SOC LAB

1. Full Adder

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a,b,cin,s,co;

    void mon()

    {

        cout<<"Inputs: "<<a<<b<<cin<<"Output: "<<s<<co<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a<<b<<cin;

    }

};

//main.cpp

#include <systemc.h>

#include "fulladder.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a,b,s,co,cin;

    fulladder fa("fulladder");

    driver drive("driver");

    monitor mon("monitor");

    fa.a(a); fa.b(b); fa.s(s); fa.cin(cin); fa.co(co);

    drive.a(a); drive.b(b);drive.cin(cin);

    mon.a(a); mon.b(b); mon.cin(cin); mon.s(s);mon.co(co);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a, "a");

    sc\_trace(Tf, b, "b");

    sc\_trace(Tf, cin, "cin");

    sc\_trace(Tf, s, "s");

    sc\_trace(Tf, co, "co");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//adder.h

#include <systemc.h>

SC\_MODULE(fulladder)

{

    sc\_in<bool> a;

    sc\_in<bool> b;

    sc\_in<bool> cin;

    sc\_out<bool> s;

    sc\_out<bool> co;

    void compute\_fa()

    {

        s.write(a.read() ^ b.read() ^ cin.read());

        co.write((a.read() & b.read()) | cin.read() & (a.read() ^ b.read()));

    }

    SC\_CTOR(fulladder)

    {

        SC\_METHOD(compute\_fa);

        sensitive<<a<<b<<cin;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a,b,cin;

    void inputs()

    {

        a.write(false);

        b.write(false);

        cin.write(false);

        wait(5, SC\_NS);

        a.write(true);

        b.write(true);

        cin.write(false);

        wait(5, SC\_NS);

        a.write(false);

        b.write(false);

        cin.write(true);

        wait(5, SC\_NS);

        a.write(true);

        b.write(true);

        cin.write(true);

        wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

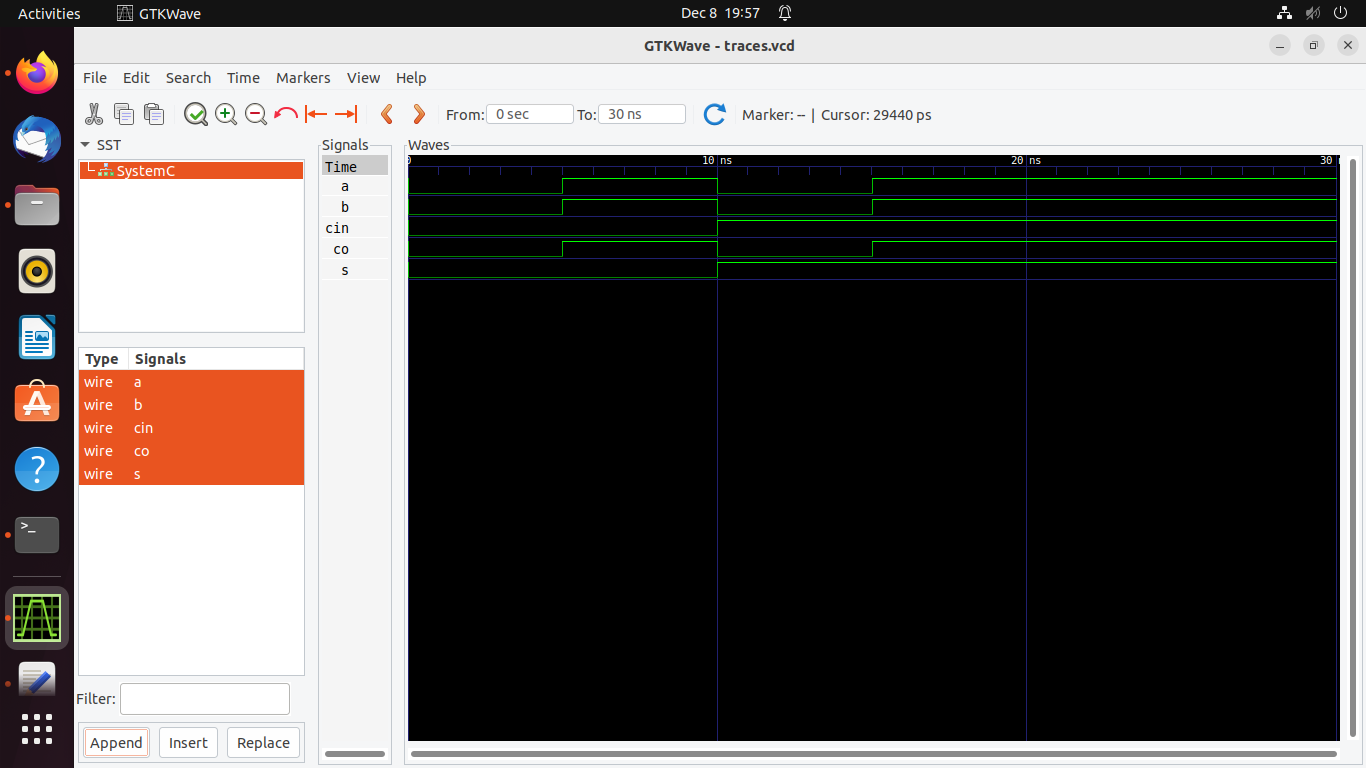
            SC\_THREAD(inputs);

            sensitive <<a <<b<<cin;

        }

};

Output:



1. Half adder

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a,b,s,c;

    void mon()

    {

        cout<<"Inputs: "<<a<<b<<"Output: "<<s<<c<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a<<b;

    }

};

#include <systemc.h>

#include "halfadder.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a,b,s,c;

    halfadder ha("halfadder");

    driver drive("driver");

    monitor mon("monitor");

    ha.a(a); ha.b(b); ha.s(s); ha.c(c);

    drive.a(a); drive.b(b);

    mon.a(a); mon.b(b); mon.c(c); mon.s(s);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a, "a");

    sc\_trace(Tf, b, "b");

    sc\_trace(Tf, c, "c");

    sc\_trace(Tf, s, "s");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//halfadder.h

