HO CHI MINH UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS

LAB 3 REPORT

SUBJECT: LOGIC SYNTHESIS (EE4449)

CLASS: TT01

SEMESTER: 231

INSTRUCTOR: Mr. NGUYỄN TUẤN HÙNG

STUDENT NAME: NGUYỄN TRƯỜNG PHÚ

STUDENT ID: 2051058

TASK:

The project is to design an asynchronous FIFO module and verify it using assertions. The memory module should be able to store 24-bit data. The memory should be able to store 32 data at most. When the memory is full, the memory should not accept any more data. When the memory is empty, the memory should not be able to read any data.

After you design the FIFO Controller and it's memories, you should write a testbench to verify it. The testbench should be able to generate all possible scenarios and check if the FIFO controller works correctly in all of them. If it does, then the testbench should pass. Otherwise, it should fail. You should have at least 10 test cases.

Lastly, create a top module name top_lab3 to connect the LFSR, FIFO controller, Memory, BCD7SEG, and a clock divider. Use the CLOCK_2SEC 0.5Hz clock that you create your self (write a clock divider module) to drive the LFSR and FIFO controller clkw. USE the CLOCK_1SEC 1 Hz that you create your self (write another clock divider module) to drive the FIFO controller clkr. Connect the output data of the memory (24-bit) and the fifo_len signal to HEX5 to HEX0 (You should use SW0 to toggle between the two). Use KEY0 to turn on write enable for the fifo controller and KEY1 for read enable. KEY2 for reset. Use LEDR0 and LEDR2 to turn on when the FIFO is full and empty, respectively.

CODES:

For this Lab, except for the Altera's built-in 2-port RAM, I have coded everything else from scratch:

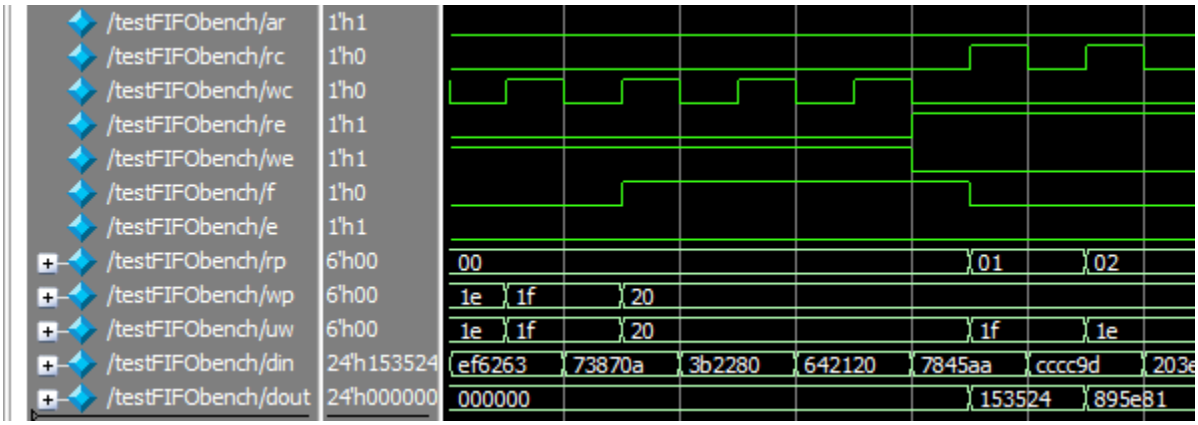
File Name	Description
`binto7seg.sv`	Converts a 4-bit number to a 7-segment display.
`clock_1sec.sv`	Generates a 1 Hz signal.
`clock_2sec.sv`	Generates a 0.5 Hz signal.
`controller.sv`	The brain of this project, where pointers, flags, and used words, are calculated.
`LFSR.sv`	24-bit Linear Feedback Shift Register.
`segsel.sv`	A simple multiplexer to select what to display.
`segssel.sv`	Wrapper for `segsel.sv` to take nibbles by nibbles directly.
`testFIFObench.sv`	Official testbench for this project. Contain 10 assertion cases.
`dual_RAM.v`	The heart of this project, where 24-bit data is stored and read.
`fFFf.v`	Connecting `dual_RAM.v` and `controller.sv` together.
`top_lab3.v`	Official top-level wrapper for `fFFf.v` to connect to DE10.
`altera_mf.v`	Altera megafunctions for testbench compatibility.

