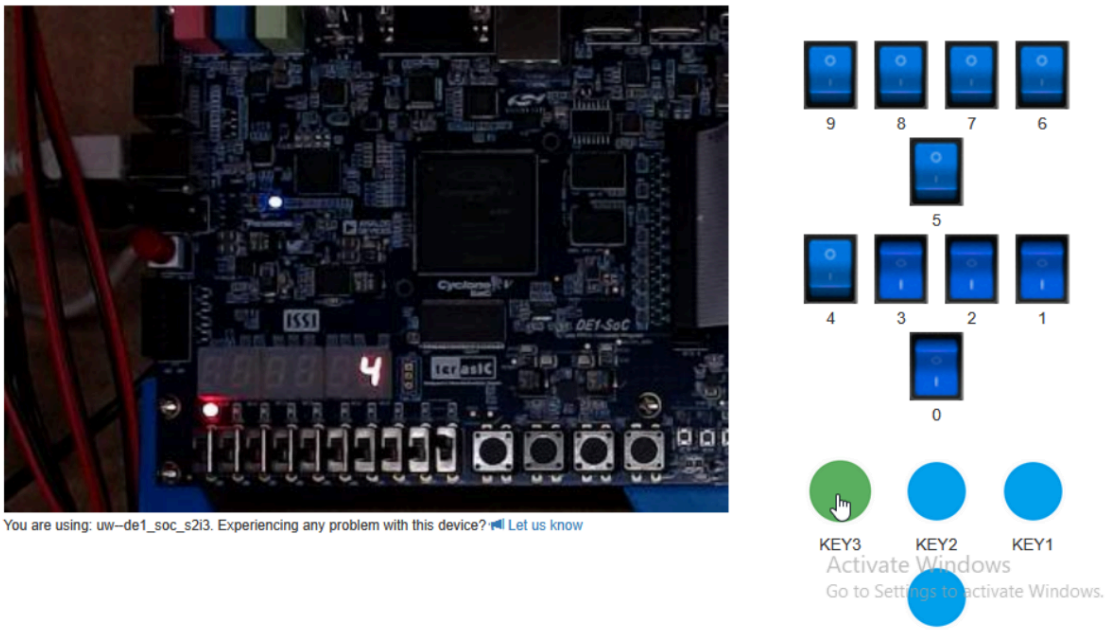


Figure 8: Task 1 deployed on LabsLand FPGA showcasing a 4 displayed on HEX0 and LEDR lighting up when four switches are in the ‘1 bit position’

Task 1 - Bit Counter:

The task 1 bestbench, bit_counting_tb, sets input SW = 10101100 and releases all keys, then asserts reset using KEY[0] = 0, then releases it. Next, it mimics pressing start with KEY[3] = 0 and releases it, which allows the module to count for 50 clock cycles. Next, it changes input to SW = 11010010, repeats the reset and start sequence, and counts another 50 clock cycles to verify output. We can see the LEDs and HEX variables show the number of one's present in each input, 4 for 10101100 and 4 for 11010010 showcasing that the module works as expected and is properly counting.

