The supported instructions are:

Instructions	Instruction Format (MAX 39 BITS)
RESET : Clear Output Register and flags and Register Files	<3 Bit Instruction> Example: 000XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Store Value to Reg File A at specified address	<3 Bit Instruction><4 bit Address><32 bit Value> Example : 00100100000111100001111000011110000111
Store Value to Reg File B at specified address	<3 Bit Instruction><4 bit Address><32 bit Value> Example: 01000100000111100001111000011110000111
Add values Op-A, Op-B using the supplied addresses	<3 Bit Instruction><4 bit Address to Register File A><4 bit Address to Register File B> Example: 01100100011
Subtract Op-A, Op-B	<3 Bit Instruction><4 bit Address to Register File A><4 bit Address to Register File B> Example: 10000100011
Bitwise-OR Op-A, Op-B	<3 Bit Instruction><4 bit Address to Register File A><4 bit Address to Register File B> Example: 10100100011
Bitwise-AND Op-A, Op-B	<3 Bit Instruction><4 bit Address to Register File A><4 bit Address to Register File B> Example: 11000100011
Shift Left Op-A by Op-B times	<3 Bit Instruction><4 bit Address to Register File A><4 bit Address to Register File B> Example: 11100100011

Our design is fully completed and has been tested on FPGA as well as through Timing Waveforms. It's fully pipelined and performs operations on the provided instruction set.
Deliverables completed:

- Our Design can operate on all the instructions provided for Tesbench testing.
- Our Design is effectively pipelined, all outputs and inputs are registered as well as the intermediate output
- Our implemented architecture will reset the program counter as well as the program once all the instructions in our instructions file (memory.dat file) are executed, so the HEX displays are cleared as well as the Registers and program counter is reset.
- Pressing reset at any point in program execution will automatically reset the program counter and the registers, as well as the HEX displays.

Description:

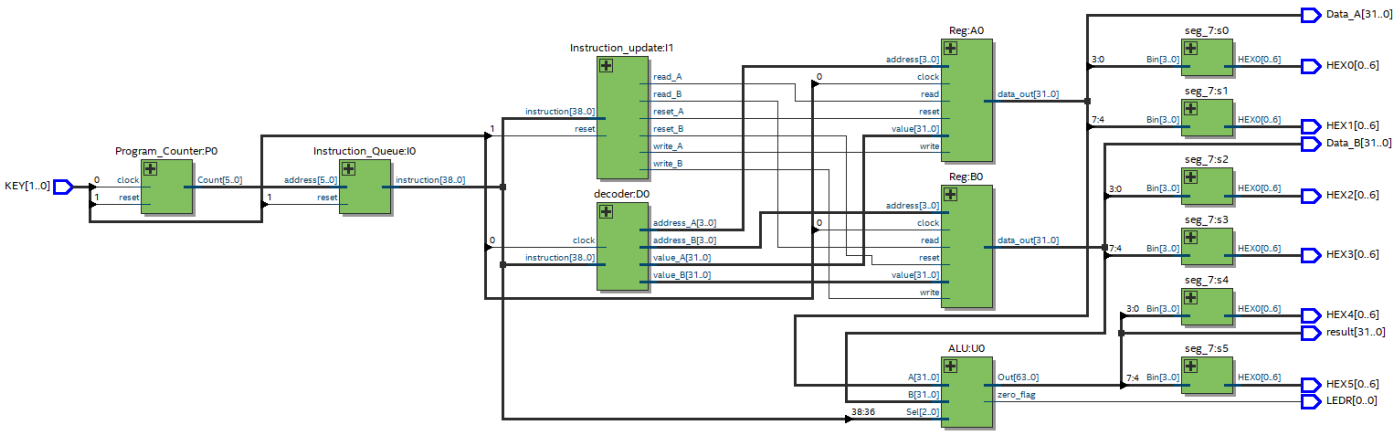
We followed the standard pipelined design for this assignment which implements a 32-bit microprocessor design on the provided Instruction set. In our top module file, we call all the individual

building blocks. We first instantiate a Program counter which returns a 5-bit value of current count. This count value is fed to our Instruction Que module. This module reads the memory.dat file and returns the instruction that is pointed by the Program Counter's 5-bit value. This instruction is then passed on to the Instruction_update module. This module will update the flags for Register operands file Register A and Register B. The register flags are required so that the Register module can use these flags to decide whether to clear the register or read a certain (memory location/row) located value or store a value.

The next module that is instantiated is the Decoder module. This module will dissect the 39-bit instruction and extract the provided data based on the type of instruction. For example, if there is an instruction to store a value on an address for Register A, then the decoder module will extract the value part and the address part from that instruction into the respective registers that are output from the decoder. These registers are passed on to the individual register modules, Register A and Register B which will update the register files according to the type of instruction and the resister flags. Finally, the ALU module is instantiated which will perform the operation on the two operands, Register A and B according to the instruction.

In our design KEY-0 is mapped to the clock and the KEY-1 is mapped to reset. Since these KEYs are active so KEY-1 is held at high for program to execute. For HEX display, HEX0 and HEX1 is used for displaying contents of Register A, HEX2 and HEX3 are used for displaying contents of Register B and finally HEX4 and HEX5 display the contents of ALU's output Regsiter.

RTL



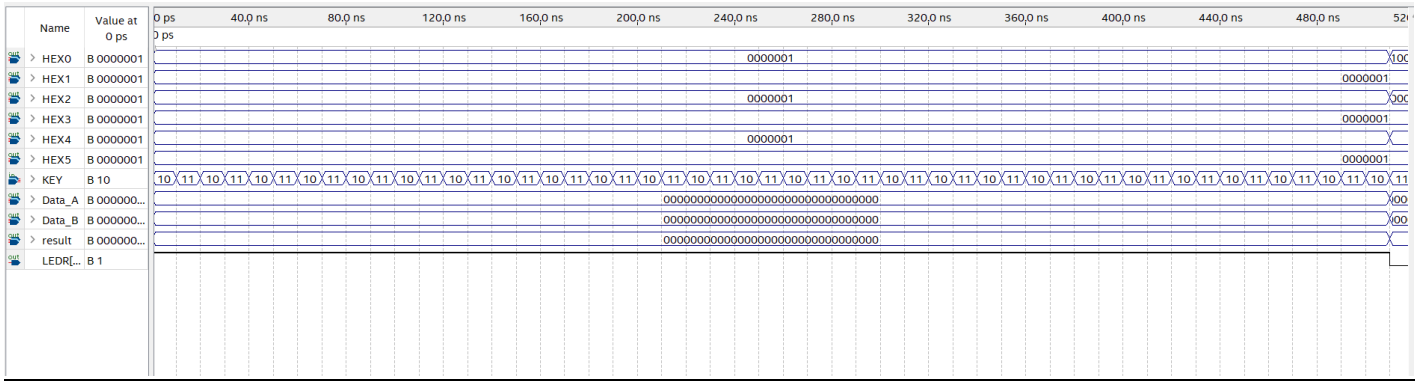
Waveform Simulation:

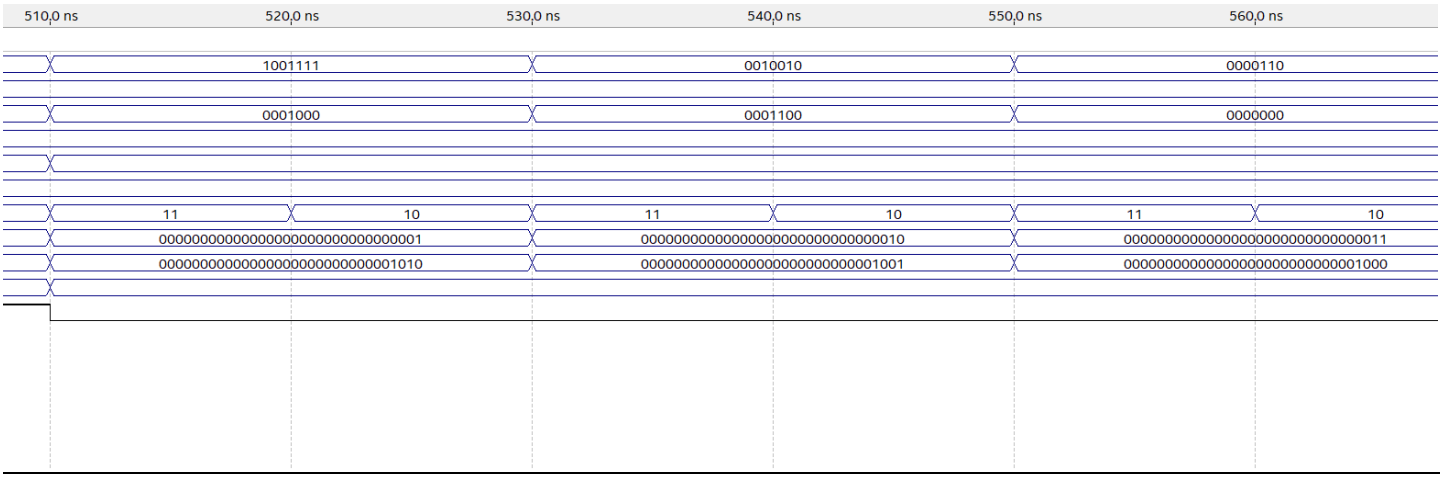
Clock Time Period: 10ns

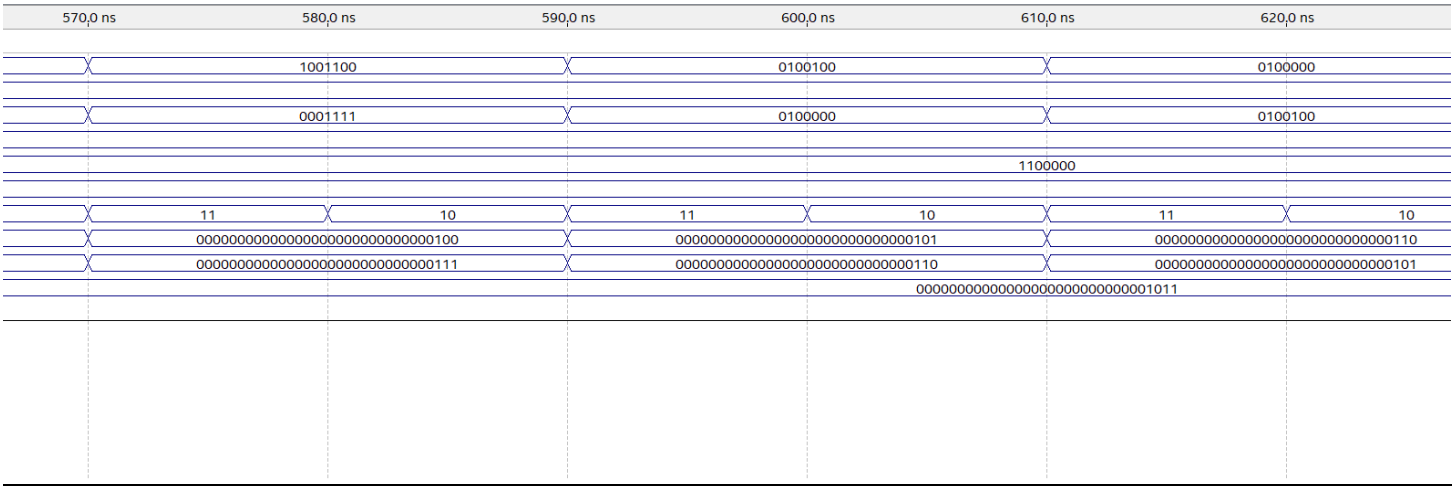
Total Simulation period: 1us

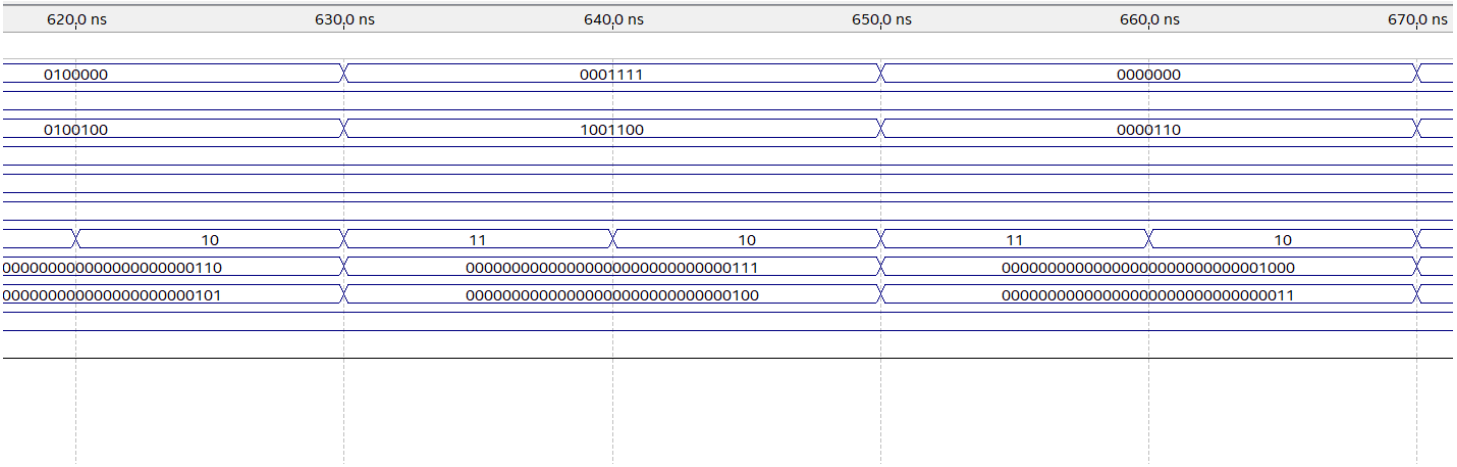
Instructions executed: All 41 instructions provided for the testbench, last instruction will automatically reset the whole program, clearing Register A and Register B as well as the ALU output Register and the program counter.

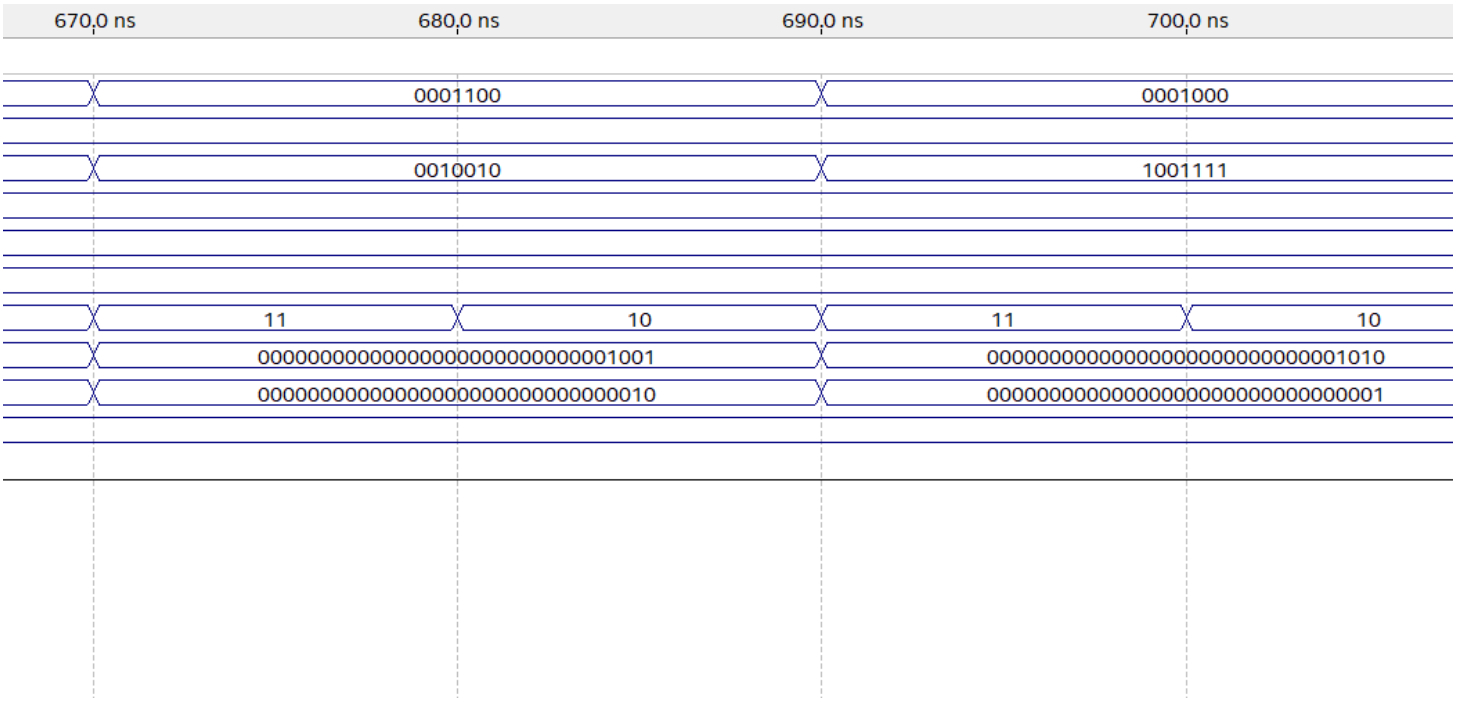
Flow of Program: To track the flow of program through simulation, we can inspect the outputs of Data_A which represents the contents of Register A and similarly Data_B and the output result after operation in the result register. The LEDR displays a flag which shows if the ALU performed any action on the two operands i.e., if the instruction required just storing or reading of values in one of the two registers. 1 value of this LEDR shows that no operation was performed by ALU i.e., it was a read/write instruction and 0 value shows that an operation was performed by ALU and the output is showed in ‘result’ register.

