Experiment 2
Sequential Synthesis and FPGA Programming

Mohammad
Mahdi
Abdolhosseini,
810198434

Homayoun
Mohseni,
810198464

Abstract— **This experiment has two goals: 1. To teach us about state machines that control devices. 2. To introduce us to FPGA devices and how to use them. We will learn about state machines, sequence detectors, Huffman coding, design simulation, synthesis, and FPGA programming through this experiment.**

1. SERIAL TRANSMITTER

In this experiment, we designed a circuit that began transmitting data from serIn to serOut when it detected the start sequence "110101" on serIn. The circuit then transmitted whatever appeared on serIn for the next 10 clock cycles, and the serOutvalid was active during this time. After transmitting 10 bits, the circuit returned to the search state for the start sequence.

1.1 ONE-PULSER

One-pulser module provided a clock-enable input for the counter and the sequence detector. We used a push-button on our board (clkPB) to create a single pulse that synchronized with the system clock. The output of the One-Pulser module connected to the clock enable input (clkEn) of the sequence detector and the counter. We wrote the Verilog description of the One-pulser module and tested our design in Modelsim.

One-pulser general structure is shown in figure 1.

The state machine is shown in figure 2 and Quartus generated state machine and transition conditions is shown in figure 3 and 4 respectively.

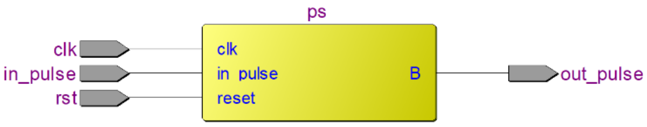


Fig.1 One-pulser module

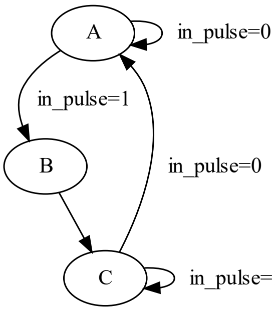


Fig.2 One-pulser state machine

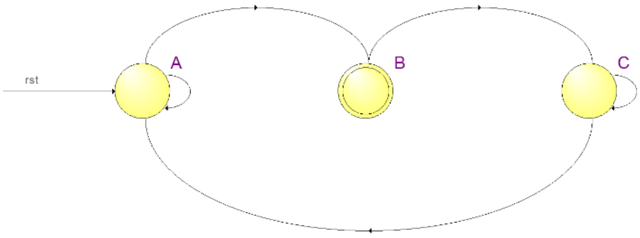
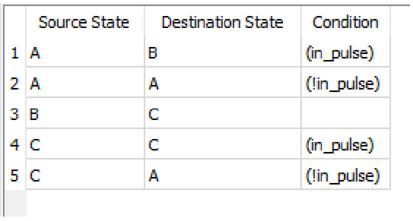


Fig.3 One-pulser Quartus generated state machine



	Source State	Destination State	Condition
1	A	B	(in_pulse)
2	A	A	(!in_pulse)
3	B	C	
4	C	C	(in_pulse)
5	C	A	(!in_pulse)

Fig.4 One-pulser Quartus generated state machine transition conditions

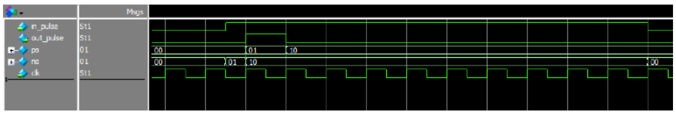


Fig.5 One-pulser simulation

1.2 ORTHOGONAL FINITE STATE MACHINE

By designing an Orthogonal FSM using a Moore state machine and a counter, the sequence detector checks for the sequence "110101" on its serIn input and activates the serOutValid output for 10 consecutive clock cycles. We designed the Moore state machine, wrote the Verilog descriptions for the transmitter circuit, and tested its correctness using a testbench in ModelSim.

