EE 671: VLSI Design

Assignment 2

Roll No. 153079029

Q–1 Using ngspice based circuit simulation, estimate the delay of a half adder and a full adder, when loaded with 4 inverters. Also evaluate the delay of a mux loaded with 4 inverters to be used in a carry select adder. (The full adder should not optimize carry generation at the cost of delaying the sum output). Use the model files provided with the previous assignment.

Ans: 1> To estimate the delay of half adder

1> Carry delay

- **Netlist:**

Half Adder Carry Delay

.include model.txt

.subckt inverter in out gnd vdd

Mn1 out in gnd gnd CMOSN L=0.4um W=0.6um ad=0.48p as=0.48p pd=2.8u ps=2.8u

Mp2 out in vdd vdd CMOSP L=0.4um W=1.5483000um ad=1.23864p as=1.23864p pd=4.6966u ps=4.6966u

.ends inverter

Mn3 1 g3 0 0 CMOSN W=1.2um L=.4um

Mn4 2 g4 1 1 CMOSN W= 1.2um L=.4um

Mp5 2 g3 8 8 CMOSP W=3.09666um L=.4um

Mp6 2 g4 8 8 CMOSP W=3.09666um L=.4um

X1 2 3 gnd 8 inverter

X2 3 4 gnd 8 inverter m=4

Vdd 8 0 3.3v

v1 g3 0 3.3v

v2 g4 0 PULSE(0 3.3 0.1n 0.1n 0.1n 10n 20n)

Cl 4 0 0.1pF

.MEAS tran delay_carry TRIG v(g4) VAL=1.65 RISE=1 TARG v(3) VAL=1.65 RISE=1

.tran 0 60n

.control

run

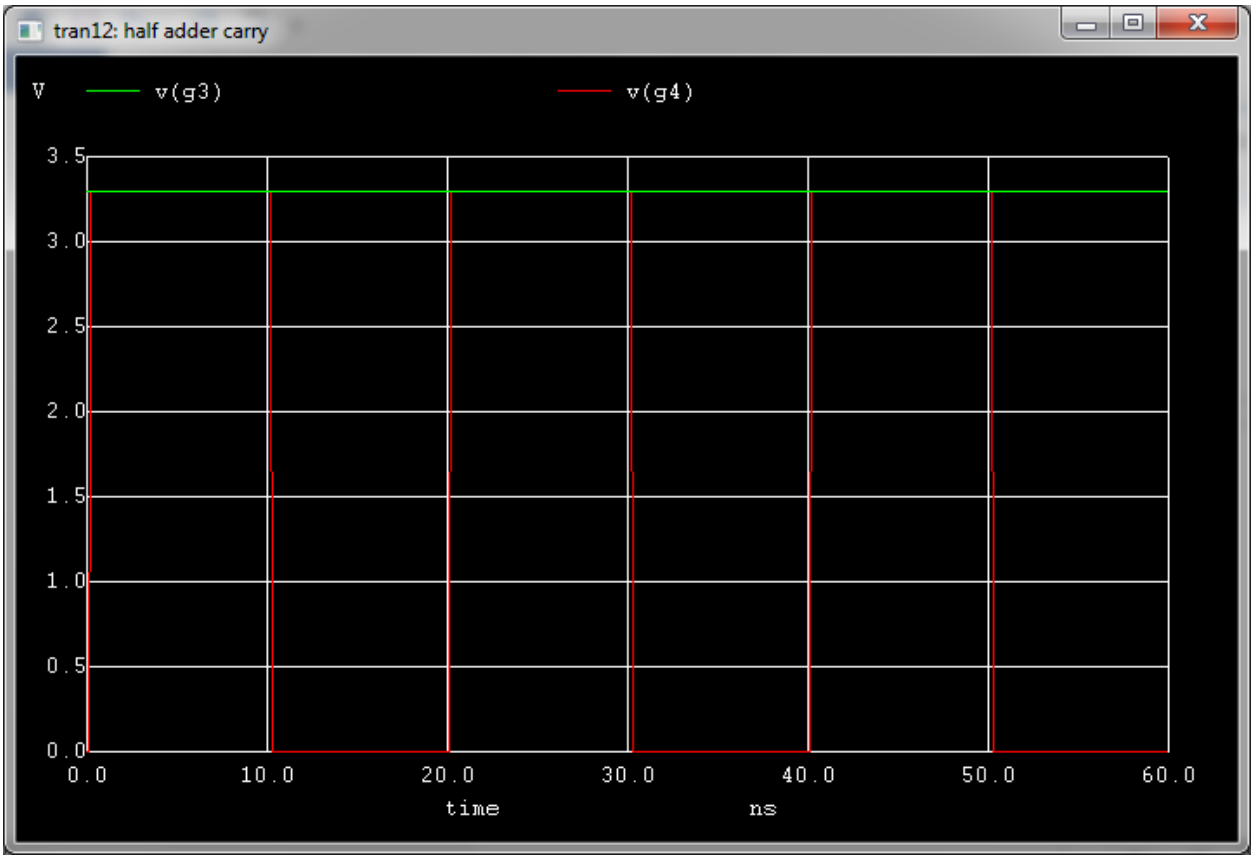
plot v(g3) v(g4)

plot v(3)

.endc

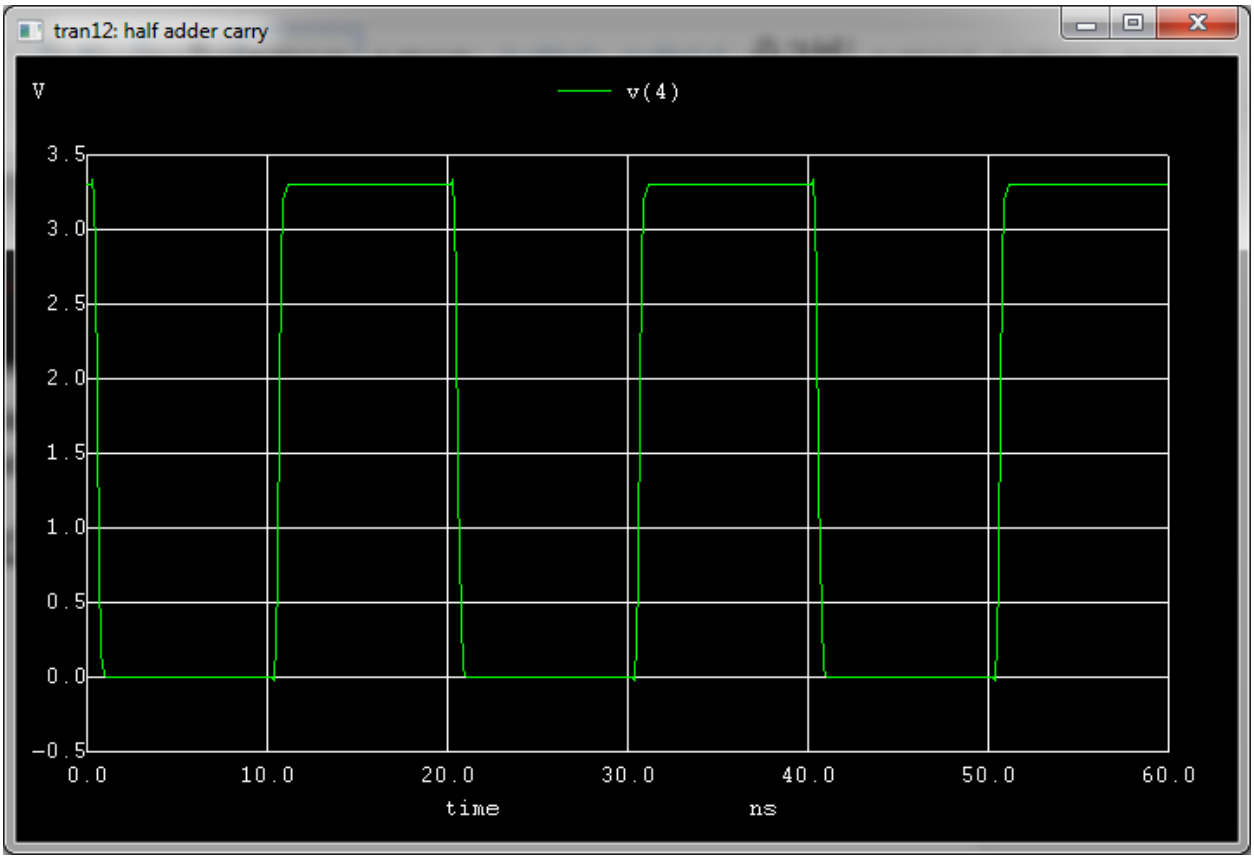
.end

- **Input Waveform**

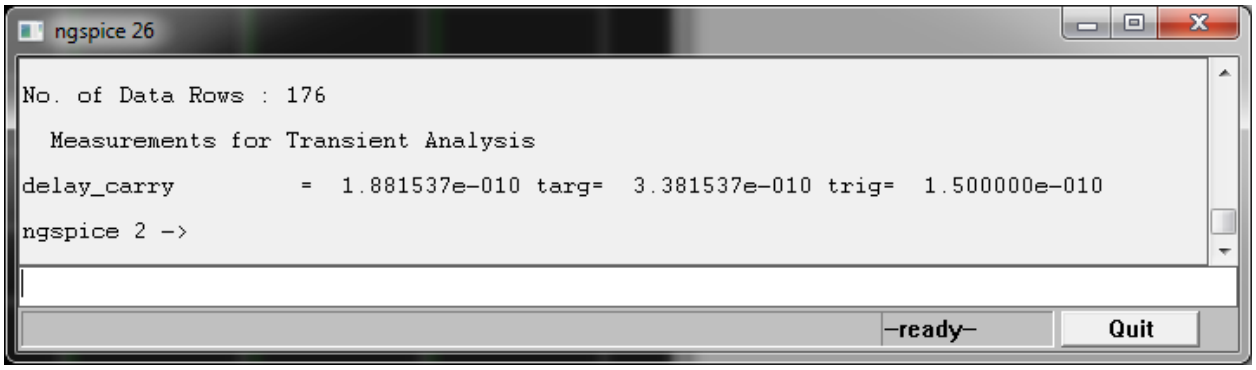


- **First input in green is kept at vdd.**
- **Second input in red is a pulse waveform.**

- **Output waveform**



- **Delay Results**



The delay of carry is = **0.1881537 nsec**.