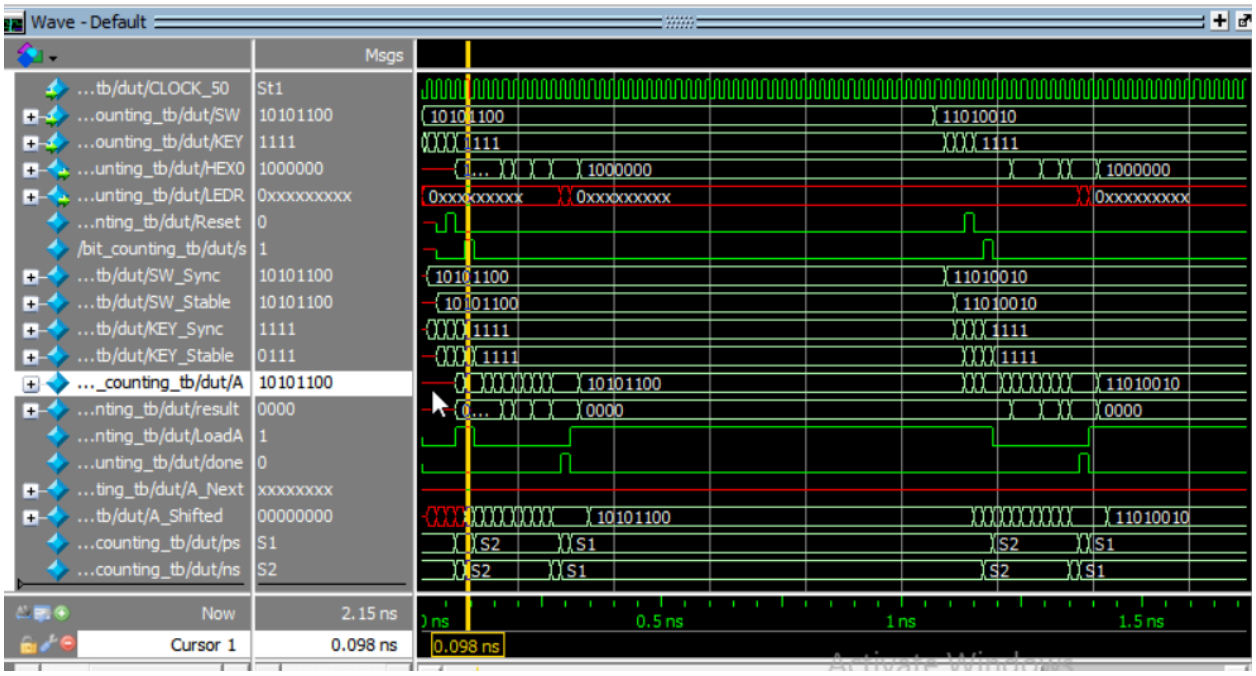


Figure 9: ModelSim Waveforms Task 1 Bit Counter, showing dut A data initialized to 10101100