#include <systemc.h>

SC\_MODULE(halfadder)

{

    sc\_in<bool> a;

    sc\_in<bool> b;

    sc\_out<bool> s;

    sc\_out<bool> c;

    void compute\_ha()

    {

        s.write(a.read() ^ b.read());

        c.write(a.read() & b.read());

    }

    SC\_CTOR(halfadder)

    {

        SC\_METHOD(compute\_ha);

        sensitive<<a<<b;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a,b;

    void inputs()

    {

        a.write(false);

        b.write(false);

        wait(5, SC\_NS);

        a.write(false);

        b.write(true);

        wait(5, SC\_NS);

        a.write(true);

        b.write(false);

        wait(5, SC\_NS);

        a.write(true);

        b.write(true);

        wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

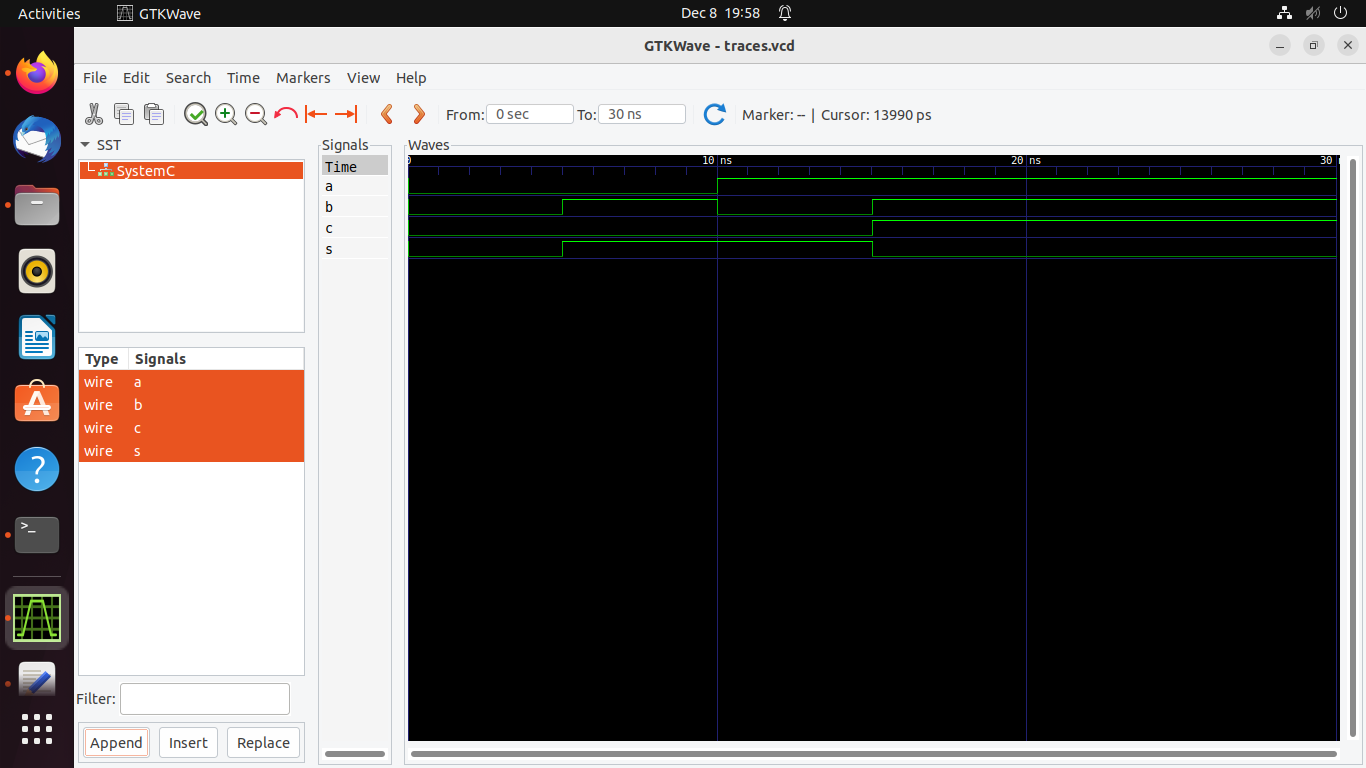
            SC\_THREAD(inputs);

            sensitive <<a <<b;

        }

};

Output:



1. Counters Ascyn

//andgate.h

#include <systemc.h>

SC\_MODULE(dff)

{

    sc\_in<bool> d;

    sc\_out<bool> q;

    sc\_in<bool> clk;

    void compute\_dff()

    {

        q.write(d.read());

    }

    SC\_CTOR(dff)

    {

        SC\_METHOD(compute\_dff);

        sensitive<<clk.pos();

    }

};

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> d,q;

    sc\_in<bool> clk;

    void mon()

    {

        cout<<"Input D: "<<d<<"Output Q:= "<<q<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<clk.pos();

    }

};

#include <systemc.h>

#include "dff.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> d,q;

    sc\_clock  testclk("testclock",10,SC\_NS,0.5);

    dff dff1("dff");

    driver drive("driver");

    monitor mon("monitor");

    dff1.d(d); dff1.q(q); dff1.clk(testclk);

    drive.d(d); drive.clk(testclk);

    mon.d(d); mon.q(q); mon.clk(testclk);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, d, "d");

    sc\_trace(Tf, q, "q");

    sc\_trace(Tf,testclk,"testclk");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> d;

    sc\_in<bool> clk;

    void inputs()

    {

        wait();

        d.write(false);

        wait();

        d.write(true);

        wait();

    }

        SC\_CTOR(driver)

