FINAL LAB REPORT

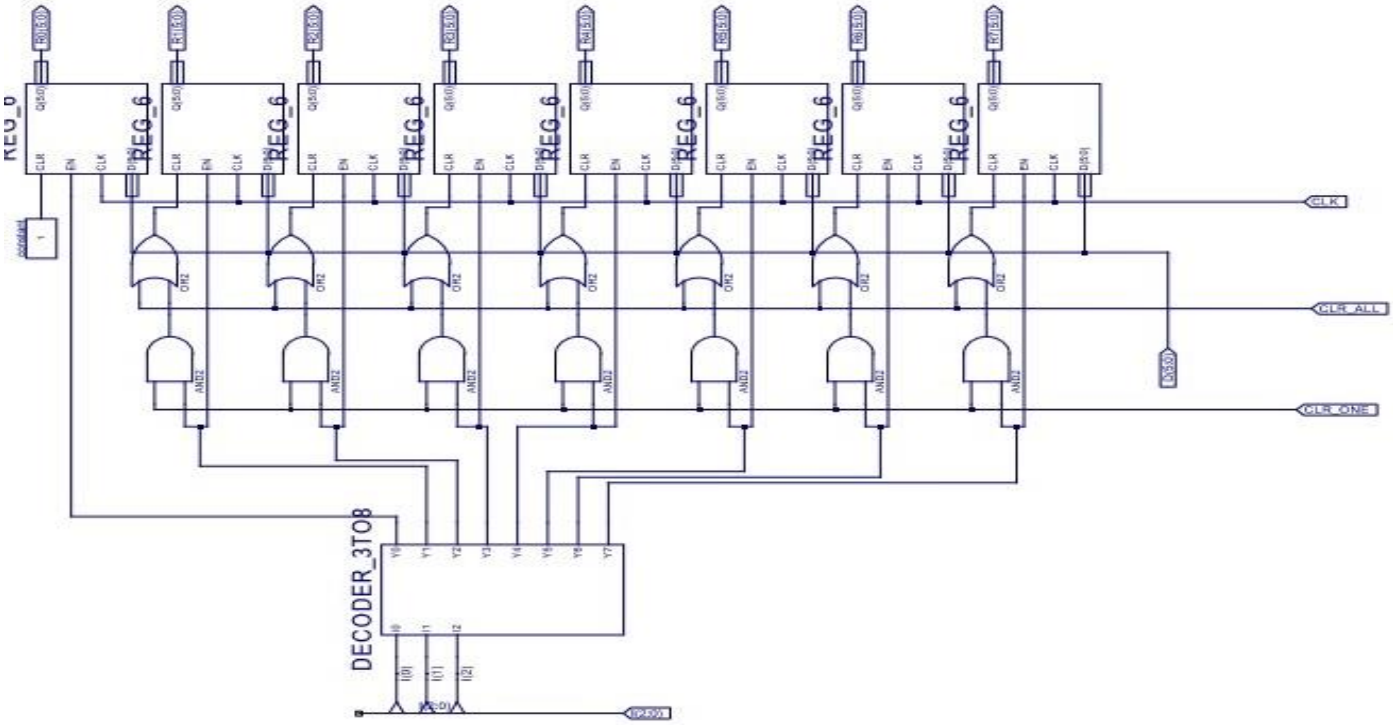
NAME	INDEX NO
W.R.A.N.M.W.RATHNAYAKE	150532E

LAB TASK

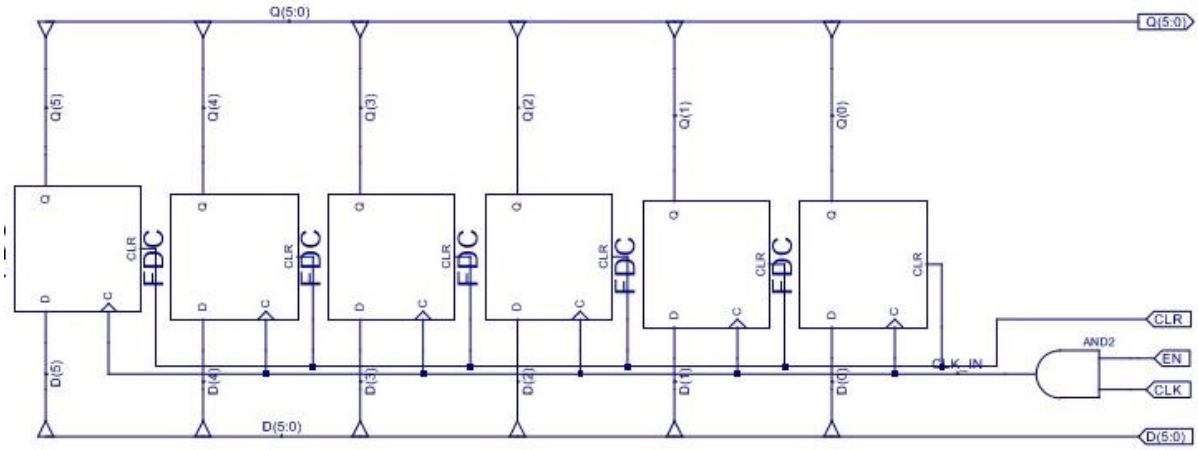
1. design and develop a 6-bit arithmetic unit that can add and subtract integers
2. decode instructions and activate necessary components on the processor
3. design and develop k-way b-bit multiplexers
4. develop and use busses
5. verify their functionality via simulation and on the development board