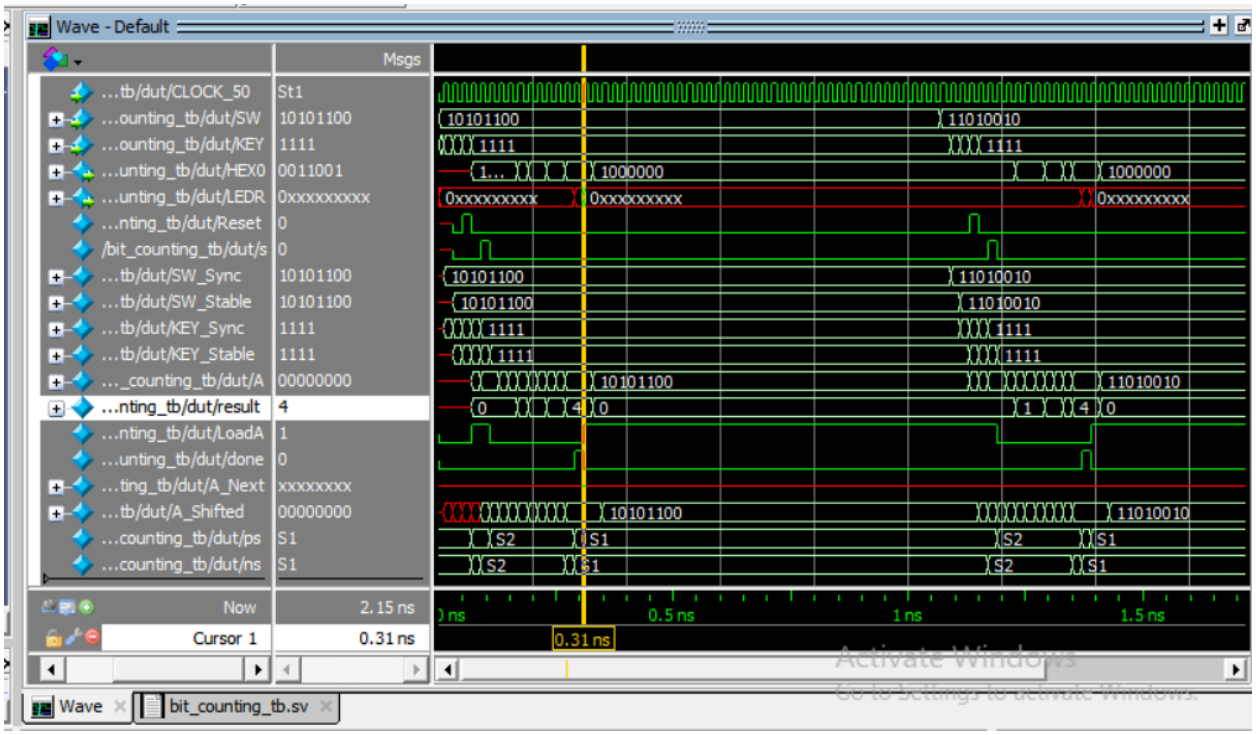


Figure 10: ModelSim Waveforms Task 1 Bit Counter, shows dut A data is all zero's after the shifting sequence and showing that 4 1's were found, showcasing correct behavior

Shift Register Testbench:

This test bench, right_shift_A_tb, loads two 8 bit values, 10101010 and 11001100, into the shift register module and lets the module shift them right over several clock cycles. It uses a 50 MHz clock and toggles LoadNewVal to test loading and shifting behavior. Looking at the wave forms, we can see the bits being shifted to the right, with the final shifted values being comprised of all zeros.

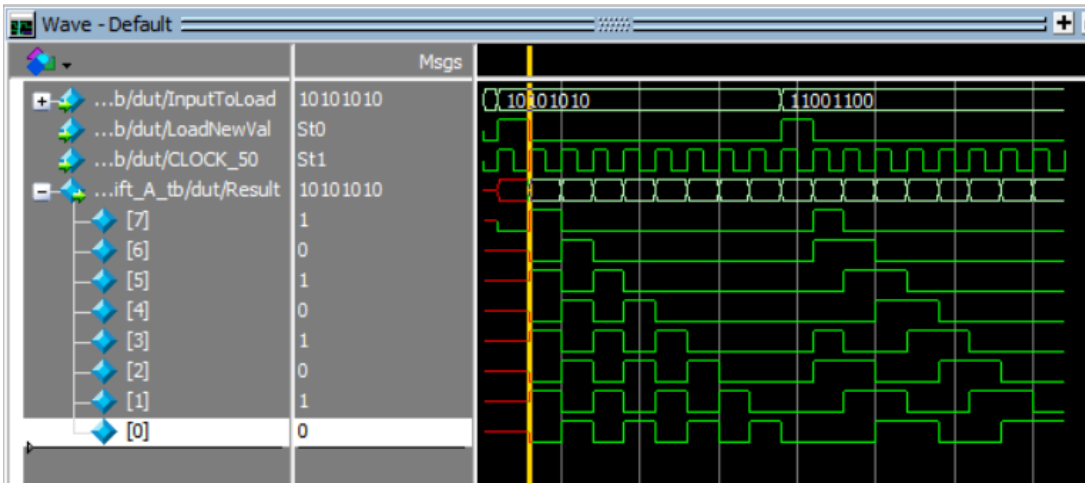


Figure 11: ModelSim Waveforms Task 1 Bit Counter, showing initialized value to 10101010