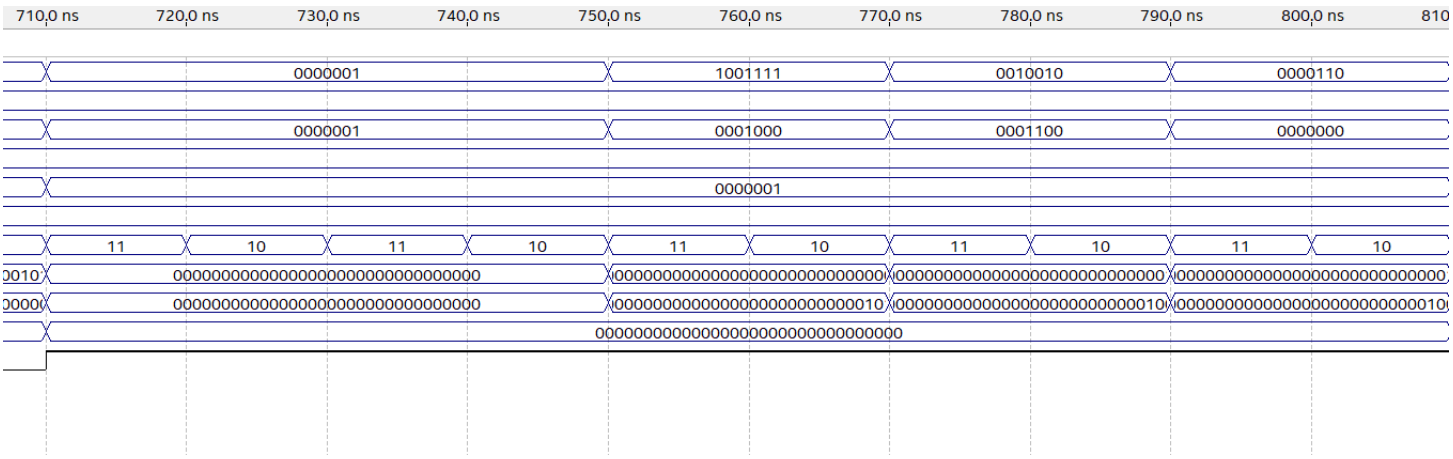

Name
Value at 0 ps
HEX0
HEX1
HEX2
HEX3
HEX4
HEX5
KEY
Data_A
Data_B
result
LEDR[...
0000001
10 11

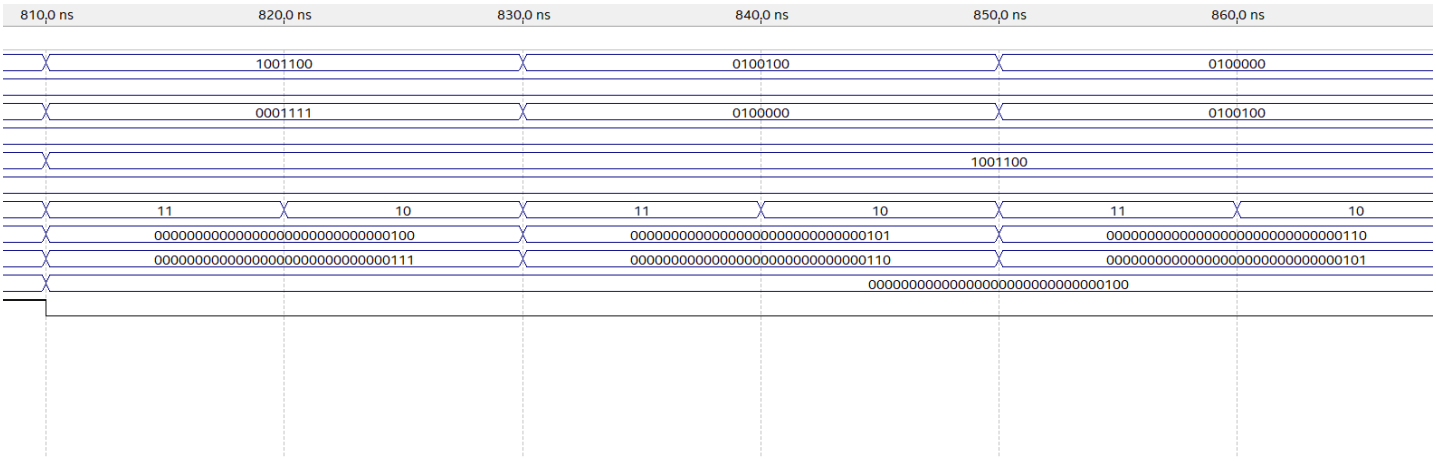

510,0 ns
520,0 ns
530,0 ns
540,0 ns
550,0 ns
560,0 ns
1001111
0010010
0000110
0001000
0001100
0000000
11
10
00000000000000000000000000000001
00000000000000000000000000000010
00000000000000000000000000000011
00000000000000000000000000001010
00000000000000000000000000001001
00000000000000000000000000001000

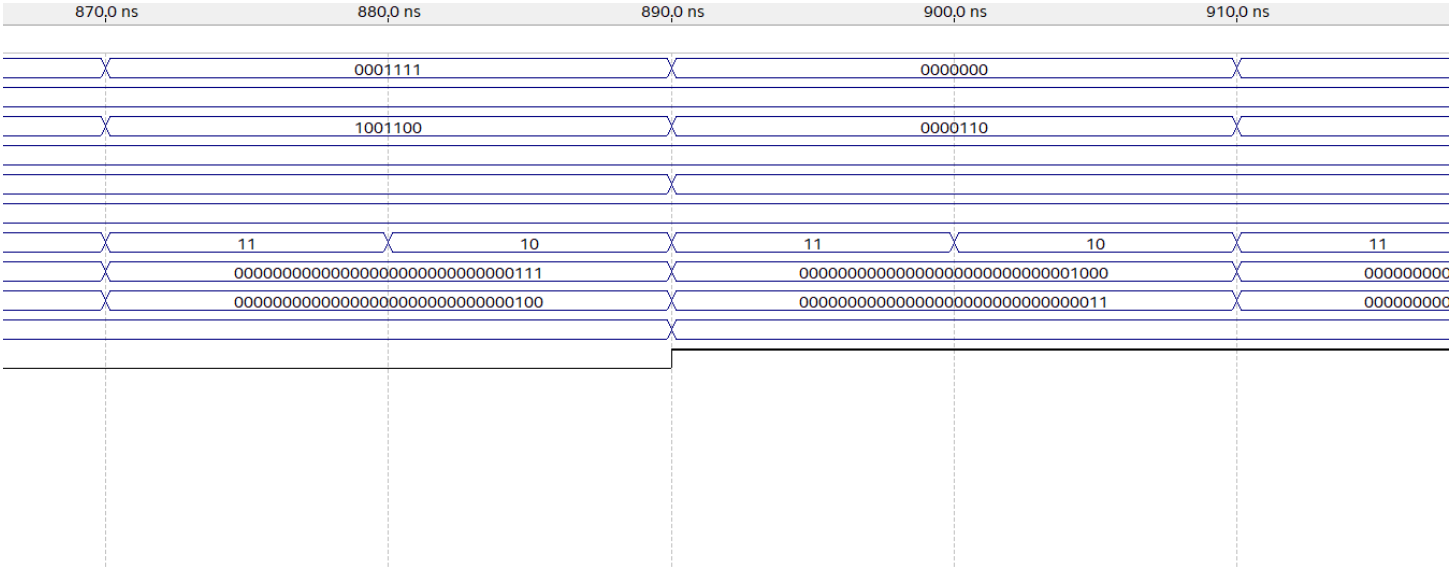

570,0 ns
580,0 ns
590,0 ns
600,0 ns
610,0 ns
620,0 ns
1001100
0100100
0100000
0001111
0100000
0100100
1100000
11
10
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000001011