TESTBENCH CASES:

Here is the list of 10 test cases I included in my testbench:

Case Number	Case Name	Case Description
1	-	Full and Empty flag must **NEVER BOTH** be 1.
2	`on_reset`	No operation can be carried when reset is raised.
3	`write_op`	On write operation, both flags should be 0, write pointer and used words should increment by 1, and read pointer should stay the same.
4	`read_op`	On read operation, both flags should be 0, write pointer should stay the same, read pointer should increment by 1, and used words should decrement by 1.
5	`dont_read_if_empty`	Nothing should happen when attempting to read from an empty FIFO.
6	`dont_write_if_full`	Nothing should happen when attempting to write to a full FIFO.
7	-	Data **MUST** be First In, First Out.
8	`both_ops_same_time`	When neither empty nor full, attempting to read and write at the same time should increment both write and read pointer by 1, and used words should stay the same.

9	`both_ops_when_empty`	When empty, attempting to read and write at the same time should be the same as Test Case 3.
10	`both_ops_when_full`	When full, attempting to read and write at the same time should be the same as Test Case 4.

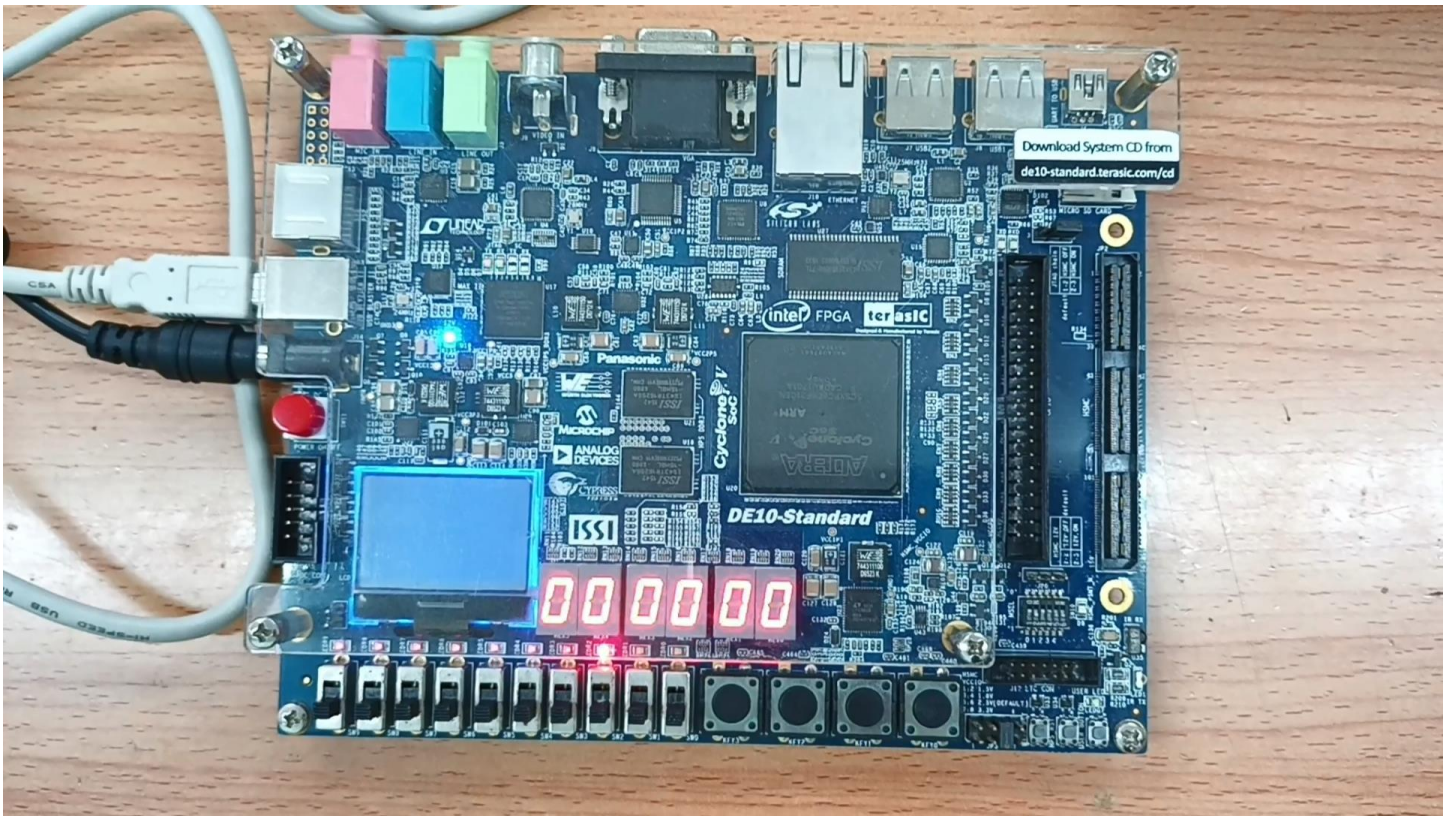



```
# @133: Test Case 1 - Full and Empty flag must NEVER BOTH be 1: PASSED
# @133: Test Case 4 - On read operation, both flags should be 0, write pointer should stay the same, read pointer should increment by 1, and used words should decrement by 1: PASSED
# @134: Test Case 7 - Data MUST be First In, First Out: PASSED
# @137: Test Case 5 - Nothing should happen when attempting to read from an empty FIFO: PASSED
# @143: Test Case 9 - When empty, attempting to read and write at the same time should be the same as Test Case 3: PASSED
# @145: Test Case 8 - When neither empty nor full, attempting to read and write at the same time should increment both write and read pointer by 1, and used words should stay the same: PASSED
# @211: Test Case 10 - When full, attempting to read and write at the same time should be the same as Test Case 4: PASSED
```