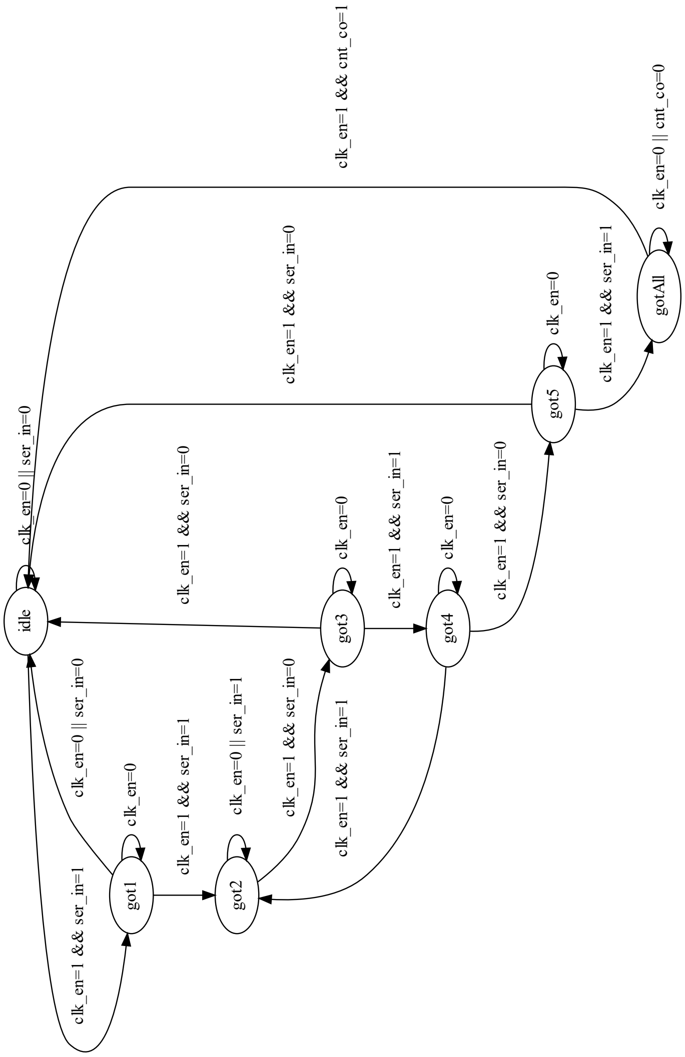


Fig.6 110101 Moore FSM

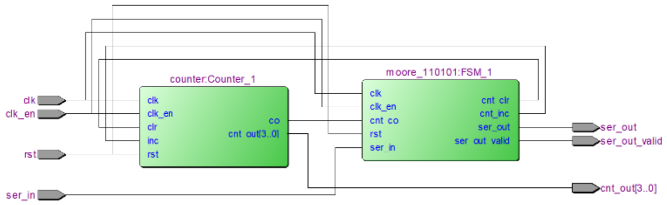


Fig.7 110101 detector module

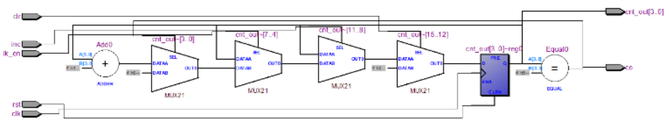


Fig.8 counter internal structure

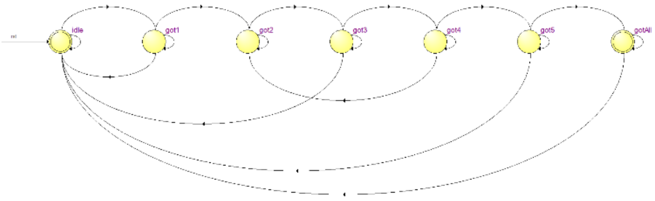
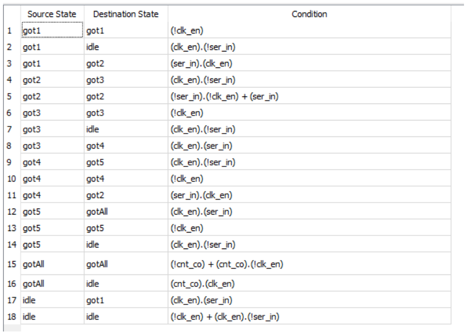