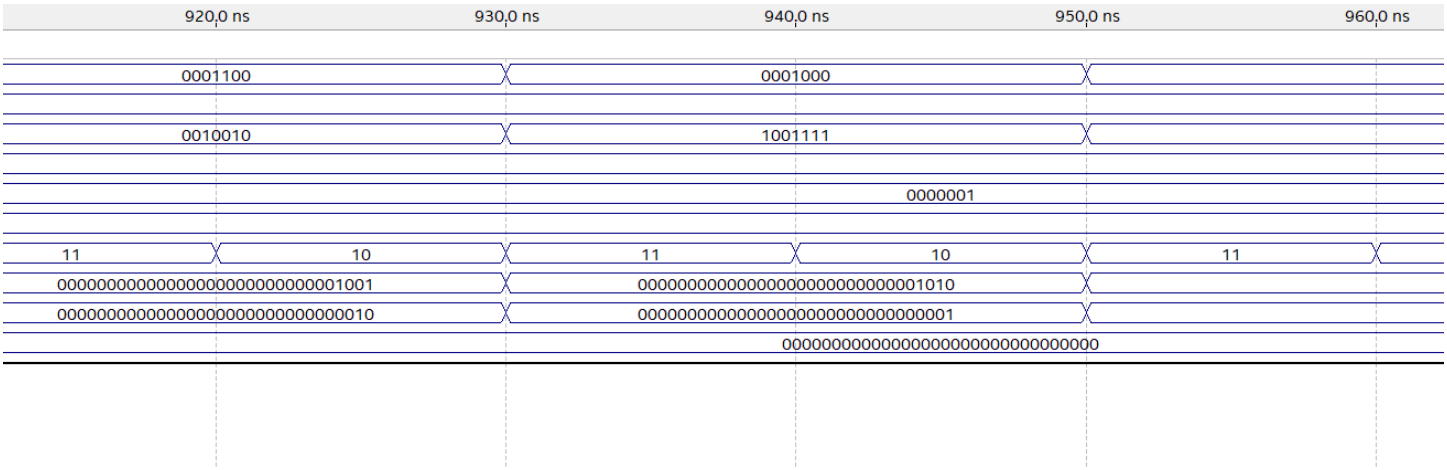

620,0 ns
630,0 ns
640,0 ns
650,0 ns
660,0 ns
670,0 ns
0100000
0001111
0000000
0100100
1001100
0000110
10
11
10
11
10
00000000000000000000110
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000101
00000000000000000000000000000100
00000000000000000000000000000011

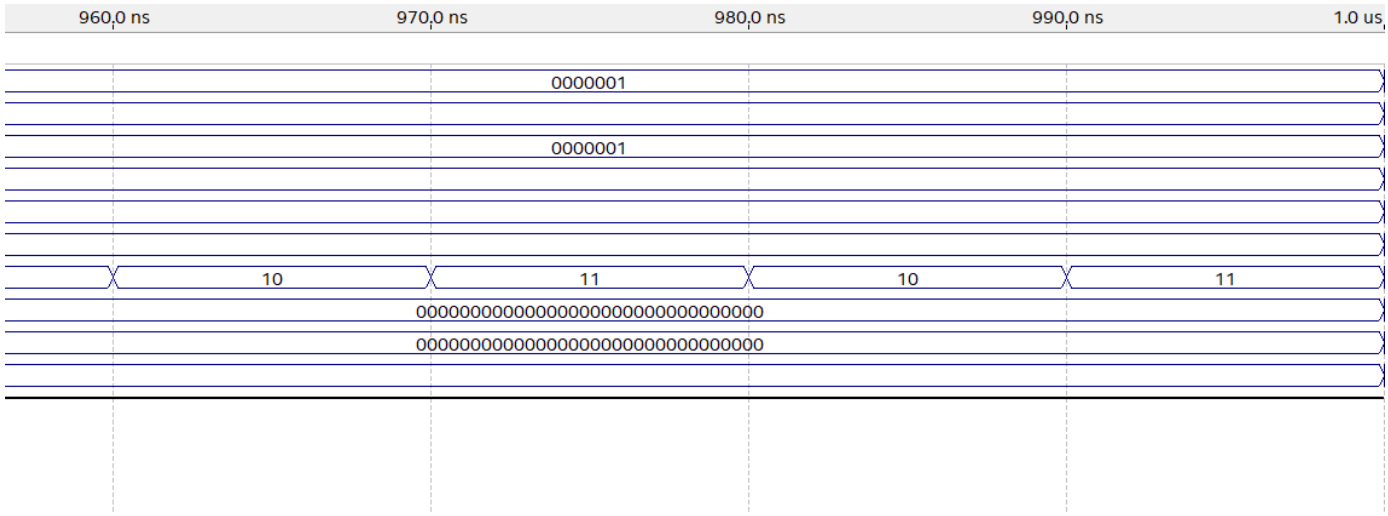

670,0 ns
680,0 ns
690,0 ns
700,0 ns
0001100
0001000
0010010
1001111
11
10
11
10
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000000010
00000000000000000000000000000001


710,0 ns
720,0 ns
730,0 ns
740,0 ns
750,0 ns
760,0 ns
770,0 ns
780,0 ns
790,0 ns
800,0 ns
0000001
1001111
0010010
0000110
0000001
0001000
0001100
0000000
0000001
11
10
11
10
11
10
11
10
11
10
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000000010
00000000000000000000000000000000

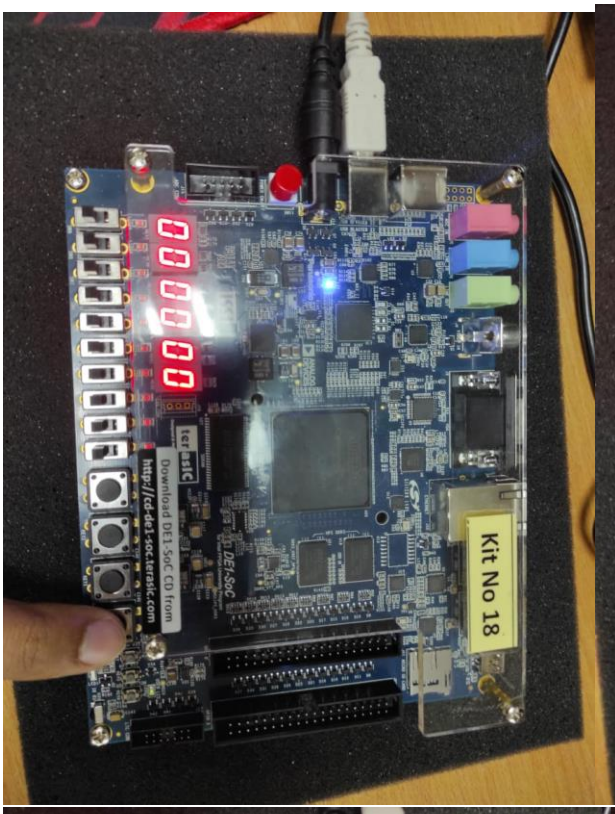

810,0 ns
820,0 ns
830,0 ns
840,0 ns
850,0 ns
860,0 ns
1001100
0100100
0100000
0001111
0100000
0100100
1001100
11
10
11
10
11
10
00000000000000000000000000000100
00000000000000000000000000000101
00000000000000000000000000000110
00000000000000000000000000000111
00000000000000000000000000000110
00000000000000000000000000000101
00000000000000000000000000000100

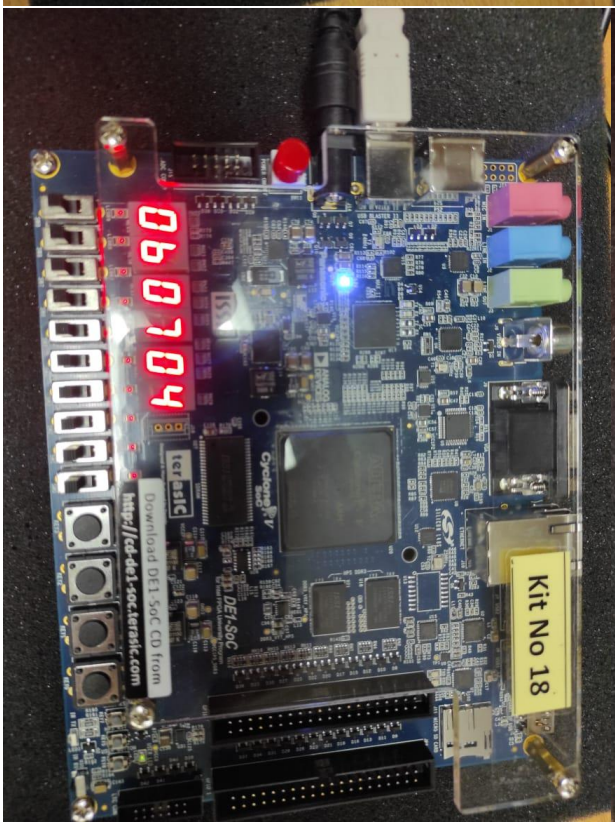

870,0 ns
880,0 ns
890,0 ns
900,0 ns
910,0 ns
0001111
0000000
1001100
0000110
11
10
11
10
11
00000000000000000000000000000111
00000000000000000000000000001000
00000000000000000000000000000100
00000000000000000000000000000011