SCHEMATIC DIAGRAMS

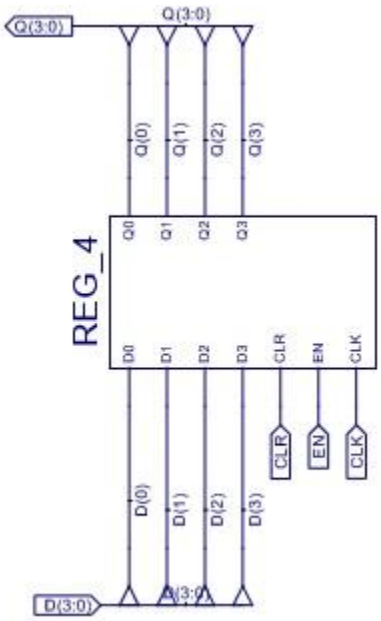
1. REGISTER BANK



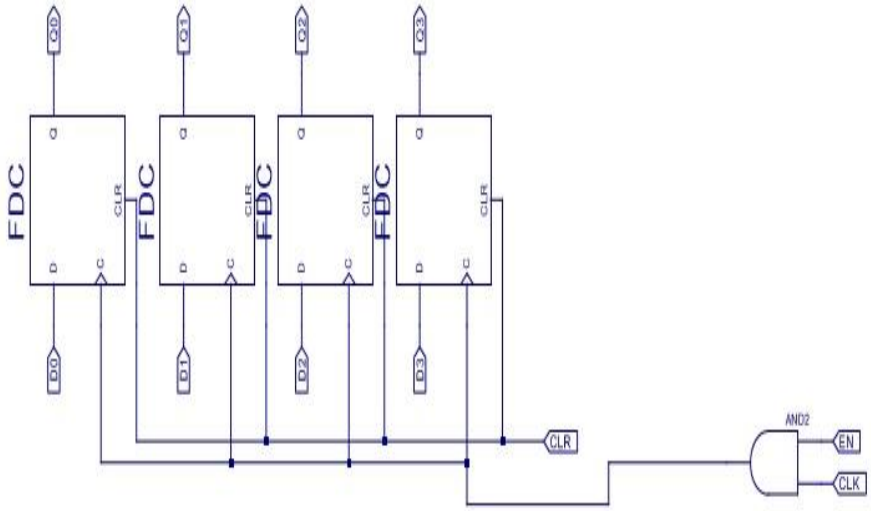
2. REGISTER 6 BITS



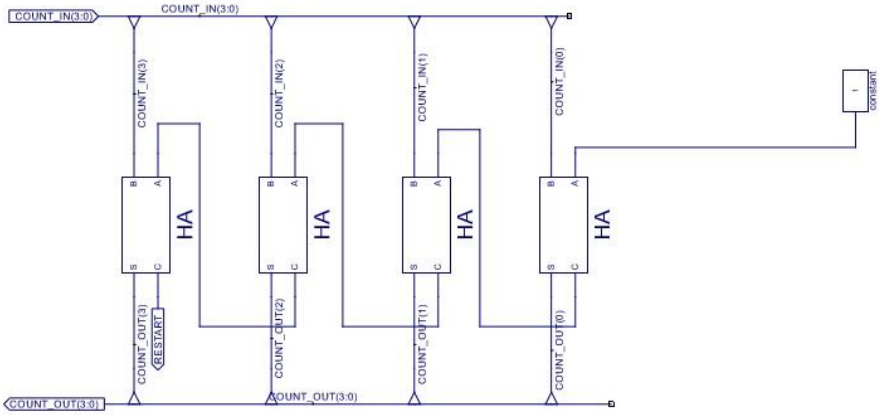
3. PROGRAM COUNTER



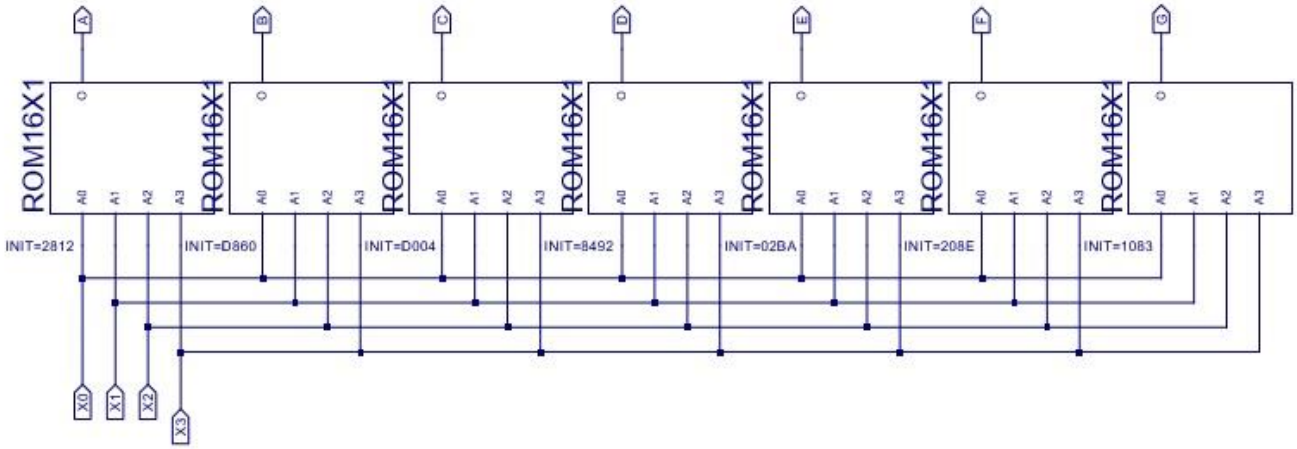
4. REGISTER 4 BITS



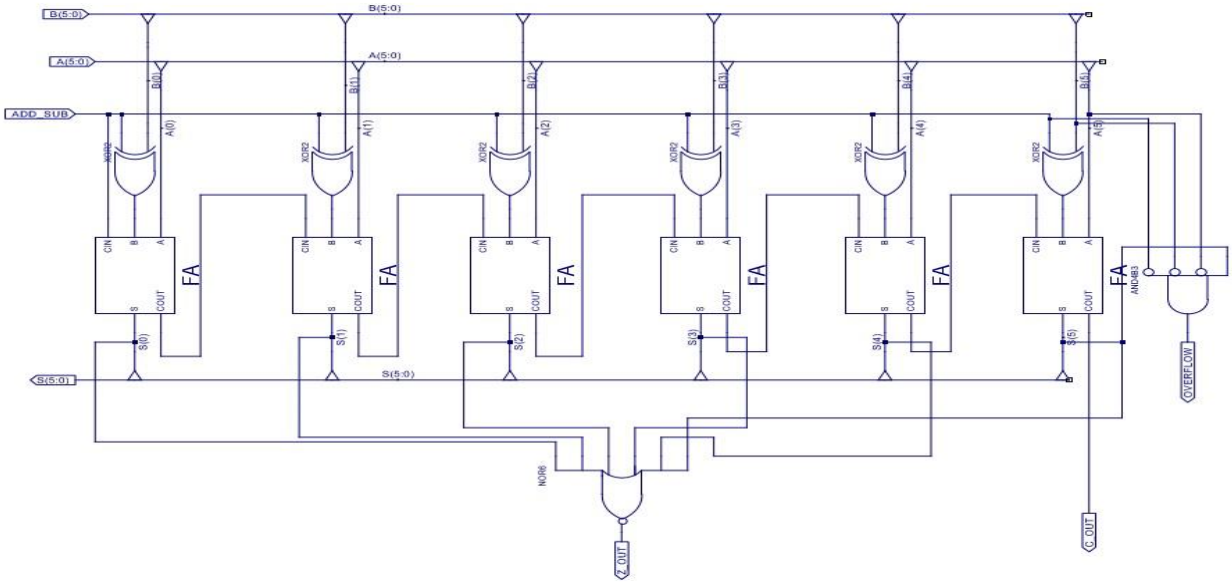
5. 4 BIT ADDER FOR COUNTER



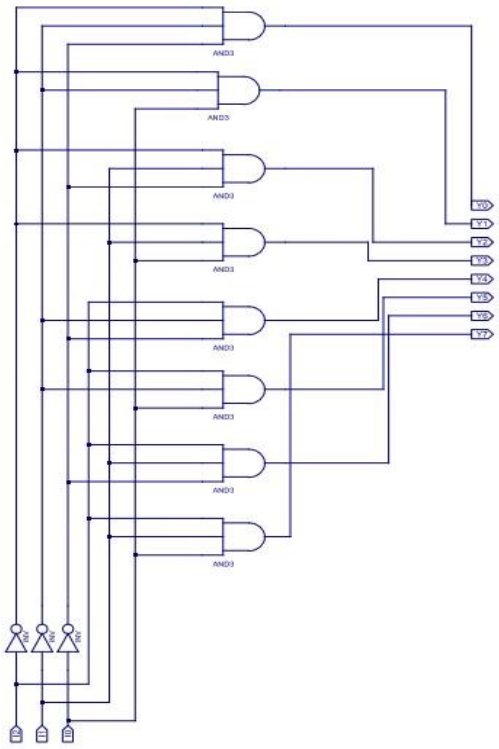
6. LUT FOR SEVEN SEGMENT DISPLAY



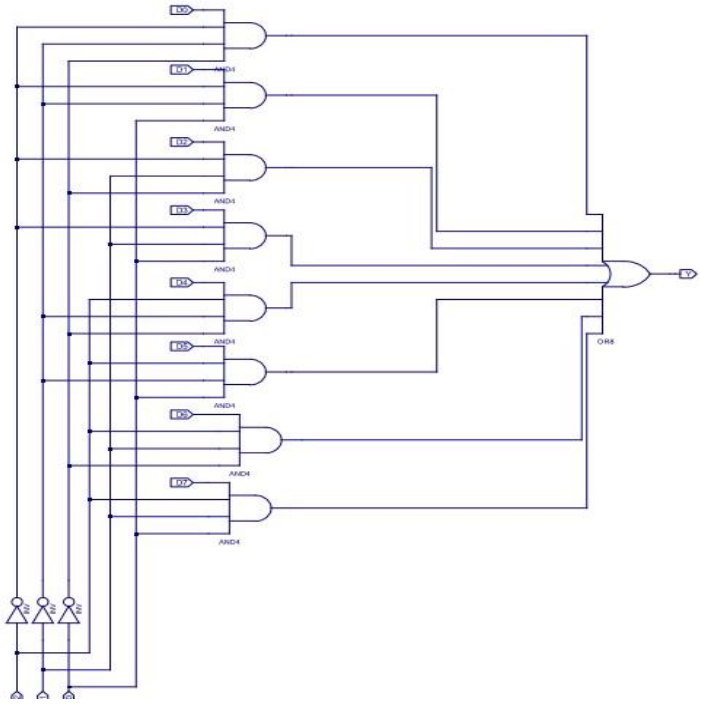
7. 6 BITS ARITHMATIC UNIT



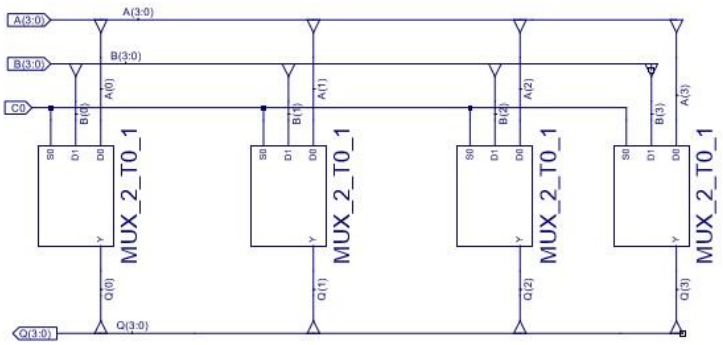
8. DECODER 3 TO 8



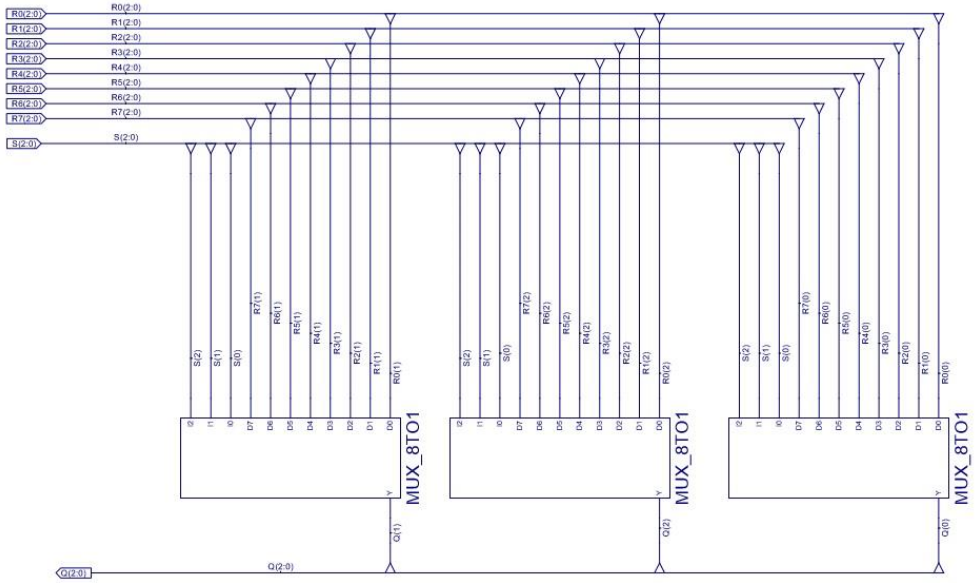
9. MUX 8 TO 1



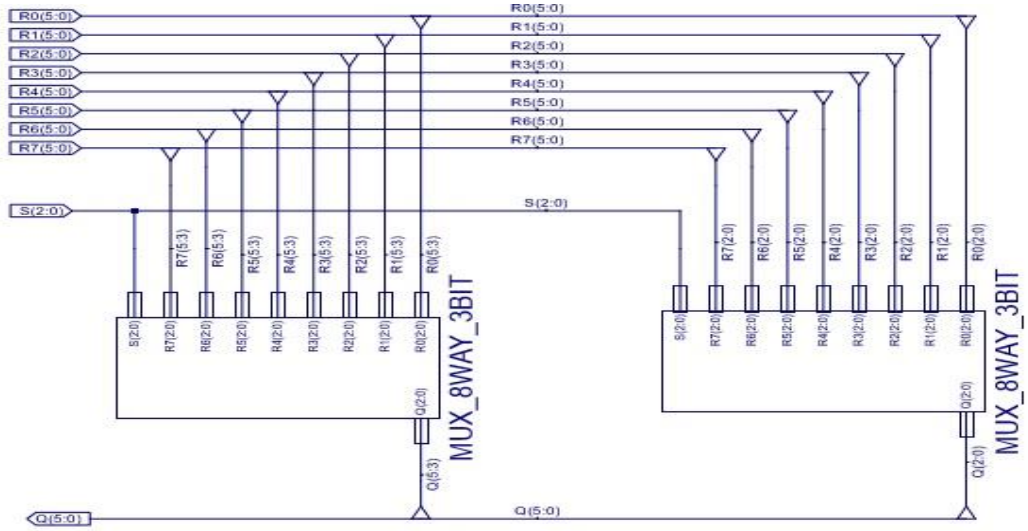
10. MUX 2 WAY 4 BITS



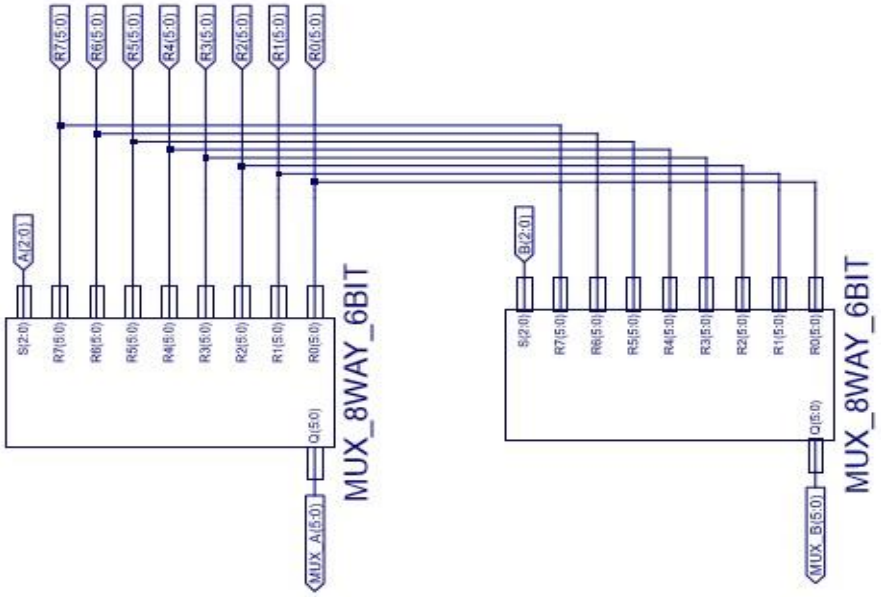
11. MUX 8 WAT 3 BITS



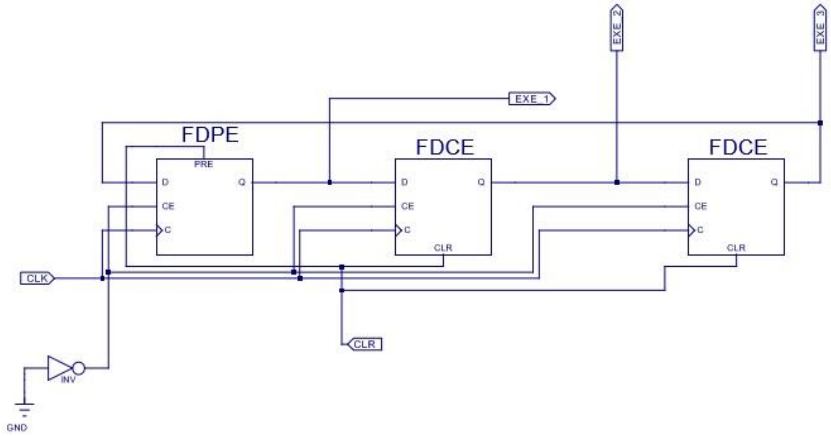