2> Sum delay

Netlist:

Half Adder Sum Delay
.include model.txt

.subckt inverter in out gnd vdd
M1 out in gnd gnd CMOSN L=0.4um W=0.6um ad=0.48p as=0.48p pd=2.8u ps=2.8u
M2 out in vdd vdd CMOSP L=0.4um W=1.5483000um ad=1.23864p as=1.23864p pd=4.6966u ps=4.6966u
.ends inverter

Mn3 b g3 0 0 CMOSN W=1.2um L=.4um
Mn4 s g4 b b CMOSN W= 1.2um L=.4um
Mn5 c g5 0 0 CMOSN W=1.2um L=.4um
Mn6 s g6 c c CMOSN W= 1.2um L=.4um

Mp7 s g6 a a CMOSP W=3.09666um L=.4um
Mp8 s g5 a a CMOSP W=3.09666um L=.4um
Mp9 a g4 8 8 CMOSP W=3.09666um L=.4um
Mp10 a g3 8 8 CMOSP W=3.09666um L=.4um

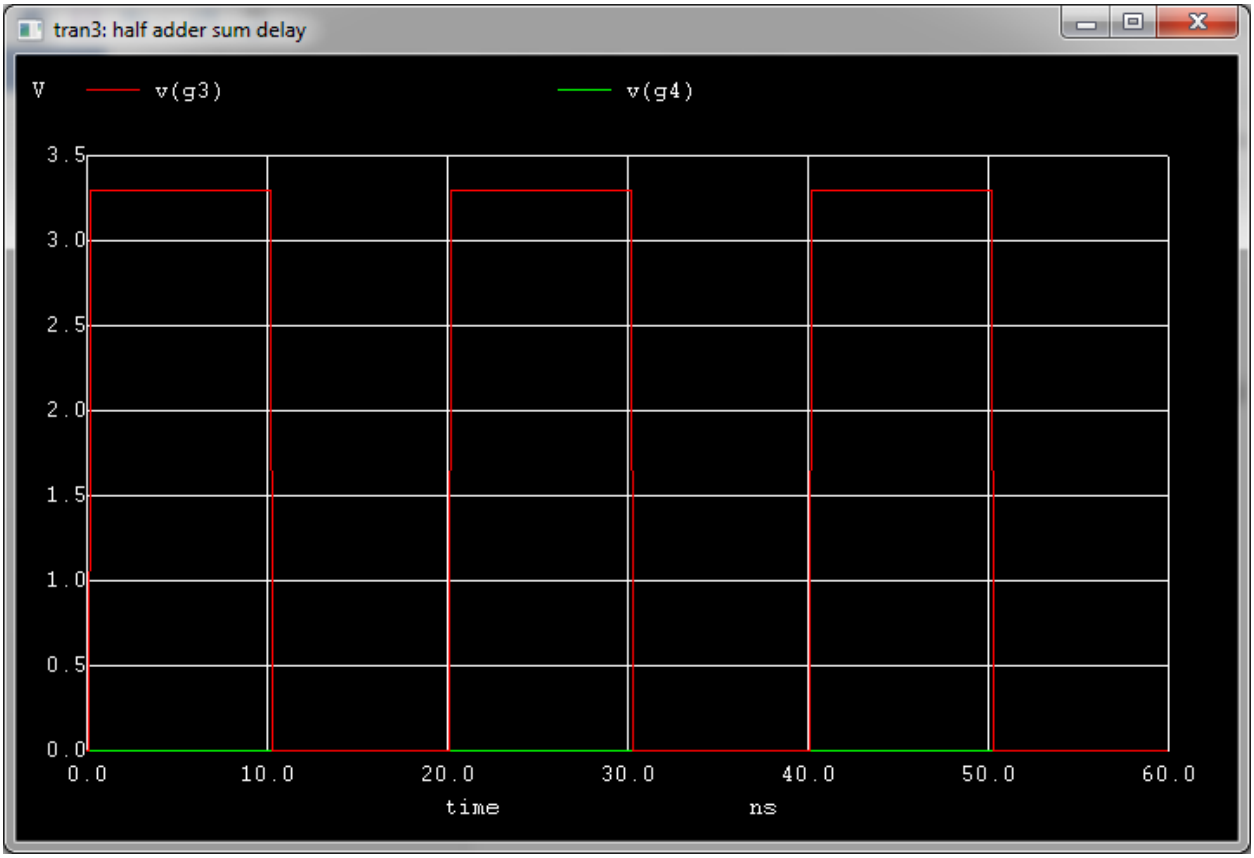
Vdd 8 0 3.3v

X1 g4 g6 0 8 inverter
X2 g3 g5 0 8 inverter
X3 s d 0 8 inverter m=4
Cl d 0 0.5pF

v1 g4 0 0V
v2 g3 0 PULSE(0 3.3 0.1n 0.1n 0.1n 10n 20n)
.MEAS tran sum_delay TRIG v(g3) VAL=1.65 RISE=1 TARG v(s) VAL=1.65 RISE=1

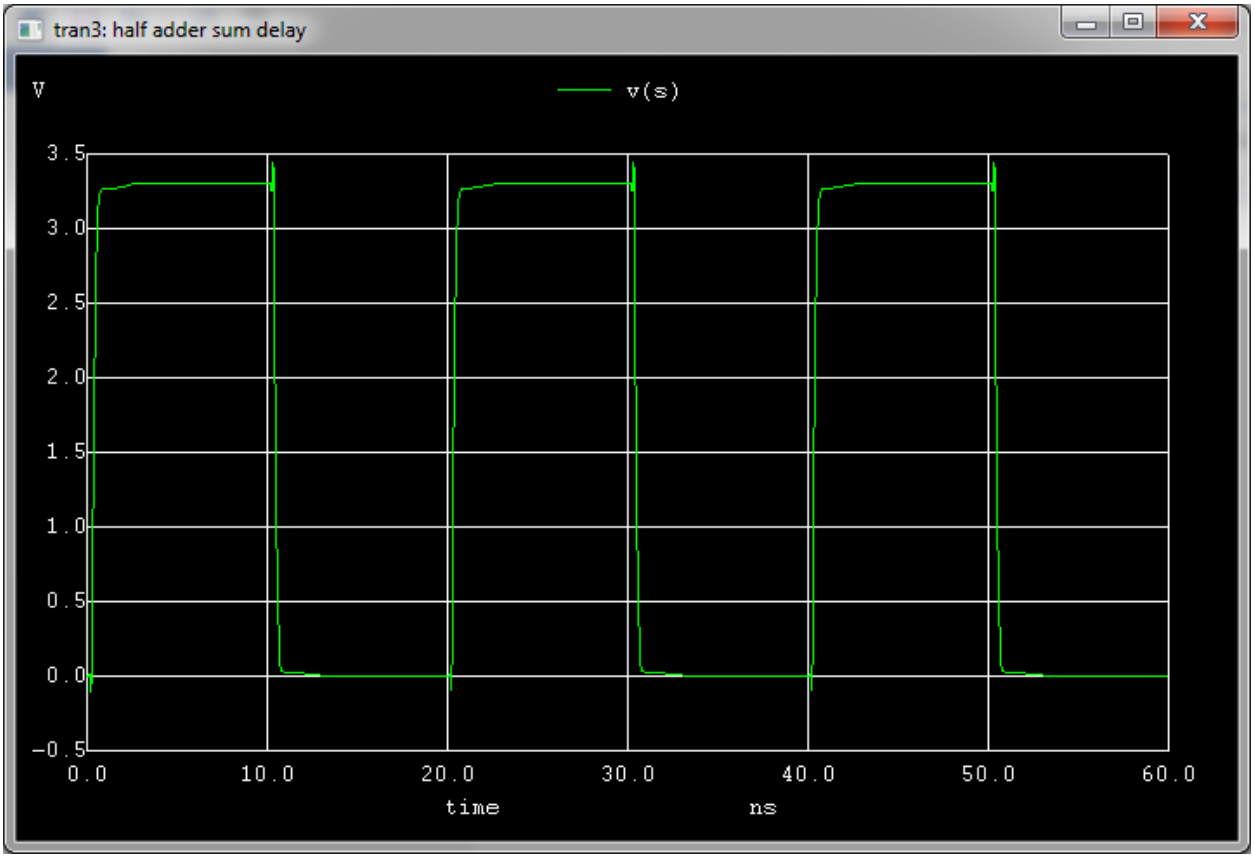
.tran 0 60n
.control
run
plot v(g4) v(g3)
plot v(s)
.endc
.end

- **Input Waveform**

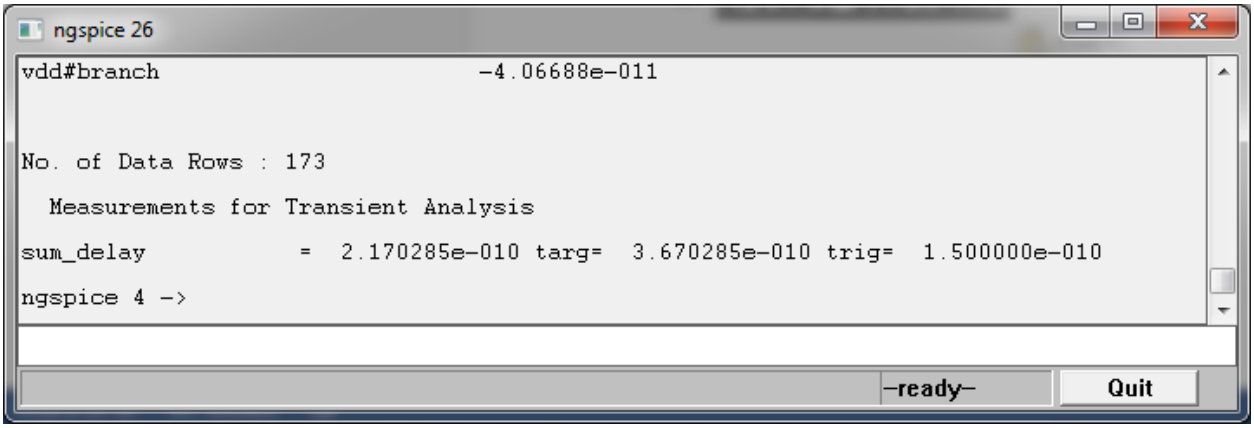


- **First input in green is kept at 0v.**
- **Second input in red is a pulse waveform.**

➢ Output waveform



- **Delay Results**



The delay of Sum is = **0.2170285 nsec**.

2> To estimate the delay of Full adder

Netlist:

```
Full Adder
.include model.txt

.subckt Inverter in out 0 vs
Mi1 out in 0 0 CMOSN L=0.4um W=0.6um ad=0.48p as=0.48p pd=2.8u ps=2.8u
Mi2 out in vs vs CMOSP L=0.4um W=1.5483000um ad=1.23864p as=1.23864p pd=4.6966u ps=4.6966u
.ends Inverter
********** full adder carry **********

vsupply vs 0 3.3

M1 1 b 0 0 cmosn l=0.4u w=1.2u
M2 1 a 0 0 cmosn l=0.4u w=1.2u
M3 2 a 0 0 cmosn l=0.4u w=1.2u
M4 3 cin 1 1 cmosn l=0.4u w=1.2u
M5 3 b 2 2 cmosn l=0.4u w=1.2u
M6 3 cin 4 4 cmosp l=0.4u w=3.0966u
M7 3 b 5 5 cmosp l=0.4u w=3.0966u
M8 4 b vs vs cmosp l=0.4u w=3.0966u
M9 4 a vs vs cmosp l=0.4u w=3.0966u
M10 5 a vs vs cmosp l=0.4u w=3.0966u



* node 3 -->>> cout bar
```

xinv 3 cout 0 vs Inverter

x1 cout out1 0 vs Inverter
x2 cout out1 0 vs Inverter
x3 cout out1 0 vs Inverter
x4 cout out1 0 vs Inverter

cout1 out1 0 0.1p

********** *full adder sum* **********

M11 6 a 0 0 cmosn l=0.4u w=1.2u
M12 6 b 0 0 cmosn l=0.4u w=1.2u
M13 6 cin 0 0 cmosn l=0.4u w=1.2u
M14 7 a 0 0 cmosn l=0.4u w=1.8u
M15 9 3 6 6 cmosn l=0.4u w=1.2u
M16 8 b 7 7 cmosn l=0.4u w=1.8u
M17 9 cin 8 8 cmosn l=0.4u w=1.8u
M18 9 3 10 10 cmosp l=0.4u w=3.0966u
M19 9 cin 11 11 cmosp l=0.4u w=4.6449u
M20 11 b 12 12 cmosp l=0.4u w=4.6449u
M21 12 a vs vs cmosp l=0.4u w=4.6449u
M22 10 a vs vs cmosp l=0.4u w=3.0966u
M23 10 b vs vs cmosp l=0.4u w=3.0966u

