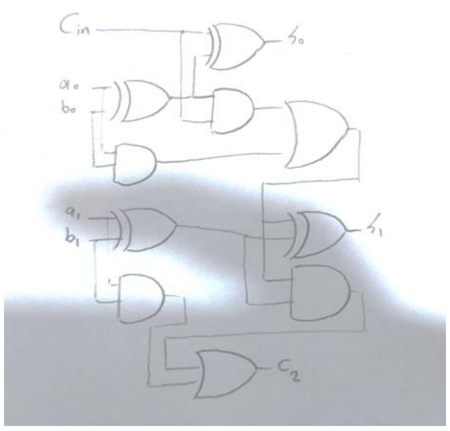
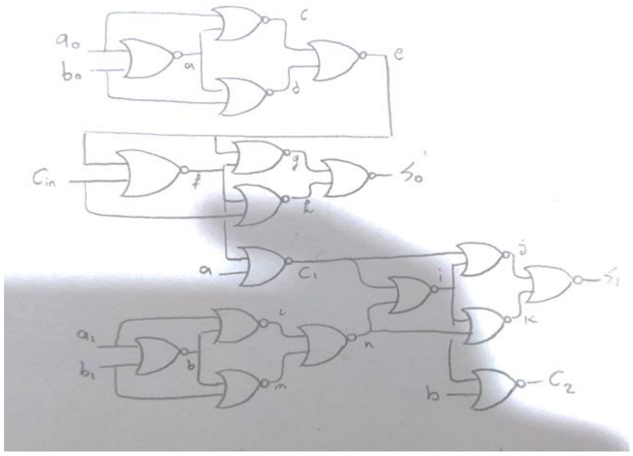
Amir Abbas Moumeni Zadeh – CA2 Report – SID : 810101529

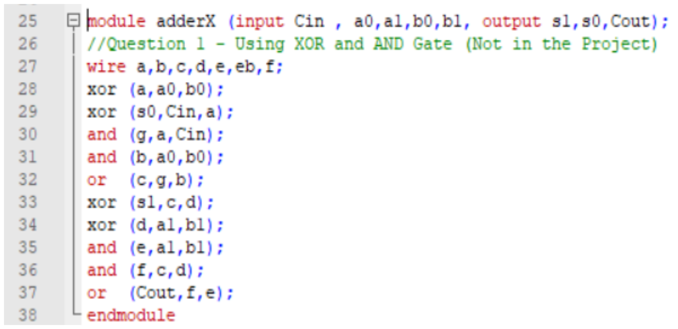
1)

We should build a 2-bit adder with carry in and carry out.

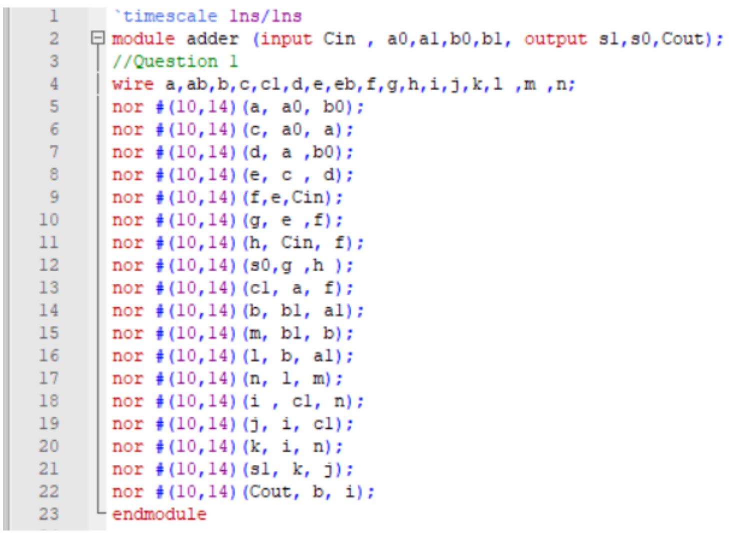
this is the original 2-bit adder but we have to convert it to all NOR gates.