12. MUX 8 WAY 6 BITS



13. MUX UNIT 8 WAY 6 BITS 2



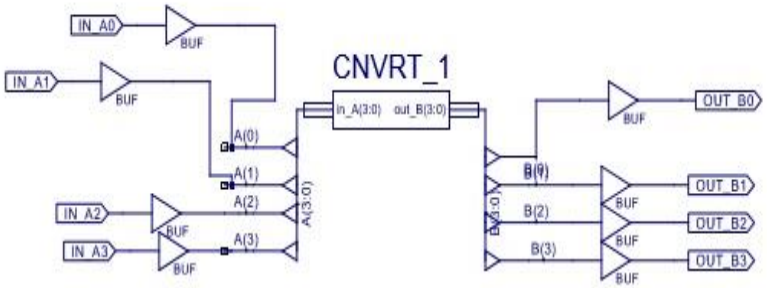
14. SEQUENCE GENERATER



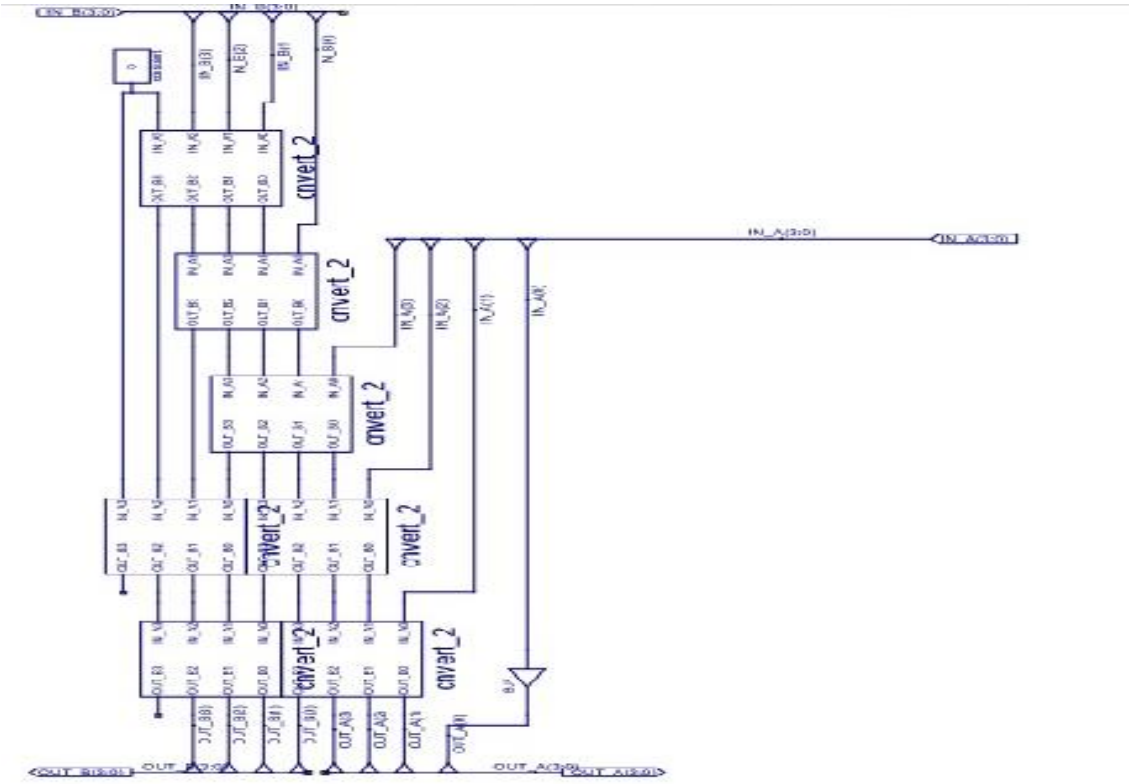
15. BINARY TO DECIMAL CONVERT UNIT 1 VERILOG MODULE

```
23
24  module convert_1(in_A,out_B);
25      input [3:0] in_A;
26      output [3:0] out_B;
27      reg [3:0] out_B;
28
29  always @ (in_A)
30  case (in_A)
31  4'b0000: out_B <=  4'b0000;
32  4'b0001 : out_B <= 4'b0001 ;
33  4'b0010 : out_B <= 4'b0010 ;
34  4'b0011 : out_B <= 4'b0011 ;
35  4'b0100 : out_B <= 4'b0100 ;
36  4'b0101 : out_B <= 4'b1000 ;
37  4'b0110 : out_B <= 4'b1001 ;
38  4'b0111 : out_B <= 4'b1010 ;
39  4'b1000 : out_B <= 4'b1011;
40  4'b1001 : out_B <= 4'b1100;
41  default: out_B <= 4'b0000;
42  endcase
43  endmodule
```

