Output of Computer Architecture Lab

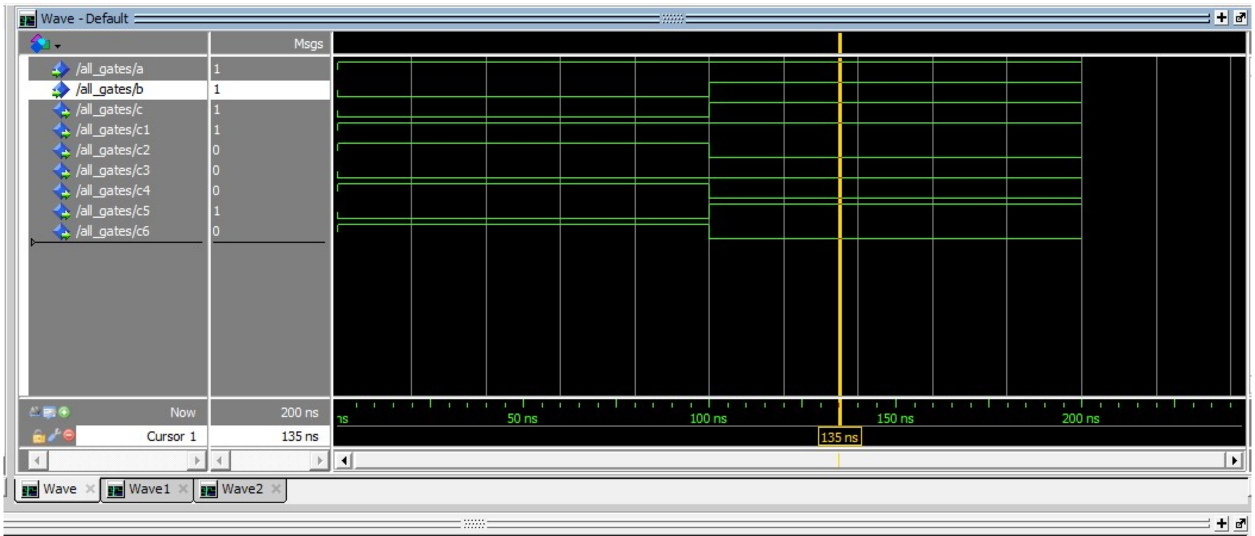

Figure: All basic gates

Output of Computer Architecture Lab

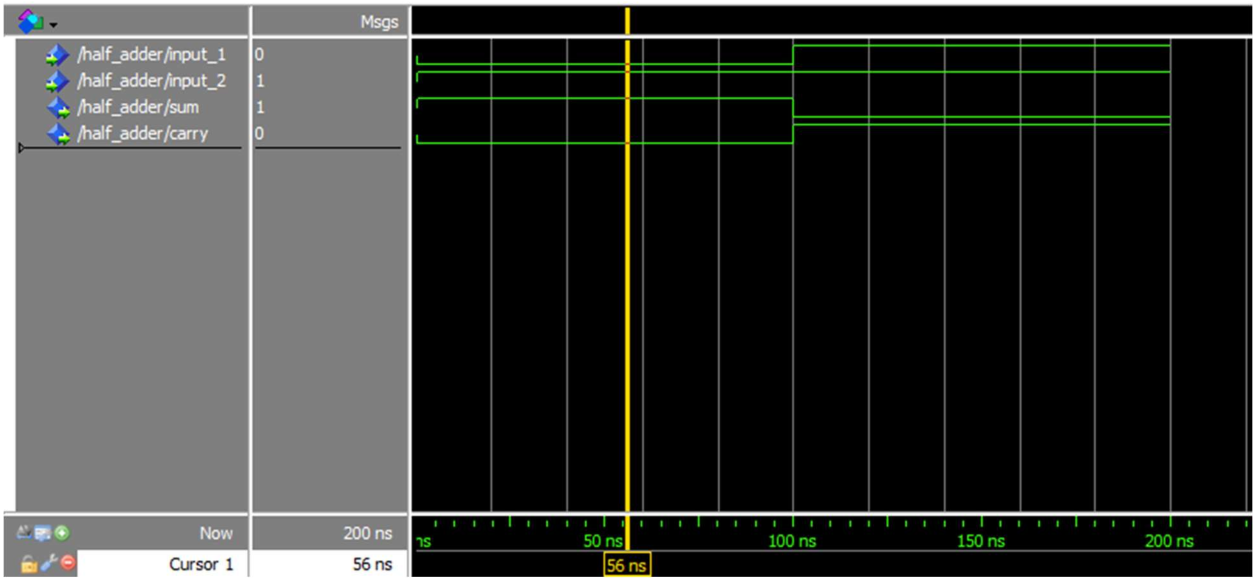

Figure: Half Adder

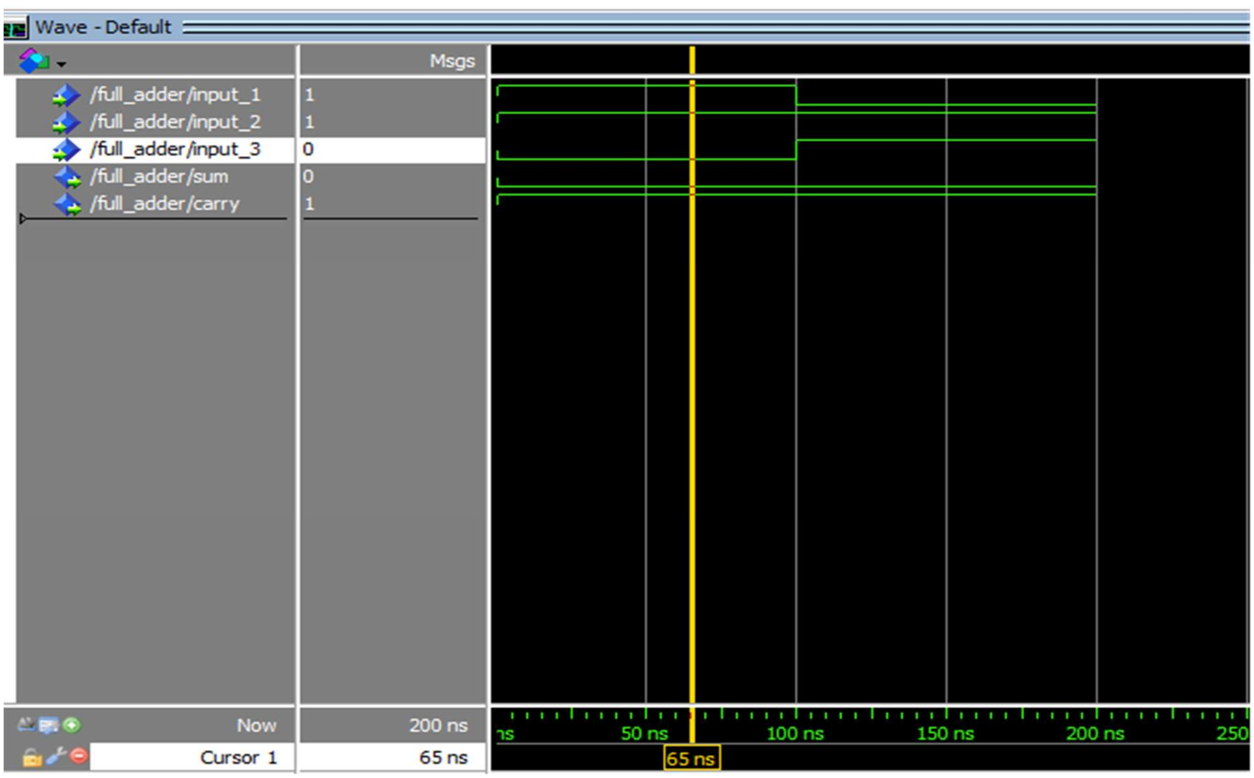

Figure: Full Adder

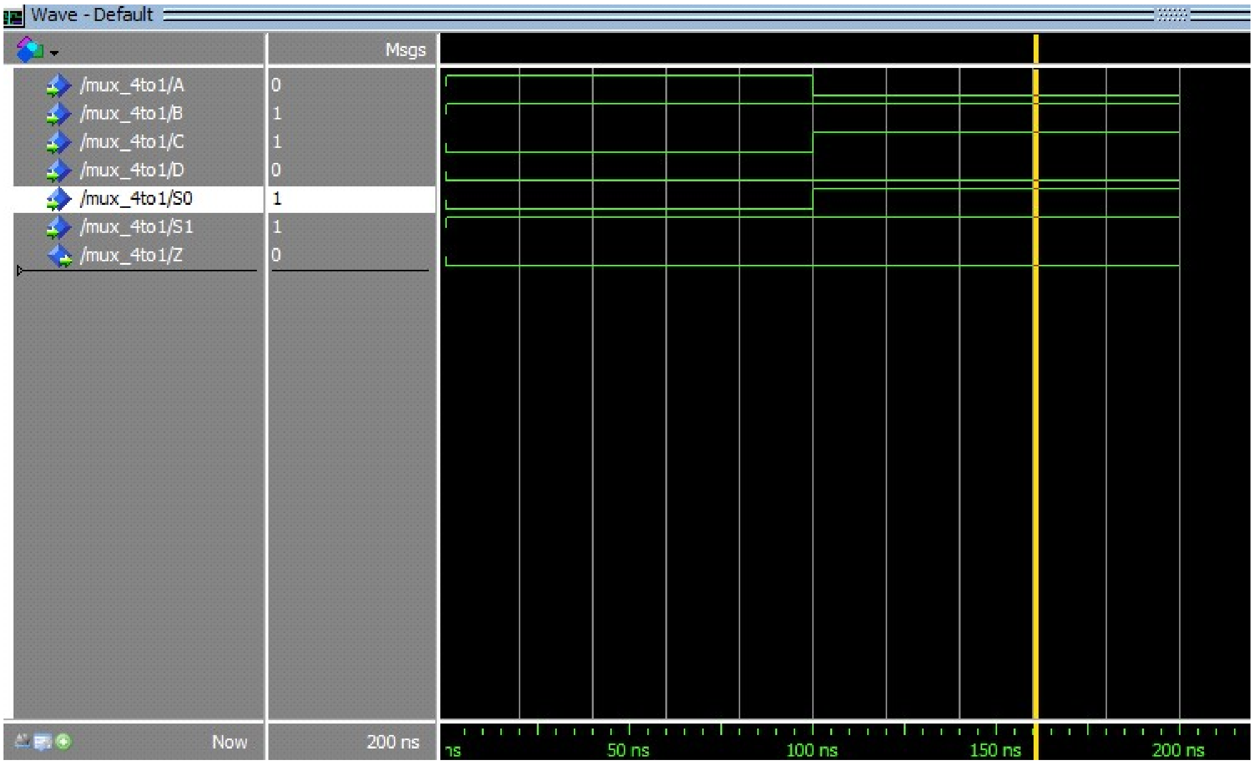

Figure: 4*1 Multiplexer

Figure: 1*4 Demultiplexer

Output of Computer Architecture Lab

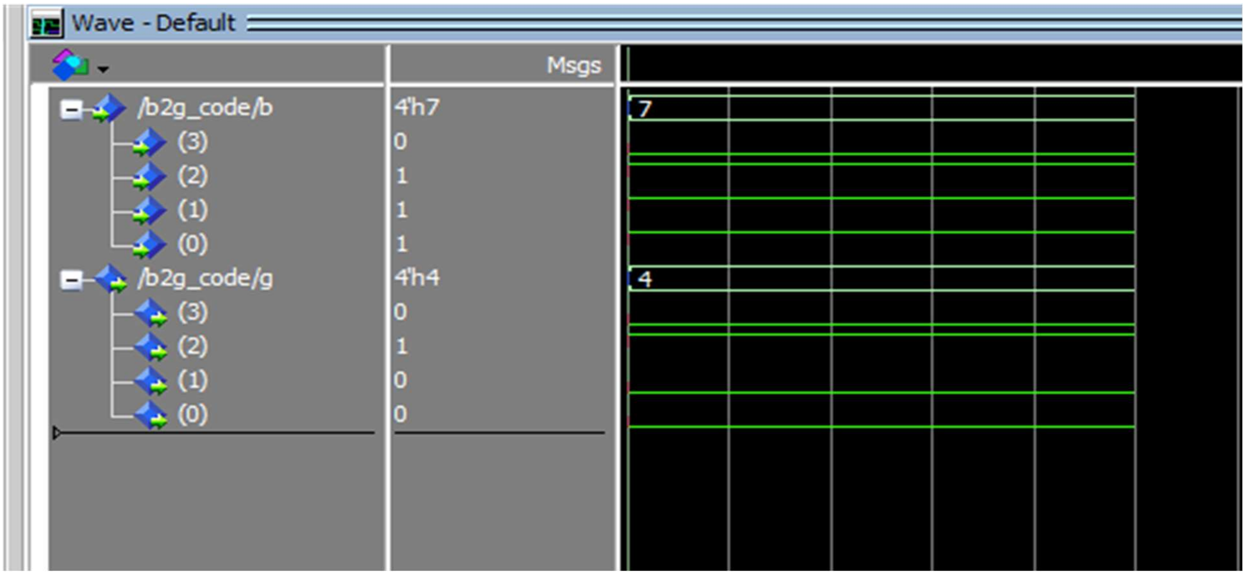