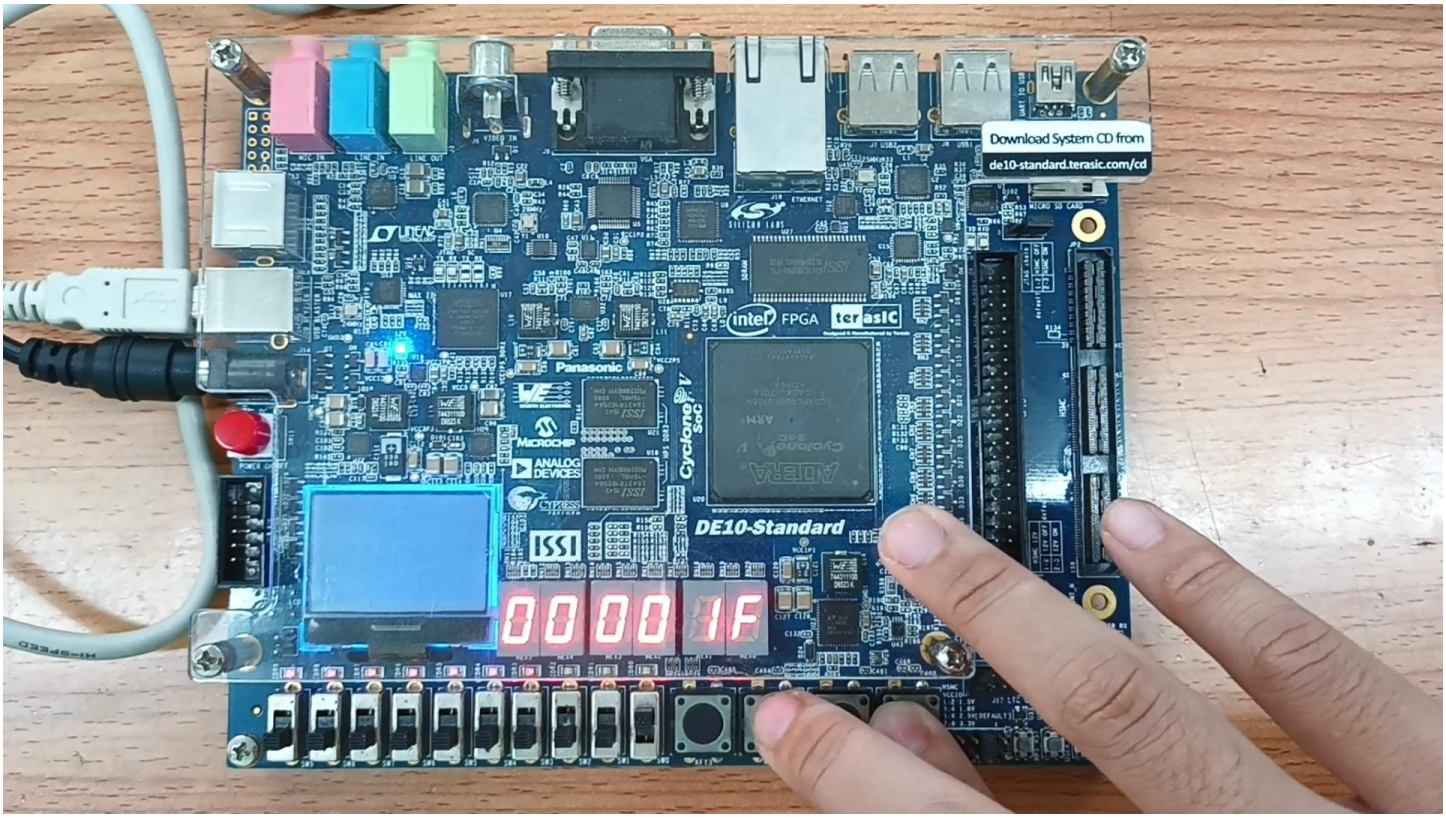
A part of my testbench's Waveform and Console Output.