        {

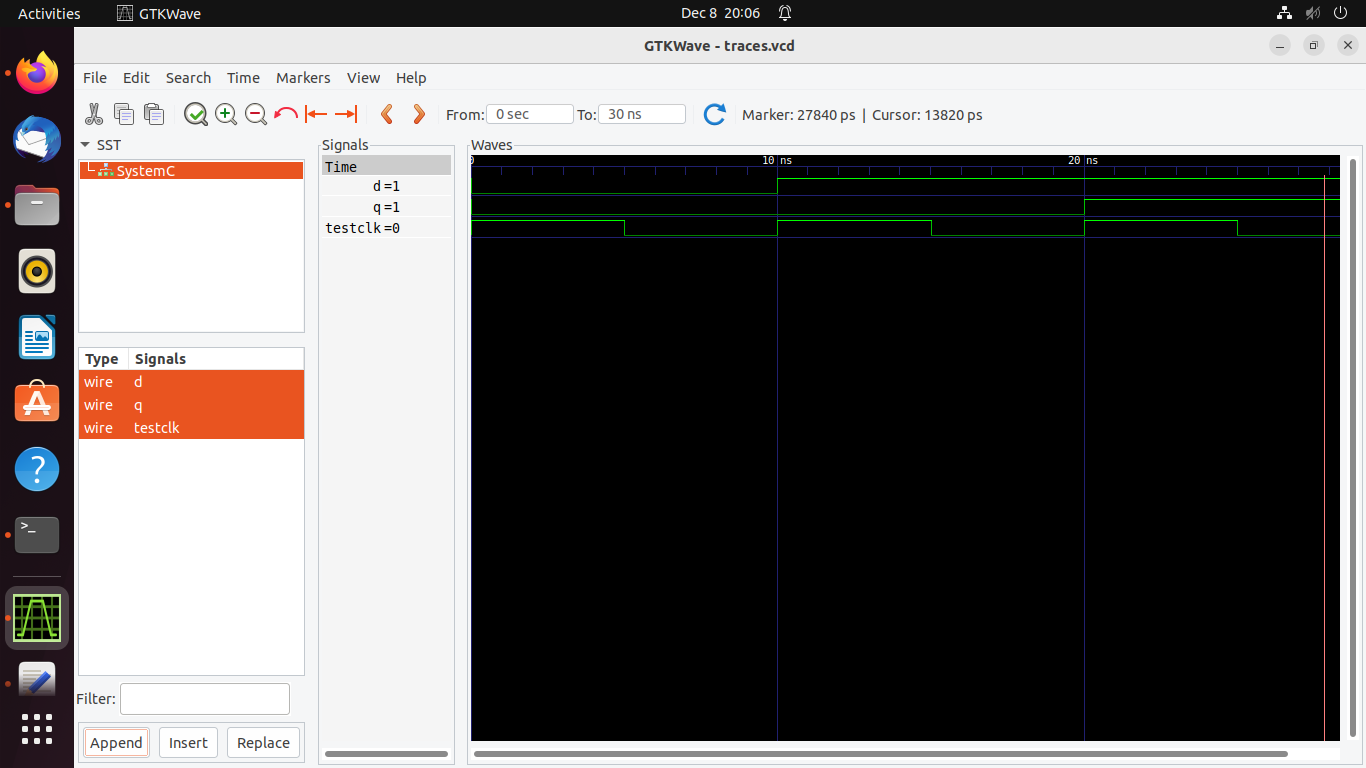
            SC\_THREAD(inputs);

            sensitive <<clk.pos();

        }

};

Output:



1. Sync Counter

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> rst,en;

    sc\_in<bool> clk;

    void inputs()

    {

        rst.write(true);

        wait(1,SC\_NS);

        rst.write(false);

        en.write(true);

        wait(1,SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<clk.pos()<<rst<<en;

        }

};

//andgate.h

#include <systemc.h>

SC\_MODULE(upcounter)

{

    sc\_in<bool> rst,en;

    sc\_out<sc\_uint<8>> counter\_out;

    sc\_in<bool> clk;

    sc\_uint<8>  count;

    void compute\_upcounter()

    {

        if (rst.read() == 1)

        {

            count =  0;

            counter\_out.write(count);

            }

            else if (en.read() == 1)

             {

              count = count + 1;

                  counter\_out.write(count);

         }

    }

    SC\_CTOR(upcounter)

    {

        SC\_METHOD(compute\_upcounter);

        sensitive<<clk.pos()<<rst;

    }

};

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> rst,en;

    sc\_in<sc\_uint<8>> counter\_out;

    sc\_in<bool> clk;

    void mon()

    {

        cout<<"Input rst: "<<rst<<"Output counter\_out:= "<<counter\_out<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<clk.pos()<<rst;

    }

};

#include <systemc.h>

#include "sync\_up.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> rst,en;

    sc\_signal<sc\_uint<8>> counter\_out;

    sc\_clock  testclk("testclock",2,SC\_NS,0.5);

    upcounter up("upcounter");

    driver drive("driver");

    monitor mon("monitor");

    up.rst(rst); up.en(en); up.clk(testclk);up.counter\_out(counter\_out);

    drive.rst(rst);drive.en(en); drive.clk(testclk);

    mon.rst(rst); mon.en(en); mon.clk(testclk);mon.counter\_out(counter\_out);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, rst, "rst");

    sc\_trace(Tf, en, "en");

    sc\_trace(Tf, counter\_out, "counter\_out");

    sc\_trace(Tf,testclk,"testclk");

