Lab 1 Report
Chenjie Luo

1. **Schematics of full adder**

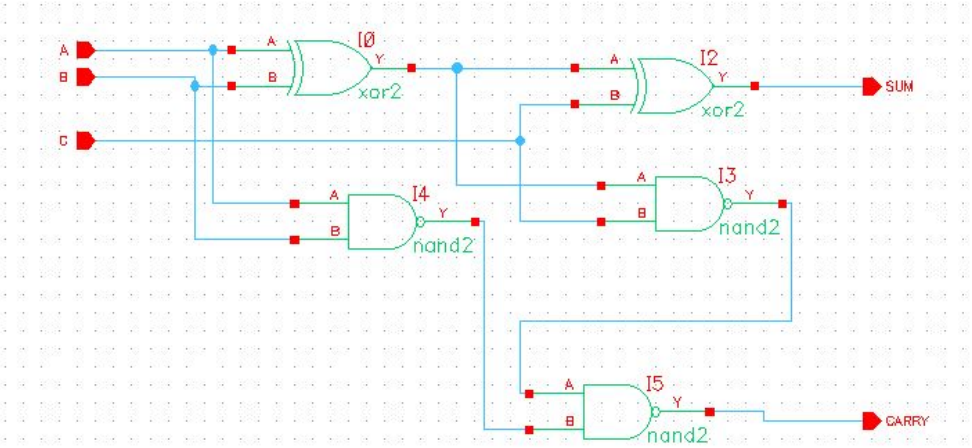

Fig.1 Full adder

A and B represent two 1 bit input while SUM represents the 1 bit output with CARRY bit. C is the carry bit from input.

```
xterm
------------------------------------
Relinquished control to SimVision...
ncsim>
ncsim> source /softwares/Linux/cadence/INCISIV102/tools/inca/files/ncsimrc
ncsim> database -open shmWave -shm -default -into shm.db
Created default SHM database shmWave
ncsim> probe -create -shm test  -all   -depth 1
Created probe 1
ncsim> run
                   0 A=0, B=0, C=0, SUM=0, CARRY=0
                  50 A=0, B=0, C=1, SUM=1, CARRY=0
                 100 A=0, B=1, C=0, SUM=1, CARRY=0
                 150 A=0, B=1, C=1, SUM=0, CARRY=1
                 200 A=1, B=0, C=0, SUM=1, CARRY=0
                 250 A=1, B=0, C=1, SUM=0, CARRY=1
                 300 A=1, B=1, C=0, SUM=0, CARRY=1
                 350 A=1, B=1, C=1, SUM=1, CARRY=1
ncsim> ^C
ncsim> exit
TOOL:   ncxlmode        10.20-s073: Exiting on Jan 25, 2019 at 11:56:52 CST  (to
tal: 00:18:34)

[lchenjie]@hera3 ~/lchenjie/cadence/fulladder_run1> (10:49:54 01/28/19)
::
```

Fig.2 Full adder simout file

Fig.3 Full adder Symbol

Verilog code for full adder:

```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.

initial
$monitor($time, " A=%b, B=%b, C=%b, SUM=%b, CARRY=%b",A, B, C, SUM, CARRY);

initial
begin

  A = 1'b0;
  B = 1'b0;
  C = 1'b0;

#50 A=1'b0; B=1'b0; C=1'b1;
#50 A=1'b0; B=1'b1; C=1'b0;
#50 A=1'b0; B=1'b1; C=1'b1;
#50 A=1'b1; B=1'b0; C=1'b0;
#50 A=1'b1; B=1'b0; C=1'b1;
#50 A=1'b1; B=1'b1; C=1'b0;
#50 A=1'b1; B=1'b1; C=1'b1;


end
```