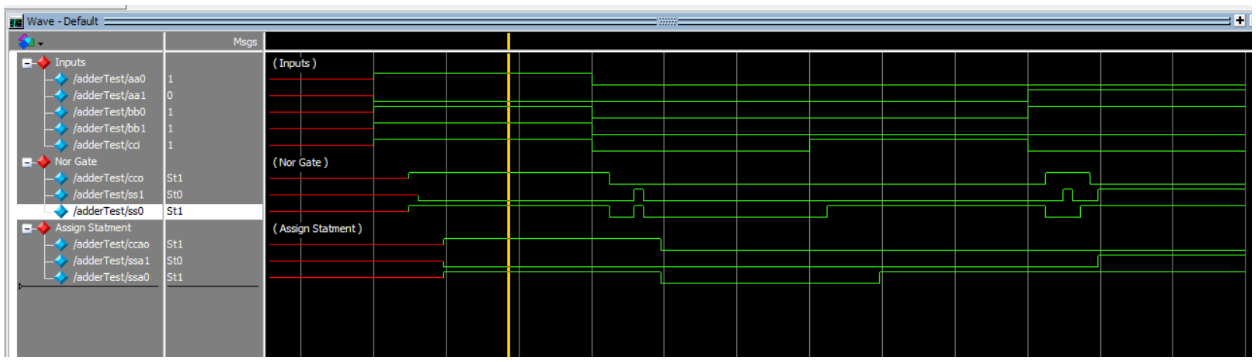




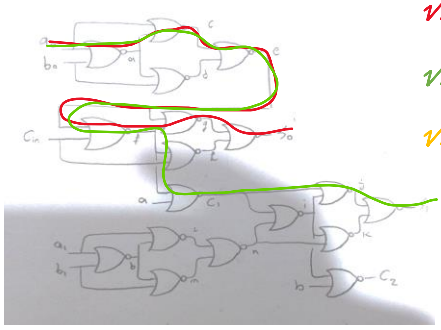
```
module adderX (input Cin , a0,a1,b0,b1, output s1,s0,Cout);
//Question 1 - Using XOR and AND Gate (Not in the Project)
wire a,b,c,d,e,eb,f;
xor (a,a0,b0);
xor (s0,Cin,a);
and (g,a,Cin);
and (b,a0,b0);
or  (c,g,b);
xor (s1,c,d);
xor (d,a1,b1);
and (e,a1,b1);
and (f,c,d);
or  (Cout,f,e);
endmodule
```



```
`timescale 1ns/1ns
module adder (input Cin , a0,a1,b0,b1, output s1,s0,Cout);
//Question 1
wire a,ab,b,c,c1,d,e,eb,f,g,h,i,j,k,l ,m ,n;
nor #(10,14)(a, a0, b0);
nor #(10,14)(c, a0, a);
nor #(10,14)(d, a ,b0);
nor #(10,14)(e, c , d);
nor #(10,14)(f,e,Cin);
nor #(10,14)(g, e ,f);
nor #(10,14)(h, Cin, f);
nor #(10,14)(s0,g ,h );
nor #(10,14)(c1, a, f);
nor #(10,14)(b, b1, a1);
nor #(10,14)(m, b1, b);
nor #(10,14)(l, b, a1);
nor #(10,14)(n, l, m);
nor #(10,14)(i , c1, n);
nor #(10,14)(j, i, c1);
nor #(10,14)(k, i, n);
nor #(10,14)(s1, k, j);
nor #(10,14)(Cout, b, i);
endmodule
```



```
Inputs
  /adderTest/aa0    1
  /adderTest/aa1    0
  /adderTest/bb0    1
  /adderTest/bb1    1
  /adderTest/cci    1
Nor Gate
  /adderTest/cco    St1
  /adderTest/ss1    St0
  /adderTest/ss0    St1
Assign Statment
  /adderTest/ccao   St1
  /adderTest/ssa1   St0
  /adderTest/ssa0   St1
```



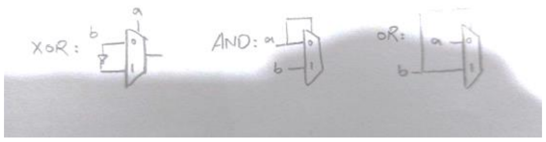
🟥 S0 Longest Path : 14 + 10 + 14 + 10 + 14 + 10 = 72ns