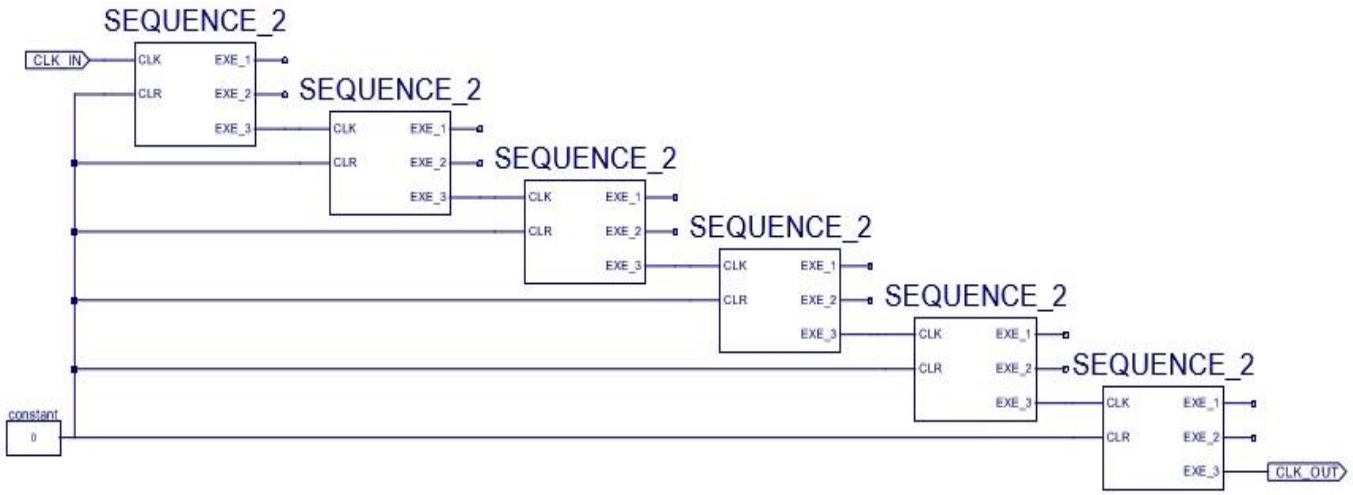
16. BIN TO DEC CONVERT



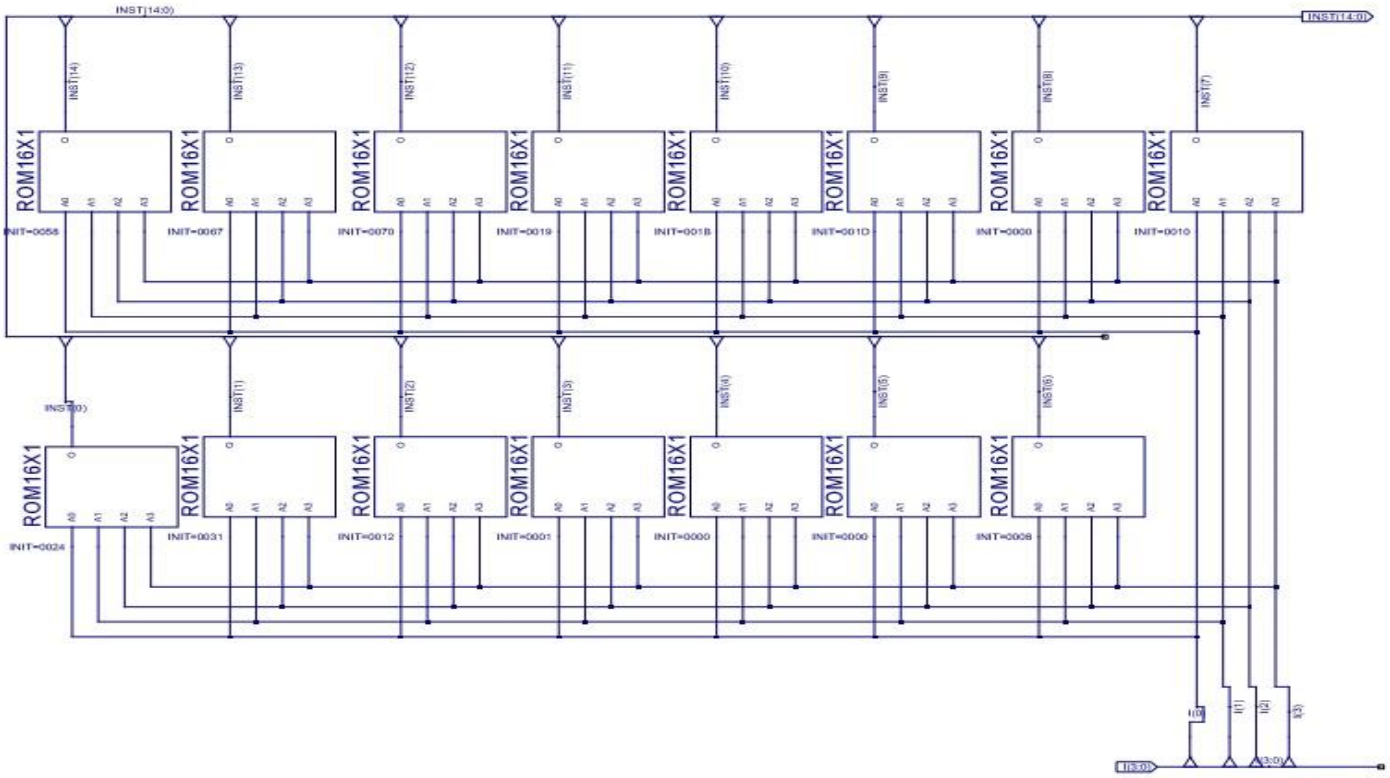
17. BIN TO DEC CONVERTER FINAL [8 BITS]