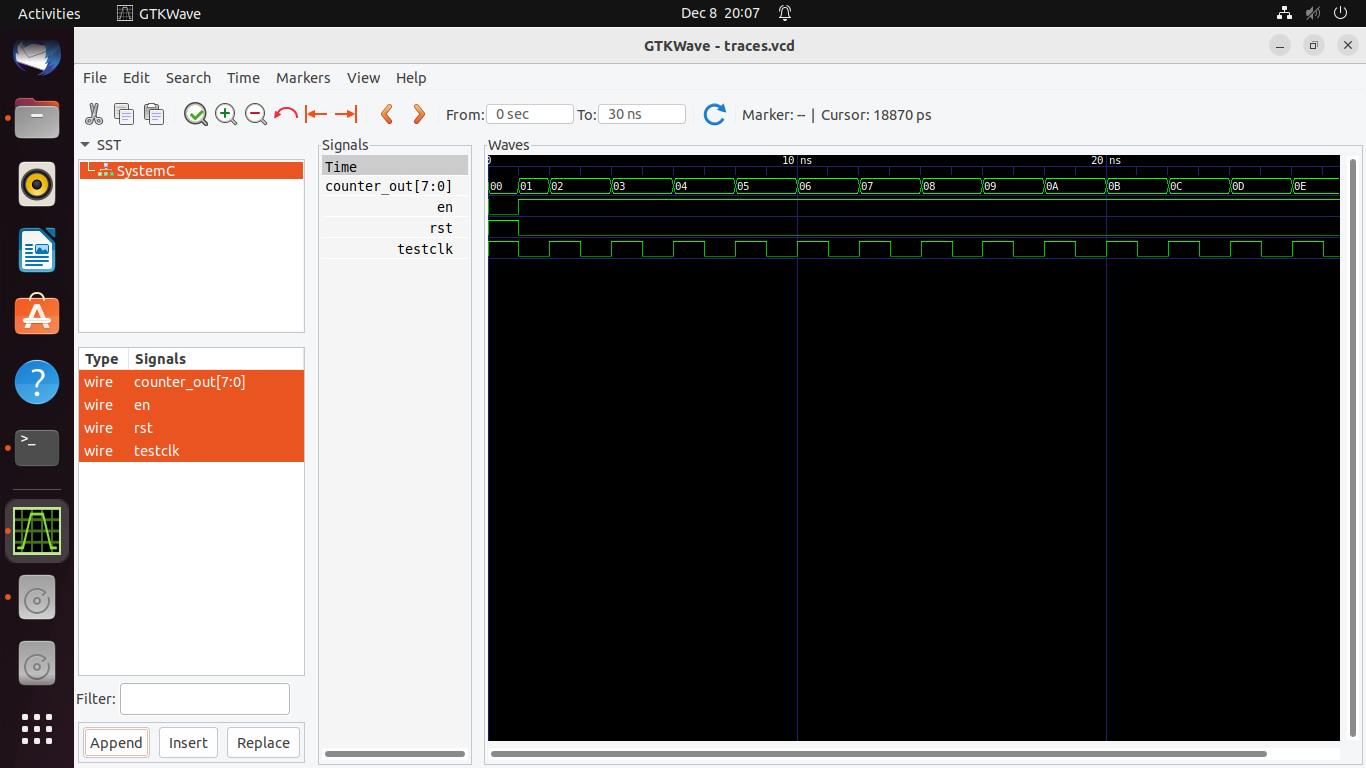
    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

Output:



1. Decoder

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> d0,d1,d2,d3,en,s0,s1;

    void mon()

    {

        cout<<"Inputs: "<<en<<s0<<s1<<"Output: "<<d0<<d1<<d2<<d3<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<en<<s0<<s1;

    }

};

#include <systemc.h>

#include "decoder.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> d0,d1,d2,d3,en,s0,s1;

    decoder dec1("decoder");

    driver drive("driver");

    monitor mon("monitor");

    dec1.d0(d0); dec1.d1(d1); dec1.d2(d2); dec1.d3(d3); dec1.en(en); dec1.s0(s0); dec1.s1(s1);

    drive.en(en); drive.s0(s0); drive.s1(s1);

    mon.d0(d0); mon.d1(d1); mon.d2(d2); mon.d3(d3); mon.en(en); mon.s0(s0); mon.s1(s1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, s0, "s0");

    sc\_trace(Tf, s1, "s1");

    sc\_trace(Tf, en, "en");

    sc\_trace(Tf, d0, "d0");

    sc\_trace(Tf, d1, "d1");

    sc\_trace(Tf, d2, "d2");

    sc\_trace(Tf, d3, "d3");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//decoder.h

#include <systemc.h>

SC\_MODULE(decoder)

{

    sc\_in<bool> s0;

    sc\_in<bool> s1;

    sc\_in<bool> en;

    sc\_out<bool> d0;

    sc\_out<bool> d1;

    sc\_out<bool> d2;

    sc\_out<bool> d3;

    void compute\_decoder()

    {

        if(en.read()==true){

            if( (s1.read()==false) && (s0.read()==false))

            {

                d0.write(true);

                d1.write(false);

                d2.write(false);

                d3.write(false);

            }

            else if((s1.read()==false) && (s0.read()==true))

            {

                d0.write(false);

                d1.write(true);

                d2.write(false);

                d3.write(false);

            }

            else if( (s1.read()==true) && (s0.read()==false))

            {

                d0.write(false);

                d1.write(false);

                d2.write(true);

                d3.write(false);

            }

            else if( (s1.read()==true) && (s0.read()==true))

            {

                d0.write(false);

                d1.write(false);

                d2.write(false);

                d3.write(true);

            }

        }

        else if(en.read()==false)

        {

            d0.write(false);

            d1.write(false);

            d2.write(false);

            d3.write(false);

        }

    }

    SC\_CTOR(decoder)

    {

        SC\_METHOD(compute\_decoder);

        sensitive<<s0<<s1<<en;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> s0,s1,en;

    void inputs()