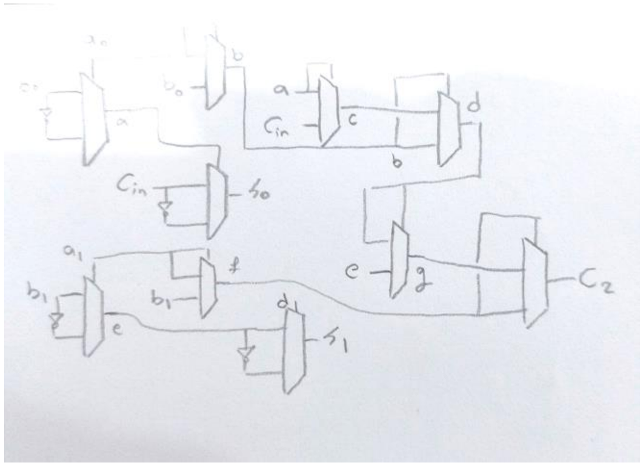
🟩 S1 Longest Path : 14 + 10 + 14 + 10 + 14 + 10 +14+10 = 96ns

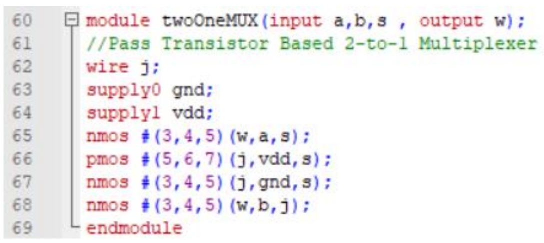
🟨 C2 Longest Path : 14 + 10 + 14 + 10 + 14 + 10 +14= 86ns

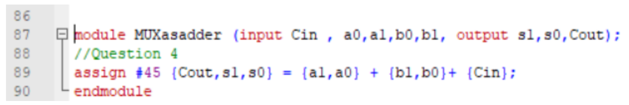
3) for implimenting this circuit with MUX we can first impliment AND, OR, XOR with MUX and then it will be easy.

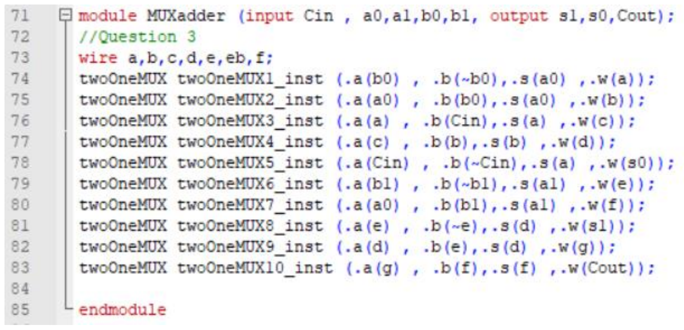
5)