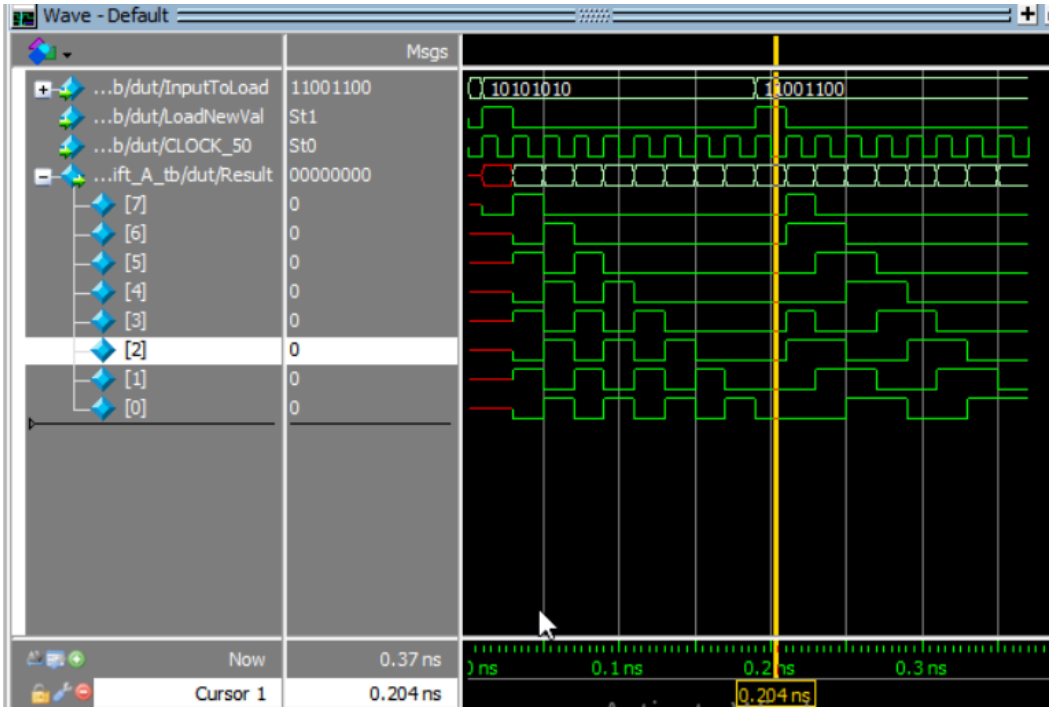


Figure 12: ModelSim Waveforms Task 1 Bit Counter, showing value after right shifting occurs now all zeros

