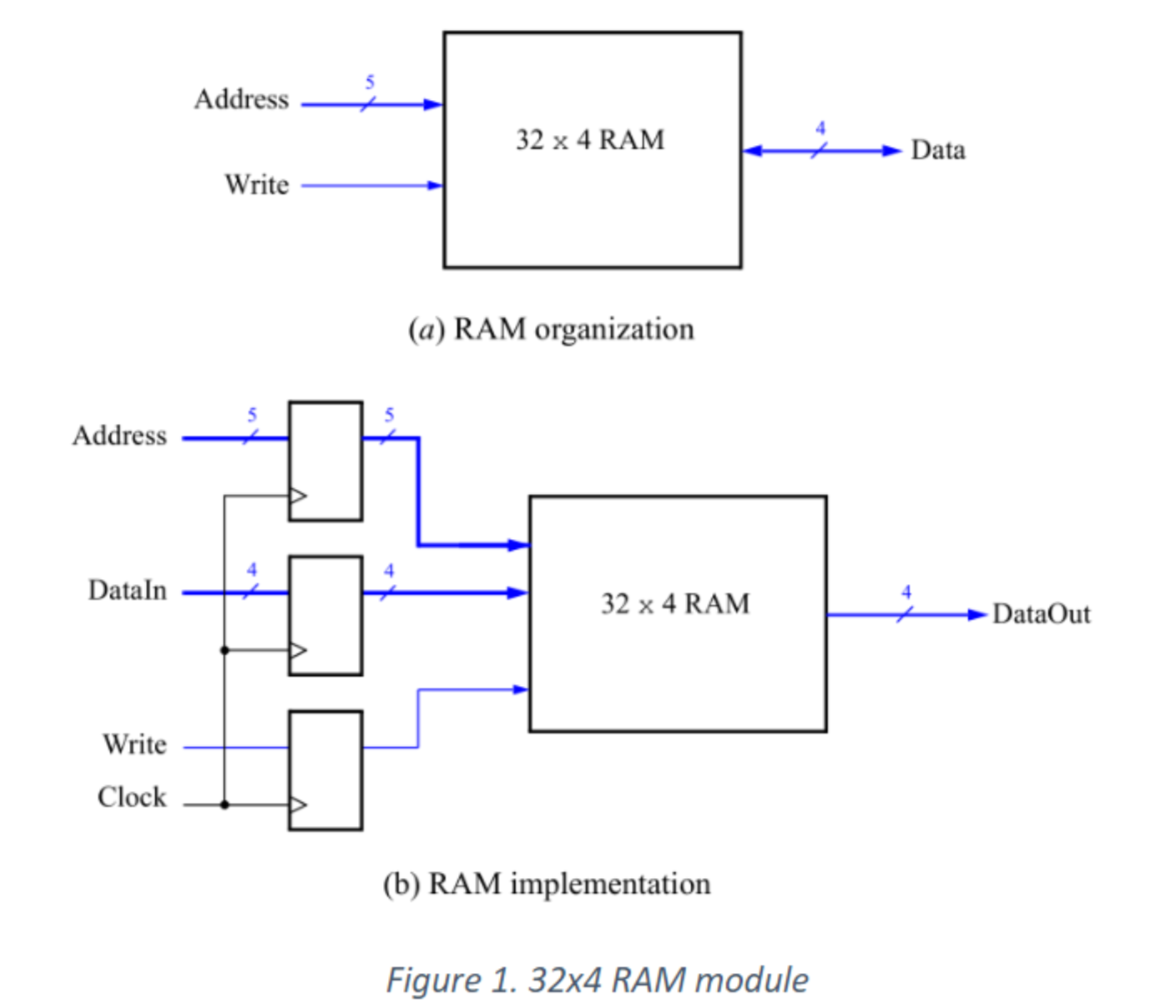
Abstract

In this lab we examined the general issues involved in implementing memory in FPGA. We created a Random Access Memory (RAM) module. We tried to create the in both ways: through IP Catalog (Pre-built modules) and built our own module from scratch. For more user interaction we displayed everything on hex displays in decimal to hexadecimal numbers.

Introduction

We created the Random Access Memory (RAM) module as shown in the Figure 1. It contains 32 four-bit words, which are accessed using five-bit address port, a four-bit data port and a write control input.