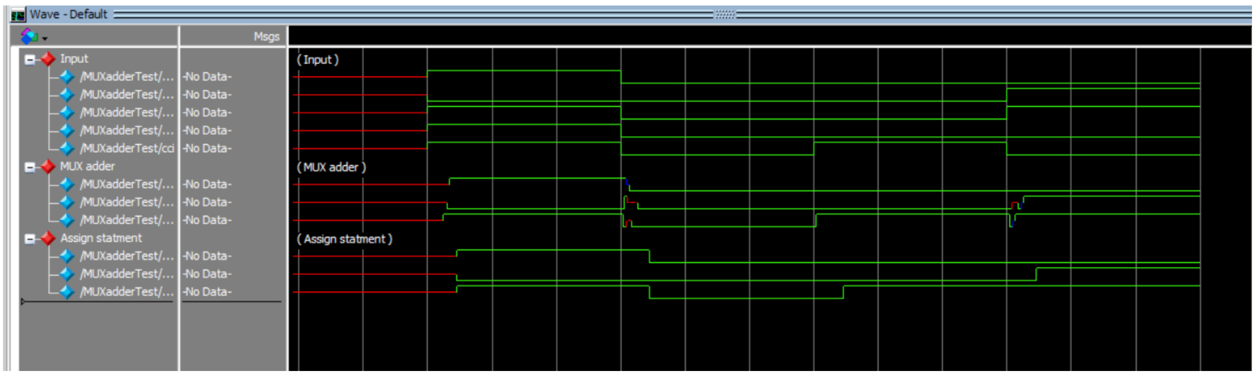




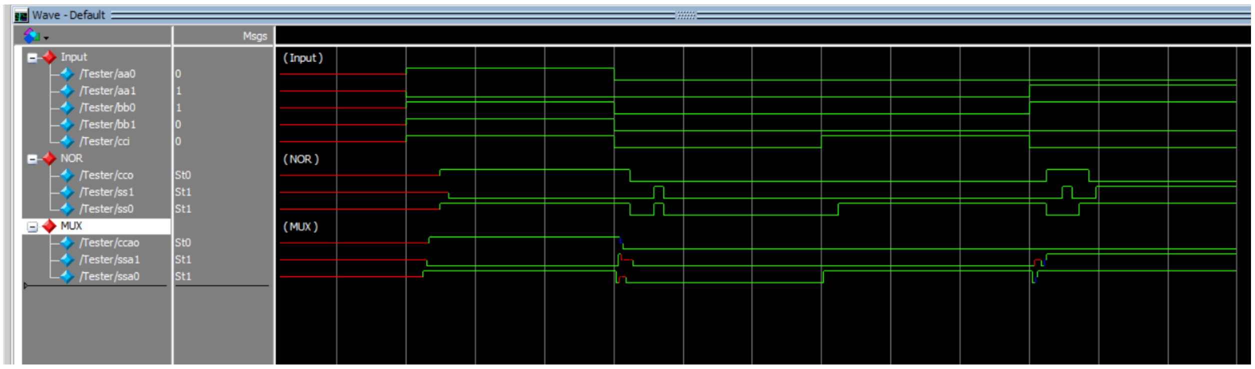

```
module twoOneMUX(input a,b,s , output w);
//Pass Transistor Based 2-to-1 Multiplexer
wire j;
supply0 gnd;
supply1 vdd;
nmos #(3,4,5)(w,a,s);
pmos #(5,6,7)(j,vdd,s);
nmos #(3,4,5)(j,gnd,s);
nmos #(3,4,5)(w,b,j);
endmodule
```


```

module MUXasadder (input Cin , a0,a1,b0,b1, output s1,s0,Cout);
//Question 4
assign #45 {Cout,s1,s0} = {a1,a0} + {b1,b0}+ {Cin};
endmodule
```