2. Schematics of 4-bit adder

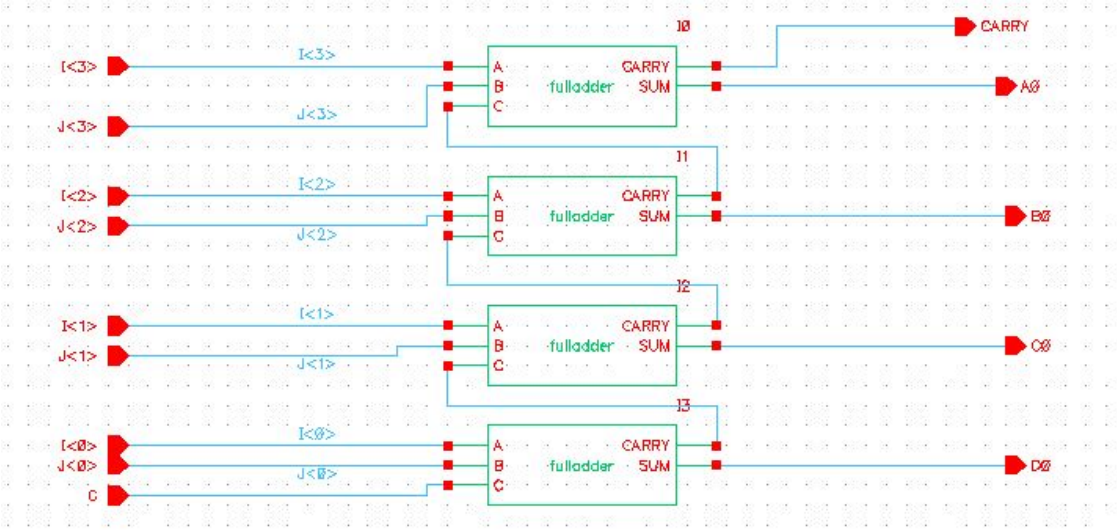

Fig.4 4-bit adder

The I<3:0> represents the first 4 bit number, while J<3:0> represents the second 4 bit number. The output is read as A0B0C0D0 with a carry bit named CARRY. C represents the carry bit from input.

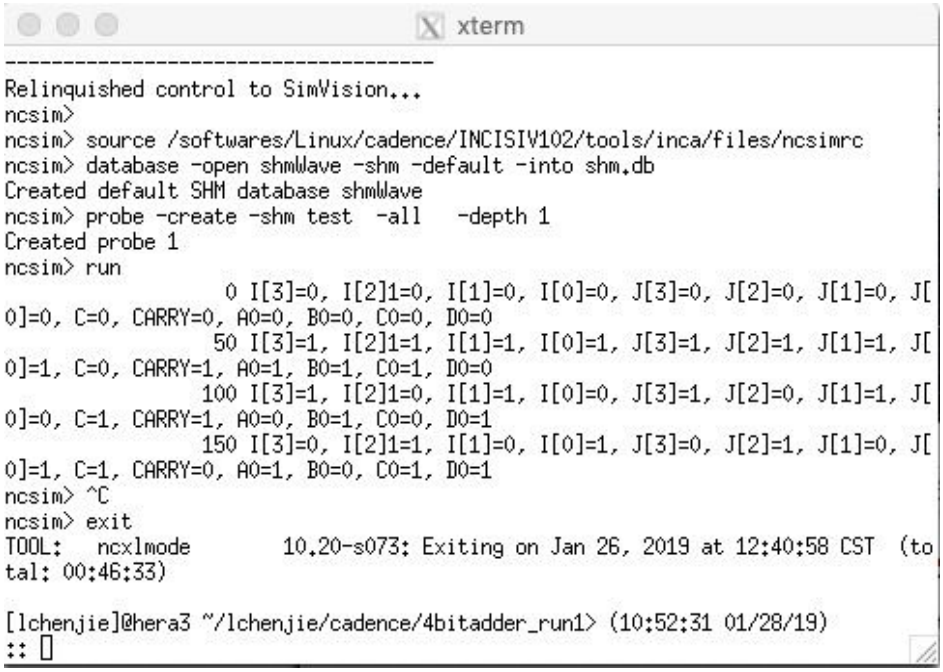

Fig.5 4-bit adder simout file

The output reads as
CARRY = 1, output = 1110
CARRY = 1, output = 0101
CARRY = 0, output = 1011