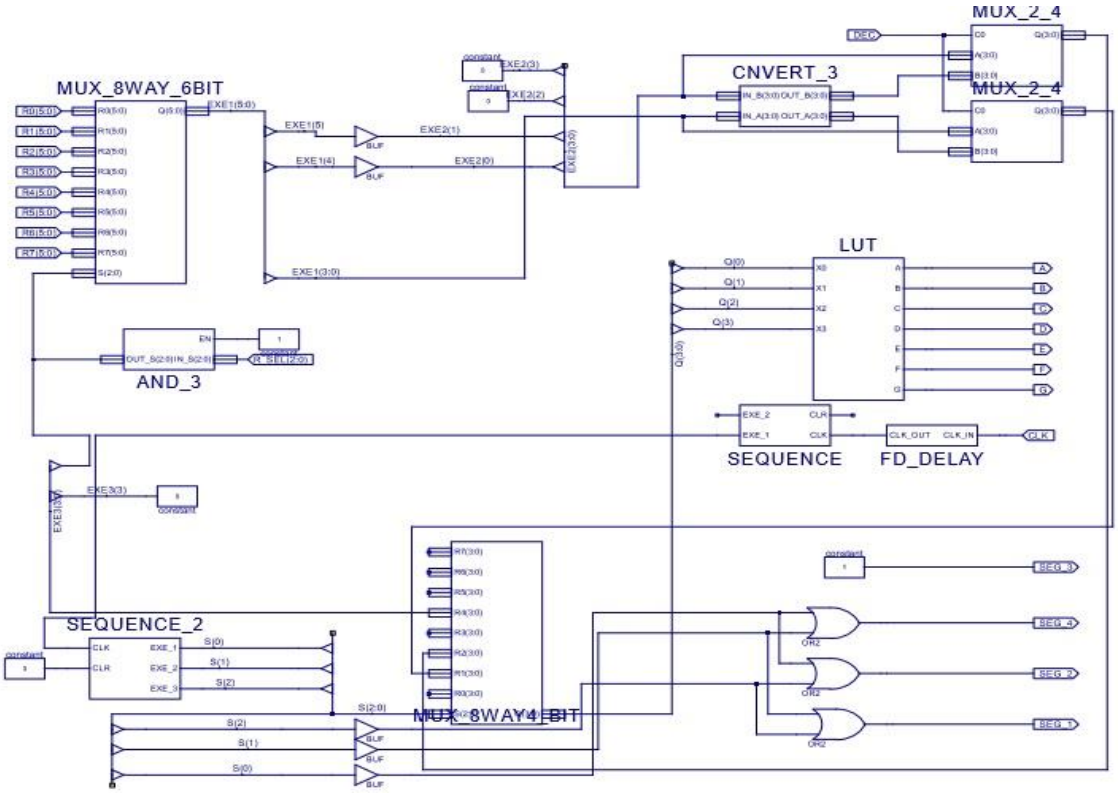
18. TIME DELAY MAKER [FD_DELAY] FOR SEVEN SEGMENTS



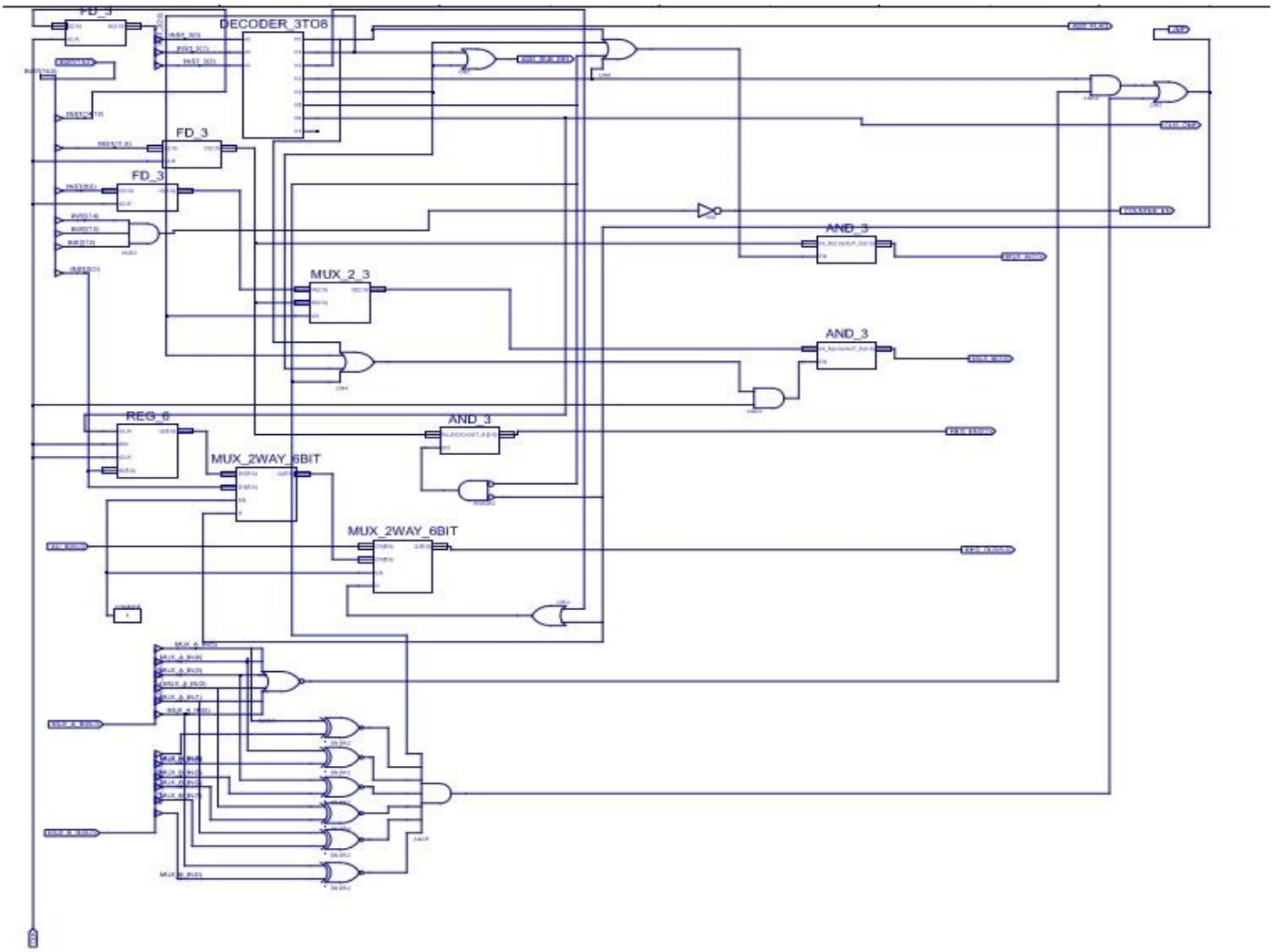
19. PROGRAM ROM 15 BITS



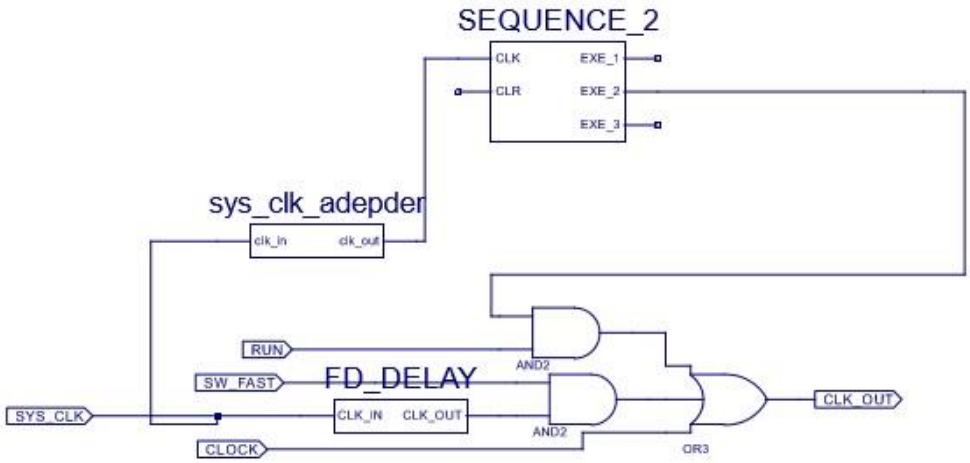
20. SEVEN SEGMENT DISPLAY ADAPTER



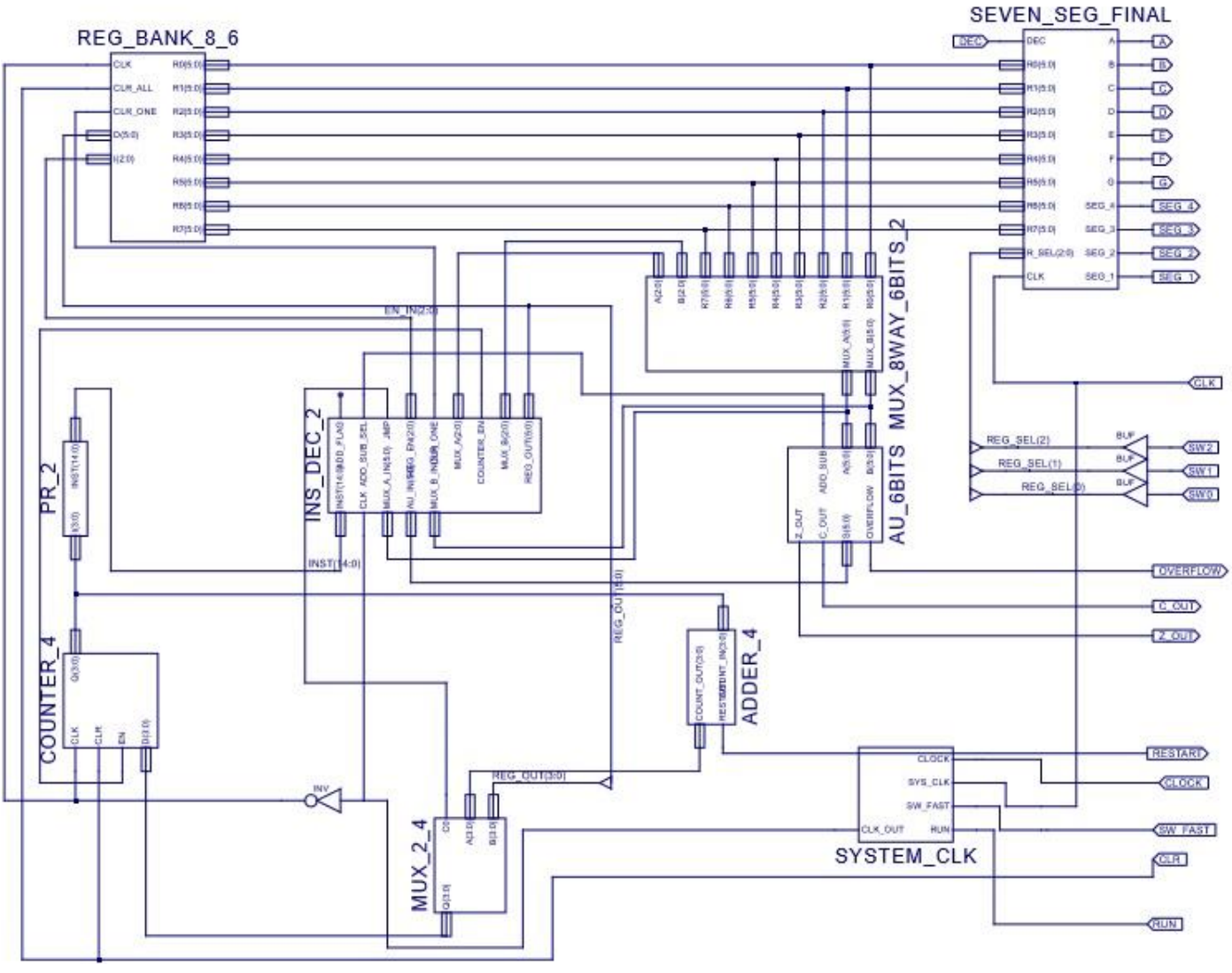
21. INSTRUCTION DECODER



22. SYS_CLOCK SWITCHER

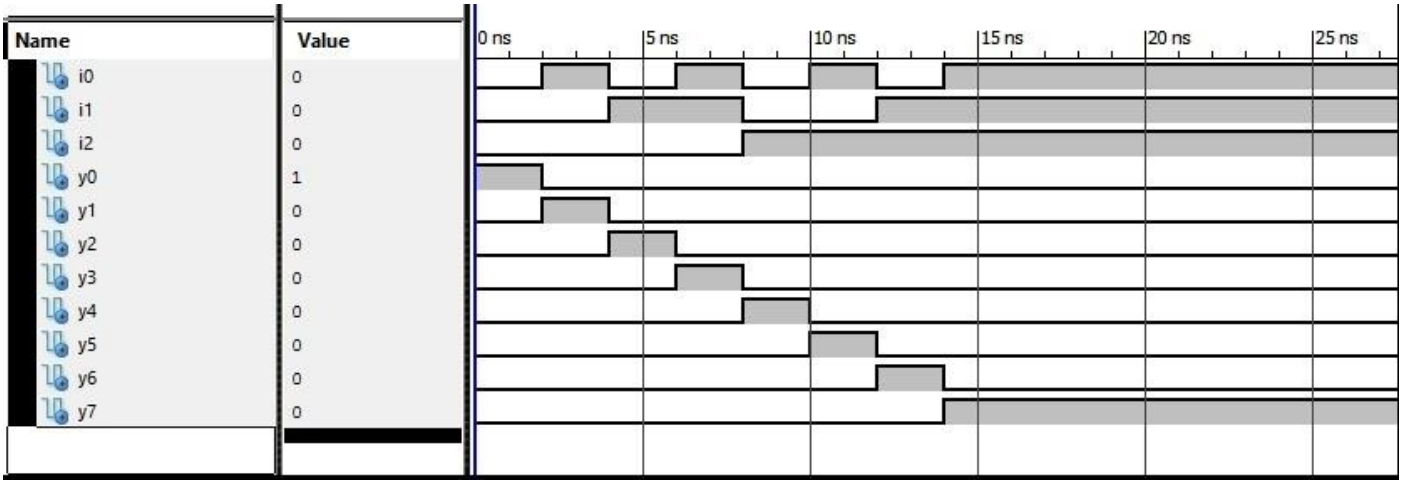