RESULT ON DE10:

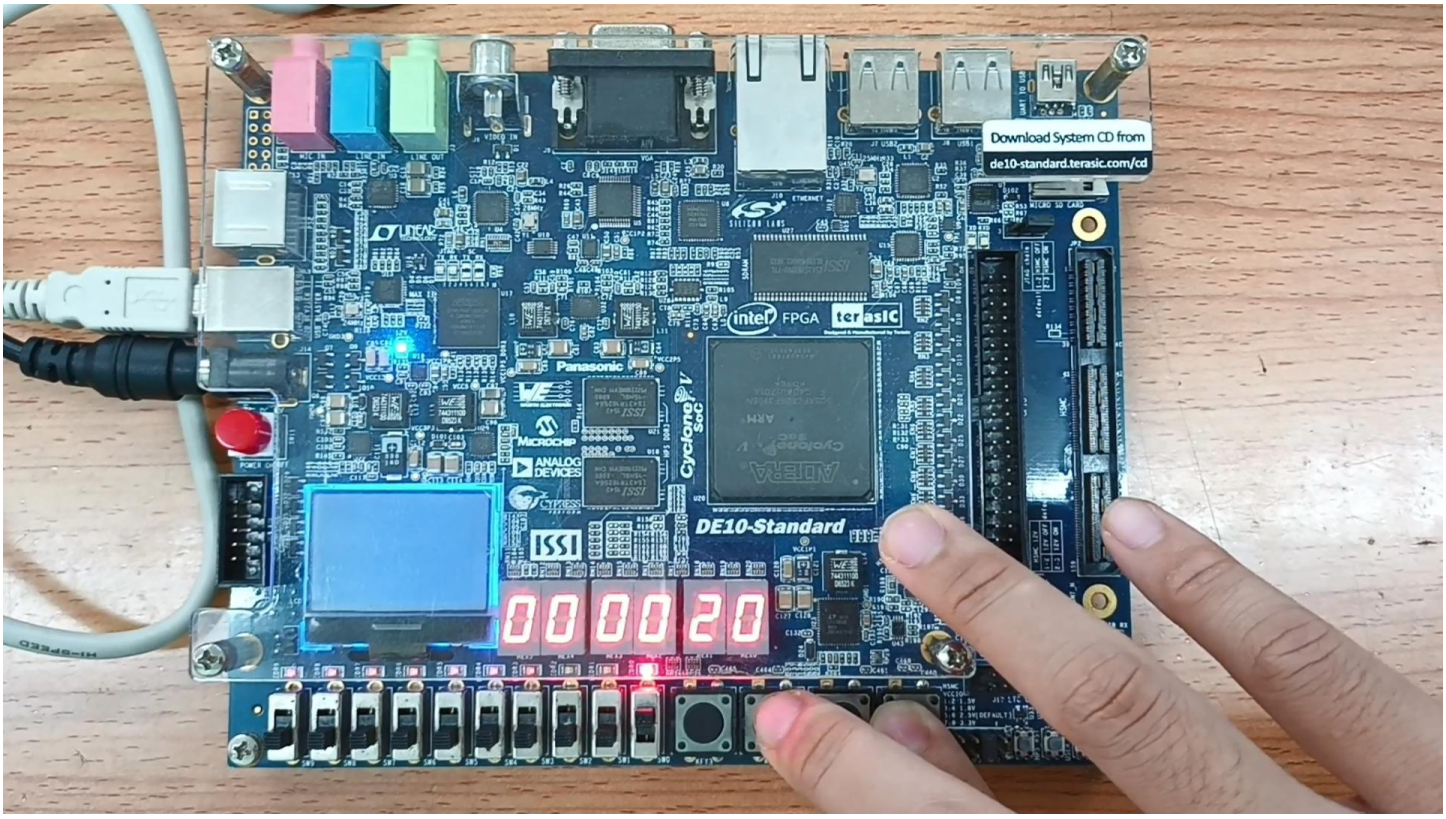
I have also included an 8-minute video demonstrating the FIFO alongside this report. Below are six notable cases:



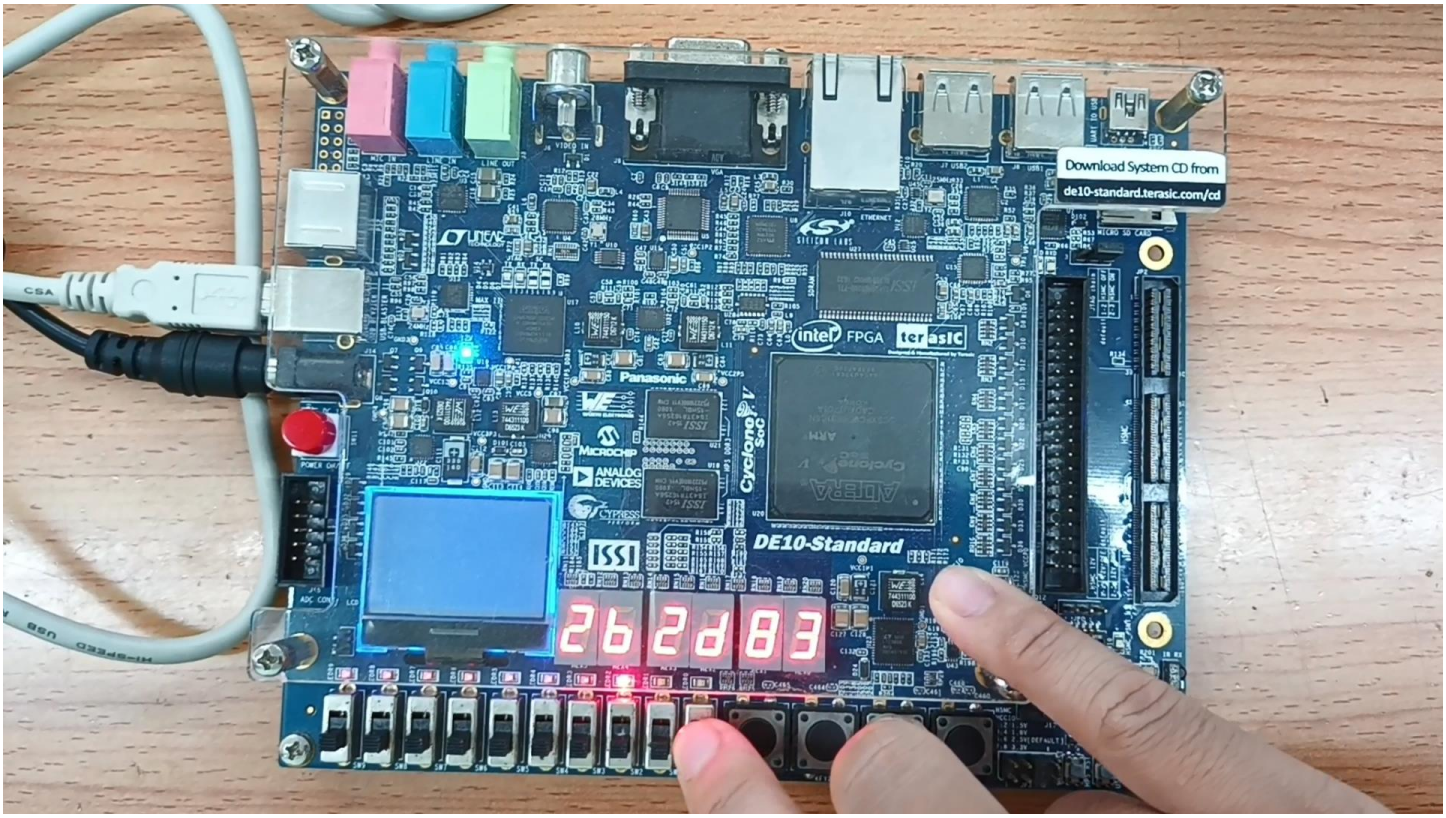
Empty gauge



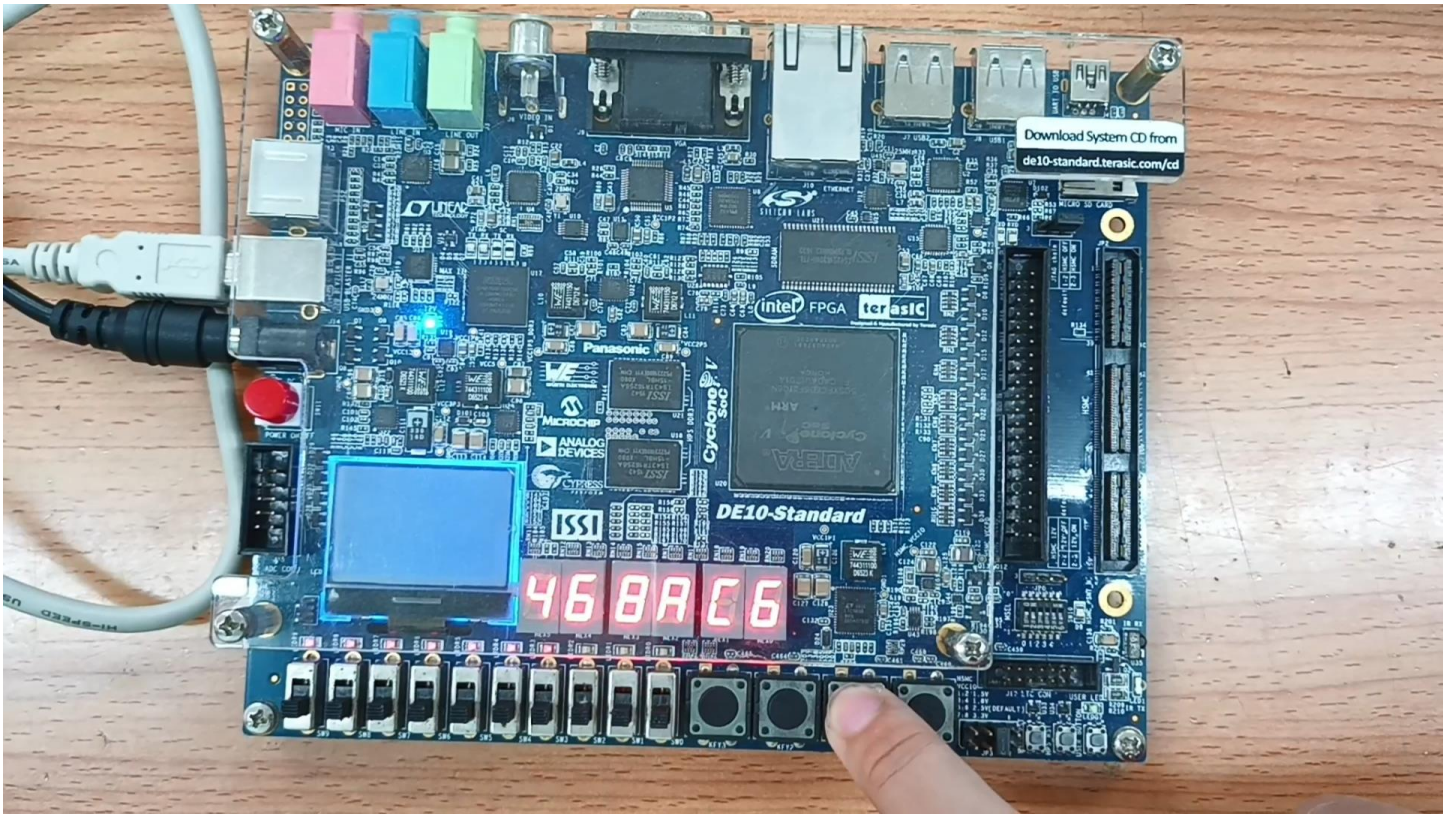