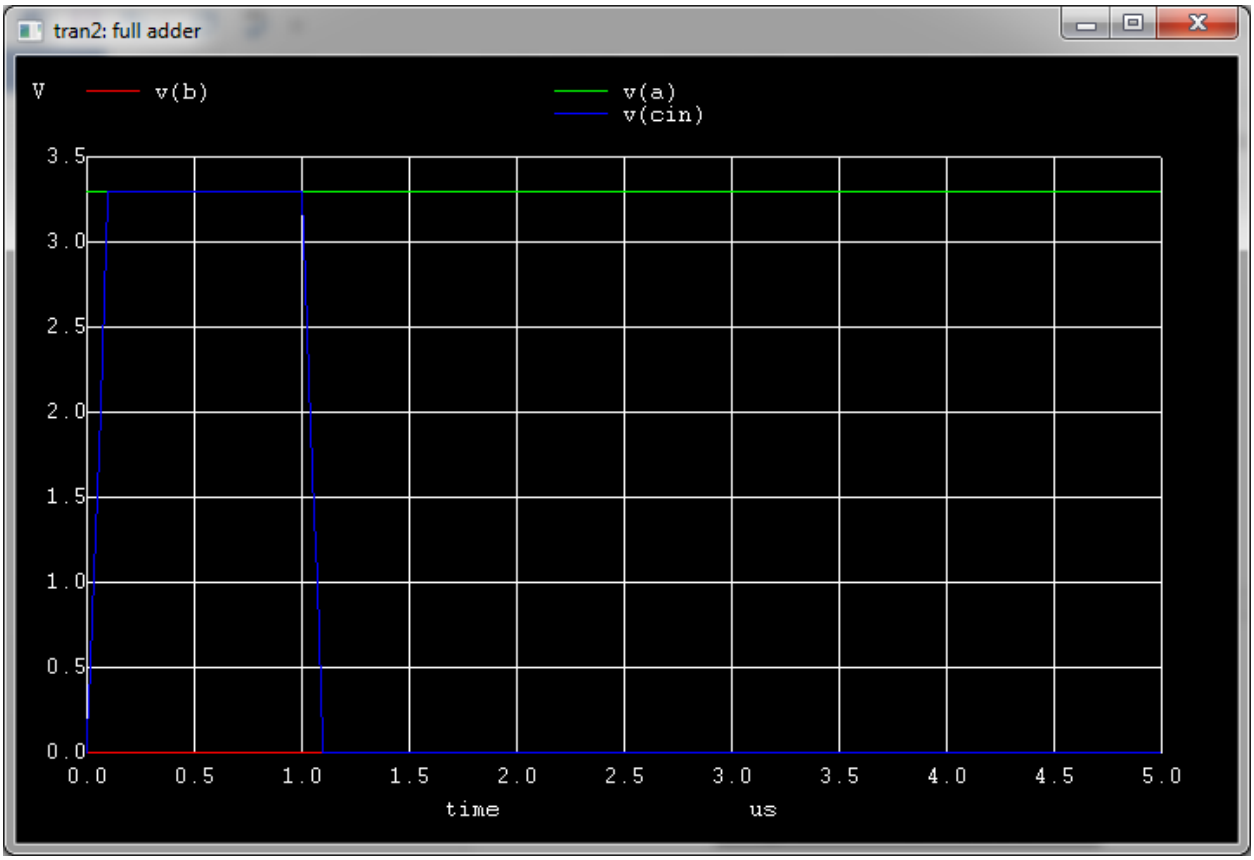
M24 10 cin vs vs cmosp l=0.4u w=3.0966u

xsum 9 sum 0 vs Inverter

x5 sum out2 0 vs Inverter
x6 sum out2 0 vs Inverter

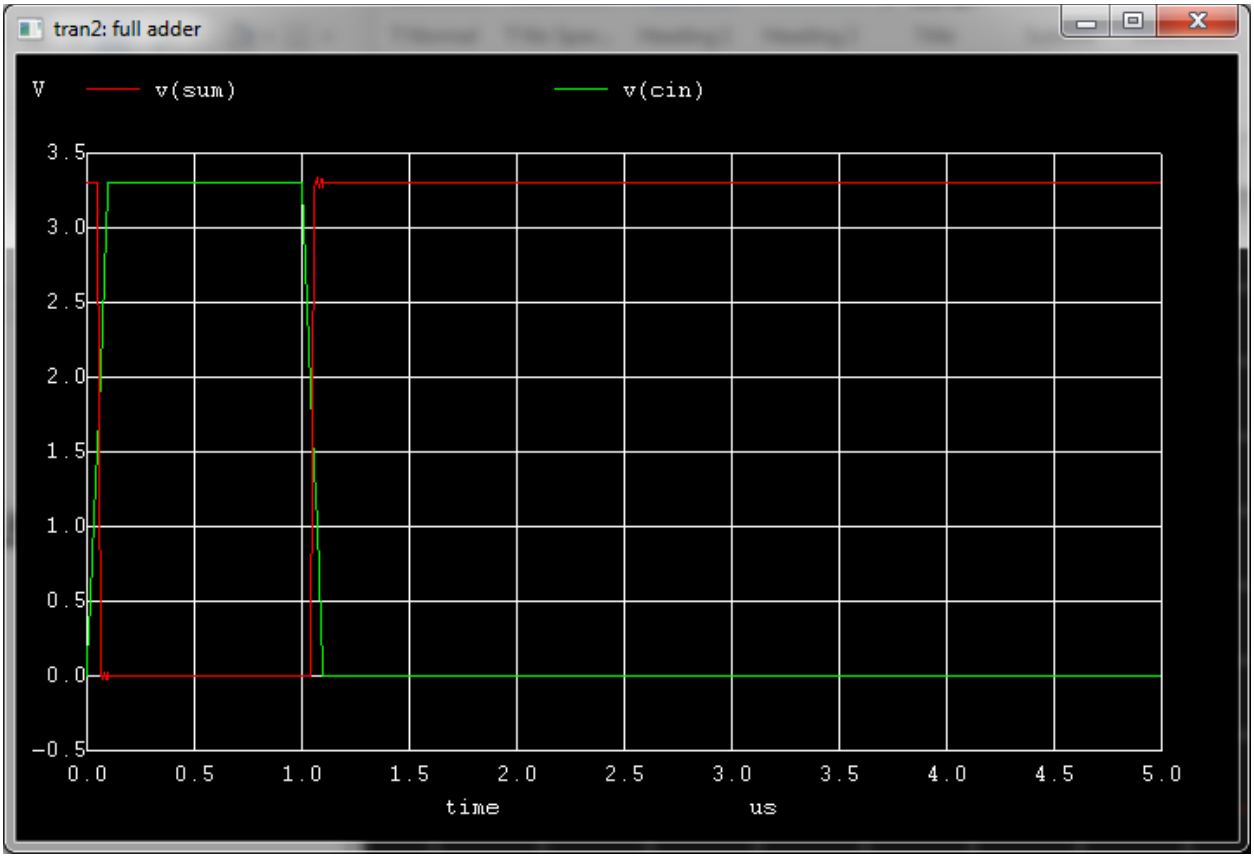
```
x7 sum out2 0 vs Inverter
x8 sum out2 0 vs Inverter

cout2 out2 0 0.1p

*v1 cin 0 dc 0v
v2 a 0 dc 3.3
v3 b 0 0
*v2 a 0 pulse(0 3.3 2n 2n 2n 5n 10n)
*v3 b 0 pulse(3.3 0 2n 2n 2n 5n 10n)
v1 cin 0 pwl(0 0 0.1us 3.3 1us 3.3 1.1us 0)
*v2 a 0 pwl(0 0 0.1ms 3.3 1ms 3.3 1.1ms 0)
*v3 b 0 pwl(0 0 0.1ms 3.3 1ms 3.3 1.1ms 0)
.tran 0 5us
.control
run
MEAS TRAN carry_delay TRIG v(cin) val=1.65 rise=1 TARG v(cout) val=1.65 rise=1
MEAS TRAN sum_delay TRIG v(cin) val=1.65 rise=1 TARG v(sum) val=1.65 fall=1
plot v(a) v(b) v(cin)
plot v(cin) v(cout)
plot v(cin) v(sum)
.endc
.end
```

- **Input Waveforms**

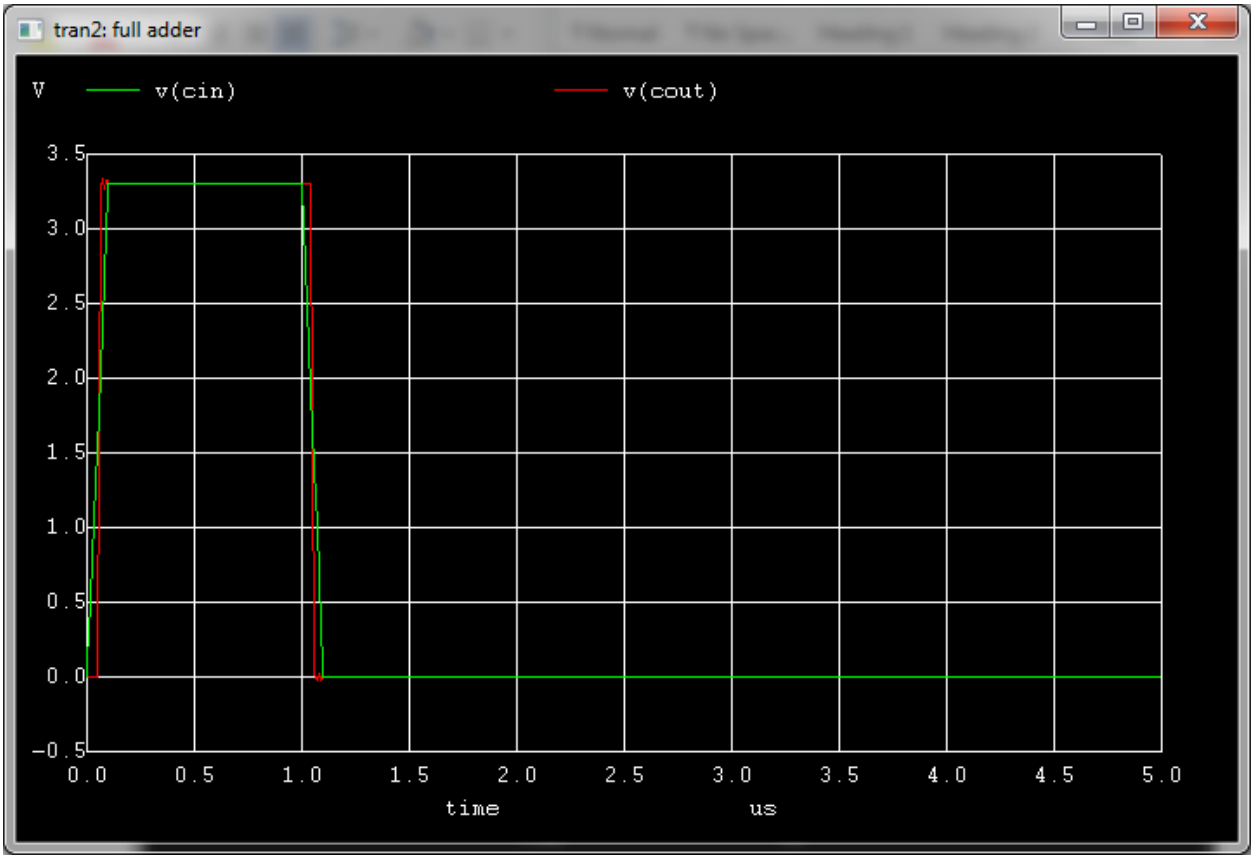


- 1st input is 0v (logic 0)
- 2nd input is 3.3v(logic 1)
- 3rd input is pwl(piecewise linear waveform)

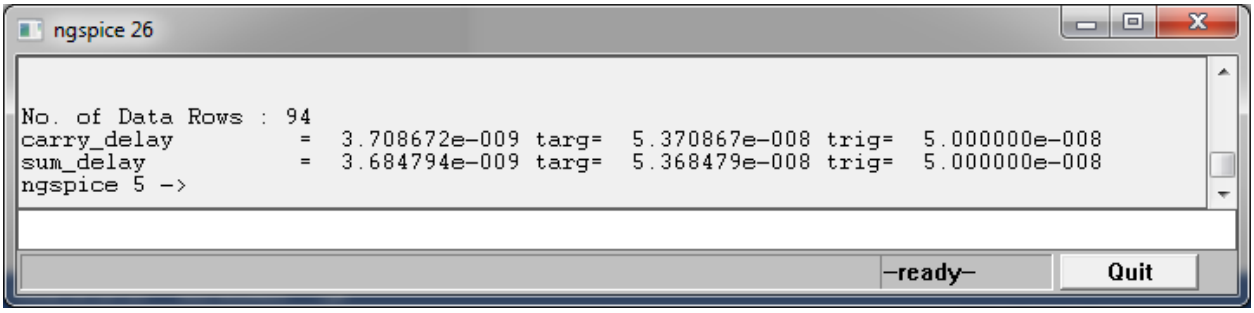
- **Waveform representing sum delay**



- **Waveform representing carry delay**



- **<u>Delay results of ngspice for sum and carry</u>**



- Sum delay shown above is = 3.684794 nsec
- Carry delay shown above is = 3.708672 nsec