Figure: Binary to Gray code converter

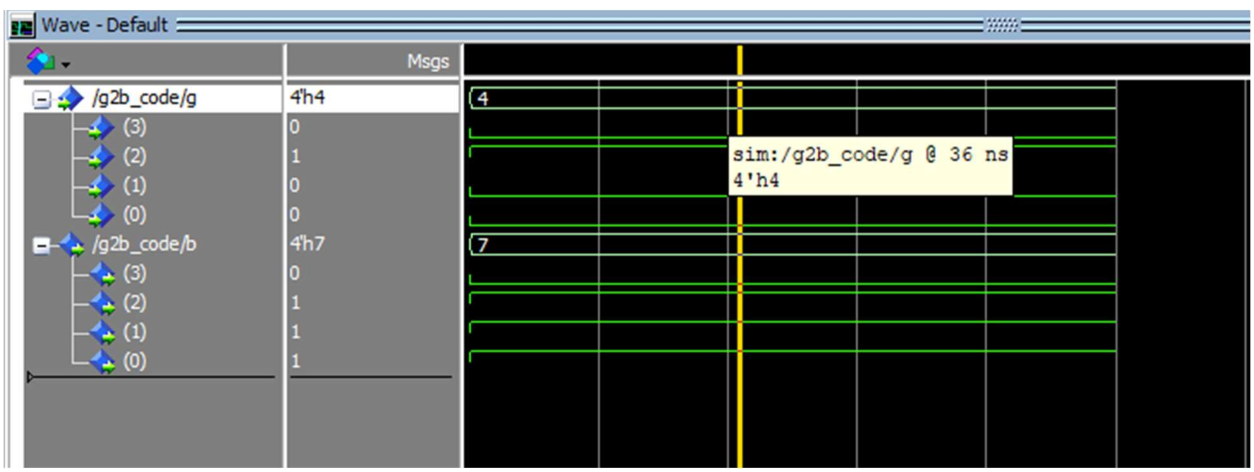

Figure: Gray to Binary code converter

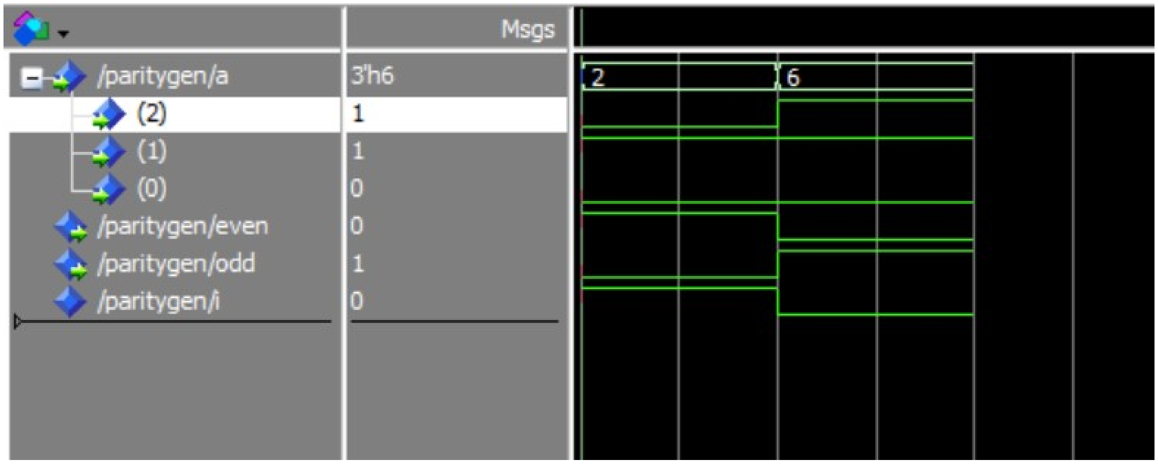

Figure: Parity Generator

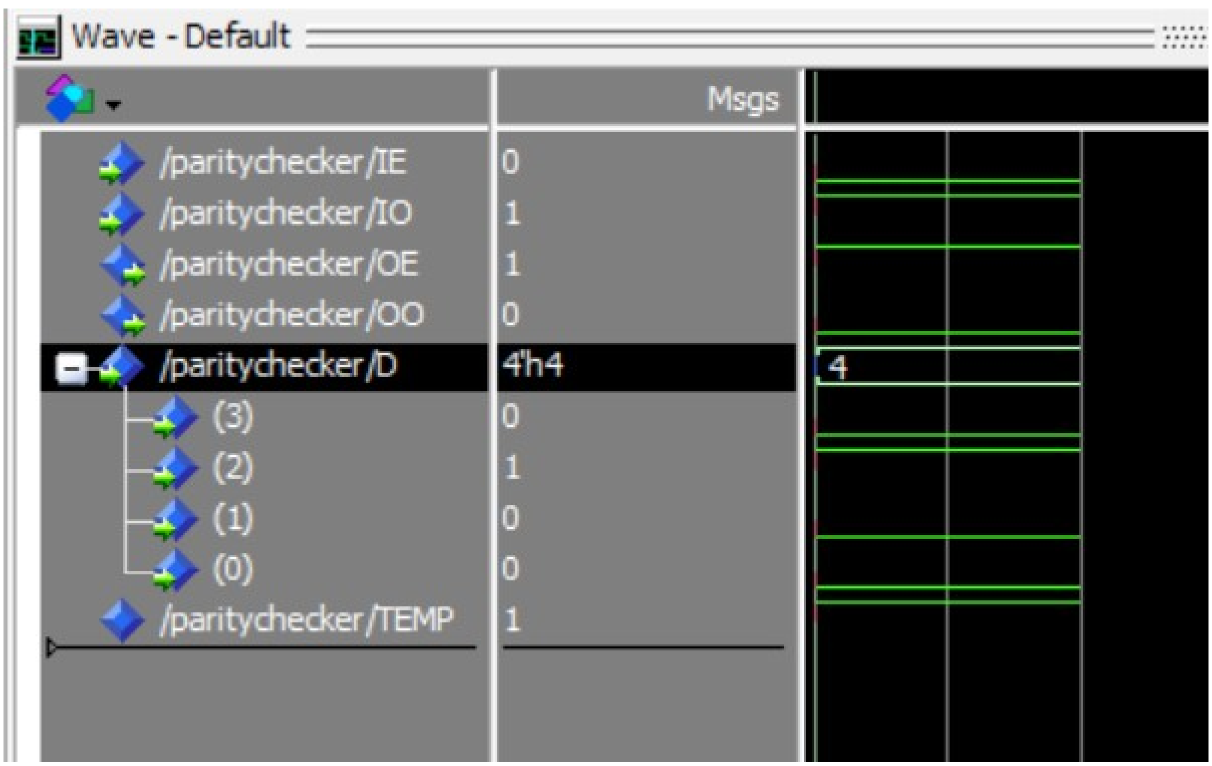

Figure: Parity Checker

Output of Computer Architecture Lab

Figure: 3*8 Decoder

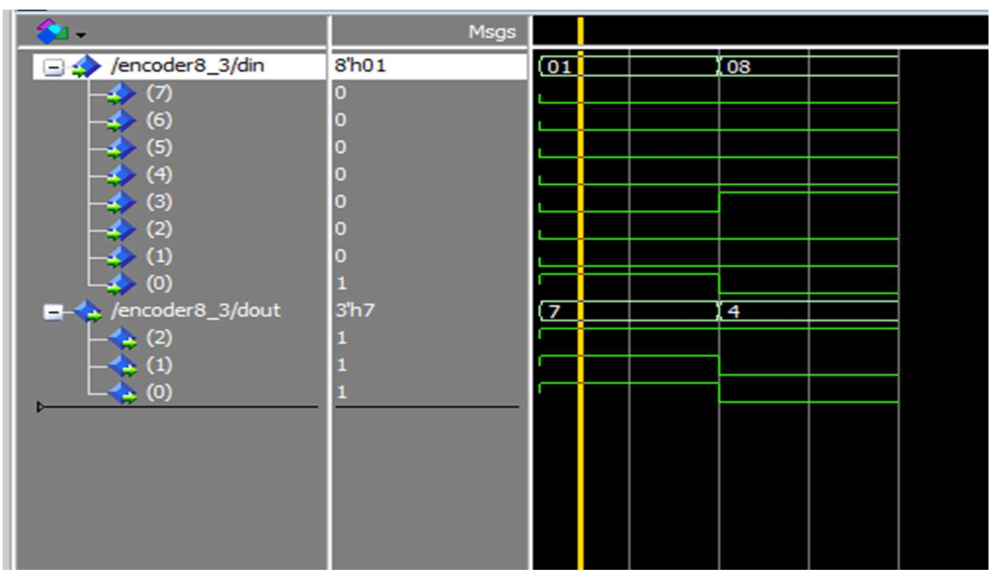

Figure: 8*3 Encoder

Output of Computer Architecture Lab

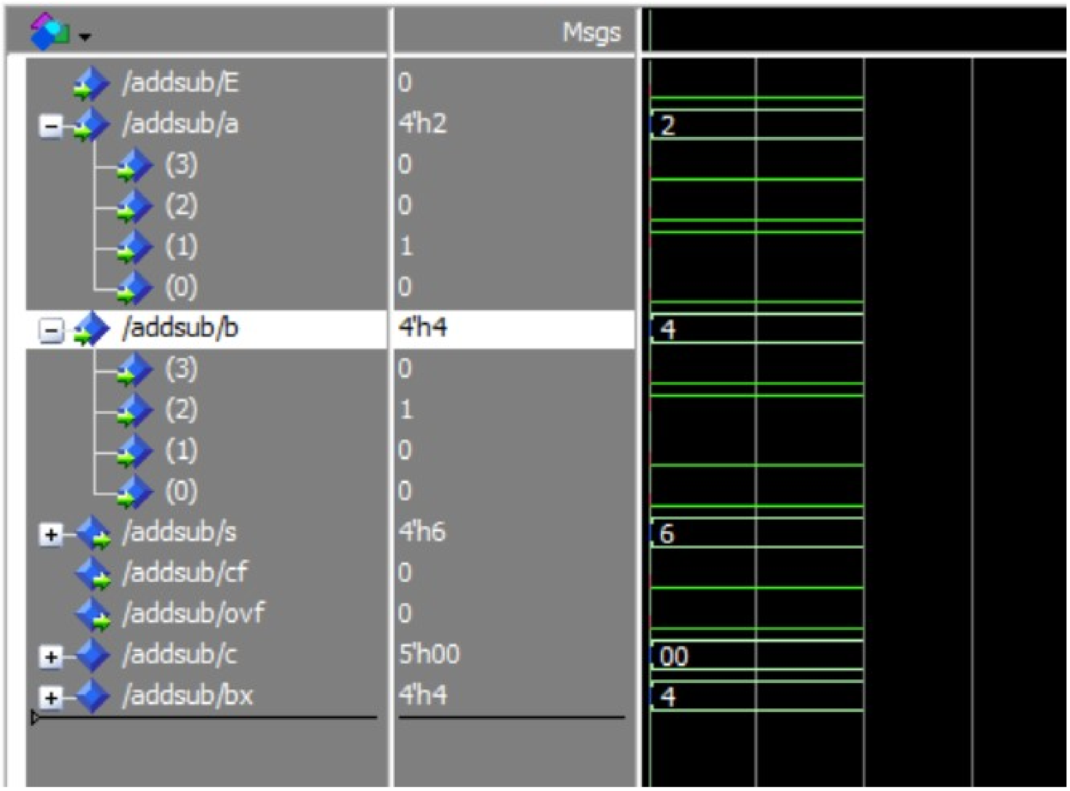