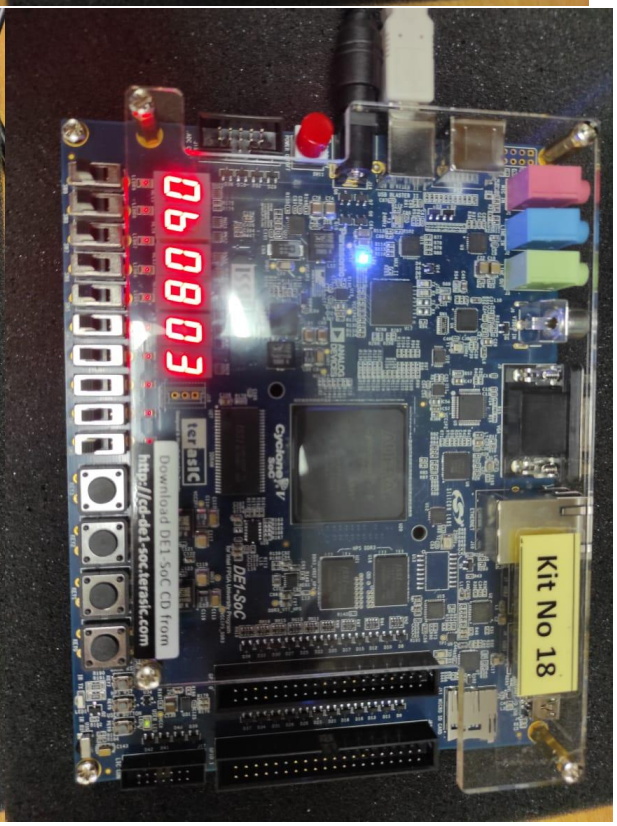

920,0 ns
930,0 ns
940,0 ns
950,0 ns
960,0 ns
0001100
0001000
0010010
1001111
0000001
11
10
11
10
11
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000000010
00000000000000000000000000000001
00000000000000000000000000000000

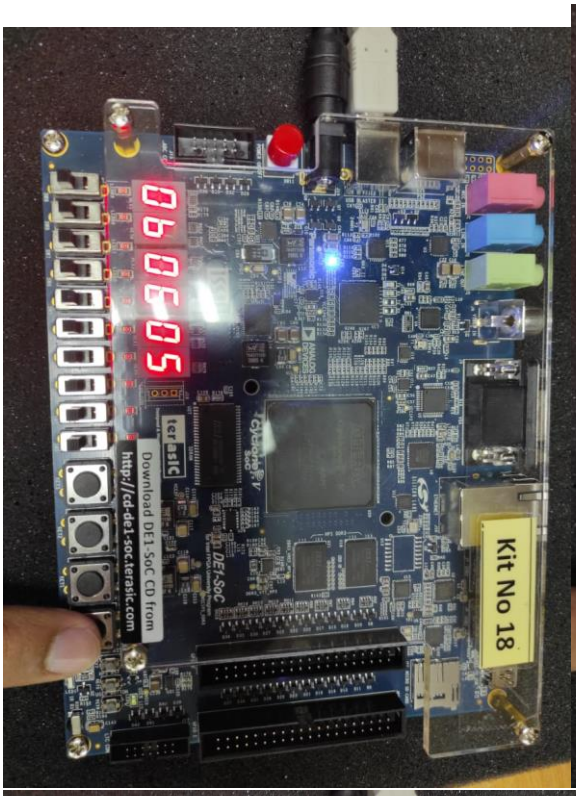

960,0 ns
970,0 ns
980,0 ns
990,0 ns
1.0 us
0000001
0000001
10
11
10
11
00000000000000000000000000000000
00000000000000000000000000000000

FPGA Demo Pictures:

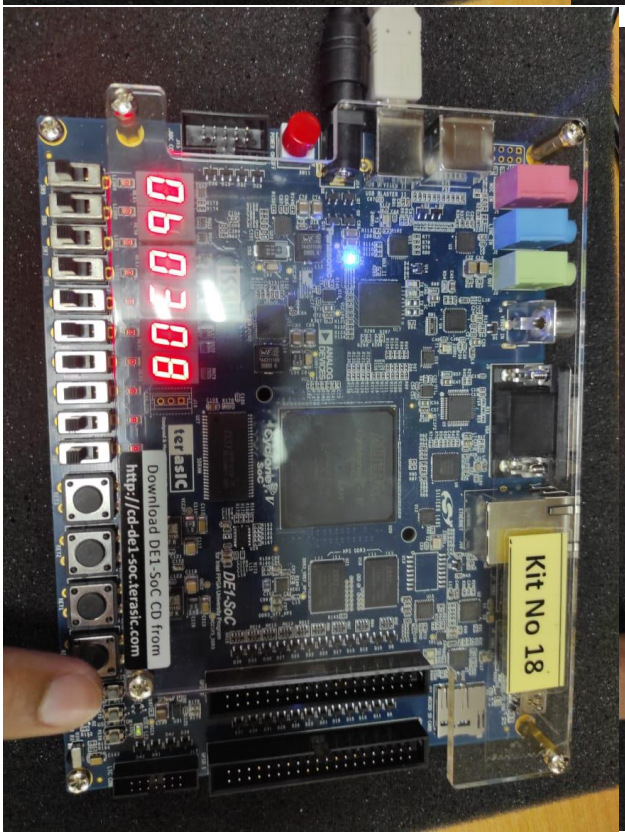


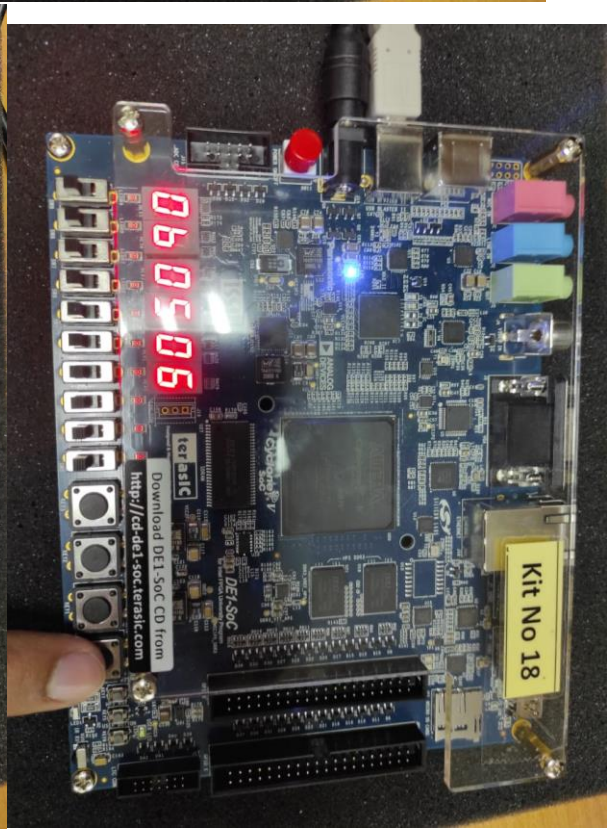


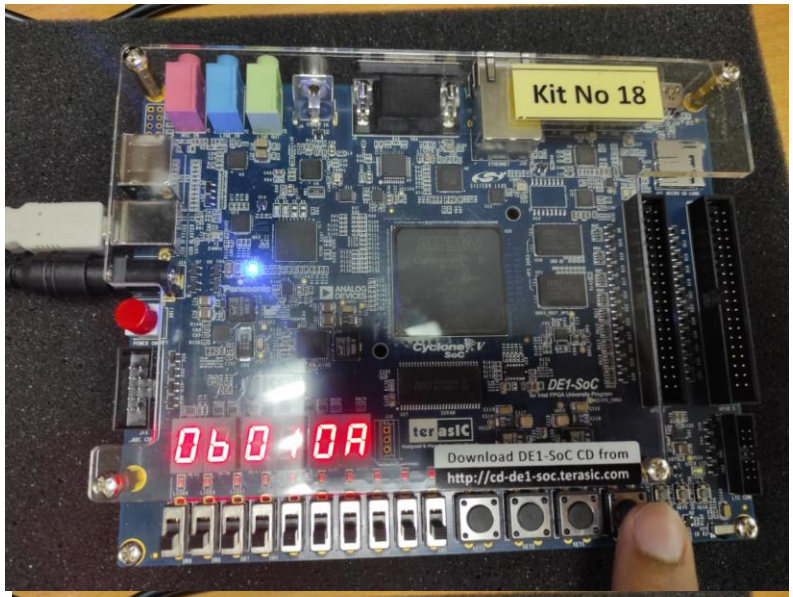


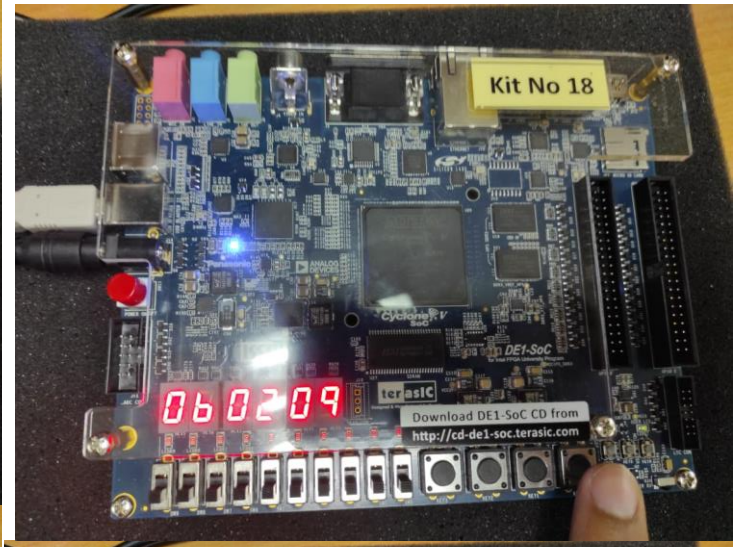

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

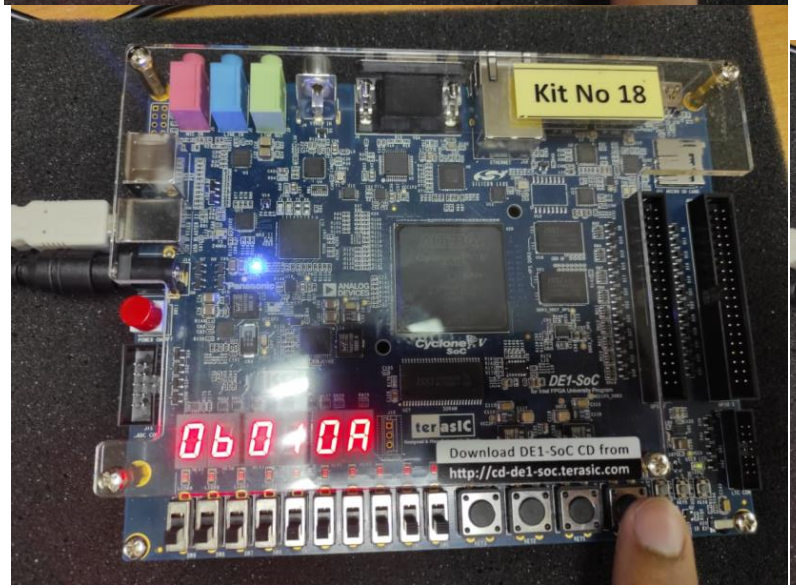

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

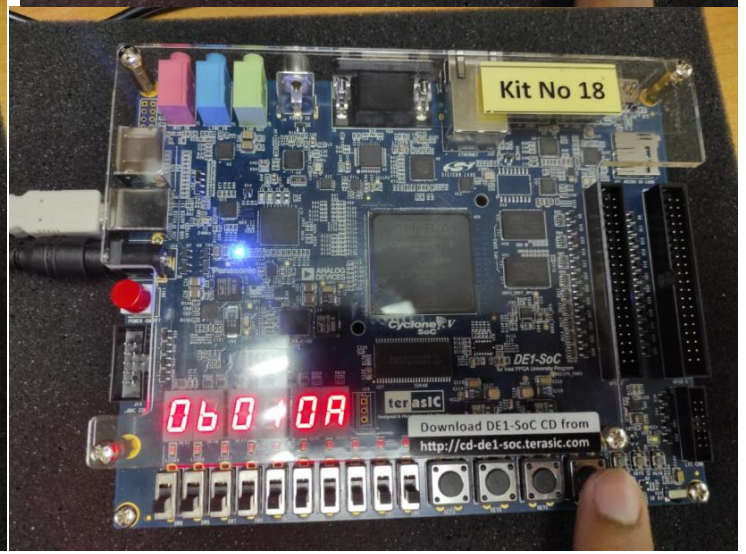

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

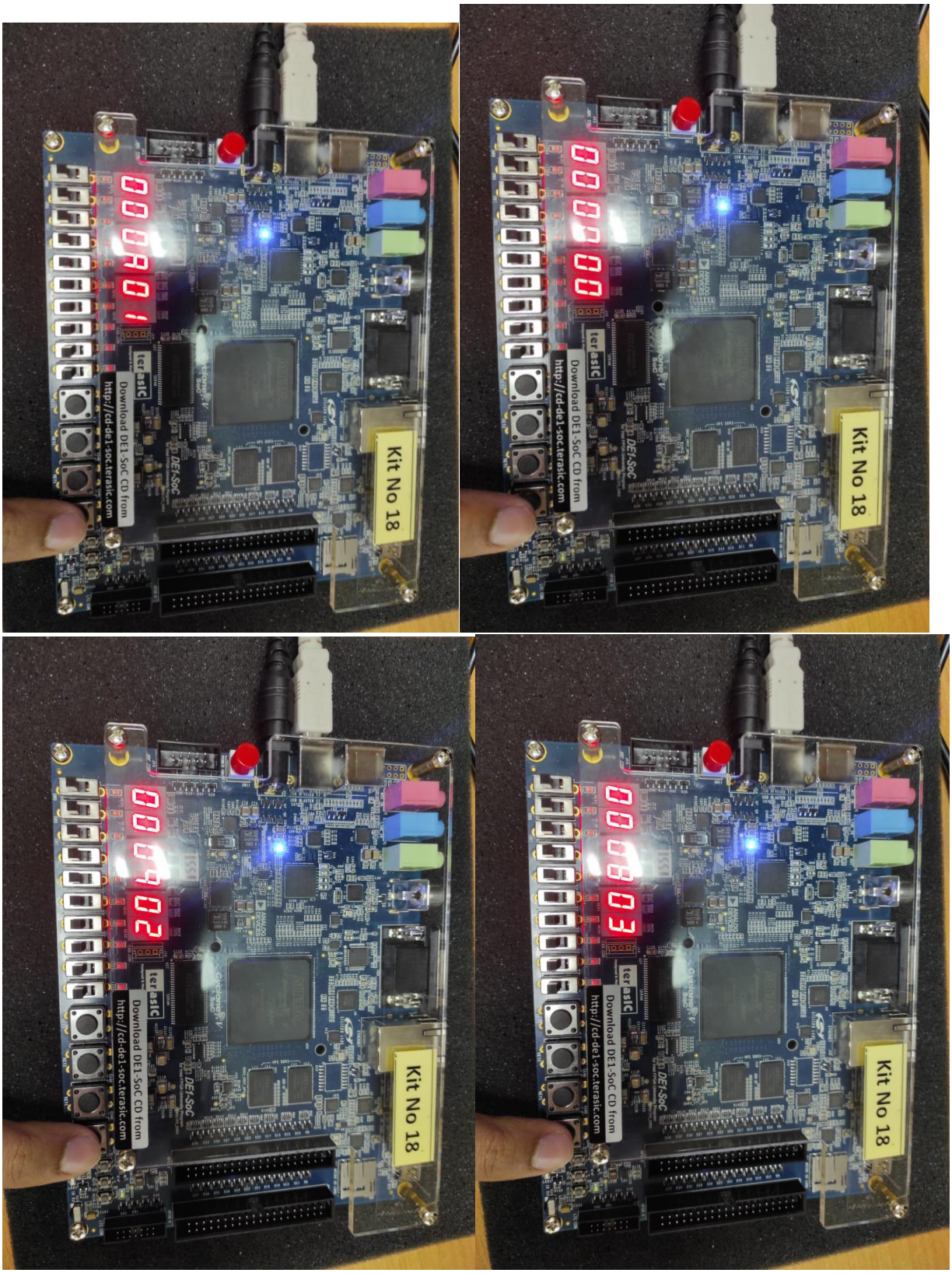

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