Fig.9 Quartus generated 110101 state machine



	Source State	Destination State	Condition
1	got1	got1	(!clk_en)
2	got1	idle	(clk_en).(!ser_in)
3	got1	got2	(ser_in).(clk_en)
4	got2	got3	(clk_en).(!ser_in)
5	got2	got2	(!ser_in).(!clk_en) + (ser_in)
6	got3	got3	(!clk_en)
7	got3	idle	(clk_en).(!ser_in)
8	got3	got4	(clk_en).(ser_in)
9	got4	got5	(clk_en).(!ser_in)
10	got4	got4	(!clk_en)
11	got4	got2	(ser_in).(clk_en)
12	got5	gotAll	(clk_en).(ser_in)
13	got5	got5	(!clk_en)
14	got5	idle	(clk_en).(!ser_in)
15	gotAll	gotAll	(!cnt_co) + (cnt_co).(!clk_en)
16	gotAll	idle	(cnt_co).(clk_en)
17	idle	got1	(clk_en).(ser_in)
18	idle	idle	(!clk_en) + (clk_en).(!ser_in)

Fig.10 Quartus generated 110101 state machine transition conditions

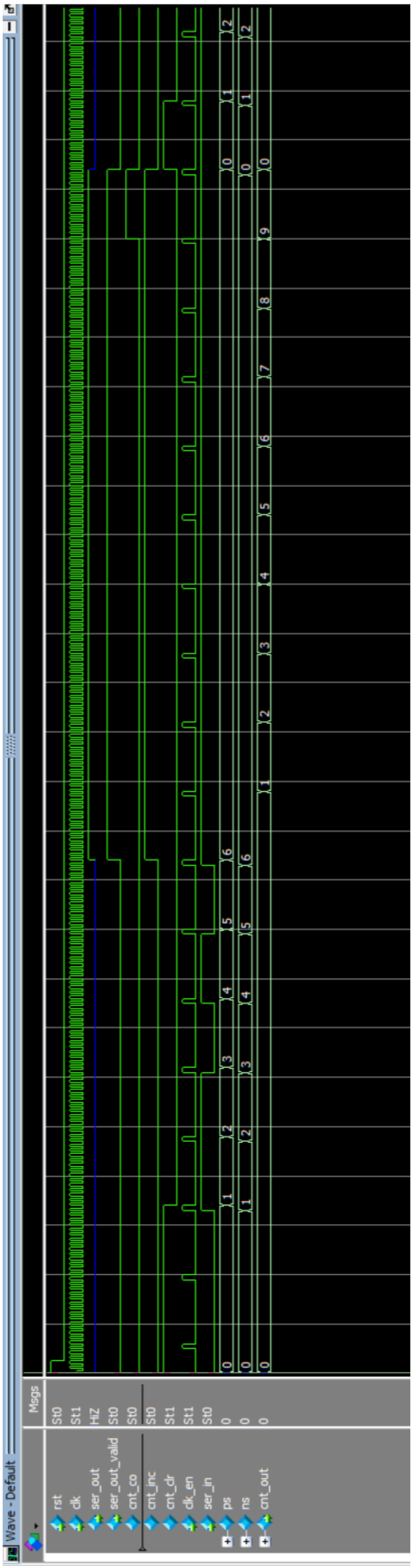


Fig.11 Orthogonal FSM wave form

1.3 Seven Segment Display

We designed a display module to show the counter output on seven segment component of the FPGA board. Each seven segment receives a 4-bit input and displayes the corresponding HEX value on its 7-bit output.

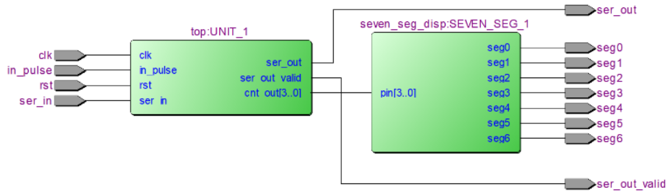


Fig.12 top level circuit including the seven segment component

Design Synthesis and FPGA Programming:

2. Serial Transmitter Implementation

For this part, we used the DE1 development board (shown in figure 3) to implement our design. We created a top-level Verilog description that connected the One-pulser, sequence detector, and seven segment display modules together.