23. PROCESSOR FINAL

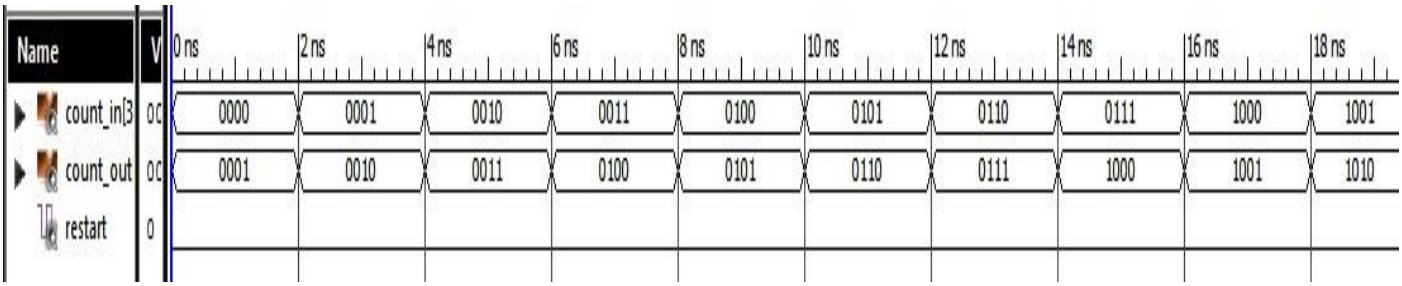


TEST BENCH TIMMING DIAGRAMS

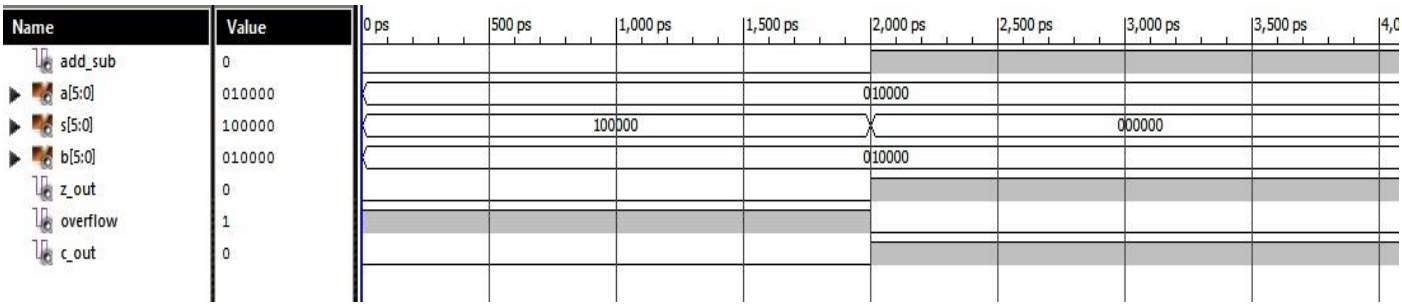
1. DECODER 3 TO 8



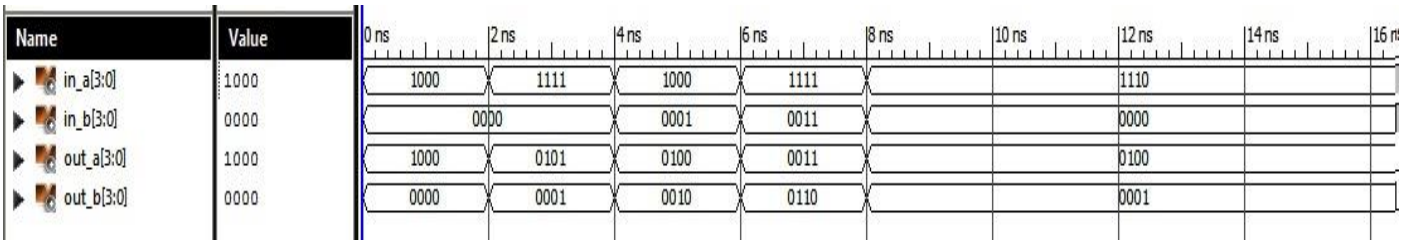
2. ADDER 4 BITS



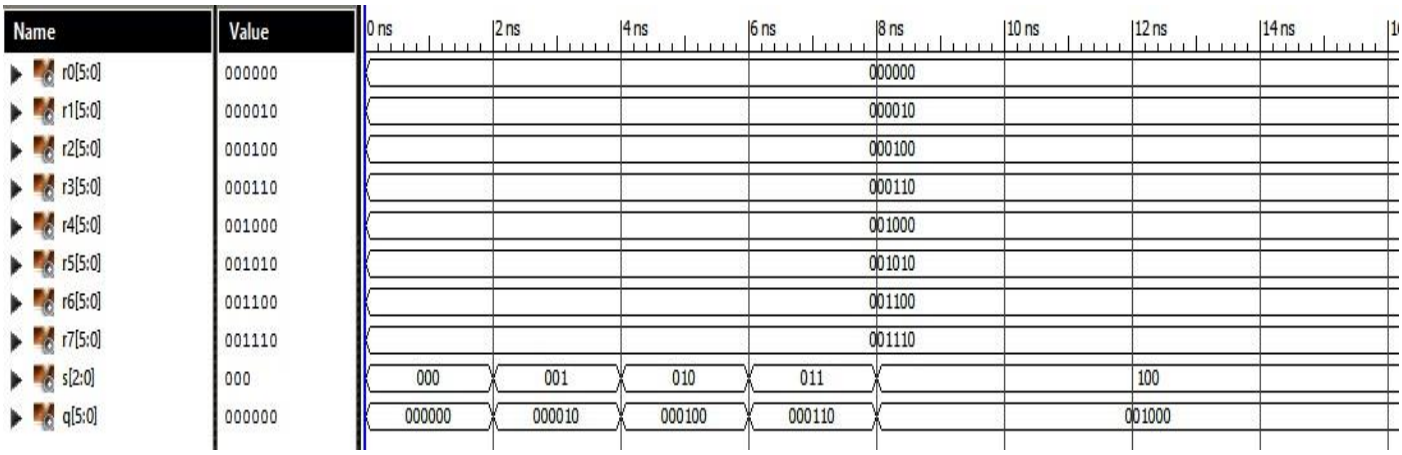
3. ARITHMATIC UNIT 6 BITS



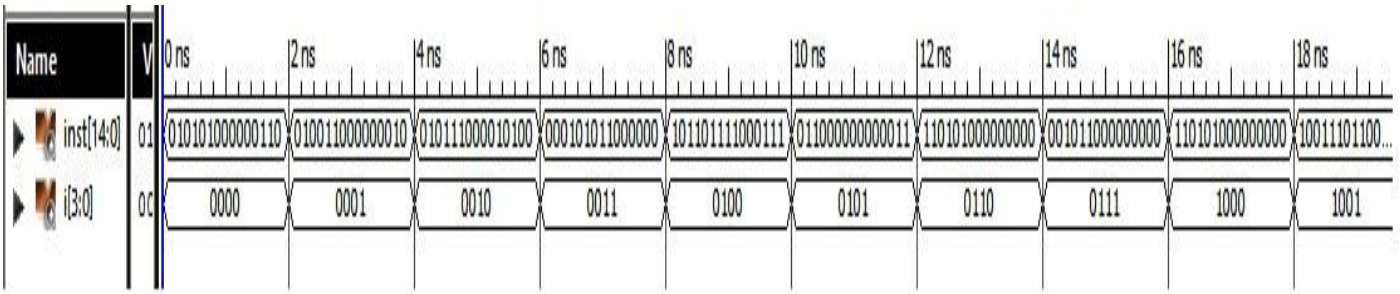
4. BIN TO DEC CONVERTER



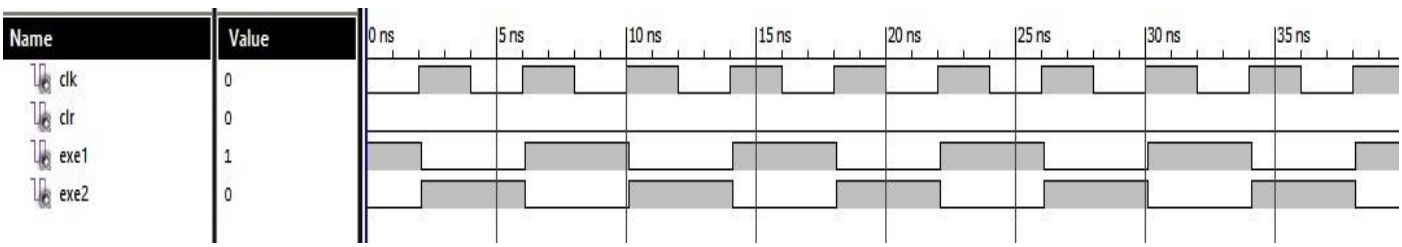