3> To estimate the delay of Multiplexer

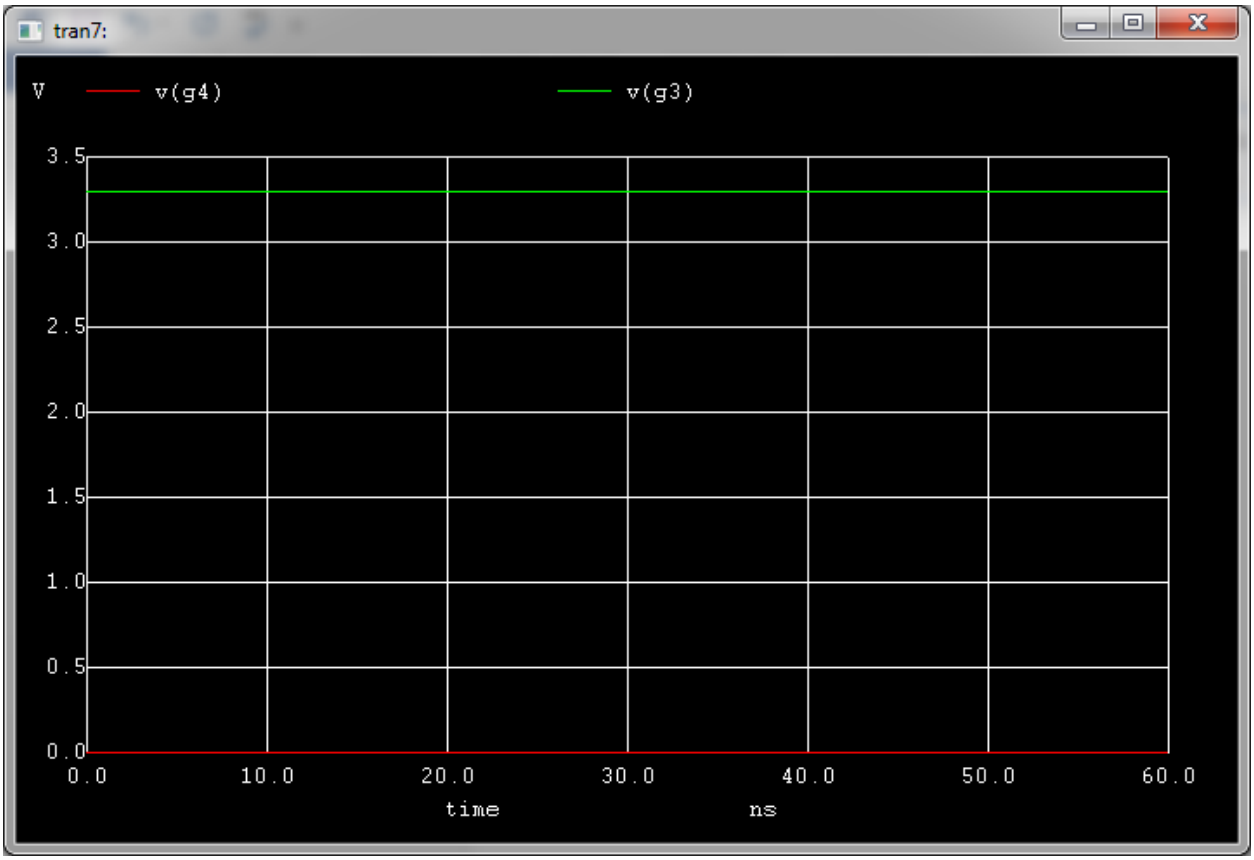
Netlist:

```
.include model.txt

Mn3 c g3 0 0 CMOSN W=1.2um L=.4um
Mn4 d g4 0 0 CMOSN W= 1.2um L=.4um
Mn5 b g5 c c CMOSN W=1.2um L=.4um
Mn6 b g6 d d CMOSN W= 1.2um L=.4um

Mp7 b g6 a a CMOSP W=3.09666um L=.4um
Mp8 b g4 a a CMOSP W=3.09666um  L=.4um
Mp9 a g5 8 8 CMOSP W=3.09666um L=.4um
Mp10 a g3 8 8 CMOSP W=3.09666um  L=.4um

Vdd 8 0 3.3V

.subckt inv in out gnd vdd
M1 out in gnd gnd CMOSN L=0.4um W=0.6um ad=0.48p as=0.48p
pd=2.8u ps=2.8u
M2 out in vdd vdd CMOSP L=0.4um W=1.5483000um
ad=1.23864p as=1.23864p pd=4.6966u ps=4.6966u
.ends inv

X2  g6 g5 0 8 inv
X1  b e 0 8 inv
X3  e out 0 8 inv m=4
Cl out 0 .5pF

Vg3 g3 0 3.3v
Vg4 g4 0 0
```

Vg6 g6 0 PULSE(0 3.3 1n 1n 1n 20n 40n)
.MEAS tran MUX_delay TRIG v(g6) VAL=1.65 Rise=1 TARG v(e) VAL=1.65 fall=1

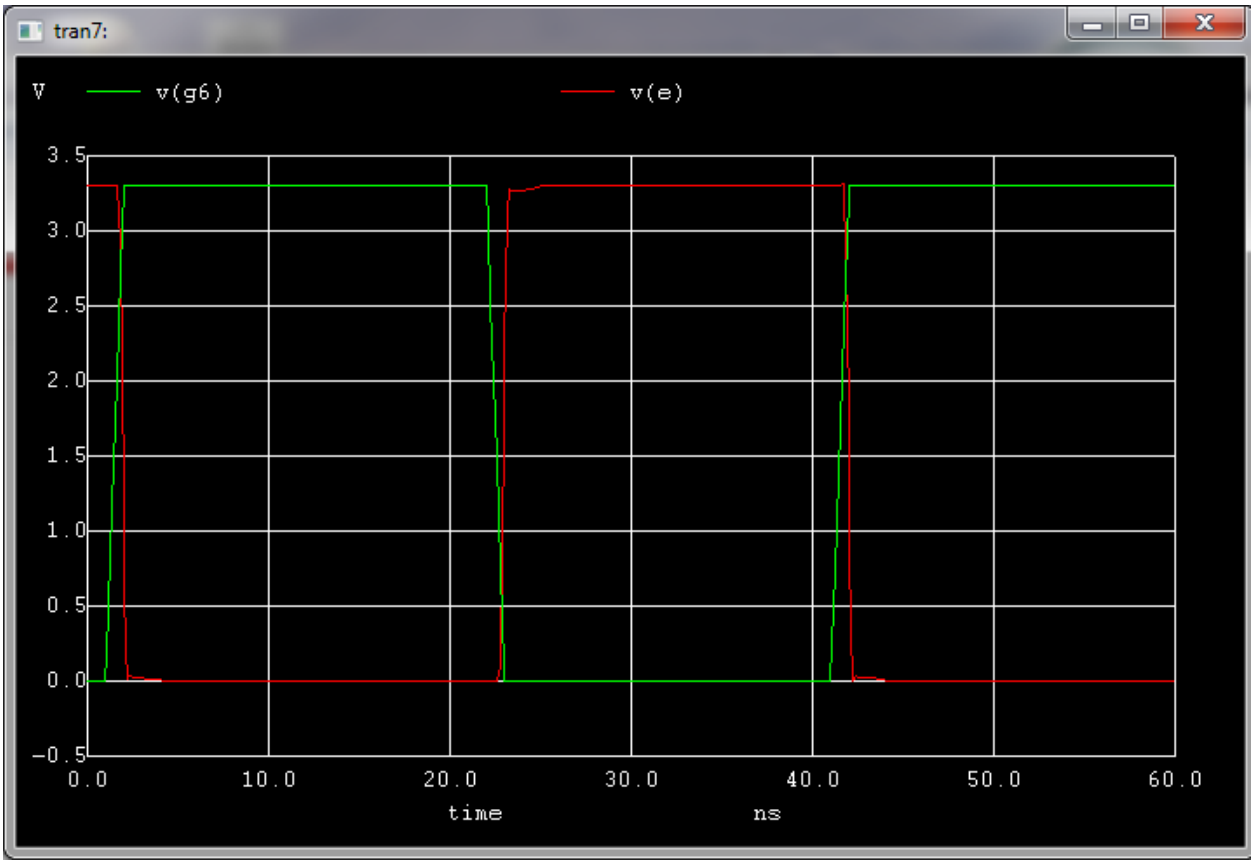
.tran 0 60n
.control
run
plot v(g3) V(g4)
plot V(g6) v(e)
.endc
.end

- **Input Waveforms**

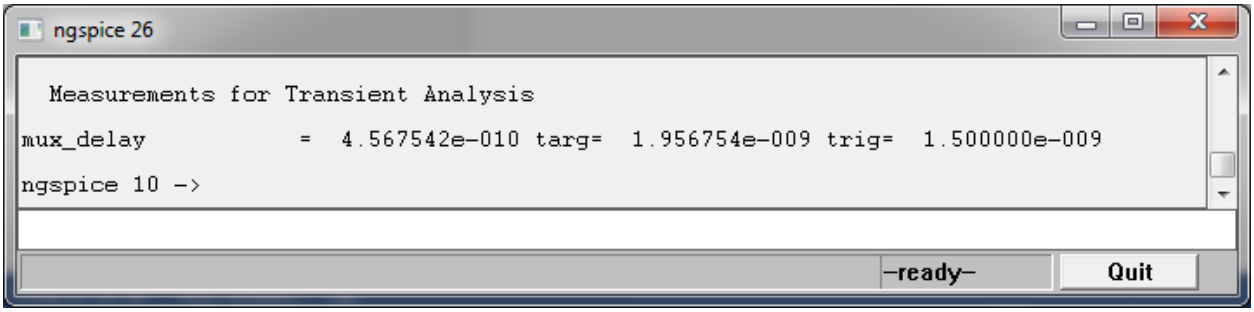


1st MUX input of I_0 = 3.3 volts
2nd MUX input of I_1= 3.3 volts

- **Output waveform of MUX:**



Delay Results of MUX:



The delay of MUX is = 4.567542 nsec

Q–2 Simulate the design of a Wallace and a Dadda multiplier for unsigned multiplication of two 8 bit numbers, generating a 16 bit product. Use a carry select architecture for the final adder with square root stacking. You can use VHDL or verilog for simulating the circuit. Your description should be synthesizable.