Figure: 2's Complement Adder-Subtractor

Output of Computer Architecture Lab

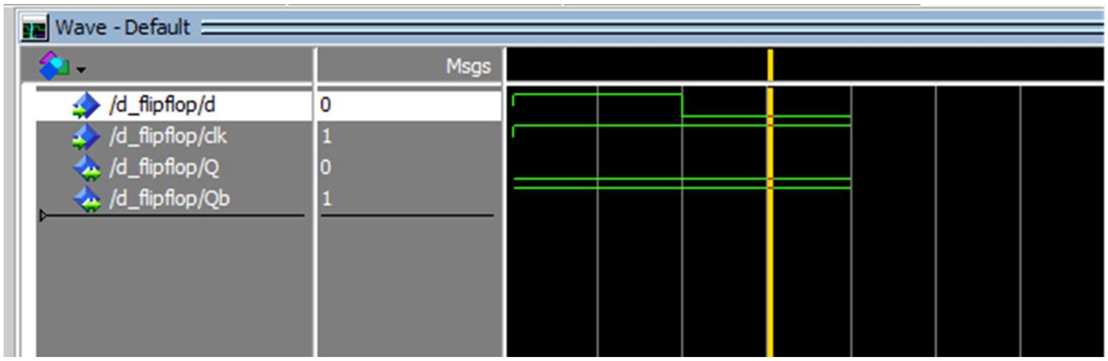

Figure: D flip-flop

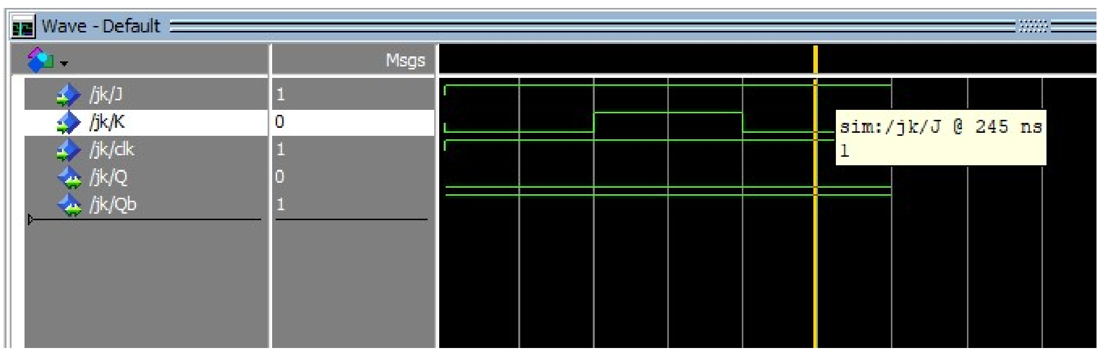

Figure: JK flip-flop

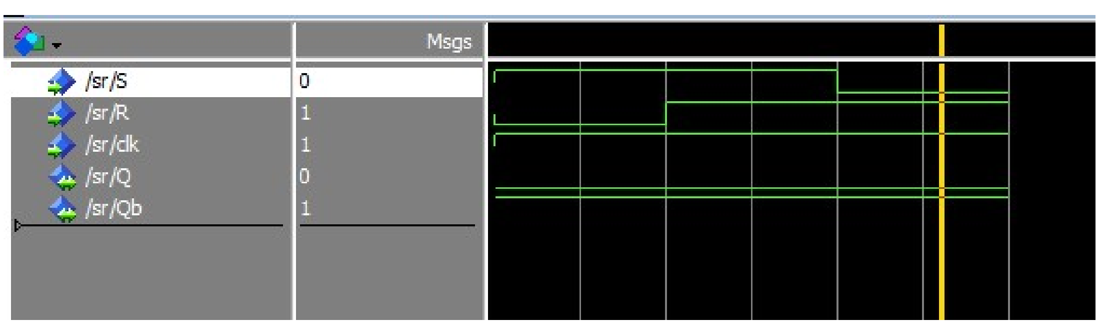

Figure: SR flip-flop

Output of Computer Architecture Lab

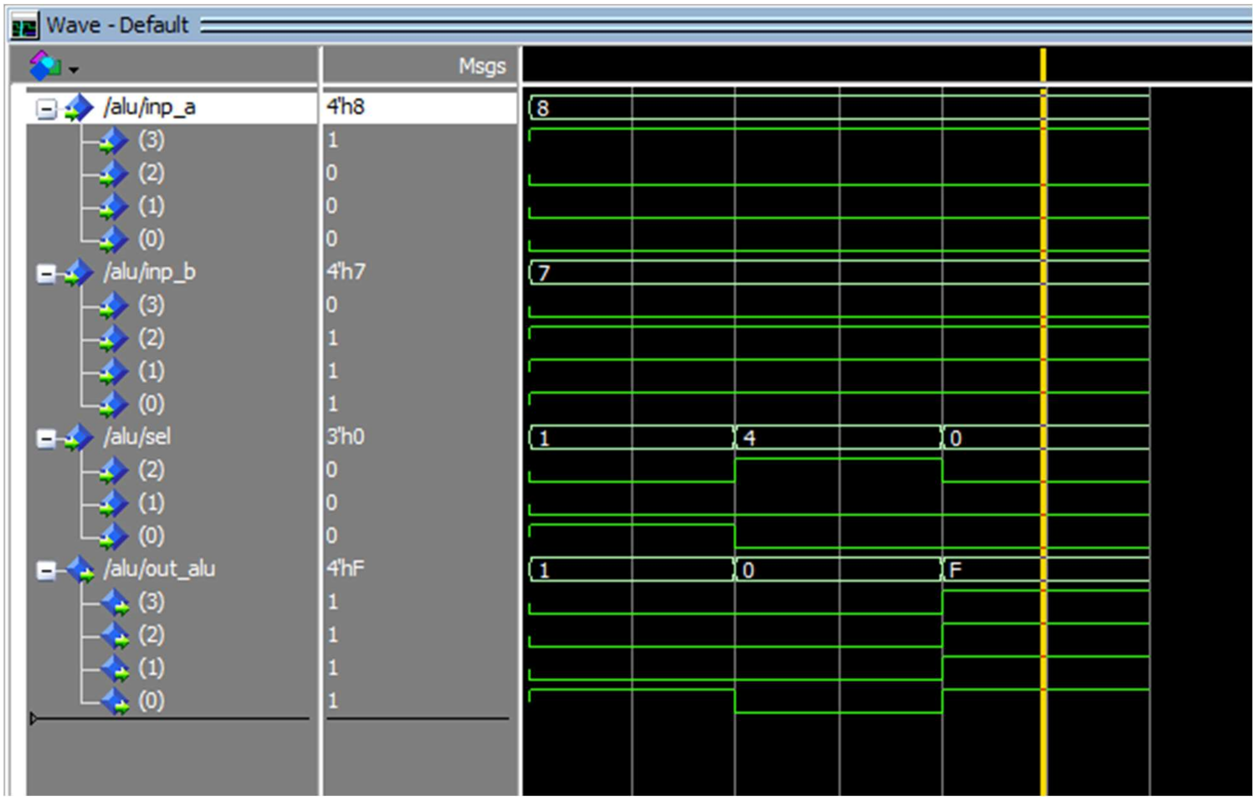

Figure: ALU

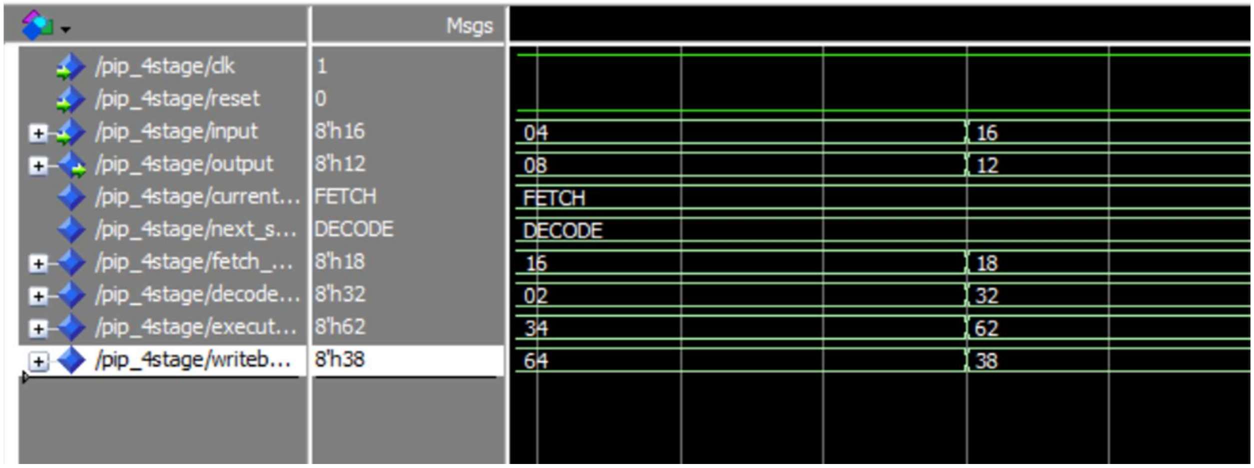

Figure: 4-stage piplining

```