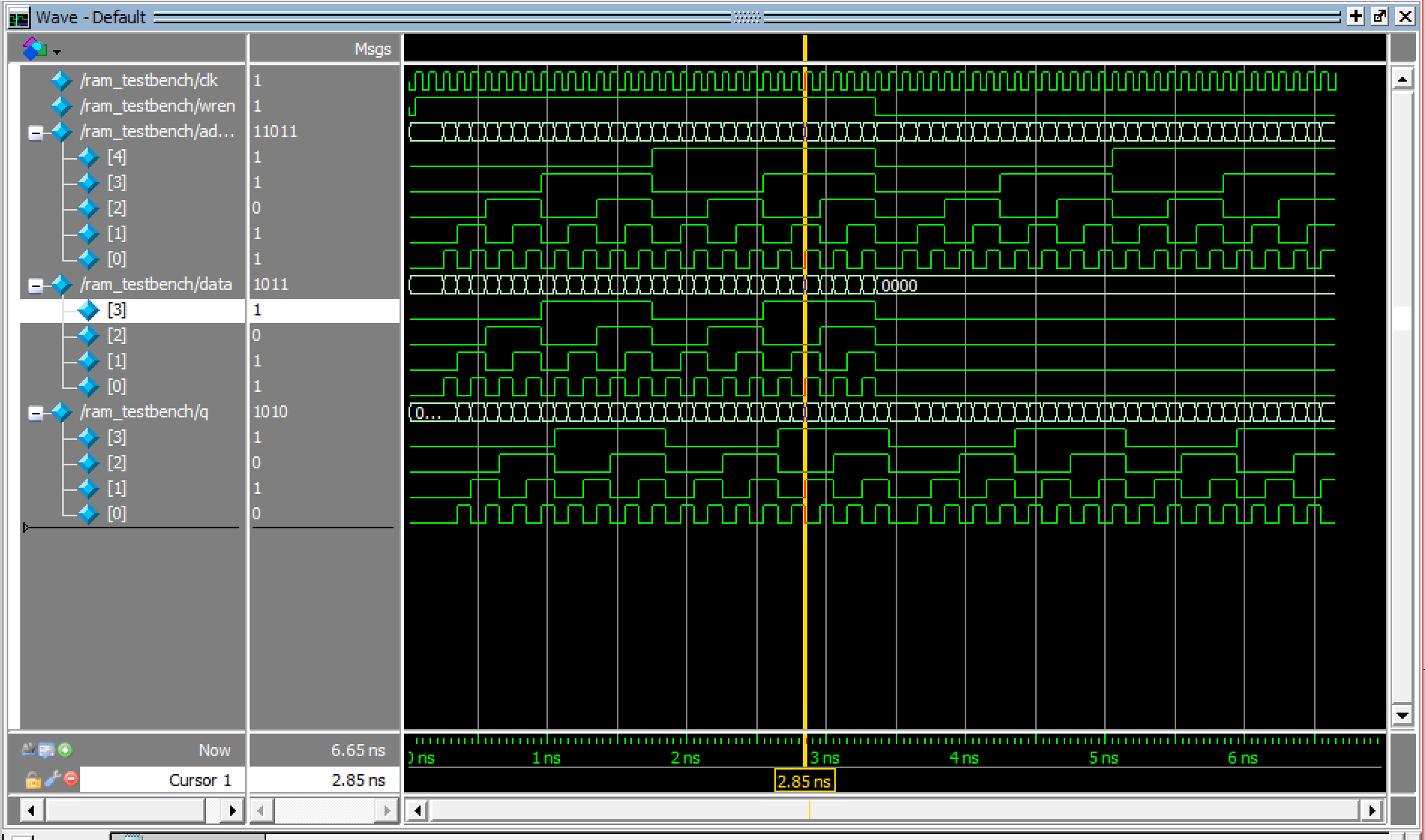
Procedures

* Task 1
  + We built a one port RAM module with the pre-built modules in libraries.
  + We implemented the memory as shown in Figure 1b.
  + Wrote a test bench to test all types of input and saw desired output with 128 bits being used by the FPGA memory blocks.
* Task 2
  + Used the same module from task one and connected to the switches and used hex display to display address, data, and q. W used KEY 0 a clock input.
  + Wrote a test bench for the display and DE1\_SoC to test all types of input and saw desired output.
* Task 3
  + This time instead of creating a memory module using the IP Catalog, we implemented the required memory by specifying its structure in System Verilog code. We did it by defining a multidimensional array (32 x 4), which has 32 words with 4 bits per word.
  + Connected all the switches and keys to the input of the RAM module and wrote the test benches for every module.
* Task 4
  + The RAM block in Figure 1 has a single port that provide the address for both the read and write operations. For this task we created a different type off memory module, in which there is one port for supplying the address for a read operation and a separate port that gives the address for a write operation.
  + The better user interaction we put write address on HEX5-4, write data on HEX1, RAM contents in HEX0 and read address on HEX3-2.
  + The inputs were same as the previous tasks, SW 3-0 to provide write data, SW 8-4 for write address, SW 9 for write enable and KEY 0 for the reset.