Ans: **Verilog Program without Delay**

Wallace:

```
// Half Adder


module HA
  (A,B,S,Cout);
  input A,B;
  output S,Cout;
  assign S=A ^ B ;
  assign Cout=A & B;
endmodule




// Full Adder
```

```
module FA
  (A,B,Cin,S,Cout);
  input A,B,Cin ;
  output S,Cout;
  assign S=A ^ B ^ Cin ;
  assign Cout=(A & B) | (B & Cin) | (A & Cin);
endmodule

`timescale 1ns / 1ps

// Wallace Multiplier

module wallace(output reg [16:0] product, input [7:0] A, B,input clock );
reg p [7:0][7:0];
wire [61:0] s ,c ;
integer i,j;

always@(A,B)

begin
  for ( i=0; i<=7; i=i+1)
    for ( j = 0; j <= 7; j = j + 1)
```

```
        p[i][j] <= A[j] & B[i];
end



module bit_2_adder(a,b,ci,s,co);
input  [1:0] a,b;
input  ci;
output [1:0] s;
output co;
wire [1:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);

assign s=ci?S1:S0;
assign co=ci?C1[1]:C0[1];

endmodule

// 3 bit adder
```

```
module bit_3_adder(a,b,ci,s,co);
input  [2:0] a,b;
input  ci;
output [2:0] s;
output co;
wire [2:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);

assign s=ci?S1:S0;
assign co=ci?C1[2]:C0[2];

endmodule


// 4 bit adder
```

```
module bit_4_adder(a,b,ci,s,co);
input  [3:0] a,b;
input  ci;
output [3:0] s;
output co;
wire [3:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);

assign s=ci?S1:S0;
assign co=ci?C1[3]:C0[3];

endmodule
```

```
// 5 bit adder

module bit_5_adder(a,b,ci,s,co);
input  [4:0] a,b;
input  ci;
output [4:0] s;
output co;
wire [4:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);
FA FA05(a[4],b[4],C0[3],S0[4],C0[4]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);
FA FA15(a[4],b[4],C1[3],S1[4],C1[4]);

assign #0.26 s=ci?S1:S0;
```

```
assign #0.26 co=ci?C1[4]:C0[4];

endmodule


// first stage
HA ha_0 (.A(p[0][1]), .B( p[1][0]), .S(s[0]), .Cout(c[0]));
FA fa_1 (.A(p[0][2]), .B( p[1][1]), .Cin( p[2][0] ), .S(s[1]), .Cout(c[1]) );
FA fa_2 (.A(p[1][2]), .B( p[2][1]), .Cin(p[3][0]), .S(s[2]), .Cout(c[2]));
FA fa_3 (.A(p[2][2]), .B( p[3][1]), .Cin( p[4][0] ), .S(s[3]), .Cout(c[3]) );
FA fa_4 (.A(p[0][5]), .B( p[1][4]), .Cin(p[2][3]), .S(s[4]), .Cout(c[4]));
FA fa_5 (.A(p[3][2]), .B( p[4][1]), .Cin( p[5][0] ), .S(s[5]), .Cout(c[5]) );
FA fa_6 (.A(p[1][5]), .B( p[2][4]), .Cin(p[3][3]),.S(s[6]), .Cout(c[6]));
FA fa_7 (.A(p[4][2]), .B( p[5][1]), .Cin( p[6][0] ), .S(s[7]), .Cout(c[7]) );
FA fa_8 (.A(p[2][5]), .B( p[3][4]),.Cin(p[4][3]), .S(s[8]), .Cout(c[8]));
FA fa_9 (.A(p[5][2]), .B( p[6][1]), .Cin( p[7][0] ), .S(s[9]), .Cout(c[9]) );
FA fa_10 (.A(p[2][6]), .B( p[3][5]), .Cin( p[4][4] ), .S(s[10]), .Cout(c[10]));
FA fa_11 (.A(p[5][3]), .B( p[6][2]), .Cin( p[7][1] ), .S(s[11]), .Cout(c[11]));
FA fa_12 (.A(p[2][7]), .B( p[3][6]), .Cin(p[4][5] ), .S(s[12]), .Cout(c[12]));
FA fa_13 (.A(p[5][4]), .B( p[6][3]), .Cin( p[7][2] ), .S(s[13]), .Cout(c[13]));
```

```
HA ha_14 (.A(p[3][7]), .B( p[4][6]), .S(s[14]), .Cout(c[14]));

FA fa_15 (.A(p[5][5]), .B(p[6][4]), .Cin( p[7][3] ), .S(s[15]), .Cout(c[15]));

FA fa_16 (.A(p[5][6]), .B( p[6][5]), .Cin( p[7][4] ), .S(s[16]), .Cout(c[16]));

FA fa_17 (.A(p[5][7]), .B( p[6][6]), .Cin( p[7][5] ), .S(s[17]), .Cout(c[17]));


//2nd stage


HA ha_18 (.A(c[0]), .B( s[1]), .S(s[18]), .Cout(c[18]));

FA fa_19 (.A(c[1]), .B( p[0][3]), .Cin( s[2] ), .S(s[19]), .Cout(c[19]));

FA fa_20 (.A(p[0][4]), .B( p[1][3]), .Cin( s[3] ), .S(s[20]), .Cout(c[20]));

FA fa_21 (.A(c[3]), .B( s[4]), .Cin( s[5] ), .S(s[21]), .Cout(c[21]));

HA ha_22 (.A(c[4]), .B( c[5]), .S(s[22]), .Cout(c[22]));

FA fa_23 (.A(p[0][6]), .B(s[6]), .Cin( s[7] ), .S(s[23]), .Cout(c[23]));

FA fa_24 (.A(c[6]), .B(c[7]), .Cin(p[0][7]), .S(s[24]), .Cout(c[24]));

FA fa_25 (.A(p[1][6]), .B(s[8]), .Cin(s[9]), .S(s[25]), .Cout(c[25]));

HA ha_26 (.A(c[8]), .B(c[9]), .S(s[26]), .Cout(c[26]));

FA fa_27 (.A(p[1][7]), .B(s[10]), .Cin(s[11] ), .S(s[27]), .Cout(c[27]));

FA fa_28 (.A(c[11]), .B( s[12]), .Cin(s[13]), .S(s[28]), .Cout(c[28]));

FA fa_29 (.A(c[13]), .B( s[14]), .Cin( s[15] ), .S(s[29]), .Cout(c[29]));

FA fa_30 (.A(c[15]), .B( p[4][7]),.Cin( s[16] ), .S(s[30]), .Cout(c[30]));

FA fa_31 (.A(c[17]), .B(p[6][7]), .Cin(p[7][6]), .S(s[31]), .Cout(c[31]));
```

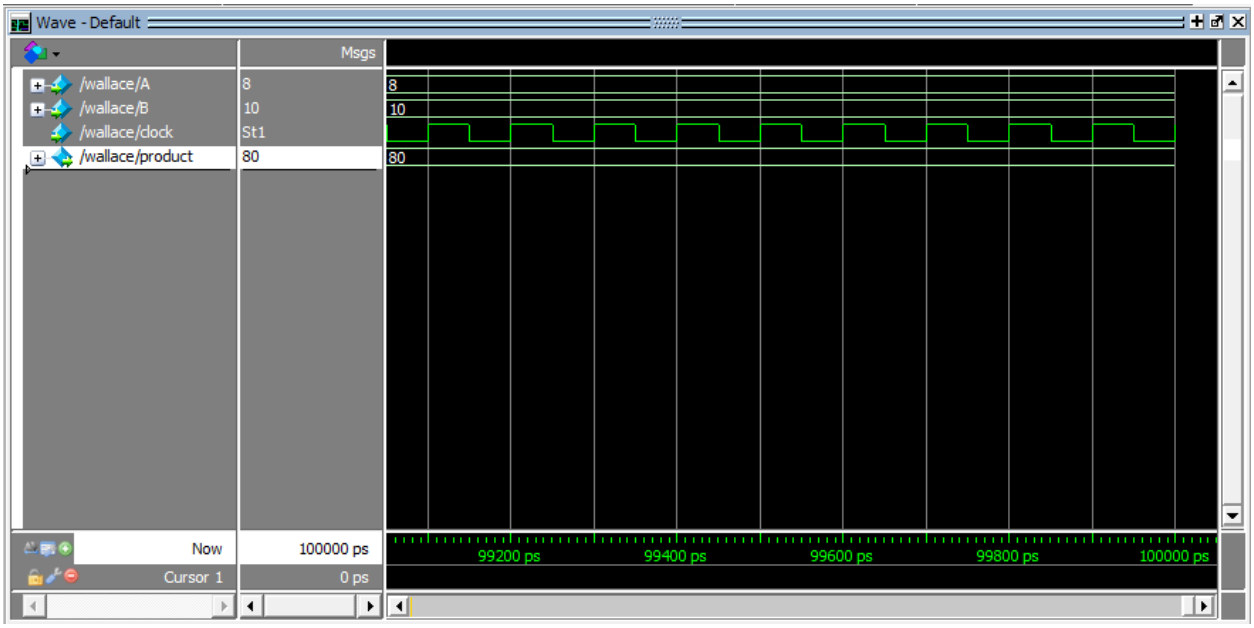
```
// 3rd stage

HA ha_32 (.A(c[18]), .B(s[19]), .S(s[32]), .Cout(c[32]));
FA fa_33 (.A(c[19]), .B(c[2]), .Cin(s[20]), .S(s[33]), .Cout(c[33]));
FA fa_34 (.A(c[21]), .B(s[22]), .Cin(s[23]), .S(s[34]), .Cout(c[34]));
FA fa_35 (.A(c[23]), .B(s[24]), .Cin(s[25]), .S(s[35]), .Cout(c[35]));
FA fa_36 (.A(c[25]), .B(s[26]), .Cin(s[27]), .S(s[36]), .Cout(c[36]));
FA fa_37 (.A(c[27]), .B(c[10]), .Cin(s[28]), .S(s[37]), .Cout(c[37]));
FA fa_38 (.A(c[28]), .B(c[12]), .Cin(s[29]), .S(s[38]), .Cout(c[38]));
FA fa_39 (.A(c[29]), .B(c[14]), .Cin(s[30]), .S(s[39]), .Cout(c[39]));
FA fa_40 (.A(c[30]), .B(c[16]), .Cin(s[17]), .S(s[40]), .Cout(c[40]));
HA ha_41 (.A(c[31]), .B(p[7][7]), .S(s[41]), .Cout(c[41]));

// 4th stage

HA ha_42 (.A(c[32]), .B( s[33]), .S(s[42]), .Cout(c[42]));
FA fa_43 (.A(c[33]), .B(c[20]), .Cin(s[21]), .S(s[43]), .Cout(c[43]));
FA fa_44 (.A(c[34]), .B(c[22]), .Cin(s[35]), .S(s[44]), .Cout(c[44]));
FA fa_45 (.A(c[35]), .B(c[24]), .Cin(s[36]), .S(s[45]), .Cout(c[45]));
FA fa_46 (.A(c[36]), .B(c[26]), .Cin(s[37]), .S(s[46]), .Cout(c[46]));
HA ha_47 (.A(c[37]), .B(s[38]), .S(s[47]), .Cout(c[47]));
HA ha_48 (.A(c[38]), .B(s[39]),  .S(s[48]), .Cout(c[48]));
```

```
HA ha_49 (.A(c[39]), .B(s[40]),  .S(s[49]), .Cout(c[49]));
HA ha_50 (.A(c[40]), .B(s[31]),  .S(s[50]), .Cout(c[50]));


HA ha_51 (.A(c[42]), .B(s[43]),  .S(s[51]), .Cout(c[51]));
FA fa_52 (.A(c[51]), .B(c[43]), .Cin(s[34]), .S(s[52]), .Cout(c[52]));
HA ha_53 (.A(c[52]), .B(s[44]), .S(s[53]), .Cout(c[53]));
FA fa_54 (.A(c[53]), .B(c[44]), .Cin(s[45]), .S(s[54]), .Cout(c[54]));
FA fa_55 (.A(c[54]), .B(c[45]), .Cin(s[46]), .S(s[55]), .Cout(c[55]));
FA fa_56 (.A(c[55]), .B(c[46]), .Cin(s[47]), .S(s[56]), .Cout(c[56]));
FA fa_57 (.A(c[56]), .B(c[47]), .Cin(s[48]), .S(s[57]), .Cout(c[57]));
FA fa_58 (.A(c[57]), .B(c[48]), .Cin(s[49]), .S(s[58]), .Cout(c[58]));
FA fa_59 (.A(c[58]), .B(c[49]), .Cin(s[50]), .S(s[59]), .Cout(c[59]));
FA fa_60 (.A(c[59]), .B(c[50]), .Cin(s[41]), .S(s[60]), .Cout(c[60]));
HA ha_61 (.A(c[60]), .B(c[41]), .S(s[61]), .Cout(c[61]));


always@(posedge clock)
begin
product                                                                 <=
{c[61],s[61],s[60],s[59],s[58],s[57],s[56],s[55],s[54],s[53],s[52],s[51],s[
42],s[32],s[18],s[0],p[0][0] };
```

end

endmodule

Simulation Result:



Input = 8 & 10, Output=80

Dadda:

```
// Half Adder

module HA
  (A,B,S,Cout);
  input A,B;
  output S,Cout;
  assign S=A ^ B ;
  assign Cout=A & B;// Half Adder
endmodule

// Full Adder

module FA
  (A,B,Cin,S,Cout);
  input A,B,Cin ;
  output S,Cout;
  assign S=A ^ B ^ Cin ;
  assign Cout=(A & B) | (B & Cin) | (A & Cin);
endmodule

`timescale 1ns / 1ps
```