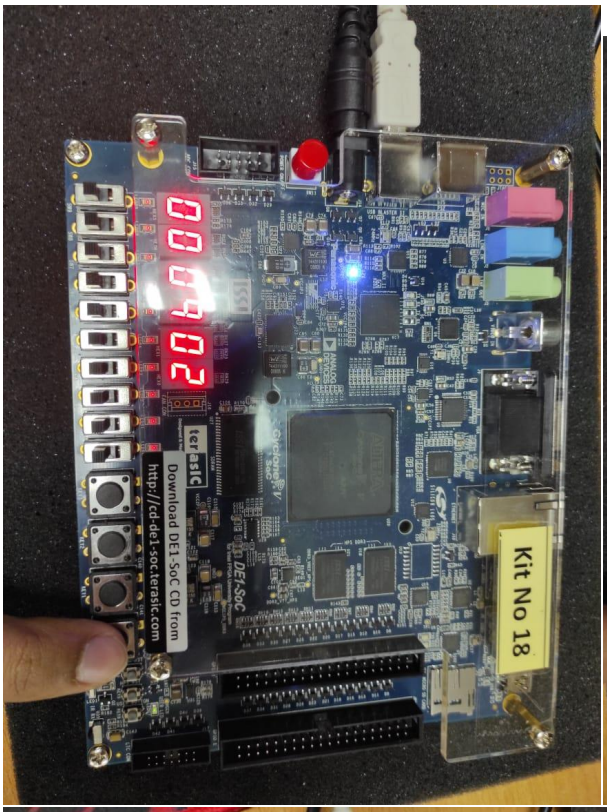

Kit No 18
0b0 0A
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

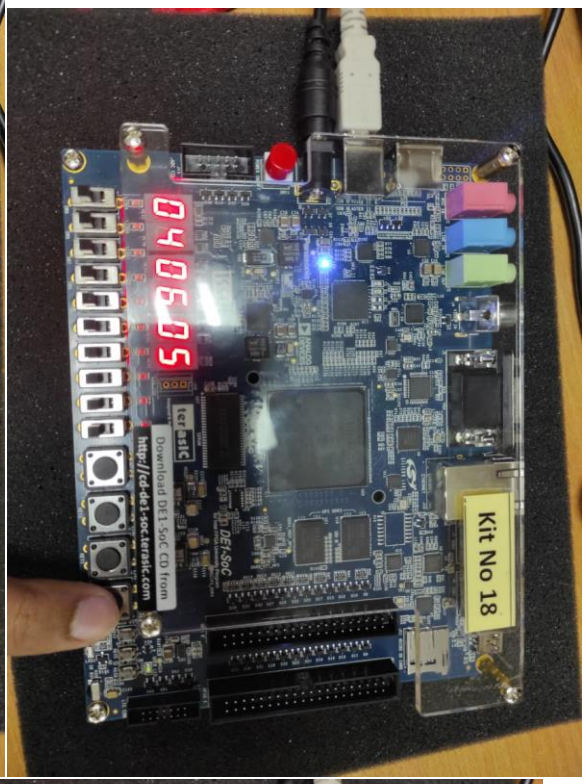

Kit No 18
0b0209
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

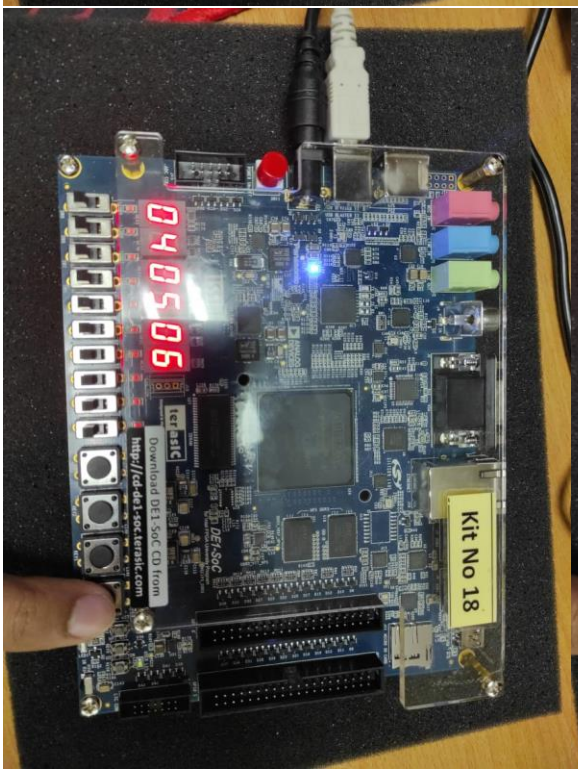

Kit No 18
0b0 0A
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

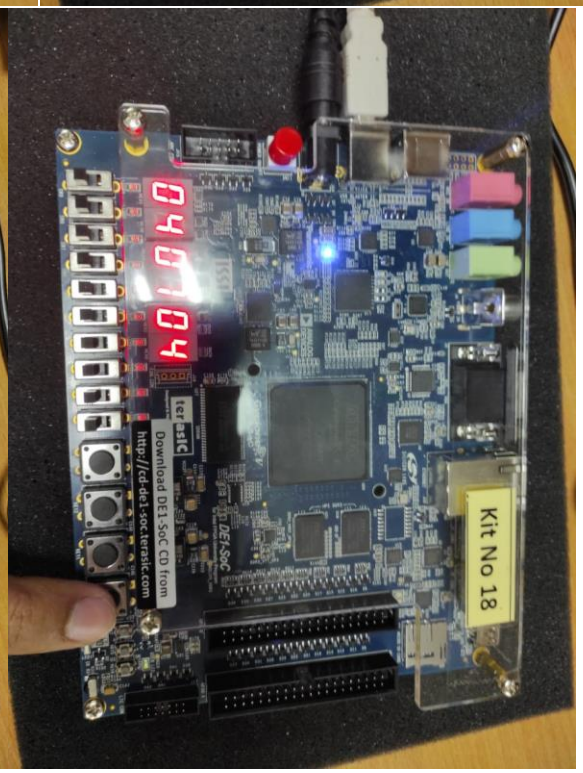

Kit No 18
0b0 0A
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

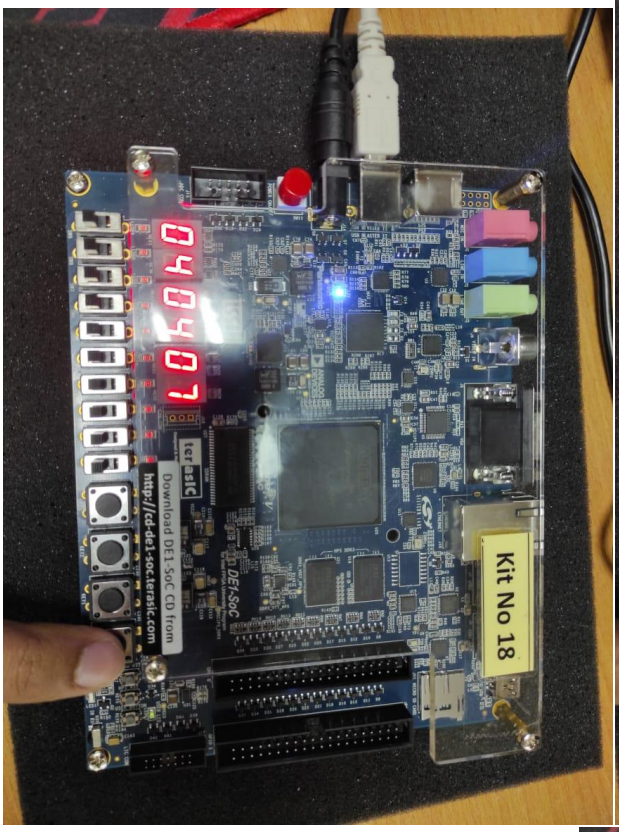

Kit No 18
Kit No 18
Kit No 18