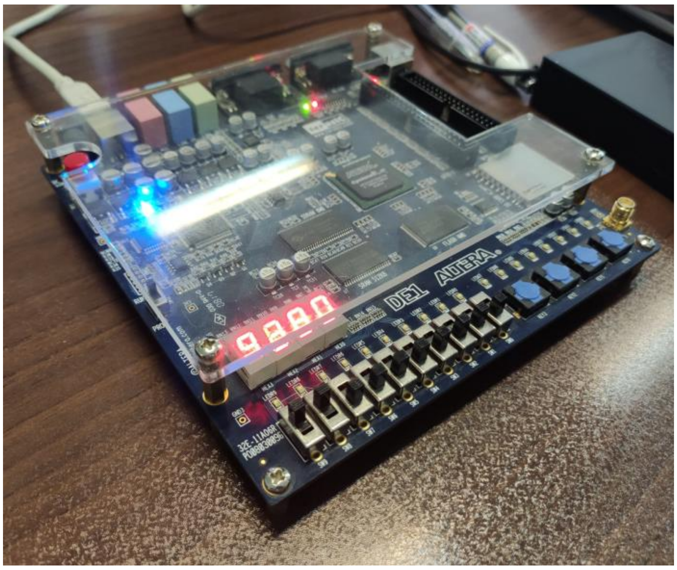
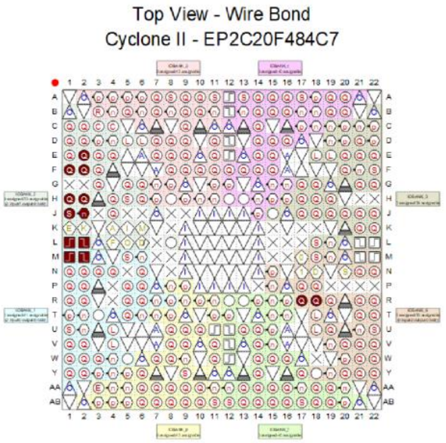


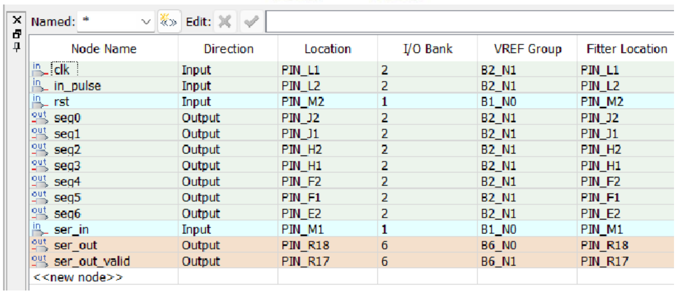
Fig.3 FPGA board

To begin, we made a Quartus project and added the top-level Verilog codes. We connected the main FPGA clock to the circuit's clock input and added a push-button to the serIn of the serial transmitter circuit and another to the input of the one-pulser circuit.

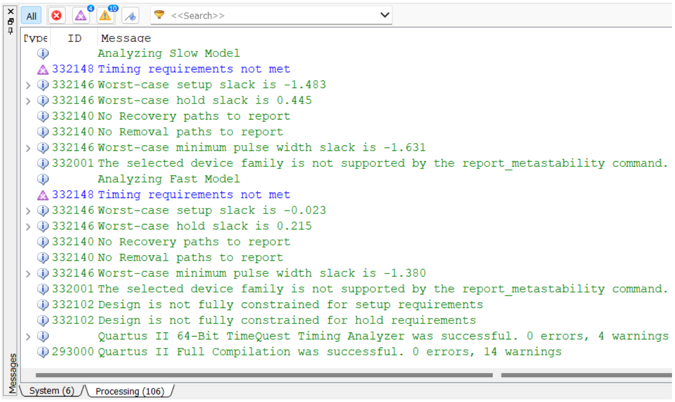
Afterwards, we performed synthesis and programmed the Cyclone II device. For output display, we used an output LED for displaying serOutvalid and one LED for serOut. We used one seven segment for displaying the counter output.