// 2 bit adder

// DADA Multiplier

```
module dada(output reg [15:0] product, input [7:0] A, B,input clock );
reg p [7:0][7:0];
wire [55:0] s ,c ;
integer i,j;

always@(A,B)

begin
  for ( i=0; i<=7; i=i+1)
    for ( j = 0; j <= 7; j = j + 1)
        p[i][j] <= A[j] & B[i];
end
```

```
module bit_2_adder(a,b,ci,s,co);
input  [1:0] a,b;
input  ci;
output [1:0] s;
output co;
wire [1:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);

assign s=ci?S1:S0;
assign co=ci?C1[1]:C0[1];

endmodule

// 3 bit adder

module bit_3_adder(a,b,ci,s,co);
```

```
input  [2:0] a,b;
input  ci;
output [2:0] s;
output co;
wire [2:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);

assign s=ci?S1:S0;
assign co=ci?C1[2]:C0[2];

endmodule


// 4 bit adder

module bit_4_adder(a,b,ci,s,co);
```

```
input  [3:0] a,b;
input  ci;
output [3:0] s;
output co;
wire [3:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);

assign s=ci?S1:S0;
assign co=ci?C1[3]:C0[3];

endmodule


// 5 bit adder
```

```
module bit_5_adder(a,b,ci,s,co);
input  [4:0] a,b;
input  ci;
output [4:0] s;
output co;
wire [4:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);
FA FA05(a[4],b[4],C0[3],S0[4],C0[4]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);
FA FA15(a[4],b[4],C1[3],S1[4],C1[4]);

assign #0.26 s=ci?S1:S0;
assign #0.26 co=ci?C1[4]:C0[4];
```

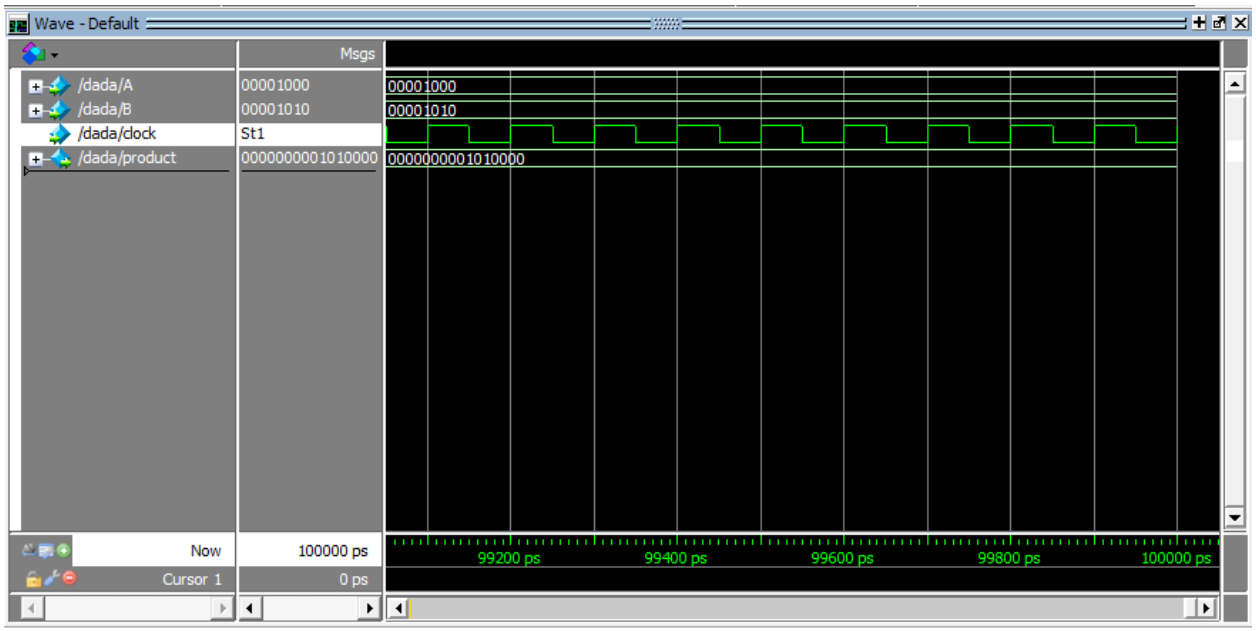
```
endmodule



// first stage
HA ha_0 (.A(p[5][1]), .B( p[6][0]), .S(s[0]), .Cout(c[0]));
FA fa_1 (.A(p[3][4]), .B( p[4][3]), .Cin( p[5][2] ), .S(s[1]),
.Cout(c[1]) );
HA ha_2 (.A(p[6][1]), .B( p[7][0]), .S(s[2]), .Cout(c[2]));
FA fa_3 (.A(p[3][5]), .B( p[4][4]), .Cin( p[5][3] ), .S(s[3]),
.Cout(c[3]) );
HA ha_4 (.A(p[6][2]), .B( p[7][1]), .S(s[4]), .Cout(c[4]));
FA fa_5 (.A(p[5][4]), .B( p[6][3]), .Cin( p[7][2] ), .S(s[5]),
.Cout(c[5]) );


//2nd stage
HA ha_6 (.A(p[3][1]), .B( p[4][0]), .S(s[6]), .Cout(c[6]));
FA fa_7 (.A(p[1][4]), .B( p[2][3]), .Cin( p[3][2] ), .S(s[7]),
.Cout(c[7]) );
HA ha_8 (.A(p[4][1]), .B( p[5][0]), .S(s[8]), .Cout(c[8]));
FA fa_9 (.A(p[0][6]), .B( p[1][5]), .Cin( p[2][4] ), .S(s[9]),
.Cout(c[9]) );
FA fa_10 (.A(p[3][3]), .B( p[4][2]), .Cin( p[5][1] ), .S(s[10]),
.Cout(c[10]));
```

```
FA fa_11 (.A(c[0]), .B( p[0][7]), .Cin( p[1][6] ), .S(s[11]),
.Cout(c[11]));

FA fa_12 (.A(p[2][5]), .B( s[1]), .Cin( s[2] ), .S(s[12]), .Cout(c[12]));

FA fa_13 (.A(c[1]), .B( c[2]), .Cin( p[1][7] ), .S(s[13]), .Cout(c[13]));

FA fa_14 (.A(p[2][6]), .B( s[3]), .Cin( s[4] ), .S(s[14]), .Cout(c[14]));

FA fa_15 (.A(c[3]), .B( c[4]), .Cin( p[2][7] ), .S(s[15]), .Cout(c[15]));

FA fa_16 (.A(p[3][6]), .B( p[4][5]), .Cin( s[5] ), .S(s[16]),
.Cout(c[16]));

FA fa_17 (.A(c[5]), .B( p[3][7]), .Cin( p[4][6] ), .S(s[17]),
.Cout(c[17]));

FA fa_18 (.A(p[5][5]), .B( p[6][4]), .Cin( p[7][3] ), .S(s[18]),
.Cout(c[18]));

FA fa_19 (.A(p[5][6]), .B( p[6][5]), .Cin( p[7][4] ), .S(s[19]),
.Cout(c[19]));


// 3rd stage

HA ha_20 (.A(p[2][1]), .B( p[3][0]), .S(s[20]), .Cout(c[20]));

FA fa_21 (.A(p[1][3]), .B( p[2][2]), .Cin( p[3][1] ), .S(s[21]),
.Cout(c[21]));

FA fa_22 (.A(p[0][5]), .B( s[7]), .Cin( s[8] ), .S(s[22]), .Cout(c[22]));

FA fa_23 (.A(c[8]), .B(s[9]), .Cin( s[10] ), .S(s[23]), .Cout(c[23]));

FA fa_24 (.A(c[10]), .B(s[11]), .Cin(s[12]), .S(s[24]), .Cout(c[24]));

FA fa_25 (.A(c[12]), .B(s[13]), .Cin(s[14]), .S(s[25]), .Cout(c[25]));

FA fa_26 (.A(c[14]), .B(s[15]), .Cin(s[16]), .S(s[26]), .Cout(c[26]));

FA fa_27 (.A(c[16]), .B(s[17]), .Cin(s[18] ), .S(s[27]), .Cout(c[27]));
```

```
FA fa_28 (.A(c[18]), .B( p[4][7]), .Cin(s[19]), .S(s[28]),
.Cout(c[28]));

FA fa_29 (.A(p[5][7]), .B( p[6][6]), .Cin( p[7][5] ), .S(s[29]),
.Cout(c[29]));


// 4th satge

HA ha_30 (.A(p[1][1]), .B( p[2][0]), .S(s[30]), .Cout(c[30]));

FA fa_31 (.A(p[0][3]), .B(p[1][2]), .Cin(s[20]), .S(s[31]),
.Cout(c[31]));

FA fa_32 (.A(c[20]), .B(p[0][4]), .Cin(s[21]), .S(s[32]), .Cout(c[32]));

FA fa_33 (.A(c[21]), .B(c[6]), .Cin(s[22]), .S(s[33]), .Cout(c[33]));

FA fa_34 (.A(c[22]), .B(c[7]), .Cin(s[23]), .S(s[34]), .Cout(c[34]));

FA fa_35 (.A(c[23]), .B(c[9]), .Cin(s[24]), .S(s[35]), .Cout(c[35]));

FA fa_36 (.A(c[24]), .B(c[11]), .Cin(s[25]), .S(s[36]), .Cout(c[36]));

FA fa_37 (.A(c[25]), .B(c[13]), .Cin(s[26]), .S(s[37]), .Cout(c[37]));

FA fa_38 (.A(c[26]), .B(c[15]), .Cin(s[27]), .S(s[38]), .Cout(c[38]));

FA fa_39 (.A(c[27]), .B(c[17]), .Cin(s[28]), .S(s[39]), .Cout(c[39]));

FA fa_40 (.A(c[28]), .B(c[19]), .Cin(s[29]), .S(s[40]), .Cout(c[40]));

FA fa_41 (.A(c[29]), .B(p[6][7]), .Cin(p[7][6]), .S(s[41]),
.Cout(c[41]));


// ripple last carry adder
```

```
bit_3_adder
add42({c[30],p[0][2],p[0][1]},{s[31],s[30],p[1][0]},{1'b0},{s[44],s[43],s[42]},{c[44]});

bit_3_adder
add43({c[33],c[32],c[31]},{s[34],s[33],s[32]},{c[44]},{s[47],s[46],s[45]},{c[47]});

bit_4_adder
add44({c[37],c[36],c[35],c[34]},{s[38],s[37],s[36],s[35]},{c[47]},{s[51],s[50],s[49],s[48]},{c[51]});

bit_4_adder
add45({c[41],c[40],c[39],c[38]},{p[7][7],s[41],s[40],s[39]},{c[51]},{s[55],s[54],s[53],s[52]},{c[55]});



always@(posedge clock)

begin

product <=
{c[55],s[55],s[54],s[53],s[52],s[51],s[50],s[49],s[48],s[47],s[46],s[45],s[44],s[43],s[42],p[0][0]};

end

endmodule
```

Simulation Result:



Input = 8 & 10, Output=80

Q–3 Now back annotate the delays of half adders and full adders from the circuit simulation done earlier.