Kit No 18

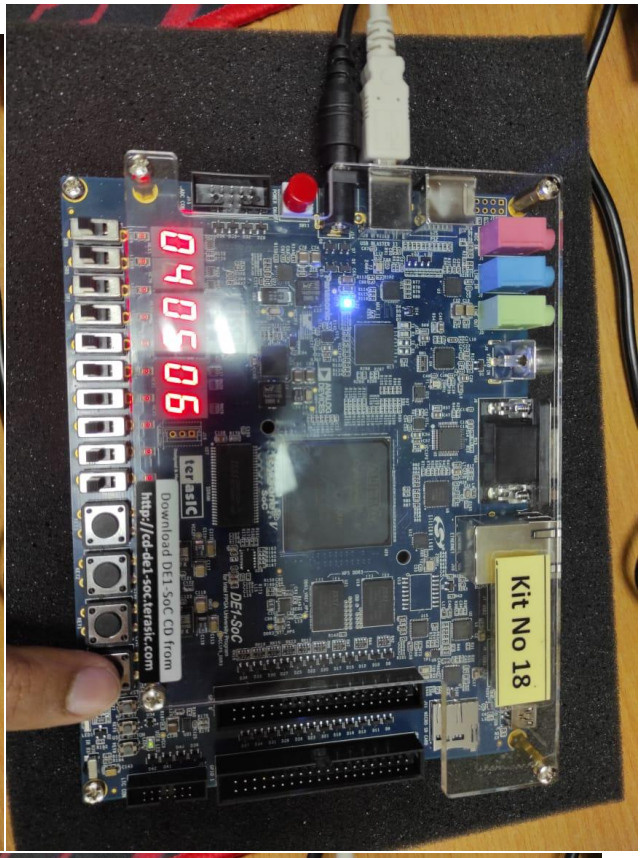

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

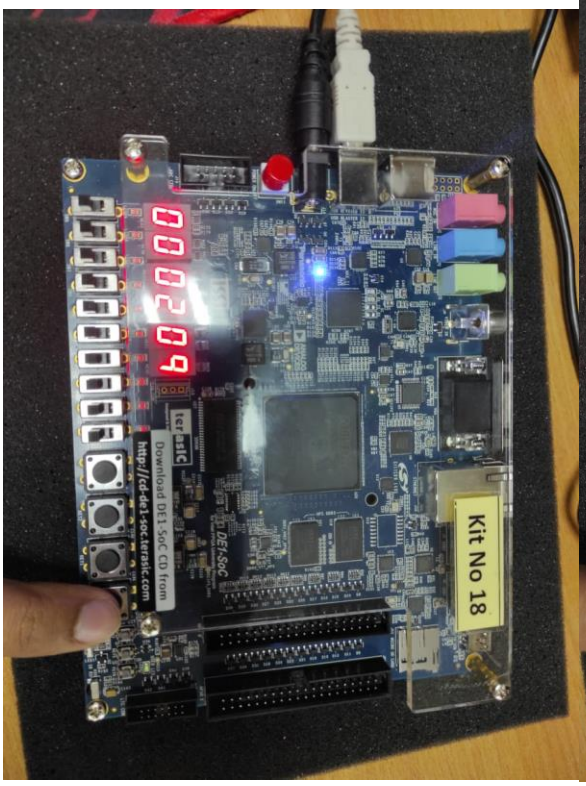

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

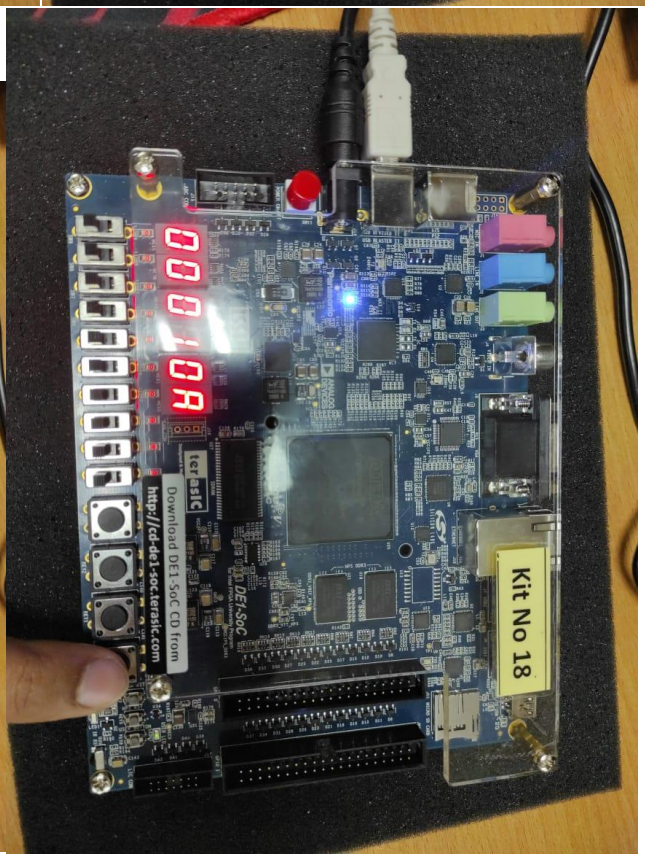

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

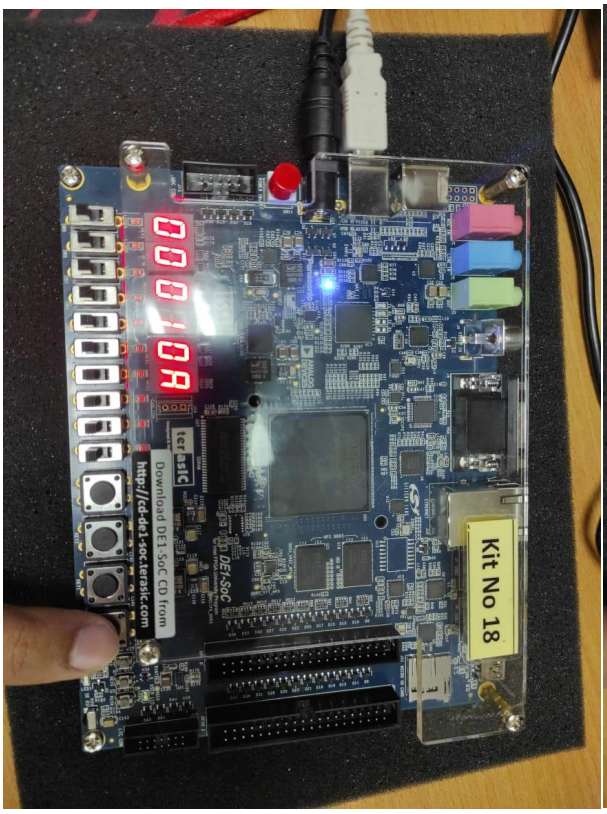

Kit No 18
Download DE1-SoC CD from
http://cd-de1-soc.terasic.com

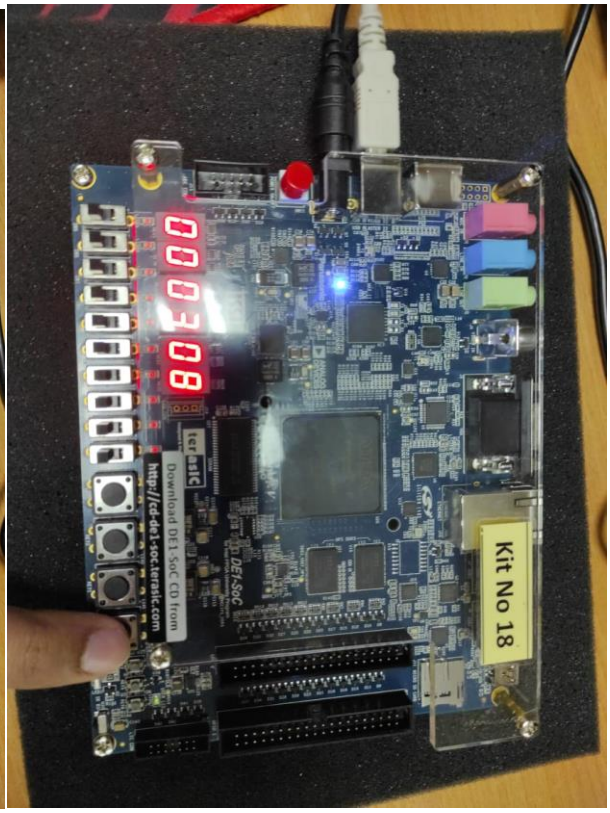

Kit No 18


Kit No 18


Kit No 18


Kit No 18


Kit No 18


Kit No 18