Program\Bsc.CSIT 3rd sem\Computer Architecture\" ; if ($?)
Booth_2_complement } ; if ($?) { .\Booth_2_complement }


Enter the first operand (x): 10
Enter the second operand (y): -5
Result of signed addition: 5
Result of signed subtraction: 15
Result of Booth addition: 5
Result of Booth subtraction: 15
```

Figure: output of Booth addition and subtraction of signed 2’s complement data

Output of Computer Architecture Lab

```
PS D:\My Programs\C Program\Bsc.CSIT 3rd sem\Computer Architecture>
cd "d:\My Programs\C Program\Bsc.CSIT 3rd sem\Computer Architecture\
" ; if ($?) { gcc RestoringDivision.c -o RestoringDivision } ; if ($
?) { .\RestoringDivision }
RESTORING DIVISION

Enter two numbers to divide
Both numbers should be less than 16

Enter the dividend: 15
Enter the divisor: 2

Expected Quotient: 7
Expected Remainder: 1


Unsigned Binary Equivalents are:
A: 01111
B: 00010
B'+1: 11110

-->
SHIFT LEFT: 00000 : 11110
-->
SUB B: 11110 : 11110
--> RESTORE
ADD B: 00000 : 11110
SHIFT LEFT: 00001 : 11100
-->
SUB B: 11111 : 11100
--> RESTORE
ADD B: 00001 : 11100
SHIFT LEFT: 00011 : 11000
-->
SUB B: 00001 : 11000
SHIFT LEFT: 00011 : 10010
-->
SUB B: 00001 : 10010
SHIFT LEFT: 00011 : 00110
-->
SUB B: 00001 : 00110
----------------------------
Sign of result: 0
Remainder: 00001
Quotient: 00111
```

Figure: output of Boot restoring division algorithm

Output of Computer Architecture Lab

```
PS D:\My Programs\C Program\Bsc.CSIT 3rd sem\Computer Architecture>
C Program\Bsc.CSIT 3rd sem\Computer Architecture\" ; if ($?) { gcc
.c -o Booth_multiplication } ; if ($?) { .\Booth_multiplication }
qn      q[n+1]          BR              AC      QR              sc
                        initial         0000    0101            4
1       0               A = A - BR      0111    0101
                        rightShift      0011    1010            3
0       1               A = A + BR      1001    1010
                        rightShift      1100    1101            2
1       0               A = A - BR      0011    1101
                        rightShift      0001    1110            1
0       1               A = A + BR      0111    1110
                        rightShift      0011    1111            0

Result = 1111
```

Figure: Output of Booth multiplication algorithm