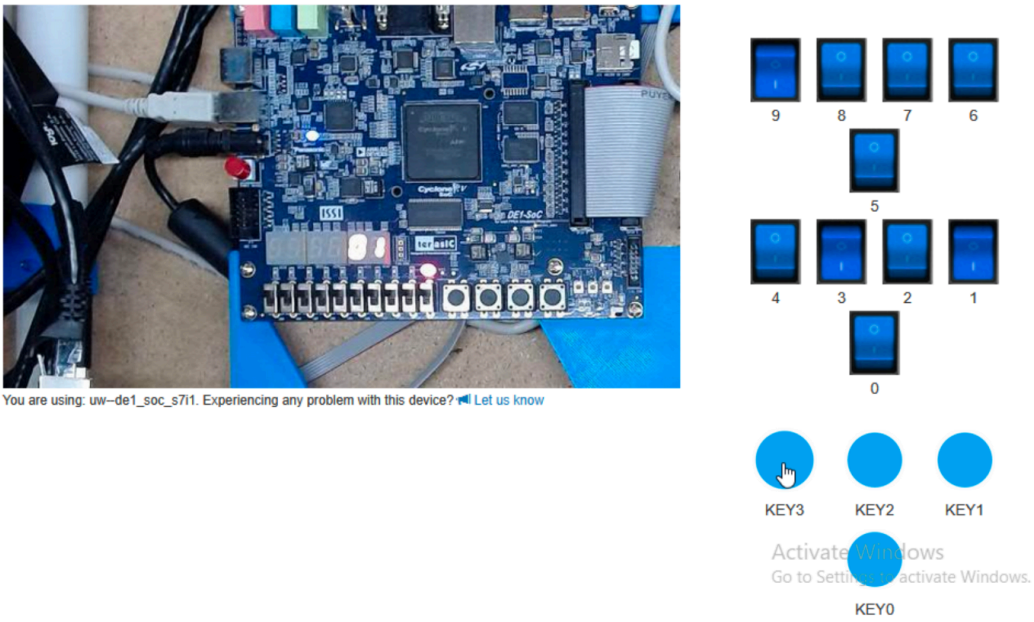


Figure 13: Task 2 deployed on LabsLand FPGA showcasing an address value of 1 displayed on HEX0 and LEDR lighting up when value ‘10’

Task 2 - Binary search:

This testbench, binary_search_tb, applies reset, then searches multiple specific values in binary search module by toggling start signal and setting appropriate SW inputs. It first searches 80, expecting address 15 as a result, then 160 at address 31, 5 at address 0, 50 at address 9, and finally 42, which is not in the array. Each value is loaded via SW. The start signal is pulsed, and the testbench waits to observe results on HEX and LED outputs. From the wave forms you can see…

This screenshot shows the test bench when number 50 was found. At time t = 4690000ps, LEDR9 and LEDR0 are both 1, showing that the search has been completed (LEDR9) and the input target number 50 has been found(LEDR0). We also see HEX0 and HEX1 display the corresponding hex values for the address 9

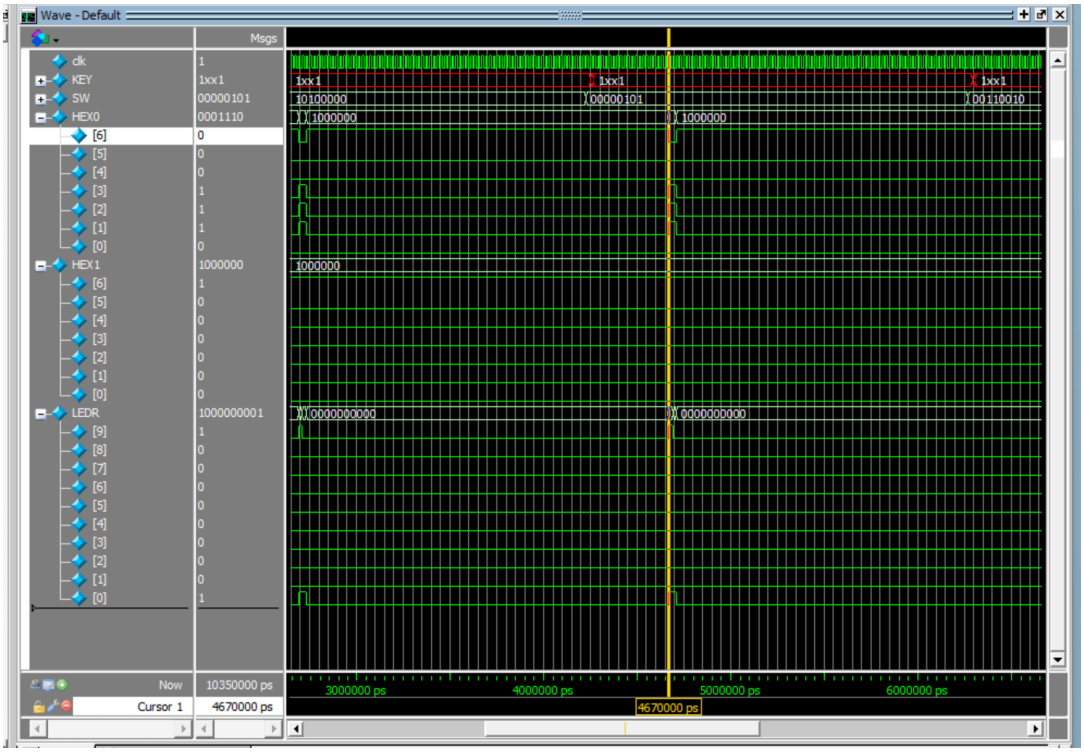


Figure 14: ModelSim Waveforms for binary search - 50 in array showing LEDR9 and LEDR0 on, with HEX0 showing 9 and HEX1 zero