Top View - Wire Bond
Cyclone II - EP2C20F484C7

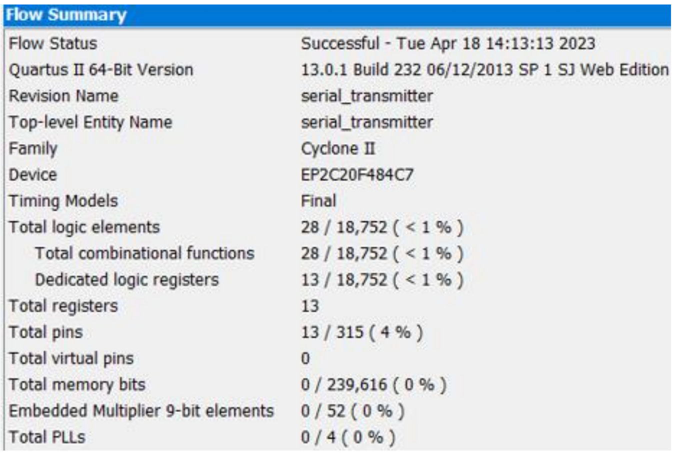


	Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location
in	clk	Input	PIN_L1	2	B2_N1	PIN_L1
in	in_pulse	Input	PIN_L2	2	B2_N1	PIN_L2
in	rst	Input	PIN_M2	1	B1_N0	PIN_M2
out	seq0	Output	PIN_J2	2	B2_N1	PIN_J2
out	seq1	Output	PIN_J1	2	B2_N1	PIN_J1
out	seq2	Output	PIN_H2	2	B2_N1	PIN_H2
out	seq3	Output	PIN_H1	2	B2_N1	PIN_H1
out	seq4	Output	PIN_F2	2	B2_N1	PIN_F2
out	seq5	Output	PIN_F1	2	B2_N1	PIN_F1
out	seq6	Output	PIN_E2	2	B2_N1	PIN_E2
in	ser_in	Input	PIN_M1	1	B1_N0	PIN_M1
out	ser_out	Output	PIN_R18	6	B6_N0	PIN_R18
out	ser_out_valid	Output	PIN_R17	6	B6_N1	PIN_R17
	<<new node>>					

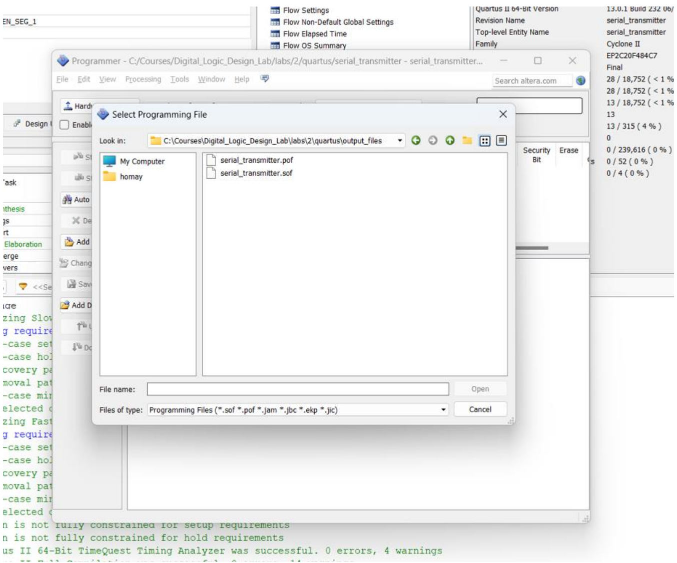


```
Type ID     Message
            Analyzing Slow Model
     332148 Timing requirements not met
   > 332146 Worst-case setup slack is -1.483
   > 332146 Worst-case hold slack is 0.445
     332140 No Recovery paths to report
     332140 No Removal paths to report
   > 332146 Worst-case minimum pulse width slack is -1.631
     332001 The selected device family is not supported by the report_metastability command.
            Analyzing Fast Model
     332148 Timing requirements not met
   > 332146 Worst-case setup slack is -0.023
   > 332146 Worst-case hold slack is 0.215
     332140 No Recovery paths to report
     332140 No Removal paths to report
   > 332146 Worst-case minimum pulse width slack is -1.380
     332001 The selected device family is not supported by the report_metastability command.
     332102 Design is not fully constrained for setup requirements
     332102 Design is not fully constrained for hold requirements
   >        Quartus II 64-Bit TimeQuest Timing Analyzer was successful. 0 errors, 4 warnings
     293000 Quartus II Full Compilation was successful. 0 errors, 14 warnings
```

Messages
System (6)
Processing (106)



Flow Summary	
Flow Status	Successful - Tue Apr 18 14:13:13 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	serial_transmitter
Top-level Entity Name	serial_transmitter
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	28 / 18,752 (< 1 %)
Total combinational functions	28 / 18,752 (< 1 %)
Dedicated logic registers	13 / 18,752 (< 1 %)
Total registers	13
Total pins	13 / 315 (4 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)



Programmer - C:/Courses/Digital_Logic_Design_Lab/labs/2/quartus/serial_transmitter - serial_transmitter...