    {

        s0.write(false);

        s1.write(false);

        en.write(true);

        wait(5, SC\_NS);

        s0.write(true);

        s1.write(false);

        en.write(true);

        wait(5, SC\_NS);

        s0.write(false);

        s1.write(true);

        en.write(false);

        wait(5, SC\_NS);

        s0.write(true);

        s1.write(true);

        en.write(true);

        wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

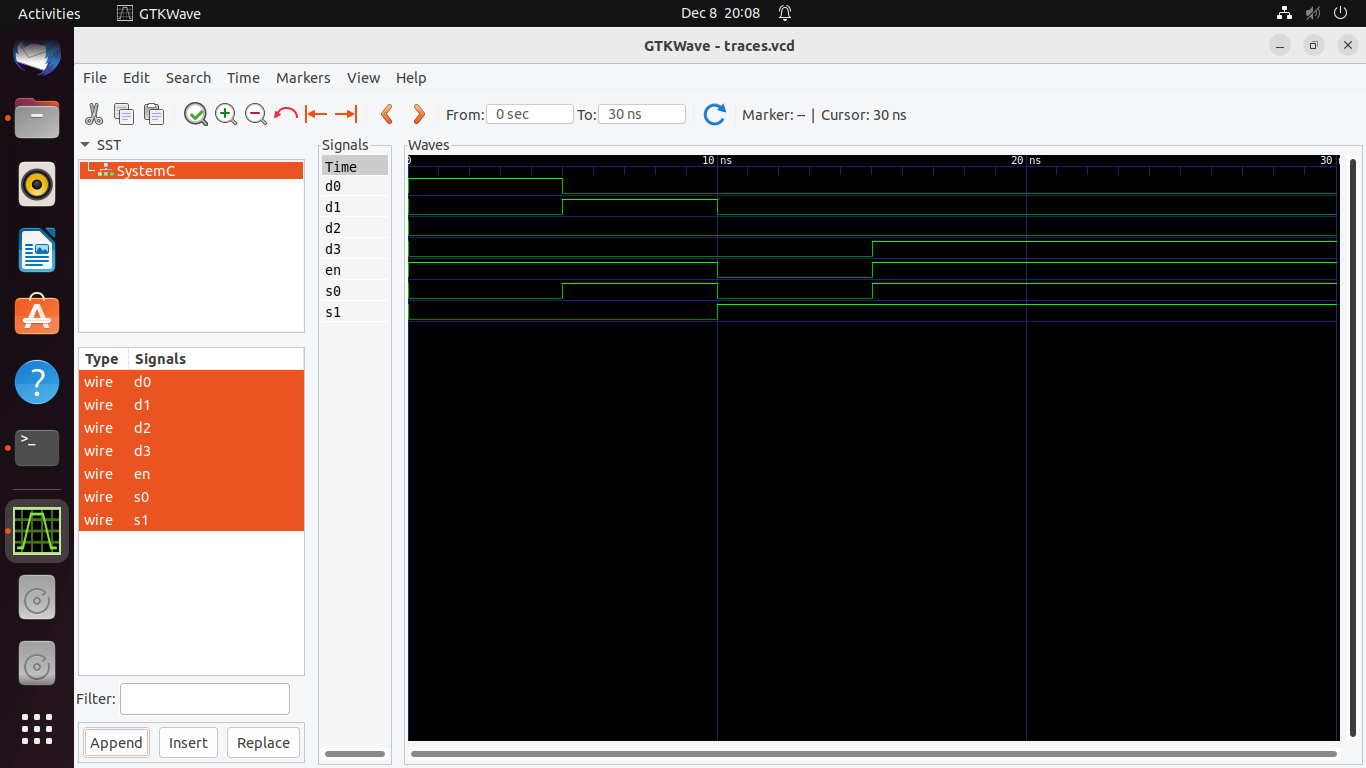
            SC\_THREAD(inputs);

            sensitive <<s0<<s1<<en;

        }

};

Output:



1. Encoder

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<sc\_lv<4>> d;

    sc\_in<sc\_lv<2>> y;

    void mon()

    {

        cout<<"Inputs: "<<d<<"Output: "<<y<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<d;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<sc\_lv<4>> d;

    void inputs()

    {

        d.write("0001");

        wait(5, SC\_NS);

        d.write("0010");

        wait(5, SC\_NS);

        d.write("1000");

        wait(5, SC\_NS);

        d.write("0000");

        wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<d;

        }

};

//encoder.h

#include <systemc.h>

SC\_MODULE(encoder)

{

    sc\_in<sc\_lv<4>> d;

    sc\_out<sc\_lv<2>> y;

    void compute\_encoder()

    {

            if( d.read()=="0001")

            {

                y.write("00");

            }

            else if( d.read()=="0010")

            {

                y.write("01");

            }

            else if( d.read()=="0100")

            {

                y.write("10");

            }

            else if( d.read()=="1000")

            {

                y.write("11");

            }

            else if( d.read()=="0000")

            {

                y.write("XX");

            }

    }

    SC\_CTOR(encoder)

    {

        SC\_METHOD(compute\_encoder);

        sensitive<<d;

    }

};

#include <systemc.h>

#include "encoder.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<sc\_lv<4>> d;

    sc\_signal<sc\_lv<2>> y;

    encoder enc("encoder");

    driver drive("driver");

    monitor mon("monitor");

    enc.d(d); enc.y(y);

    drive.d(d);

    mon.d(d); mon.y(y);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, d, "d");

    sc\_trace(Tf, y, "y");

    /\*sc\_trace(Tf, en, "en");

    sc\_trace(Tf, d0, "d0");

    sc\_trace(Tf, d1, "d1");

    sc\_trace(Tf, d2, "d2");