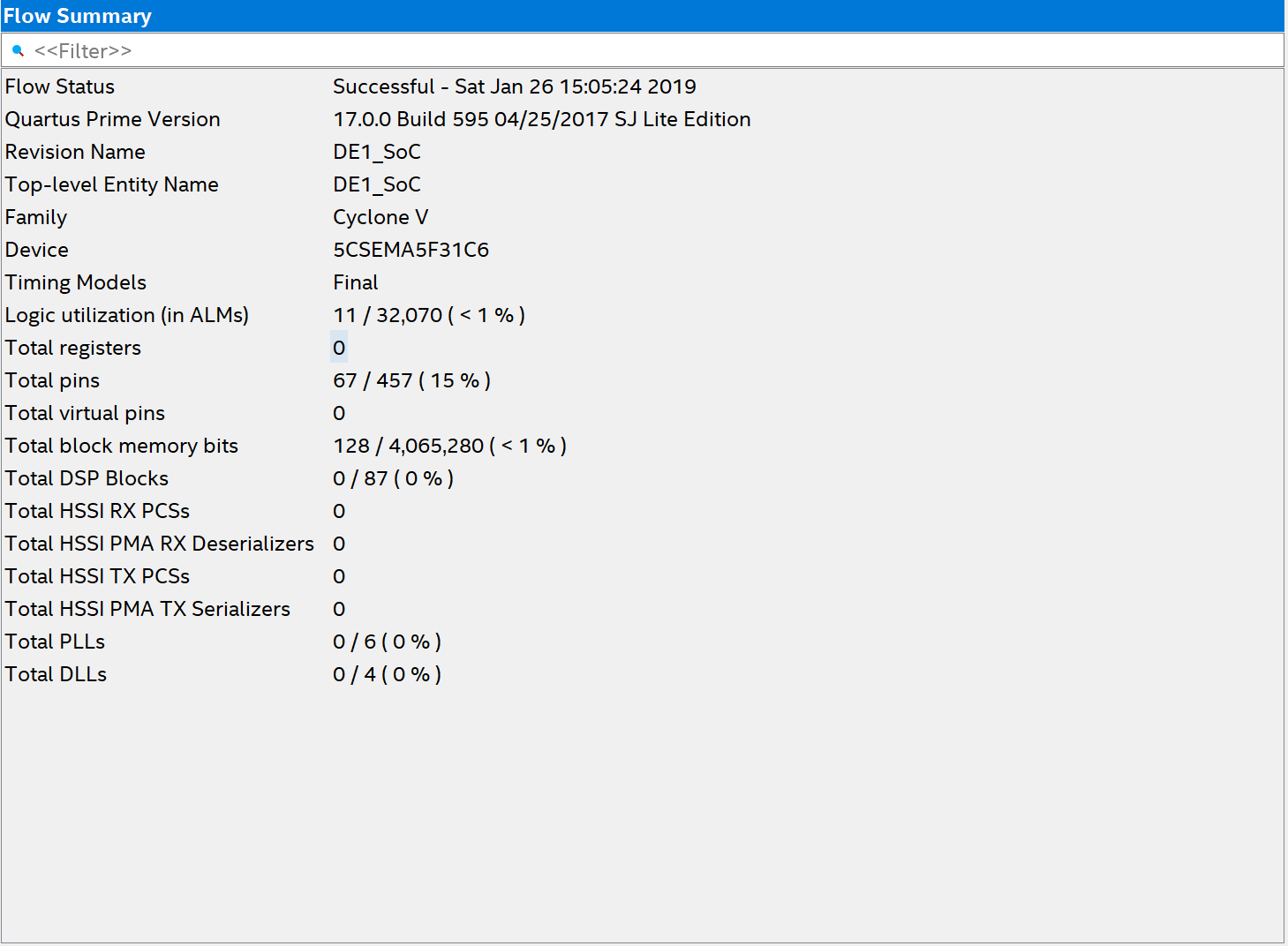
Results

The car enters from the right. So, SW[0] is a and SW[1] is b. Also, the reset switch is SW[9].