Gauge almost full



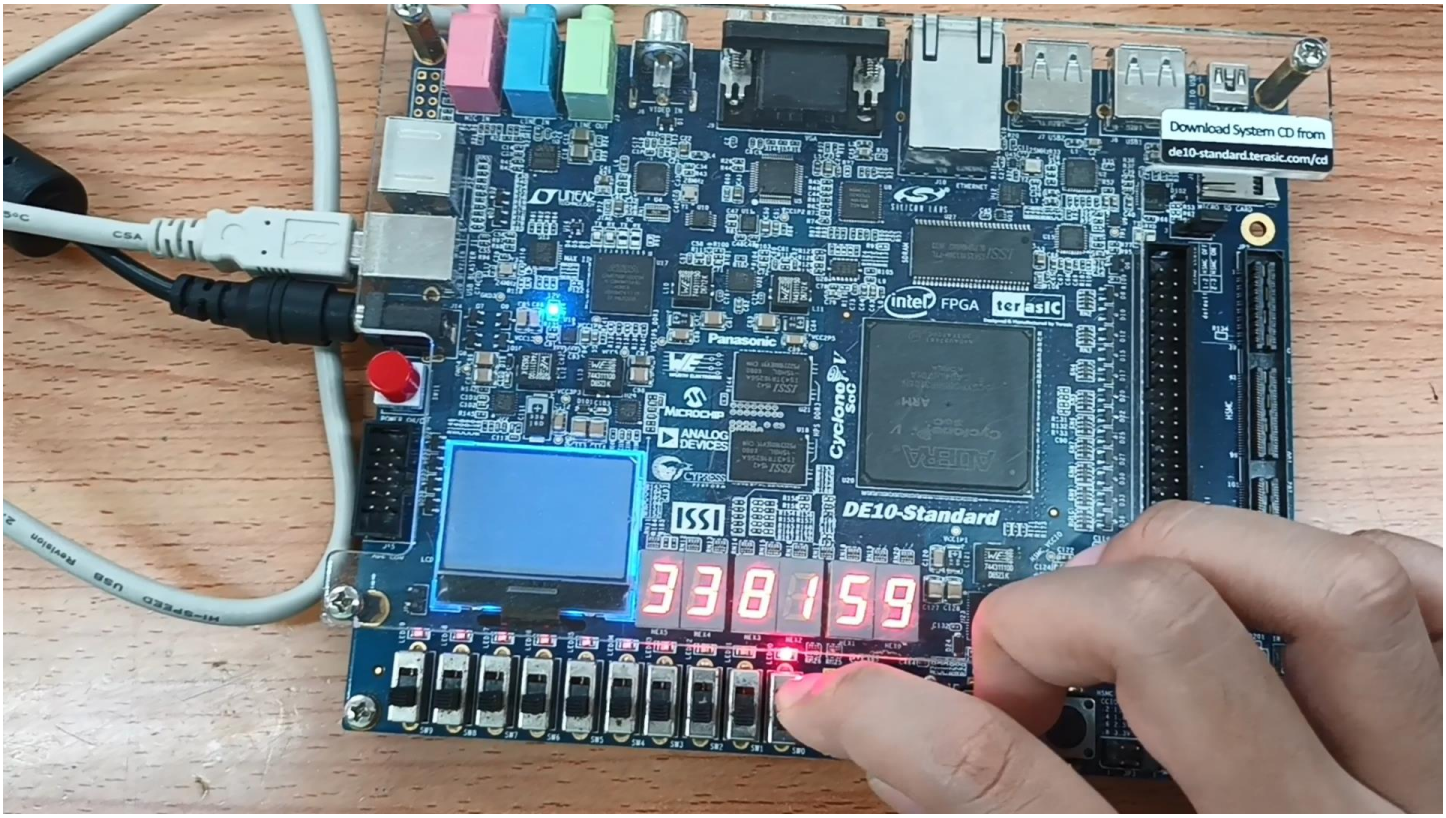
Gauge full



The last data read



Some data read in the middle



The most recent data read (NOT the first)