    sc\_trace(Tf, d3, "d3");\*/

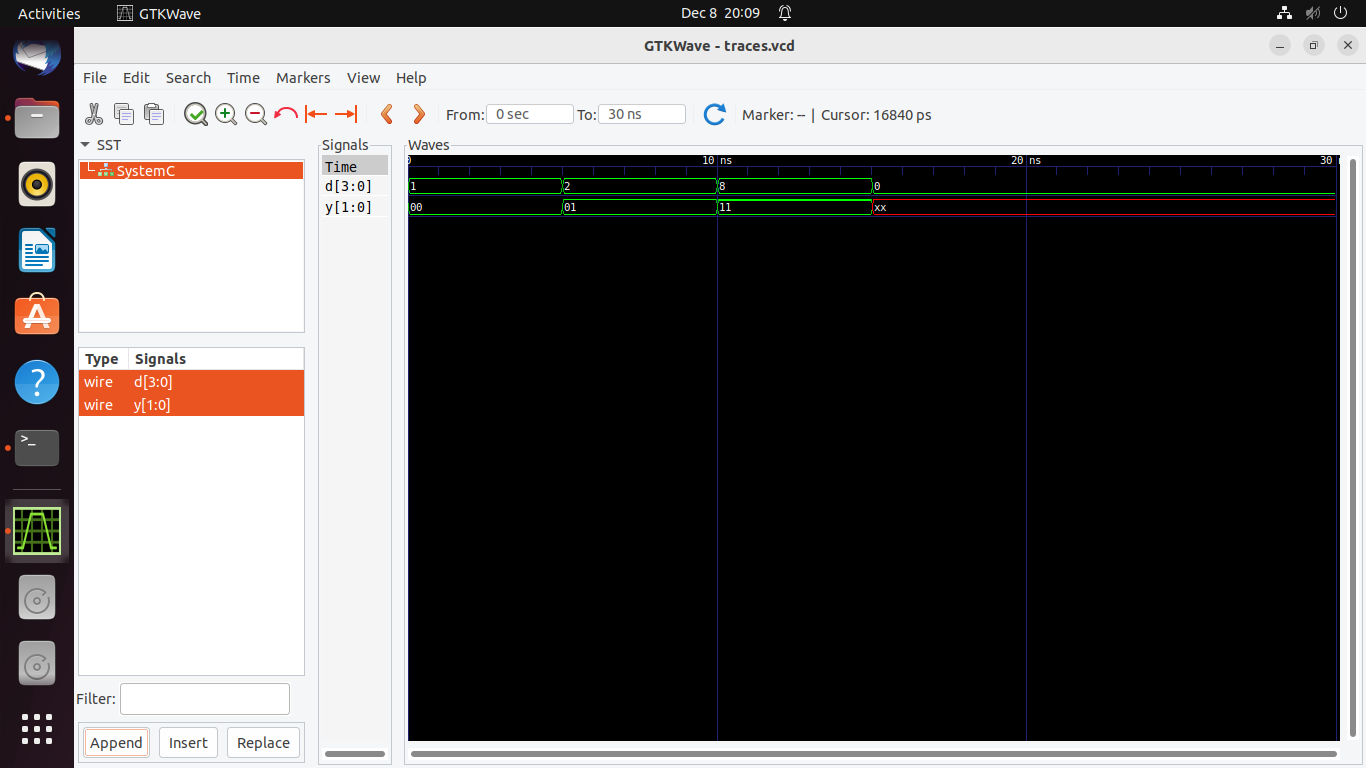
    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

Output:



1. FIFO

#include "systemc.h"

#include "fifo.h"

int sc\_main(int argc, char\* argv[])

{

    example\_fifo ex\_fifo("ex\_fifo0");

    sc\_start(10, SC\_NS);

    return 0;

}

#include "systemc.h"

SC\_MODULE(example\_fifo) {

// Declare the FIFO

sc\_fifo<int> packet\_fifo;

void fiforead(void) {

int val;

for (;;){

    wait(3, SC\_NS);

    packet\_fifo.read(val);

    cout<< sc\_time\_stamp() << ": Read " << val<<endl;

    cout<< sc\_time\_stamp()<<"No.of Data in FIFO "<<packet\_fifo.num\_available()<<endl;

    }

}

void fifowrite(void) {

int val= 10;

for (;;) {

    wait(2, SC\_NS);

    val++;

    packet\_fifo.write(val);

    cout<< sc\_time\_stamp() << ": wrote " << val<< "  No. of Free Slots in FIFO "<<packet\_fifo.num\_free()<<endl;

    }

}

SC\_CTOR(example\_fifo) :packet\_fifo(10)

{

    SC\_THREAD(fifowrite);

    SC\_THREAD(fiforead);

    }

};

1. D flipflop

//dff.h

#include <systemc.h>

SC\_MODULE(dff)

{

    sc\_in<bool> d;

    sc\_out<bool> q;

    sc\_in<bool> clk;

    void compute\_dff()

    {

        q.write(d.read());

    }

    SC\_CTOR(dff)

    {

        SC\_METHOD(compute\_dff);

        sensitive<<clk.pos();

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> d;

    sc\_in<bool> clk;

    void inputs()

    {

        wait();

        d.write(false);

        wait();

        d.write(true);

        wait();

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<clk.pos();

        }

};

#include <systemc.h>

#include "dff.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> d,q;

    sc\_clock  testclk("testclock",10,SC\_NS,0.5);

    dff dff1("dff");

    driver drive("driver");

    monitor mon("monitor");

    dff1.d(d); dff1.q(q); dff1.clk(testclk);

    drive.d(d); drive.clk(testclk);

    mon.d(d); mon.q(q); mon.clk(testclk);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, d, "d");

    sc\_trace(Tf, q, "q");

    sc\_trace(Tf,testclk,"testclk");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> d,q;

    sc\_in<bool> clk;

    void mon()

    {

        cout<<"Input D: "<<d<<"Output Q:= "<<q<<endl;

    }

    SC\_CTOR(monitor)