* Task 1:
  + ram\_testbench waveform

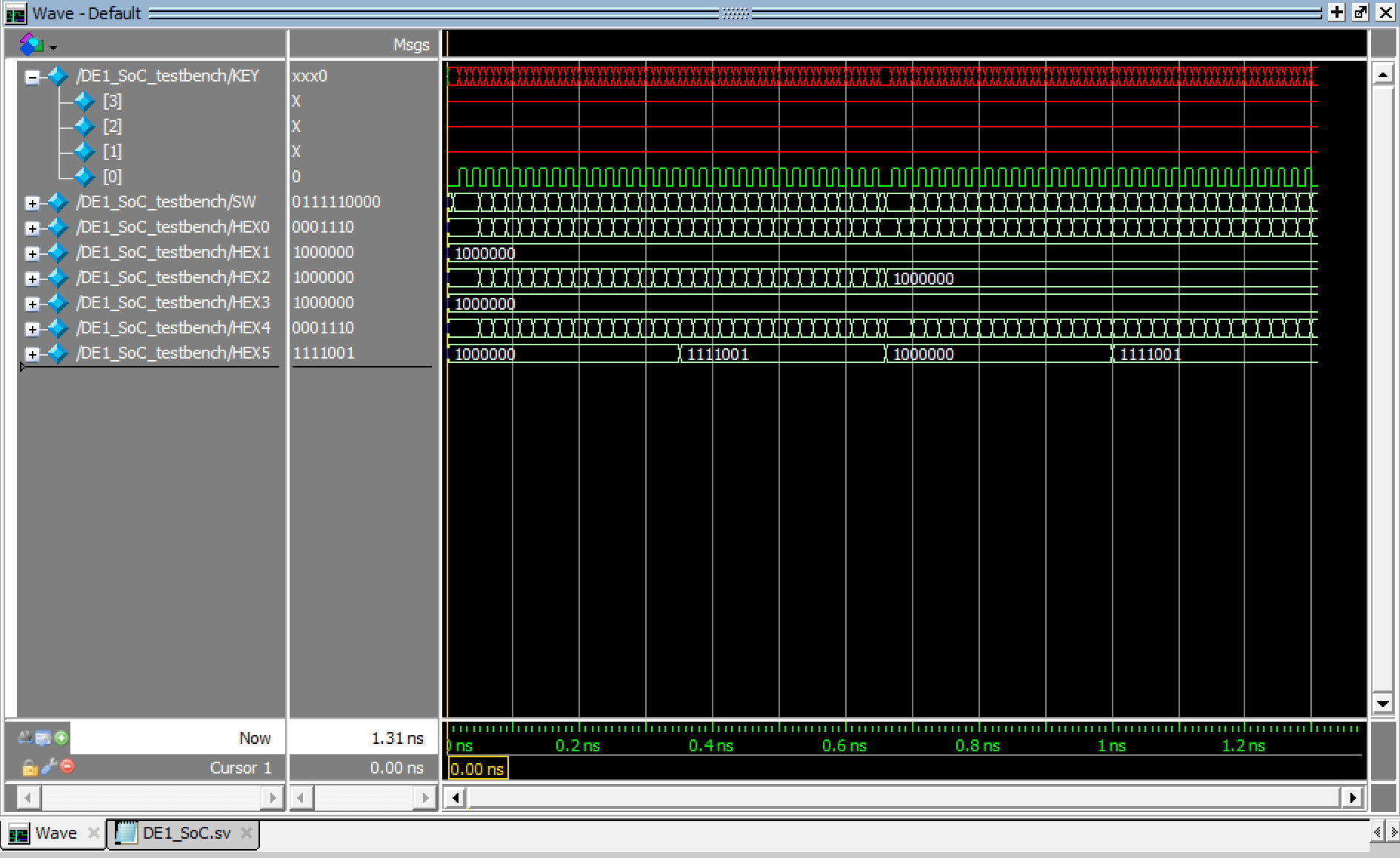


* + Task 1 and 2 were in the same Quartus project so their flow summary is the same.