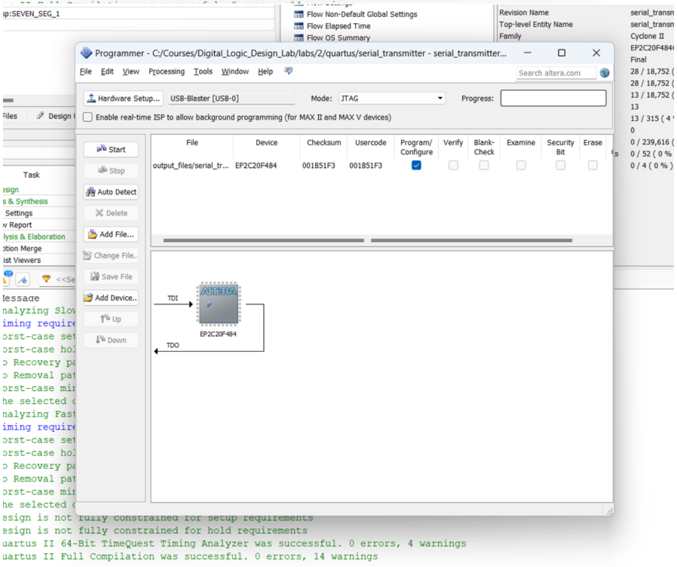
Select Programming File

Look in: C:\Courses\Digital_Logic_Design_Lab\labs\2\quartus\output_files

My Computer
homay

serial_transmitter.pof
serial_transmitter.sof

File name:
Files of type: Programming Files (*.sof *.pof *.jam *.jbc *.ekp *.jic)



Programmer - C:/Courses/Digital_Logic_Design_Lab/labs/2/quartus/serial_transmitter - serial_transmitter...

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress:

☐ Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase
output_files/serial_tr...	EP2C20F484	001B51F3	001B51F3	☑	☐	☐	☐	☐	☐

To apply serial input, we pushed the corresponding push-button for the serIn input and then pressed the ClkPB push button once.