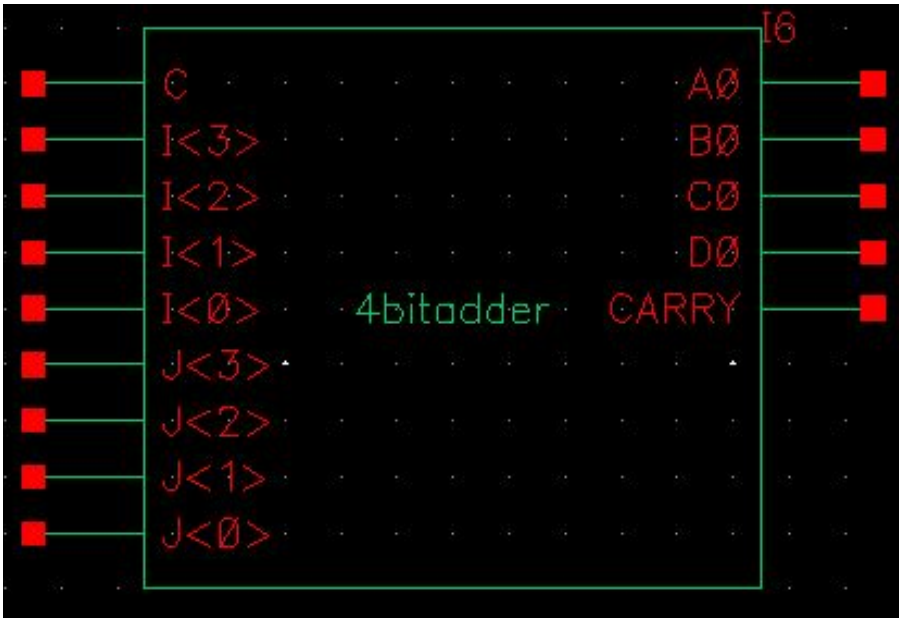

Fig. 6 4 bit adder Symbol

Verilog code for 4-bit adder:

```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.
initial
$monitor($time," I[3]=%b, I[2]1=%b, I[1]=%b, I[0]=%b, J[3]=%b, J[2]=%b, J[1]=%b, J[0]=%b,
C=%b, CARRY=%b, A0=%b, B0=%b, C0=%b, D0=%b", I[3], I[2], I[1], I[0], J[3], J[2], J[1], J[0], C,
CARRY, A0, B0, C0, D0);
initial
begin
I[3]=1'b0; I[2]=1'b0; I[1]=1'b0; I[0]=1'b0; J[3]=1'b0; J[2]=1'b0; J[1]=1'b0; J[0]=1'b0; C=1'b0;
#50 I[3] = 1'b1; I[2] = 1'b1; I[1] = 1'b1; I[0] = 1'b1; C=1'b0; J[3]=1'b1; J[2]=1'b1; J[1]=1'b1; J[0]=
1'b1;  //1111+1111 + CARRY IN 0
#50 I[3] = 1'b1; I[2] = 1'b0; I[1] = 1'b1; I[0] = 1'b0; C=1'b1; J[3]=1'b1; J[2]=1'b0; J[1]=1'b1; J[0]=
1'b0;  //1010+1010 + CARRY IN 1
#50 I[3] = 1'b0; I[2] = 1'b1; I[1] = 1'b0; I[0] = 1'b1; C=1'b1; J[3]=1'b0; J[2]=1'b1; J[1]=1'b0; J[0]=
1'b1;  //0101+0101 + CARRY IN 1

End
```