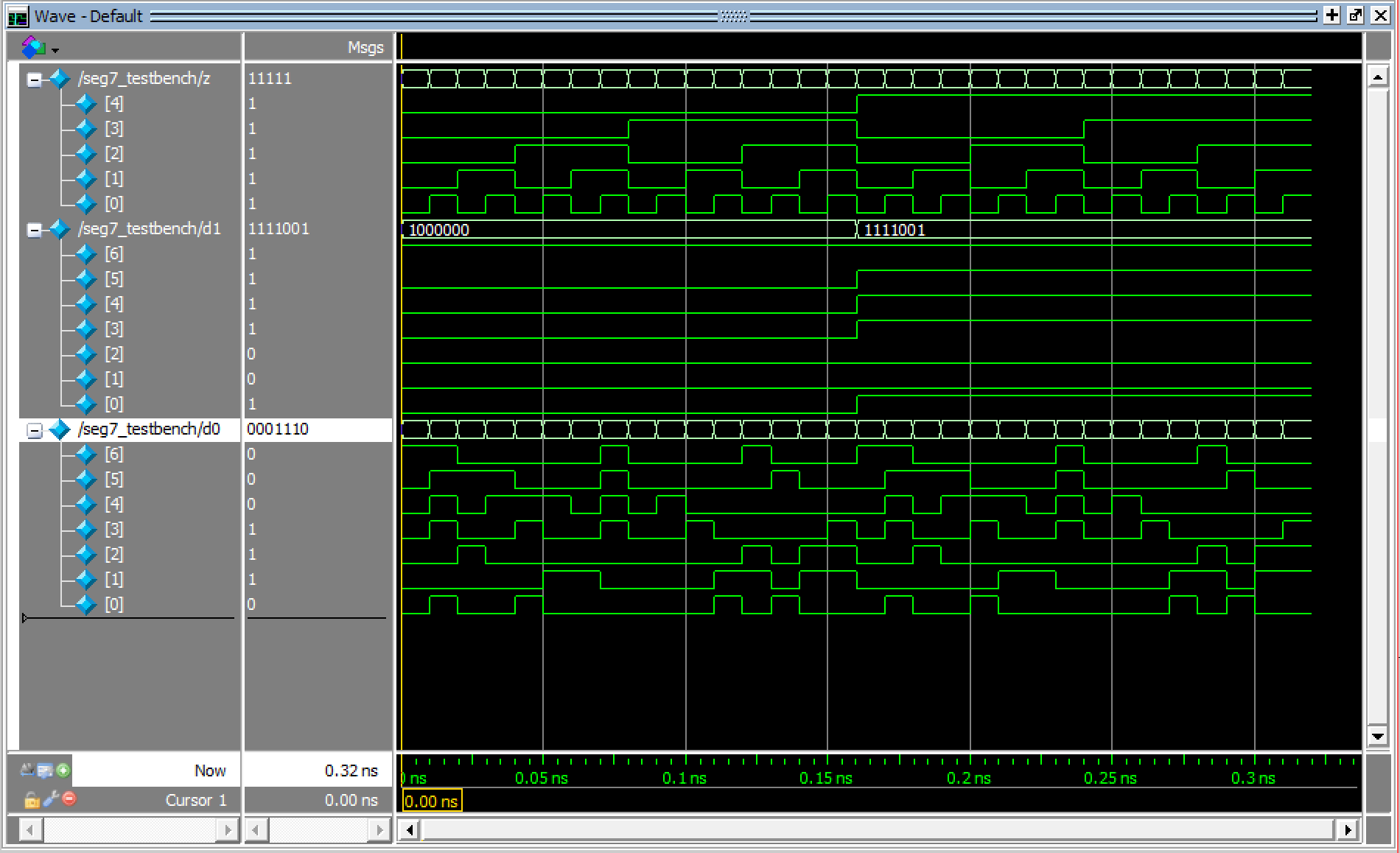


* Task 2:

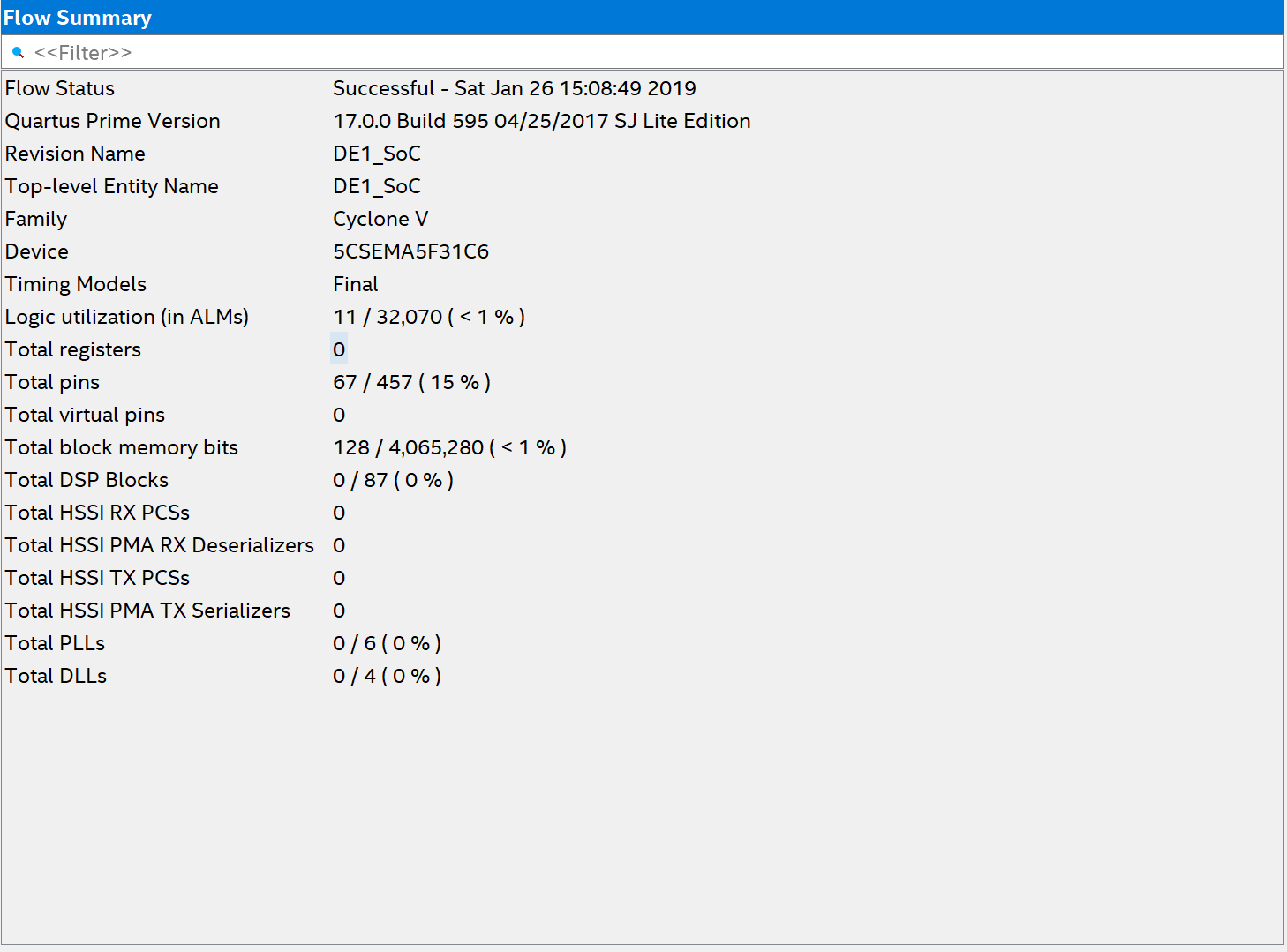
* + DE1\_SoC\_testbench waveform



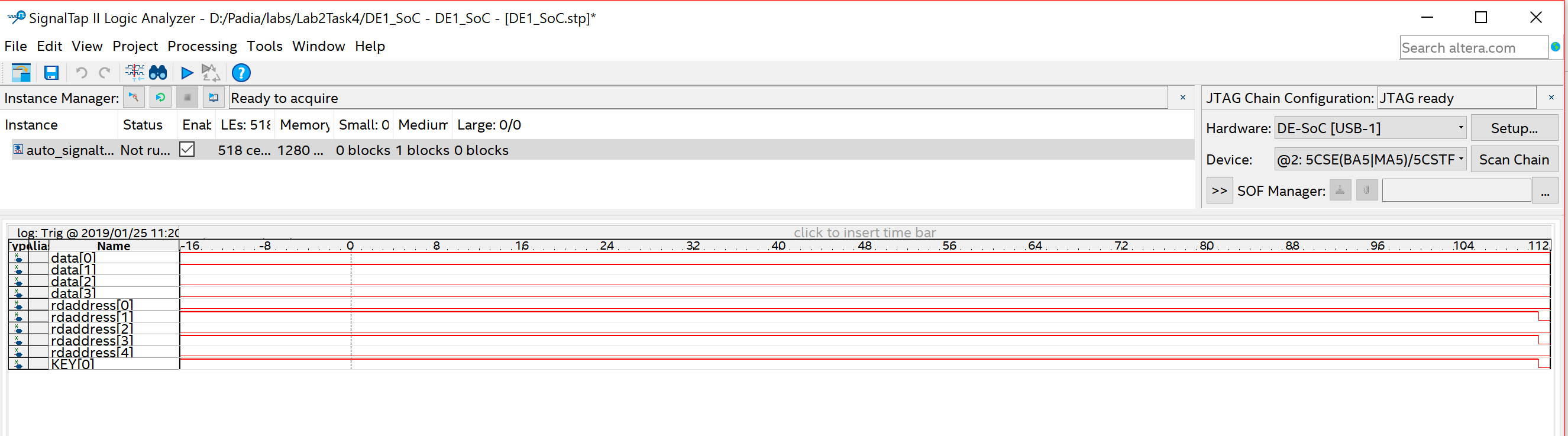
* + seg7\_testbench waveform



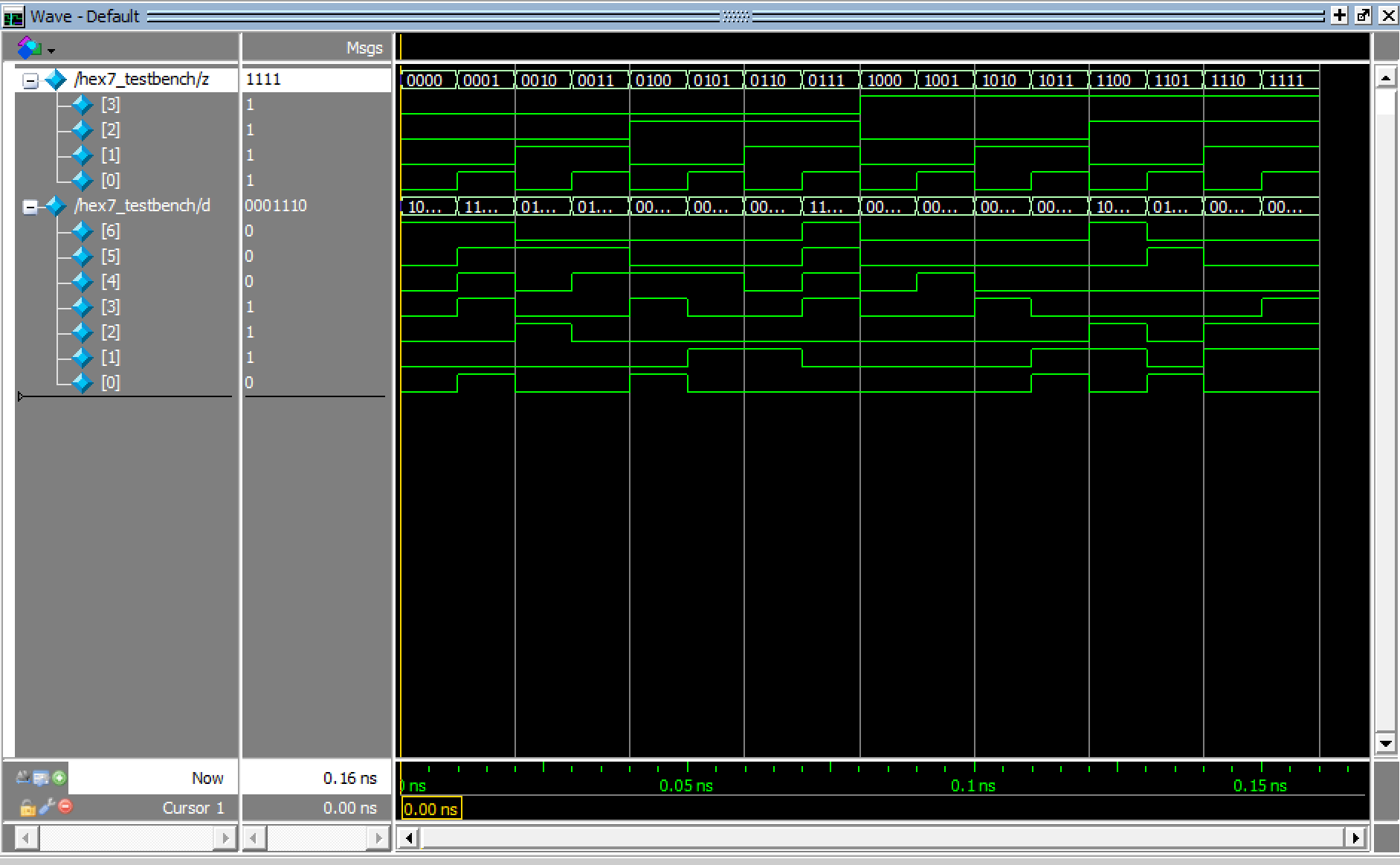
* Task 3
  + All the waveforms were the same as the previous task because we only the ram32x4.sv file which we made from the scratch. The ram32x4\_tetbench has the same waveform as the ram\_testbench in task 1.
  + Flow Summary