5. MUX 8 TO 1



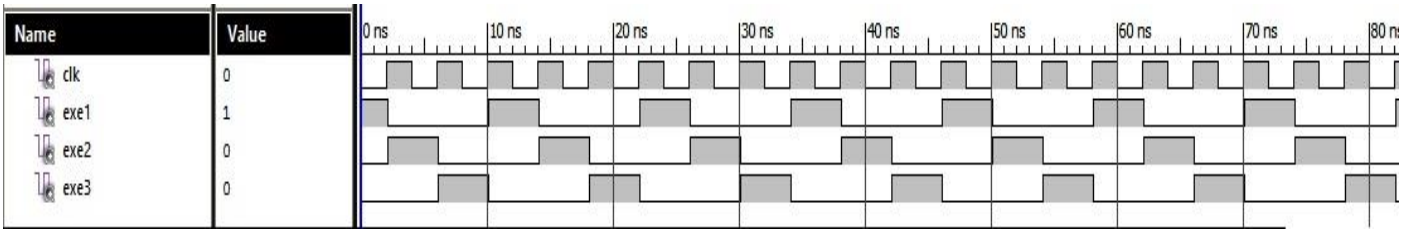
6. PROGRAM ROM



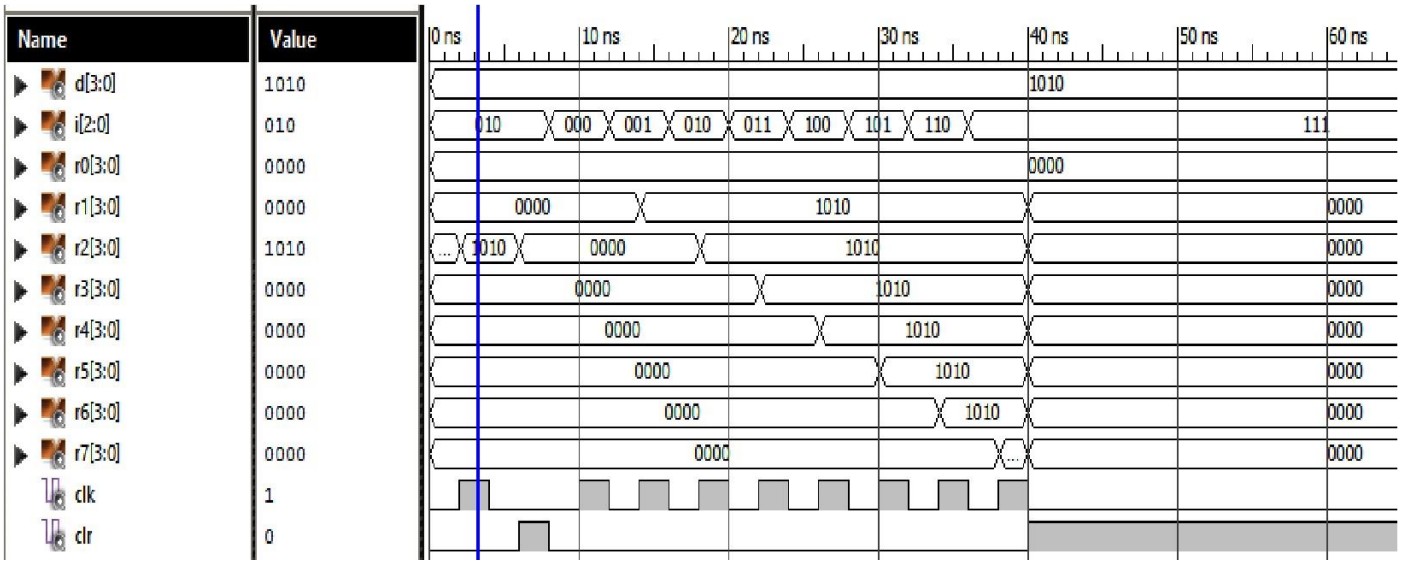
7. SEQUENCE GENARATER 1



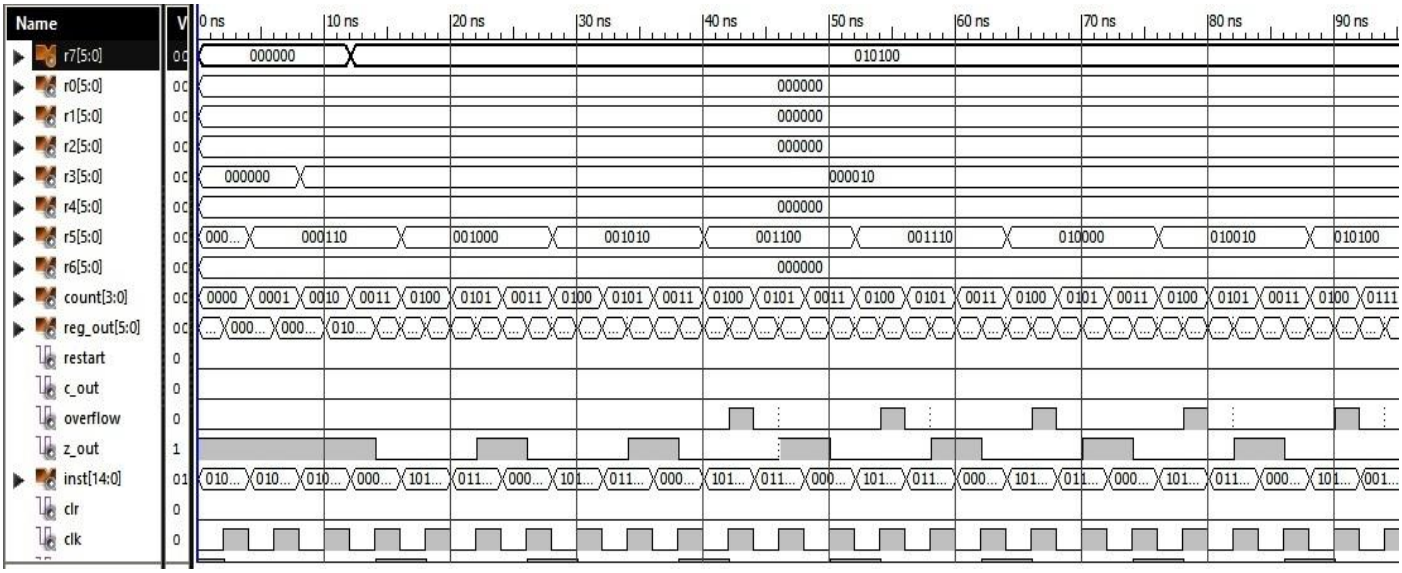
8. SEQUENCE GENARETER 2



9. REGISTER BANK



10.FINAL 6 BIT PROCESSOR



SIMMULATING ASSEMBLY CODES

1. PROGRAM 1

0 - MOVI,R7,10 [0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0]