3. Schematics of 3. 8-bit adder

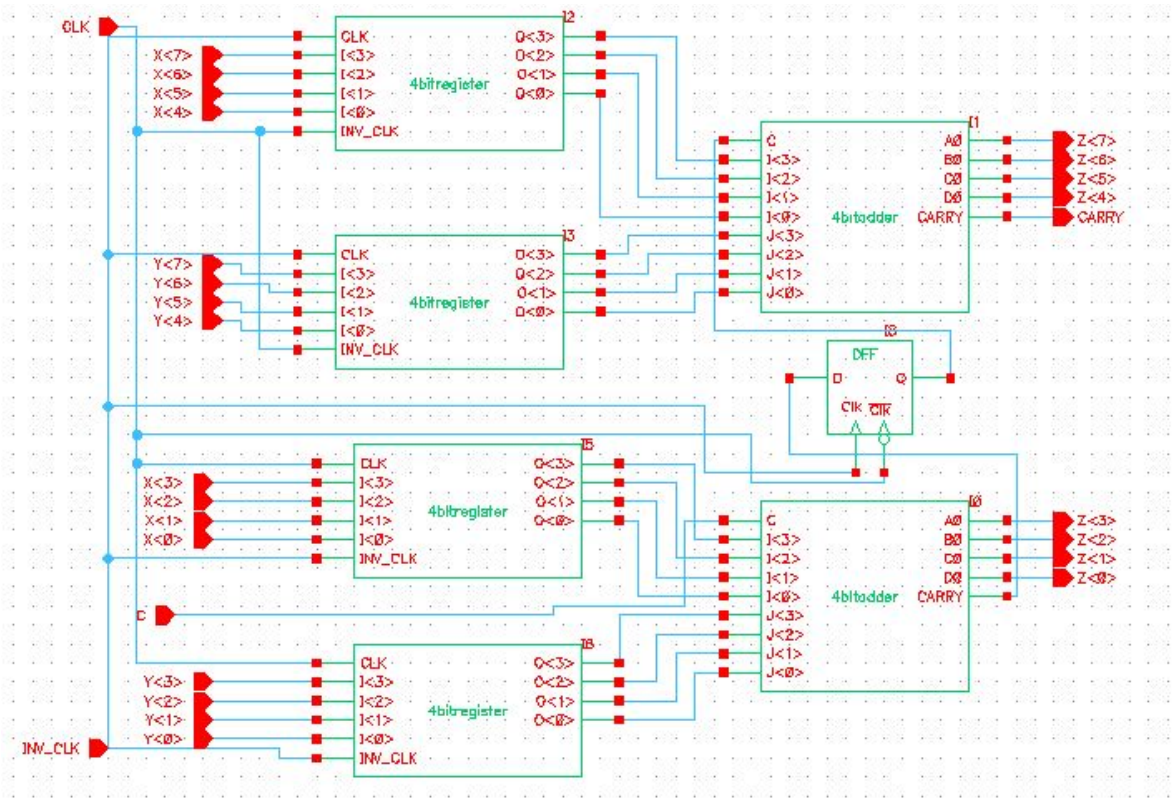

Fig.7 8-bit adder design

The X<7:0> represents the first 8 bit number, while Y<7:0> represents the second 8 bit number. The output is Z<7:0> with a carry bit named CARRY. C represents the carry bit from input.