This screenshot of the testbench for binary_search_tb shows where the search was completed, but the number 42 wasn't found as it was not in the array. At time 6710000ps, LEDR9 is 1, showing the search is done, but LEDR0 is 0, saying it wasn't found, and the hex values for both HEX0 and HEX1 are 0

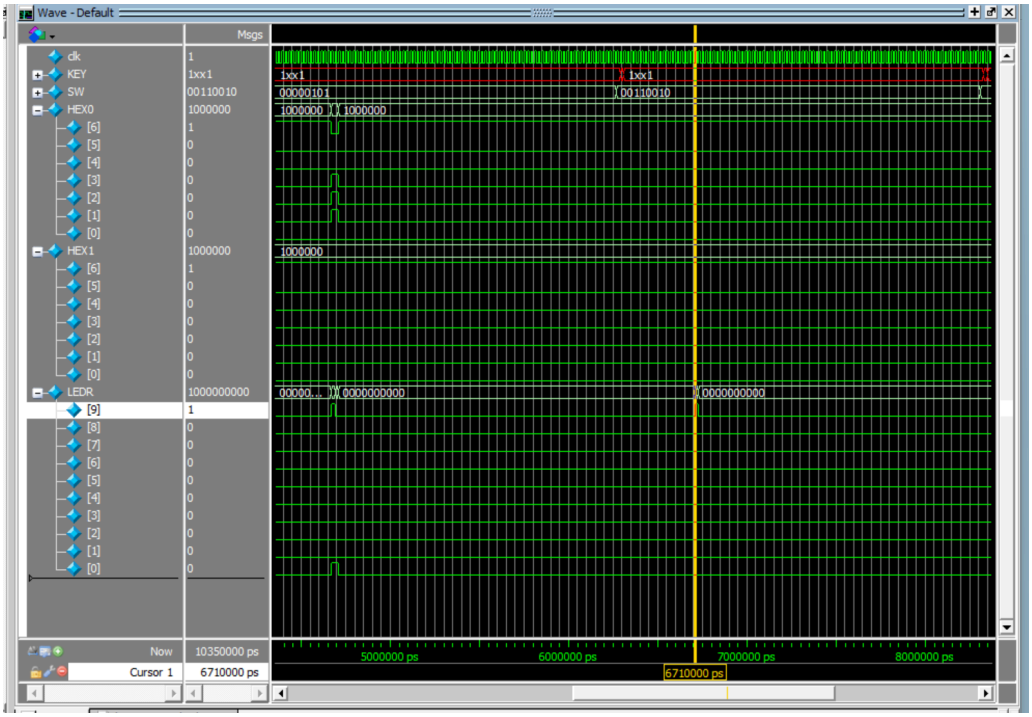


Figure 15: ModelSim Waveforms for binary search - not in array showing LEDR9 on but LEDR0 off, with both HEX values at 0

Flow Summary

Flow Summary

<<Filter>>

Flow Status	Successful - Tue Nov 04 17:45:09 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	DE1_SoC
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	34
Total pins	67
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0

Flow Summary

<<Filter>>

Flow Status	Successful - Tue Nov 04 21:19:27 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	binary_search_top
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	35
Total pins	37
Total virtual pins	0
Total block memory bits	256
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Flow Summaries for Task 1 & 2

Flow Summary

<<Filter>>

Flow Status	Successful - Tue Nov 04 21:21:34 2025
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	DE1_SoC
Top-level Entity Name	lab_top
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	73
Total pins	67
Total virtual pins	0
Total block memory bits	256
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Flow summaries for top level integrated module