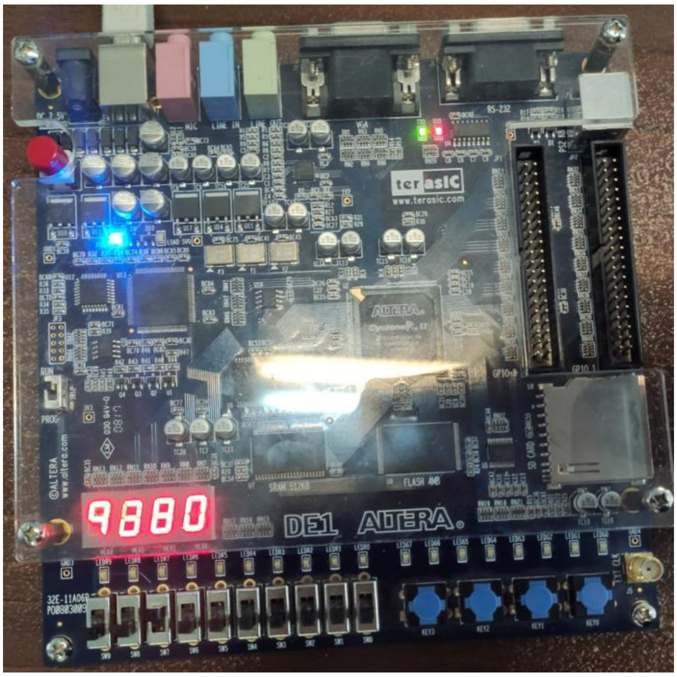


Fig.4 reset before starting

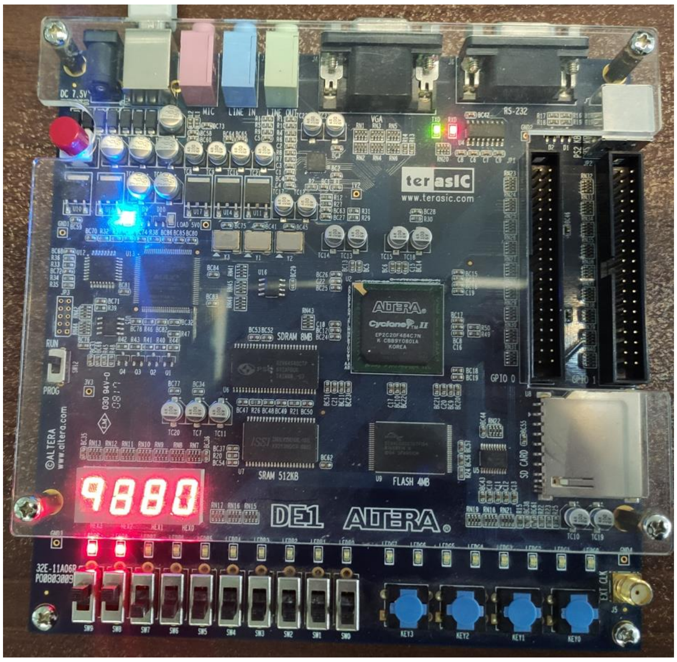


Fig.5 sequence is detected

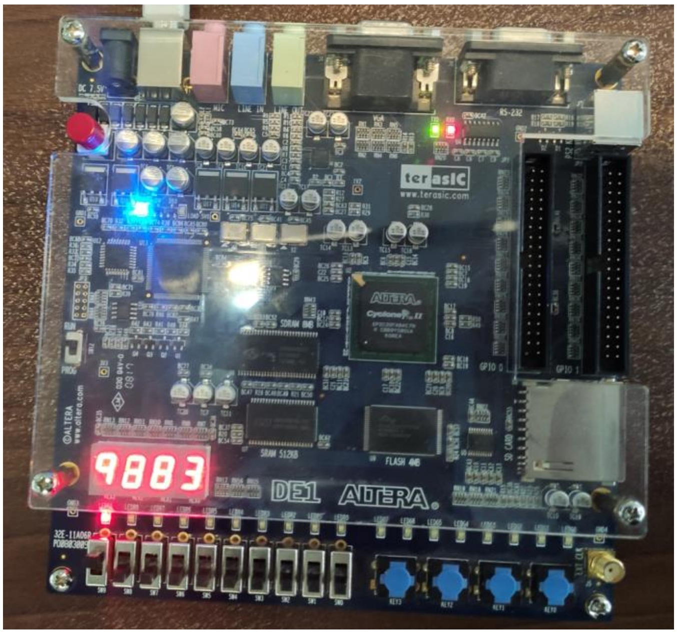


Fig.6 3rd number received

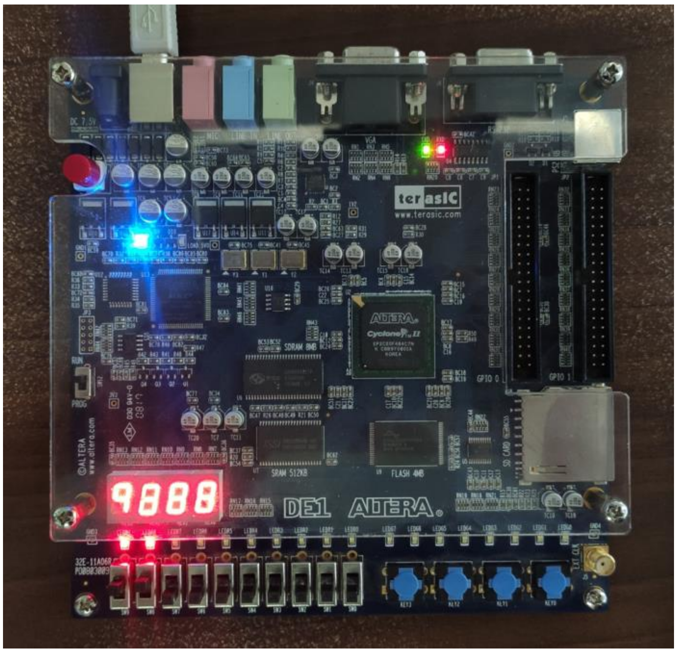


Fig.7 8th number received