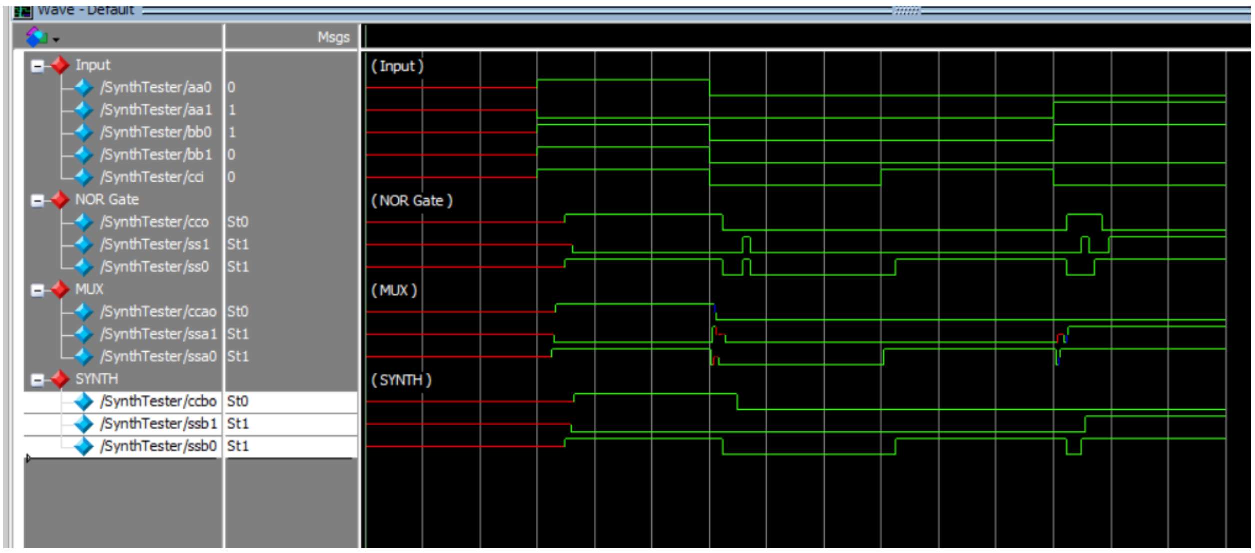

```
module MUXadder (input Cin , a0,a1,b0,b1, output s1,s0,Cout);
//Question 3
wire a,b,c,d,e,eb,f;
twoOneMUX twoOneMUX1_inst (.a(b0) , .b(~b0),.s(a0) ,.w(a));
twoOneMUX twoOneMUX2_inst (.a(a0) , .b(b0),.s(a0) ,.w(b));
twoOneMUX twoOneMUX3_inst (.a(a) , .b(Cin),.s(a) ,.w(c));
twoOneMUX twoOneMUX4_inst (.a(c) , .b(b),.s(b) ,.w(d));
twoOneMUX twoOneMUX5_inst (.a(Cin) , .b(~Cin),.s(a) ,.w(s0));
twoOneMUX twoOneMUX6_inst (.a(b1) , .b(~b1),.s(a1) ,.w(e));
twoOneMUX twoOneMUX7_inst (.a(a0) , .b(b1),.s(a1) ,.w(f));
twoOneMUX twoOneMUX8_inst (.a(e) , .b(~e),.s(d) ,.w(s1));
twoOneMUX twoOneMUX9_inst (.a(d) , .b(e),.s(d) ,.w(g));
twoOneMUX twoOneMUX10_inst (.a(g) , .b(f),.s(f) ,.w(Cout));

endmodule
```


Wave - Default
Msgs
Input
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
/MUXadderTest/cci -No Data-
MUX adder
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
Assign statment
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
/MUXadderTest/... -No Data-
(Input)
(MUX adder)
(Assign statment)

5)



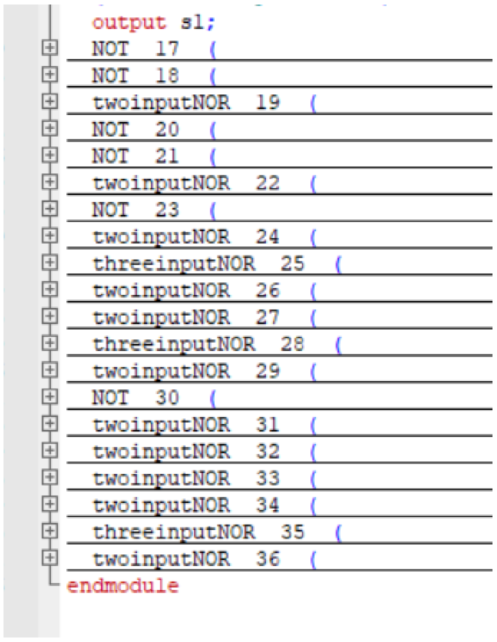
7)



11 Two input NOR Gates

3 Three Input NOR Gates

6 Invertors



```
  output s1;
  NOT  17  (
  NOT  18  (
  twoinputNOR  19  (
  NOT  20  (
  NOT  21  (
  twoinputNOR  22  (
  NOT  23  (
  twoinputNOR  24  (
  threeinputNOR  25  (
  twoinputNOR  26  (
  twoinputNOR  27  (
  threeinputNOR  28  (
  twoinputNOR  29  (
  NOT  30  (
  twoinputNOR  31  (
  twoinputNOR  32  (
  twoinputNOR  33  (
  twoinputNOR  34  (
  threeinputNOR  35  (
  twoinputNOR  36  (
endmodule
```