Ans: Verilog Program with Delay

Wallace:

```
// Half Adder

module HA
  (A,B,S,Cout);
  output S,Cout;
  input A,B;
  assign #0.2955 Cout=A & B;
  assign #0.79 S=A ^ B ;

endmodule




// Full Adder

module FA
```

```
  (A,B,Cin,S,Cout);
  output S,Cout;
  input A,B,Cin ;
  assign #3.67603 Cout=(A & B) | (B & Cin) | (A & Cin);
  assign #3.652009 S=A ^ B ^ Cin ;
endmodule

`timescale 1ns / 1ps

// Wallace Multiplier

module wallace(output reg [16:0] product, input [7:0] A, B,input
clock );
reg p [7:0][7:0];
wire [61:0] s ,c ;
integer i,j;

always@(A,B)

begin
  for ( i=0; i<=7; i=i+1)
    for ( j = 0; j <= 7; j = j + 1)
        p[i][j] <= A[j] & B[i];
```

```
end


module bit_2_adder(a,b,ci,s,co);
input  [1:0] a,b;
input  ci;
output [1:0] s;
output co;
wire [1:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);

assign s=ci?S1:S0;
assign co=ci?C1[1]:C0[1];

endmodule

// 3 bit adder
```

```
module bit_3_adder(a,b,ci,s,co);
input  [2:0] a,b;
input  ci;
output [2:0] s;
output co;
wire [2:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);

assign s=ci?S1:S0;
assign co=ci?C1[2]:C0[2];

endmodule


// 4 bit adder
```

```
module bit_4_adder(a,b,ci,s,co);
input  [3:0] a,b;
input  ci;
output [3:0] s;
output co;
wire [3:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);

assign s=ci?S1:S0;
assign co=ci?C1[3]:C0[3];

endmodule
```

```
// 5 bit adder

module bit_5_adder(a,b,ci,s,co);
input  [4:0] a,b;
input  ci;
output [4:0] s;
output co;
wire [4:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);
FA FA05(a[4],b[4],C0[3],S0[4],C0[4]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);
FA FA15(a[4],b[4],C1[3],S1[4],C1[4]);

assign #0.26 s=ci?S1:S0;
assign #0.26 co=ci?C1[4]:C0[4];
```

```
endmodule



// first stage
HA ha_0 (.A(p[0][1]), .B( p[1][0]), .S(s[0]), .Cout(c[0]));
FA fa_1 (.A(p[0][2]), .B( p[1][1]), .Cin( p[2][0] ), .S(s[1]),
.Cout(c[1]) );
FA fa_2 (.A(p[1][2]), .B( p[2][1]), .Cin(p[3][0]), .S(s[2]),
.Cout(c[2]));
FA fa_3 (.A(p[2][2]), .B( p[3][1]), .Cin( p[4][0] ), .S(s[3]),
.Cout(c[3]) );
FA fa_4 (.A(p[0][5]), .B( p[1][4]), .Cin(p[2][3]), .S(s[4]),
.Cout(c[4]));
FA fa_5 (.A(p[3][2]), .B( p[4][1]), .Cin( p[5][0] ), .S(s[5]),
.Cout(c[5]) );
FA fa_6 (.A(p[1][5]), .B( p[2][4]), .Cin(p[3][3]),.S(s[6]), .Cout(c[6]));
FA fa_7 (.A(p[4][2]), .B( p[5][1]), .Cin( p[6][0] ), .S(s[7]),
.Cout(c[7]) );
FA fa_8 (.A(p[2][5]), .B( p[3][4]),.Cin(p[4][3]), .S(s[8]), .Cout(c[8]));
FA fa_9 (.A(p[5][2]), .B( p[6][1]), .Cin( p[7][0] ), .S(s[9]),
.Cout(c[9]) );
FA fa_10 (.A(p[2][6]), .B( p[3][5]), .Cin( p[4][4] ), .S(s[10]),
.Cout(c[10]));
FA fa_11 (.A(p[5][3]), .B( p[6][2]), .Cin( p[7][1] ), .S(s[11]),
.Cout(c[11]));
```

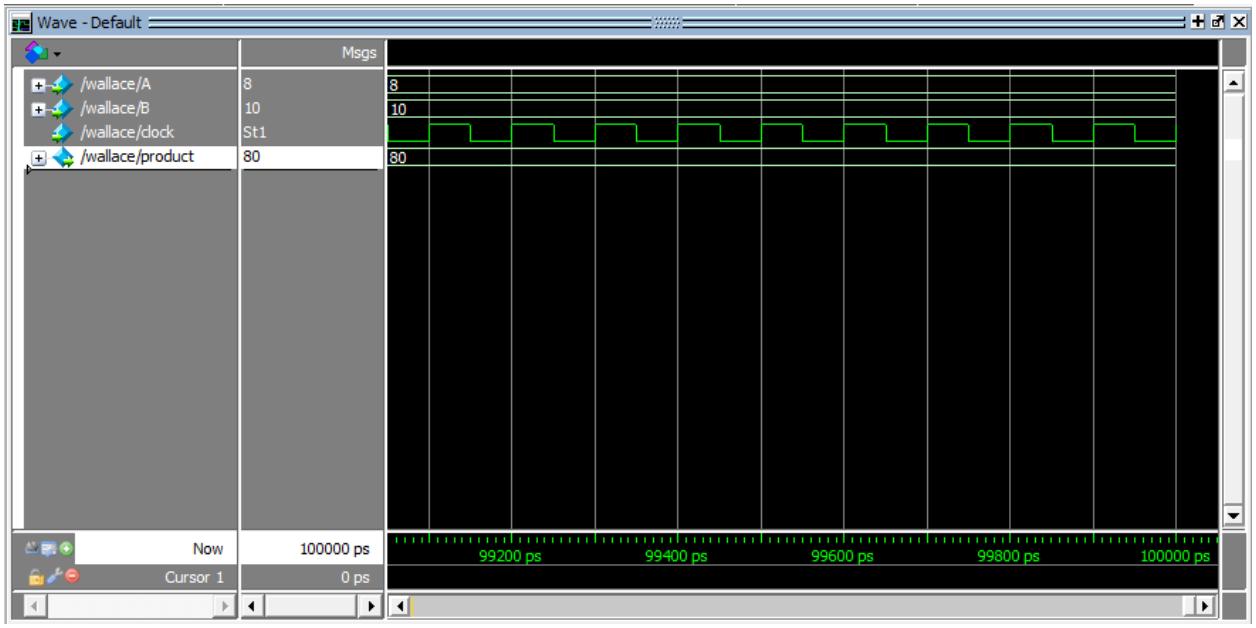
```
FA fa_12 (.A(p[2][7]), .B( p[3][6]), .Cin(p[4][5] ), .S(s[12]),
.Cout(c[12]));

FA fa_13 (.A(p[5][4]), .B( p[6][3]), .Cin( p[7][2] ), .S(s[13]),
.Cout(c[13]));

HA ha_14 (.A(p[3][7]), .B( p[4][6]), .S(s[14]), .Cout(c[14]));

FA fa_15 (.A(p[5][5]), .B(p[6][4]), .Cin( p[7][3] ), .S(s[15]),
.Cout(c[15]));

FA fa_16 (.A(p[5][6]), .B( p[6][5]), .Cin( p[7][4] ), .S(s[16]),
.Cout(c[16]));

FA fa_17 (.A(p[5][7]), .B( p[6][6]), .Cin( p[7][5] ), .S(s[17]),
.Cout(c[17]));


//2nd stage


HA ha_18 (.A(c[0]), .B( s[1]), .S(s[18]), .Cout(c[18]));

FA fa_19 (.A(c[1]), .B( p[0][3]), .Cin( s[2] ), .S(s[19]), .Cout(c[19]));

FA fa_20 (.A(p[0][4]), .B( p[1][3]), .Cin( s[3] ), .S(s[20]),
.Cout(c[20]));

FA fa_21 (.A(c[3]), .B( s[4]), .Cin( s[5] ), .S(s[21]), .Cout(c[21]));

HA ha_22 (.A(c[4]), .B( c[5]), .S(s[22]), .Cout(c[22]));

FA fa_23 (.A(p[0][6]), .B(s[6]), .Cin( s[7] ), .S(s[23]), .Cout(c[23]));

FA fa_24 (.A(c[6]), .B(c[7]), .Cin(p[0][7]), .S(s[24]), .Cout(c[24]));

FA fa_25 (.A(p[1][6]), .B(s[8]), .Cin(s[9]), .S(s[25]), .Cout(c[25]));

HA ha_26 (.A(c[8]), .B(c[9]), .S(s[26]), .Cout(c[26]));
```

```
FA fa_27 (.A(p[1][7]), .B(s[10]), .Cin(s[11] ), .S(s[27]),
.Cout(c[27]));

FA fa_28 (.A(c[11]), .B( s[12]), .Cin(s[13]), .S(s[28]), .Cout(c[28]));

FA fa_29 (.A(c[13]), .B( s[14]), .Cin( s[15] ), .S(s[29]), .Cout(c[29]));

FA fa_30 (.A(c[15]), .B( p[4][7]),.Cin( s[16] ), .S(s[30]),
.Cout(c[30]));

FA fa_31 (.A(c[17]), .B(p[6][7]), .Cin(p[7][6]), .S(s[31]),
.Cout(c[31]));


// 3rd stage


HA ha_32 (.A(c[18]), .B(s[19]), .S(s[32]), .Cout(c[32]));

FA fa_33 (.A(c[19]), .B(c[2]), .Cin(s[20]), .S(s[33]), .Cout(c[33]));

FA fa_34 (.A(c[21]), .B(s[22]), .Cin(s[23]), .S(s[34]), .Cout(c[34]));

FA fa_35 (.A(c[23]), .B(s[24]), .Cin(s[25]), .S(s[35]), .Cout(c[35]));

FA fa_36 (.A(c[25]), .B(s[26]), .Cin(s[27]), .S(s[36]), .Cout(c[36]));

FA fa_37 (.A(c[27]), .B(c[10]), .Cin(s[28]), .S(s[37]), .Cout(c[37]));

FA fa_38 (.A(c[28]), .B(c[12]), .Cin(s[29]), .S(s[38]), .Cout(c[38]));

FA fa_39 (.A(c[29]), .B(c[14]), .Cin(s[30]), .S(s[39]), .Cout(c[39]));

FA fa_40 (.A(c[30]), .B(c[16]), .Cin(s[17]), .S(s[40]), .Cout(c[40]));

HA ha_41 (.A(c[31]), .B(p[7][7]), .S(s[41]), .Cout(c[41]));


// 4th stage
```

```
HA ha_42 (.A(c[32]), .B( s[33]), .S(s[42]), .Cout(c[42]));
FA fa_43 (.A(c[33]), .B(c[20]), .Cin(s[21]), .S(s[43]), .Cout(c[43]));
FA fa_44 (.A(c[34]), .B(c[22]), .Cin(s[35]), .S(s[44]), .Cout(c[44]));
FA fa_45 (.A(c[35]), .B(c[24]), .Cin(s[36]), .S(s[45]), .Cout(c[45]));
FA fa_46 (.A(c[36]), .B(c[26]), .Cin(s[37]), .S(s[46]), .Cout(c[46]));
HA ha_47 (.A(c[37]), .B(s[38]), .S(s[47]), .Cout(c[47]));
HA ha_48 (.A(c[38]), .B(s[39]),  .S(s[48]), .Cout(c[48]));
HA ha_49 (.A(c[39]), .B(s[40]),  .S(s[49]), .Cout(c[49]));
HA ha_50 (.A(c[40]), .B(s[31]),  .S(s[50]), .Cout(c[50]));


HA ha_51 (.A(c[42]), .B(s[43]),  .S(s[51]), .Cout(c[51]));
FA fa_52 (.A(c[51]), .B(c[43]), .Cin(s[34]), .S(s[52]), .Cout(c[52]));
HA ha_53 (.A(c[52]), .B(s[44]), .S(s[53]), .Cout(c[53]));
FA fa_54 (.A(c[53]), .B(c[44]), .Cin(s[45]), .S(s[54]), .Cout(c[54]));
FA fa_55 (.A(c[54]), .B(c[45]), .Cin(s[46]), .S(s[55]), .Cout(c[55]));
FA fa_56 (.A(c[55]), .B(c[46]), .Cin(s[47]), .S(s[56]), .Cout(c[56]));
FA fa_57 (.A(c[56]), .B(c[47]), .Cin(s[48]), .S(s[57]), .Cout(c[57]));
FA fa_58 (.A(c[57]), .B(c[48]), .Cin(s[49]), .S(s[58]), .Cout(c[58]));
FA fa_59 (.A(c[58]), .B(c[49]), .Cin(s[50]), .S(s[59]), .Cout(c[59]));
FA fa_60 (.A(c[59]), .B(c[50]), .Cin(s[41]), .S(s[60]), .Cout(c[60]));
HA ha_61 (.A(c[60]), .B(c[41]), .S(s[61]), .Cout(c[61]));
```

```
always@(posedge clock)
begin
product <=
{c[61],s[61],s[60],s[59],s[58],s[57],s[56],s[55],s[54],s[53],s[52],s[51]
,s[42],s[32],s[18],s[0],p[0][0] };
end
endmodule
```

Simulation result:



Input = 8 & 10, Output=80

Dadda:

```
// Half Adder


module HA
  (A,B,S,Cout);
  input A,B;
  output S,Cout;
  assign #0.2170285 S=A ^ B ;            //delay of sum for half adder
  assign #0.1881537 Cout=A & B;          // delay of carry for half
adder
endmodule




// Full Adder


module FA
  (A,B,Cin,S,Cout);
  input A,B,Cin ;
  output S,Cout;
  assign #3.684794 S=A ^ B ^ Cin ;    //delay of sum for full adder
```

```
  assign #3.708672 Cout=(A & B) | (B & Cin) | (A & Cin); ////delay of sum for full adder
endmodule

`timescale 1ns / 1ps



// 2 bit adder









// DADA Multiplier

module dada(output reg [15:0] product, input [7:0] A, B,input clock );
reg p [7:0][7:0];
wire [55:0] s ,c ;
integer i,j;

always@(A,B)
```

```
begin
  for ( i=0; i<=7; i=i+1)
    for ( j = 0; j <= 7; j = j + 1)
        p[i][j] <= A[j] & B[i];
end


module bit_2_adder(a,b,ci,s,co);
input  [1:0] a,b;
input  ci;
output [1:0] s;
output co;
wire [1:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);

assign s=ci?S1:S0;
assign co=ci?C1[1]:C0[1];
```

```
endmodule

// 3 bit adder

module bit_3_adder(a,b,ci,s,co);
input  [2:0] a,b;
input  ci;
output [2:0] s;
output co;
wire [2:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);

assign s=ci?S1:S0;
assign co=ci?C1[2]:C0[2];
```

```
endmodule


// 4 bit adder

module bit_4_adder(a,b,ci,s,co);
input  [3:0] a,b;
input  ci;
output [3:0] s;
output co;
wire [3:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);

assign s=ci?S1:S0;
```

```
assign co=ci?C1[3]:C0[3];

endmodule


// 5 bit adder

module bit_5_adder(a,b,ci,s,co);
input  [4:0] a,b;
input  ci;
output [4:0] s;
output co;
wire [4:0] S1,S0,C0,C1;

FA FA01(a[0],b[0],1'b0,S0[0],C0[0]);
FA FA02(a[1],b[1],C0[0],S0[1],C0[1]);
FA FA03(a[2],b[2],C0[1],S0[2],C0[2]);
FA FA04(a[3],b[3],C0[2],S0[3],C0[3]);
FA FA05(a[4],b[4],C0[3],S0[4],C0[4]);

FA FA11(a[0],b[0],1'b1,S1[0],C1[0]);
FA FA12(a[1],b[1],C1[0],S1[1],C1[1]);
FA FA13(a[2],b[2],C1[1],S1[2],C1[2]);
```

```
FA FA14(a[3],b[3],C1[2],S1[3],C1[3]);
FA FA15(a[4],b[4],C1[3],S1[4],C1[4]);

assign #0.26 s=ci?S1:S0;
assign #0.26 co=ci?C1[4]:C0[4];

endmodule



// first stage
HA ha_0 (.A(p[5][1]), .B( p[6][0]), .S(s[0]), .Cout(c[0]));
FA fa_1 (.A(p[3][4]), .B( p[4][3]), .Cin( p[5][2] ), .S(s[1]),
.Cout(c[1]) );
HA ha_2 (.A(p[6][1]), .B( p[7][0]), .S(s[2]), .Cout(c[2]));
FA fa_3 (.A(p[3][5]), .B( p[4][4]), .Cin( p[5][3] ), .S(s[3]),
.Cout(c[3]) );
HA ha_4 (.A(p[6][2]), .B( p[7][1]), .S(s[4]), .Cout(c[4]));
FA fa_5 (.A(p[5][4]), .B( p[6][3]), .Cin( p[7][2] ), .S(s[5]),
.Cout(c[5]) );

//2nd stage
HA ha_6 (.A(p[3][1]), .B( p[4][0]), .S(s[6]), .Cout(c[6]));
```

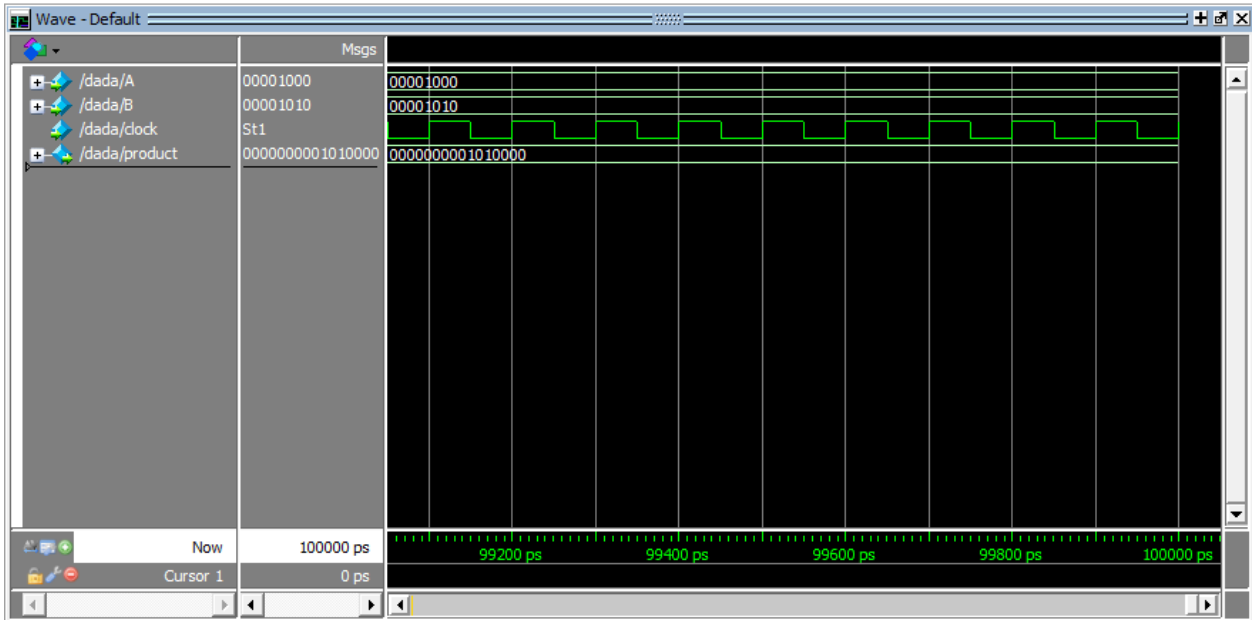
```
FA fa_7 (.A(p[1][4]), .B( p[2][3]), .Cin( p[3][2] ), .S(s[7]),
.Cout(c[7]) );

HA ha_8 (.A(p[4][1]), .B( p[5][0]), .S(s[8]), .Cout(c[8]));

FA fa_9 (.A(p[0][6]), .B( p[1][5]), .Cin( p[2][4] ), .S(s[9]),
.Cout(c[9]) );

FA fa_10 (.A(p[3][3]), .B( p[4][2]), .Cin( p[5][1] ), .S(s[10]),
.Cout(c[10]));

FA fa_11 (.A(c[0]), .B( p[0][7]), .Cin( p[1][6] ), .S(s[11]),
.Cout(c[11]));

FA fa_12 (.A(p[2][5]), .B( s[1]), .Cin( s[2] ), .S(s[12]), .Cout(c[12]));

FA fa_13 (.A(c[1]), .B( c[2]), .Cin( p[1][7] ), .S(s[13]), .Cout(c[13]));

FA fa_14 (.A(p[2][6]), .B( s[3]), .Cin( s[4] ), .S(s[14]), .Cout(c[14]));

FA fa_15 (.A(c[3]), .B( c[4]), .Cin( p[2][7] ), .S(s[15]), .Cout(c[15]));

FA fa_16 (.A(p[3][6]), .B( p[4][5]), .Cin( s[5] ), .S(s[16]),
.Cout(c[16]));

FA fa_17 (.A(c[5]), .B( p[3][7]), .Cin( p[4][6] ), .S(s[17]),
.Cout(c[17]));

FA fa_18 (.A(p[5][5]), .B( p[6][4]), .Cin( p[7][3] ), .S(s[18]),
.Cout(c[18]));

FA fa_19 (.A(p[5][6]), .B( p[6][5]), .Cin( p[7][4] ), .S(s[19]),
.Cout(c[19]));


// 3rd stage

HA ha_20 (.A(p[2][1]), .B( p[3][0]), .S(s[20]), .Cout(c[20]));

FA fa_21 (.A(p[1][3]), .B( p[2][2]), .Cin( p[3][1] ), .S(s[21]),
.Cout(c[21]));
```

```
FA fa_22 (.A(p[0][5]), .B( s[7]), .Cin( s[8] ), .S(s[22]), .Cout(c[22]));

FA fa_23 (.A(c[8]), .B(s[9]), .Cin( s[10] ), .S(s[23]), .Cout(c[23]));

FA fa_24 (.A(c[10]), .B(s[11]), .Cin(s[12]), .S(s[24]), .Cout(c[24]));

FA fa_25 (.A(c[12]), .B(s[13]), .Cin(s[14]), .S(s[25]), .Cout(c[25]));

FA fa_26 (.A(c[14]), .B(s[15]), .Cin(s[16]), .S(s[26]), .Cout(c[26]));

FA fa_27 (.A(c[16]), .B(s[17]), .Cin(s[18] ), .S(s[27]), .Cout(c[27]));

FA fa_28 (.A(c[18]), .B( p[4][7]), .Cin(s[19]), .S(s[28]),
.Cout(c[28]));

FA fa_29 (.A(p[5][7]), .B( p[6][6]), .Cin( p[7][5] ), .S(s[29]),
.Cout(c[29]));


// 4th satge

HA ha_30 (.A(p[1][1]), .B( p[2][0]), .S(s[30]), .Cout(c[30]));

FA fa_31 (.A(p[0][3]), .B(p[1][2]), .Cin(s[20]), .S(s[31]),
.Cout(c[31]));

FA fa_32 (.A(c[20]), .B(p[0][4]), .Cin(s[21]), .S(s[32]), .Cout(c[32]));

FA fa_33 (.A(c[21]), .B(c[6]), .Cin(s[22]), .S(s[33]), .Cout(c[33]));

FA fa_34 (.A(c[22]), .B(c[7]), .Cin(s[23]), .S(s[34]), .Cout(c[34]));

FA fa_35 (.A(c[23]), .B(c[9]), .Cin(s[24]), .S(s[35]), .Cout(c[35]));

FA fa_36 (.A(c[24]), .B(c[11]), .Cin(s[25]), .S(s[36]), .Cout(c[36]));

FA fa_37 (.A(c[25]), .B(c[13]), .Cin(s[26]), .S(s[37]), .Cout(c[37]));

FA fa_38 (.A(c[26]), .B(c[15]), .Cin(s[27]), .S(s[38]), .Cout(c[38]));

FA fa_39 (.A(c[27]), .B(c[17]), .Cin(s[28]), .S(s[39]), .Cout(c[39]));

FA fa_40 (.A(c[28]), .B(c[19]), .Cin(s[29]), .S(s[40]), .Cout(c[40]));
```

```
FA fa_41 (.A(c[29]), .B(p[6][7]), .Cin(p[7][6]), .S(s[41]),
.Cout(c[41]));


// ripple last carry adder


bit_3_adder
add42({c[30],p[0][2],p[0][1]},{s[31],s[30],p[1][0]},{1'b0},{s[44],s[4
3],s[42]},{c[44]});

bit_3_adder
add43({c[33],c[32],c[31]},{s[34],s[33],s[32]},{c[44]},{s[47],s[46],s[
45]},{c[47]});

bit_4_adder
add44({c[37],c[36],c[35],c[34]},{s[38],s[37],s[36],s[35]},{c[47]},{s[
51],s[50],s[49],s[48]},{c[51]});

bit_4_adder
add45({c[41],c[40],c[39],c[38]},{p[7][7],s[41],s[40],s[39]},{c[51]},{
s[55],s[54],s[53],s[52]},{c[55]});




always@(posedge clock)

begin

product <=
{c[55],s[55],s[54],s[53],s[52],s[51],s[50],s[49],s[48],s[47],s[46],s[45]
,s[44],s[43],s[42],p[0][0]};

end
```

endmodule

Simulation Result:



Input = 8 & 10, Output=80