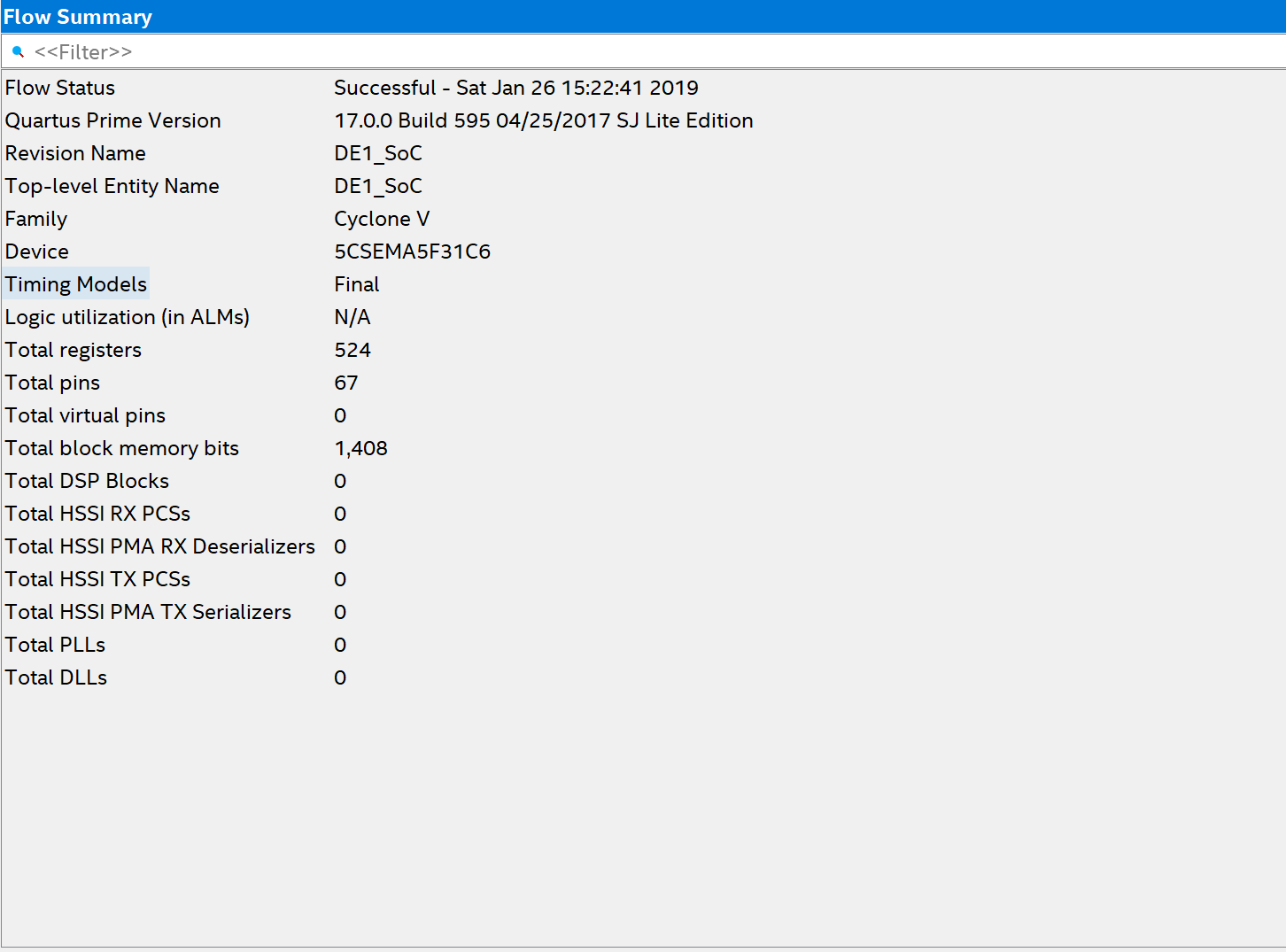
* Task 4
  + Signal Tap II



* + Hex7\_testbench



* + Flow summary



Analysis

This lab covered general issues involved in implementing memory in FPGA. We learned to create a Random Access Memory (RAM) module through IP Catalog (Pre-built modules) and built our own module from scratch.

Conclusion

This lab took a total time of 18 hours including reading, planning, design, coding, debugging, testing etc. Problems were faced during creating a Signal Tap II functionality and debugging Verilog. In this we learnt how to make a RAM module in FPGA using prebuilt modules. Also, we understood what a RAM module is and its basic functionality.