    {

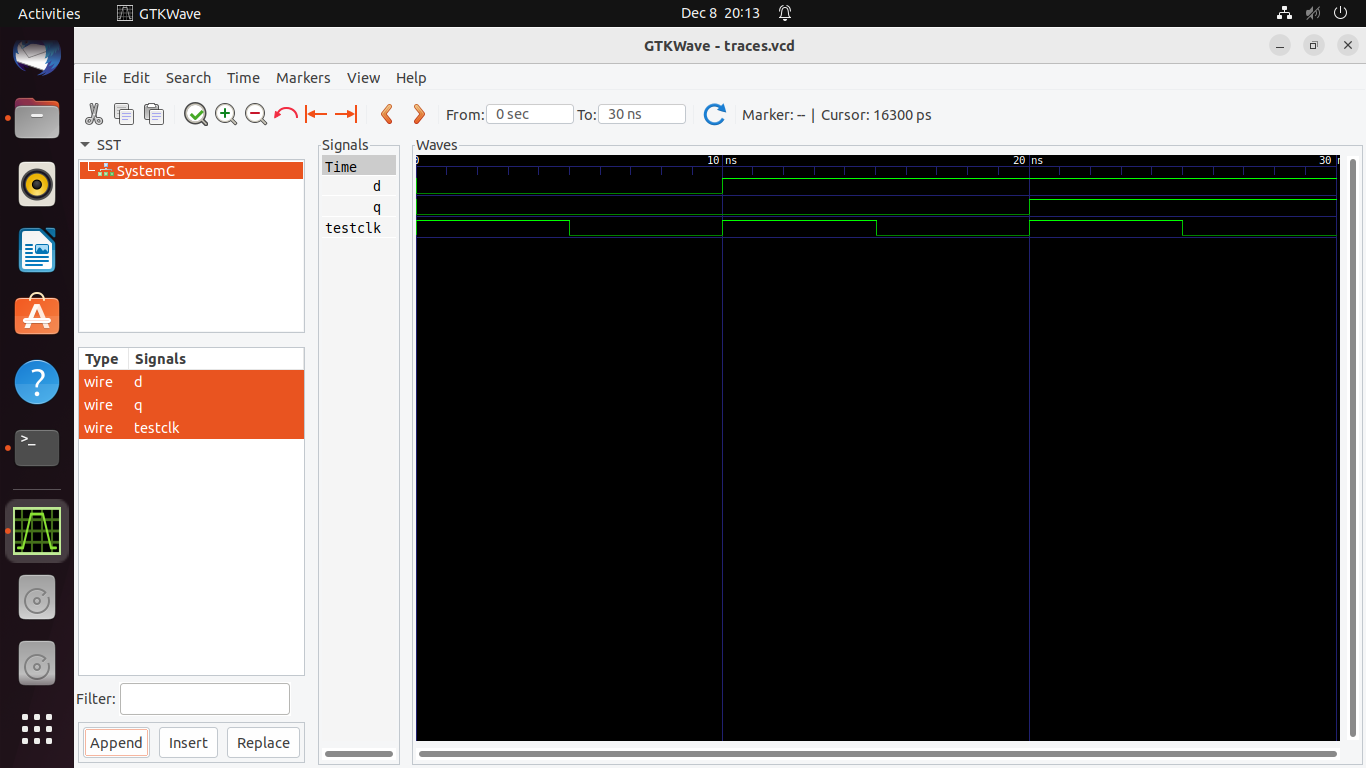
        SC\_METHOD(mon);

        sensitive<<clk.pos();

    }

};

Output:



1. T flipflop

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> t,q;

    sc\_in<bool> clk;

    void mon()

    {

        cout<<"Input T: "<<t<<"Output Q:= "<<q<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<clk.pos();

    }

};

//andgate.h

#include <systemc.h>

SC\_MODULE(tff)

{

    sc\_in<bool> t;

    sc\_out<bool> q;

    sc\_in<bool> clk;

    void compute\_tff()

    {

        q.write(!(t.read()));

    }

    SC\_CTOR(tff)

    {

        SC\_METHOD(compute\_tff);

        sensitive<<clk.pos();

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> t;

    sc\_in<bool> clk;

    void inputs()

    {

        wait();

        t.write(false);

        wait();

        t.write(true);

        wait();

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<clk.pos();

        }

};

#include <systemc.h>

#include "tff.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> t,q;

    sc\_clock  testclk("testclock",10,SC\_NS,0.5);

    tff tff1("tff");

    driver drive("driver");

    monitor mon("monitor");

    tff1.t(t); tff1.q(q); tff1.clk(testclk);

    drive.t(t); drive.clk(testclk);

    mon.t(t); mon.q(q); mon.clk(testclk);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, t, "t");

    sc\_trace(Tf, q, "q");

    sc\_trace(Tf,testclk,"testclk");

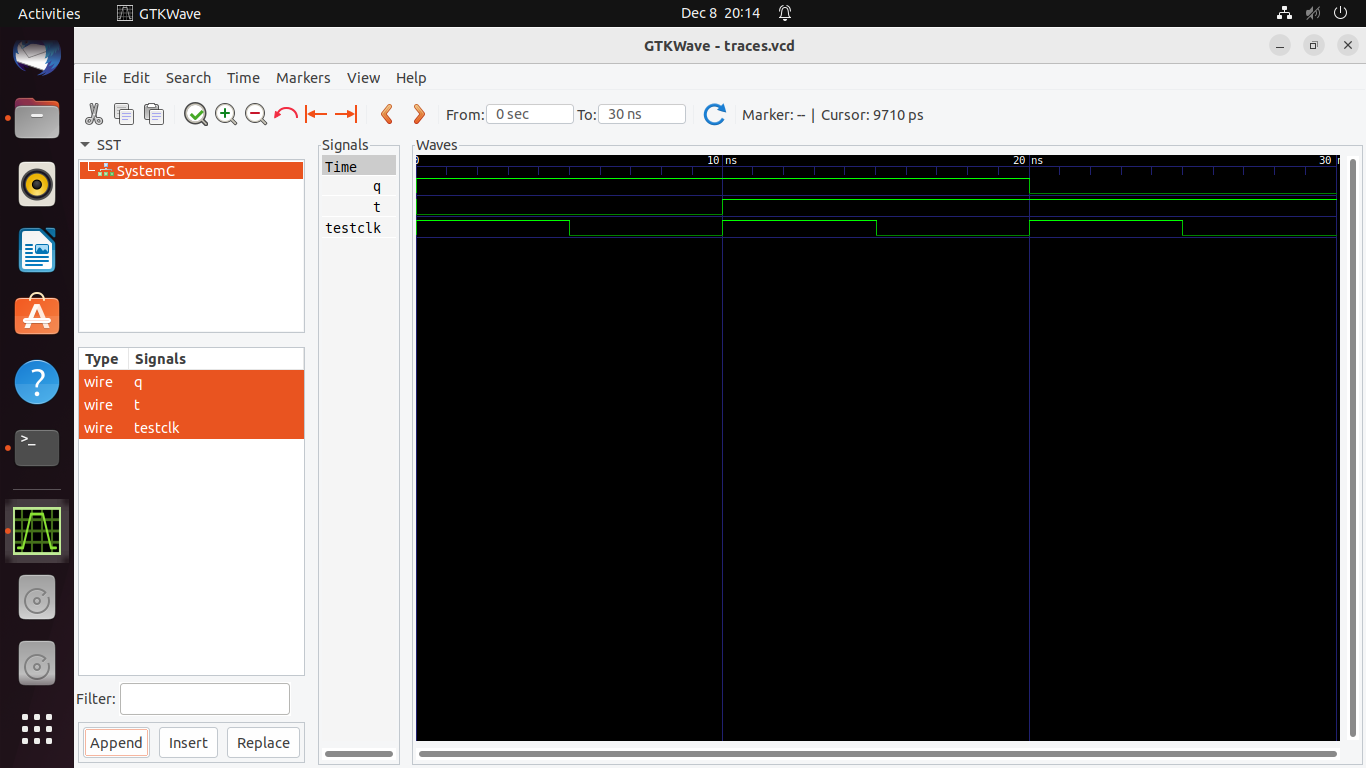
    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