1 - MOVI,R2,4 [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]

2 - MOVI,R1,1 [0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]

3 - SUB,R7,R1 [1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]

4 - COMP,R7,R2,6 [1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0]

5 - JZR,R0,3 [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]

6 - END [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

2. PROGRAM 2

0 - MOVI,R4,10 [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]

1 - MOVI,R3,50 [0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0]

2 - MOVI,R6,5 [0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]

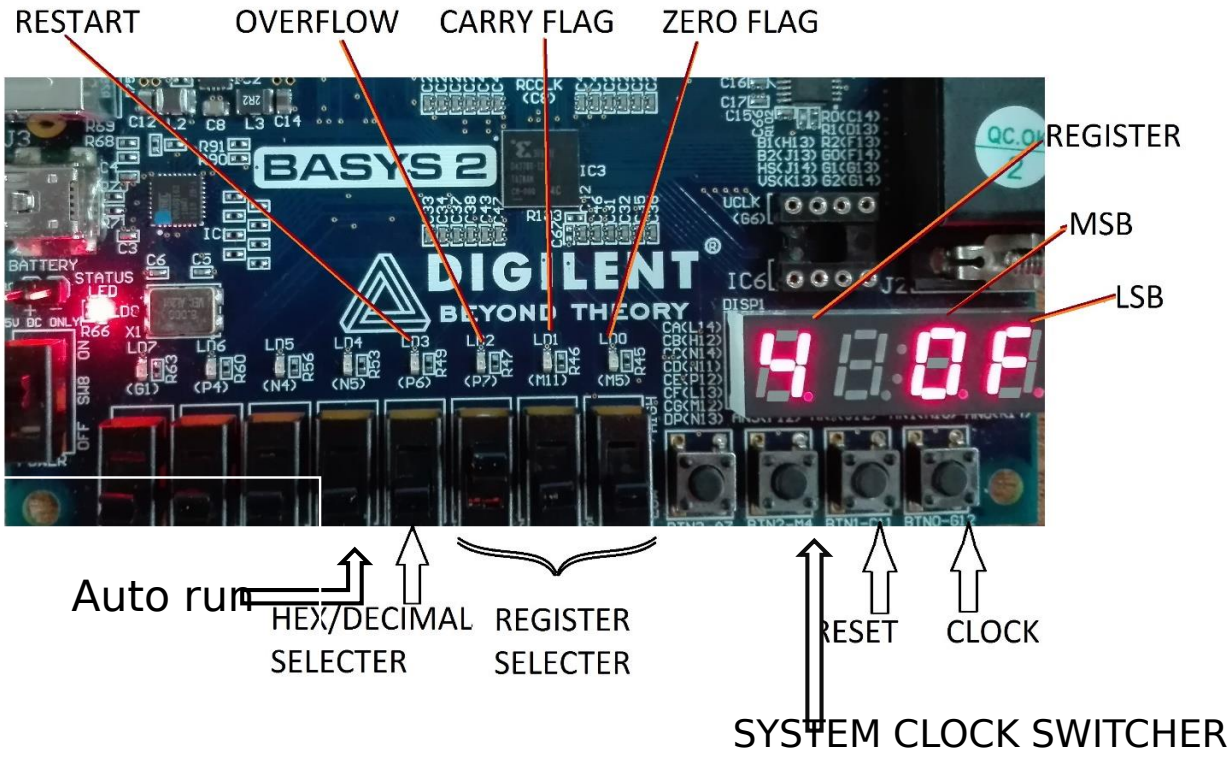
3 - ADD,R4,R6 [0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]

4 - COMP,R4,R3,6 [1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0]

5 - JZR,R0,3 [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]

6 - END [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

CP-132 BOARD CONFIGURATION



PROCESSOR SPECIFICATIONS

1. SUPPORT 8 INSTRUCTIONS AND 6 BIT REGISTERS

		INSTRUCTION[14:0]														
DECCODE NUM	INSTRUCTION	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ADD	0	0	0	R(A)	R(A)	R(A)	R(B)	R(B)	R(B)	0	0	0	0	0	0
1	NEG	0	0	1	R(A)	R(A)	R(A)	0	0	0	0	0	0	0	0	0
2	MOVI	0	1	0	R(A)	R(A)	R(A)	0	0	0	D	D	D	D	D	D
3	JZR	0	1	1	R(A)	R(A)	R(A)	0	0	0	0	0	D	D	D	D
4	SUB	1	0	0	R(A)	R(A)	R(A)	R(B)	R(B)	R(B)	0	0	0	0	0	0
5	COMP	1	0	1	R(A)	R(A)	R(A)	R(B)	R(B)	R(B)	0	0	D	D	D	D
6	CLR	1	1	0	R(A)	R(A)	R(A)	0	0	0	0	0	0	0	0	0
7	END	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

1. ADD : ADD REGISTER A AND REGISTER B ,THEN ANSWER GOES TO REGISTER
2. NEG : NEGETION OF REGISTER A SAVE IN ITSELF
3. MOVI : MOVE INTEGER VALUE D UP TO 63 ,MOVES TO REGISTER A
4. JZR : JUMP TO GIVEN D VALUE'TH CODE LINE IF REGISTER A IS ZERO
5. SUB : SUBTRACT REGISTER A FROM REGISTER B ,THEN ANSWER GOES TO REG
6. COMP : JUMP TO GIVEN D VALUE'TH CODE LINE IF REGISTER A IS EQUAL TO RE
7. CLR : CLEAR REGISTER A
8. END : END PROGRAM ,THEN NOTHING CHANGES OCCUR IN SYSTEM WITH CLO PULSES

2. WORK TO INTEGER VALUES UP TO 63 WITH 6 BIT AU AND REGISTERS
3. DISPLAY EXACT REGISTER VALUE HEXA DECIMAL OR DECIMAL CORRECTLY
4. ABILITY TO CHECK ALL REGISTER VALUES ON SEVEN SEGMENT DISPLAY BY SWITHING S
5. CONVERT ANY ANSWER TO DECIMAL BY SW 4
6. ALL INSTRUCTIONS ARE RUN WITH EXACT ONE CLOCK PULSE
7. ABILITY TO EXECUTE 16 CODE LINES BY THE USAGE OF 4 BIT ADDER
8. ABILITY TO OBSERVE ZERO, CARRY OUT, OVERFLOW BY FLAGS LED[0,1,2]
9. SWITCH TO SYSTEM CLOCK ONCE AND EXECUTE THE CODE HIGH FAST PUSHING BTN 2
10. SWITCH TO AUTORUN PROCEDURE BY SW 5.

CONCLUSIONS

1. WE CAN DEVELOP A SIMPLE NANO PROCESSOR BY THE COMBINATION OF BASIC LOGIC
   FLOP, AND ROM S.
2. CLOCK OF THE SYSTEM CAN BE OVERWRITE TO A PUSH BUTTON SWITCH
3. THERE EXISTS CONSIDEREBLE DELAY OF LOGIC GATES IN COMBINATIONS THAT EFFECT
   CIRCUIT TIMMING
4. BY USING GATED CLOCK TO DIFFER THE TIMING OF VARIOUS ELEMENTS OF CIRCUIT, W
   MANAGE THE ELEMENTS DELAY
5. VERY HIGH FREAQUENCY OF SIMMULATING BOARD IS COULDN’T DERECTELY USE FOR A
   MULTIPLE SEVEN SEGMENTS AND MAIN EXCECUTION.
6. USAGE OF FLIP-FLOPS TO REDUSE SYSTEM CLOCK RATE IS THE METHOD OF SYSTEM CL
7. THE GATES, FLIP-FLOPS, ROM S, AND INPUT OUTPUT MARKERS OF SIMMULATING BOARI
   LIMITED
8. SYSTEM PUSH BUTTONS CAN DEVELOP MORE THAN ONE CLOCK PULSE FOR AN ONE PU