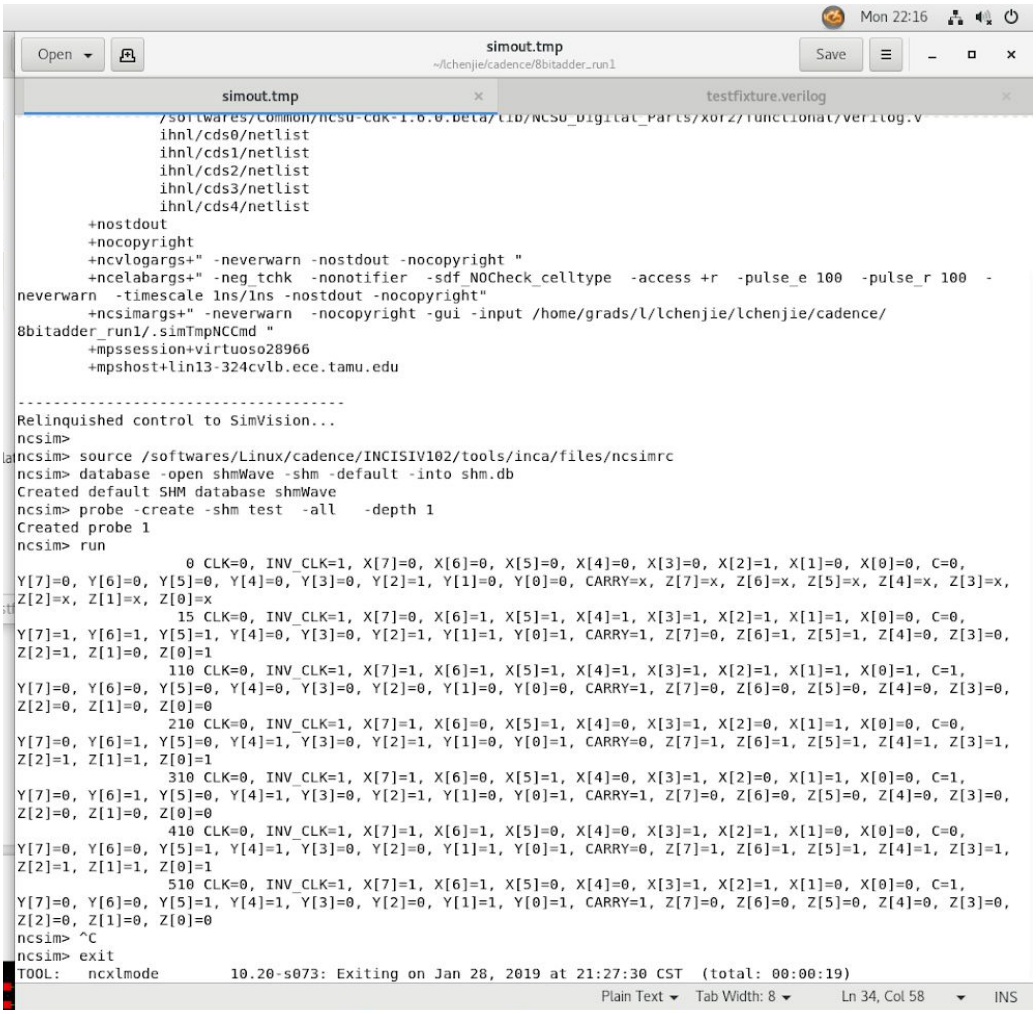

Fig.8 8-bit adder simout file

The output is as follows:

CARRY = 1, output = 01100101

CARRY = 1, output = 00000000

CARRY = 0, output = 11111111

CARRY = 1, output = 00000000

CARRY = 0, output = 11111111

CARRY = 1, output = 00000000

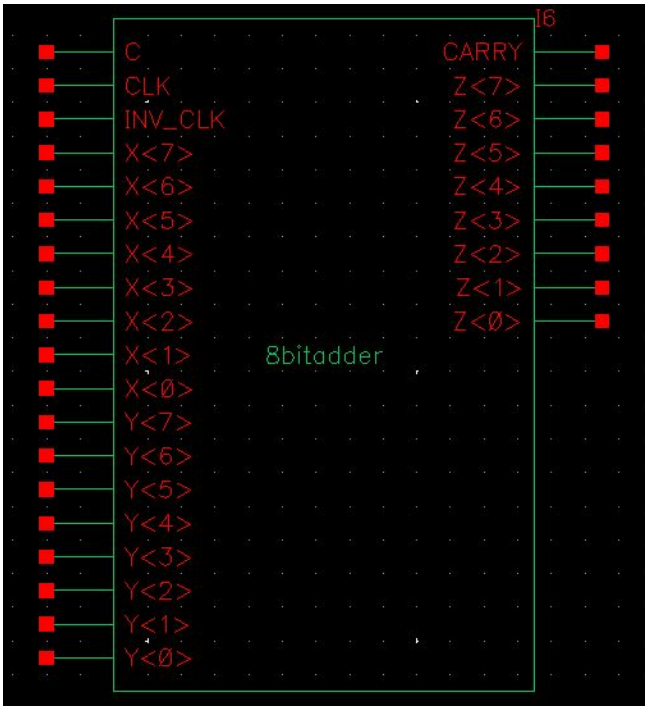

Fig. 9 8 bit adder Symbol

Verilog code for 8-bit adder:

```
// Verilog stimulus file.
// Please do not create a module in this file.


// Default verilog stimulus.

always @(negedge CLK)
begin
$display($time, " CLK=%b, INV_CLK=%b, X[7]=%b, X[6]=%b, X[5]=%b, X[4]=%b, X[3]=%b,
X[2]=%b, X[1]=%b, X[0]=%b, C=%b, Y[7]=%b, Y[6]=%b, Y[5]=%b, Y[4]=%b, Y[3]=%b, Y[2]=%b,
Y[1]=%b, Y[0]=%b, CARRY=%b, Z[7]=%b, Z[6]=%b, Z[5]=%b, Z[4]=%b, Z[3]=%b, Z[2]=%b,
Z[1]=%b, Z[0]=%b", CLK, INV_CLK, X[7], X[6], X[5], X[4], X[3], X[2], X[1], X[0], C, Y[7], Y[6],
Y[5], Y[4], Y[3], Y[2], Y[1], Y[0], CARRY, Z[7], Z[6], Z[5], Z[4], Z[3], Z[2], Z[1], Z[0]);
end

initial
begin

  C = 1'b0;
  CLK = 1'b0;
  INV_CLK = 1'b1;
  X[0] = 1'b0;
  X[1] = 1'b0;
  X[2] = 1'b1;
  X[3] = 1'b0;
  X[4] = 1'b0;
  X[5] = 1'b0;
  X[6] = 1'b0;
  X[7] = 1'b0;
  Y[0] = 1'b0;
  Y[1] = 1'b0;
  Y[2] = 1'b1;
  Y[3] = 1'b0;
  Y[4] = 1'b0;
  Y[5] = 1'b0;
  Y[6] = 1'b0;
  Y[7] = 1'b0;

#5 X[7]=1'b0; X[6]=1'b1; X[5]=1'b1; X[4]=1'b1; X[3]=1'b1; X[2]=1'b1; X[1]=1'b1; X[0]=1'b0;
Y[7]=1'b1; Y[6]=1'b1; Y[5]=1'b1; Y[4]=1'b0; Y[3]=1'b0; Y[2]=1'b1; Y[1]=1'b1; Y[0]=1'b1;C=0;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
```

```
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal


#85 X[7]=1'b1; X[6]=1'b1; X[5]=1'b1; X[4]=1'b1; X[3]=1'b1; X[2]=1'b1; X[1]=1'b1; X[0]=1'b1;
Y[7]=1'b0; Y[6]=1'b0; Y[5]=1'b0; Y[4]=1'b0; Y[3]=1'b0; Y[2]=1'b0; Y[1]=1'b0; Y[0]=1'b0;C=1;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal


#90 X[7]=1'b1; X[6]=1'b0; X[5]=1'b1; X[4]=1'b0; X[3]=1'b1; X[2]=1'b0; X[1]=1'b1; X[0]=1'b0;
Y[7]=1'b0; Y[6]=1'b1; Y[5]=1'b0; Y[4]=1'b1; Y[3]=1'b0; Y[2]=1'b1; Y[1]=1'b0; Y[0]=1'b1;C=0;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal


#90 X[7]=1'b1; X[6]=1'b0; X[5]=1'b1; X[4]=1'b0; X[3]=1'b1; X[2]=1'b0; X[1]=1'b1; X[0]=1'b0;
Y[7]=1'b0; Y[6]=1'b1; Y[5]=1'b0; Y[4]=1'b1; Y[3]=1'b0; Y[2]=1'b1; Y[1]=1'b0; Y[0]=1'b1;C=1;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal


#90 X[7]=1'b1; X[6]=1'b1; X[5]=1'b0; X[4]=1'b0; X[3]=1'b1; X[2]=1'b1; X[1]=1'b0; X[0]=1'b0;
Y[7]=1'b0; Y[6]=1'b0; Y[5]=1'b1; Y[4]=1'b1; Y[3]=1'b0; Y[2]=1'b0; Y[1]=1'b1; Y[0]=1'b1;C=0;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal


#90 X[7]=1'b1; X[6]=1'b1; X[5]=1'b0; X[4]=1'b0; X[3]=1'b1; X[2]=1'b1; X[1]=1'b0; X[0]=1'b0;
Y[7]=1'b0; Y[6]=1'b0; Y[5]=1'b1; Y[4]=1'b1; Y[3]=1'b0; Y[2]=1'b0; Y[1]=1'b1; Y[0]=1'b1;C=1;
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //positive edge for CLK signal
#5 CLK=~CLK; INV_CLK=~INV_CLK;  //negative edge for CLK signal

end
```