Output:



1. FSM

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<sc\_lv<4>> d;

    void inputs()

    {

        d.write("0001");

        wait(5, SC\_NS);

        d.write("0010");

        wait(5, SC\_NS);

        d.write("1000");

        wait(5, SC\_NS);

        d.write("0000");

        wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<d;

        }

};

case 1:

if (

in.read

next\_state

= 1;out=

else{

next\_state

= 2;out=

break;

case 2:

if (

in.read

next\_state

= 1;out=

else{

next\_state

= 0;out=

break;

default:

{

next\_state

= 0;out=

break;

}

}

SC\_CTOR(

fsm

SC\_METHOD(

update\_state

sensitive <<

clk.pos rst

SC\_METHOD(

ns\_logic

sensitive <<

clk.pos ()<< rst

}};

rst.write

(

in.write

(

wait();

in.write

(

wait();

in.write

(

wait();

in.write

(

wait();

}

SC\_CTOR(driver)

{

SC\_THREAD(inputs);

sensitive <<

clk.pos

}

};

#include "

systemc.h

SC\_MODULE(driver)

{

sc\_out

bool rst,in

sc\_in

bool clk

void inputs()

{

rst.write

(

in.write

(

wait();

rst.write

(

in.write

(

wait();

#include "

systemc.h

SC\_MODULE(monitor)

{

sc\_in

bool > rst , in,out

sc\_in

bool > clk

void mon()

{

while (true){

cout

<< sc\_time\_stamp

cout

<< " rst rst.read

cout

<< " clk clk.read

cout

<< " in in.read

cout

<< " out out.read () endl

wait();

}}

SC\_CTOR(monitor)

{

SC\_THREAD(mon);

sensitive <<

clk.pos

}};

#include "

systemc.h

#include "

driver.h

#include "

monitor.h

#include "

fsm.h

int

sc\_main int argc , char\*

argv

{

sc\_signal

bool > rst,in,out

sc\_clock

clk ("TestClock",10,SC\_NS,

fsm

f\_sm state\_machine

driver drive("driver");

monitor mon("monitor");

f\_sm.rst

rst

f\_sm.in(in);

f\_sm.clk

clk

f\_sm.out

(

drive.rst

rst

drive.in(in);

drive.clk

clk

mon.rst

rst

mon.in(in);

mon.out

(

mon.clk

clk

//Waveform

sc\_trace\_file

Tf

Tf

=

sc\_create\_vcd\_trace\_file ("traces

sc\_trace

Tf , rst , rst

sc\_trace

Tf , in, "

sc\_trace

Tf , clk , clk

sc\_trace

Tf , out, "

sc\_trace

Tf , f\_sm.state ," sc\_start (100,SC\_ sc\_close\_vcd\_trace\_file Tf); return 0;

1. Mux

#include <systemc.h>

#include "mux.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a,b,c,d,f,s0,s1;

    mux mux1("mux");

    driver drive("driver");

    monitor mon("monitor");

    mux1.a(a); mux1.b(b); mux1.f(f); mux1.c(c); mux1.d(d); mux1.s0(s0); mux1.s1(s1);

    drive.a(a); drive.b(b);  drive.c(c); drive.d(d); drive.s0(s0); drive.s1(s1);

    mon.a(a); mon.b(b); mon.f(f); mon.c(c); mon.d(d); mon.s0(s0); mon.s1(s1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a, "a");

    sc\_trace(Tf, b, "b");

    sc\_trace(Tf, c, "c");

    sc\_trace(Tf, d, "d");

    sc\_trace(Tf, f, "f");

    sc\_trace(Tf, s0, "s0");

    sc\_trace(Tf, s1, "s1");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//mux.h

#include <systemc.h>

SC\_MODULE(mux)

{

    sc\_in<bool> s0;

    sc\_in<bool> s1;

    sc\_in<bool> a;

    sc\_in<bool> b;

    sc\_in<bool> c;

    sc\_in<bool> d;

    sc\_out<bool> f;

    void compute\_mux()

    {

        if((s1.read()==false) && (s0.read()==false))

        {

            f.write(a.read());

        }

        else if((s1.read()==false) && (s0.read()==true))

        {

            f.write(b.read());

        }

        if((s1.read()==true) && (s0.read()==false))

        {

            f.write(c.read());

        }

        if((s1.read()==true) && (s0.read()==true))

        {

            f.write(d.read());

        }

    }

    SC\_CTOR(mux)

    {

        SC\_METHOD(compute\_mux);

        sensitive<<s0<<s1<<a<<b<<c<<d;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a,b,c,d,s0,s1;

    void inputs()

    {

        a.write(true);

        b.write(false);

        c.write(true);

        d.write(false);

        s0.write(false);

        s1.write(false);wait(5, SC\_NS);

        a.write(true);

        b.write(false);

        c.write(true);

        d.write(false);

        s0.write(true);

        s1.write(false);wait(5, SC\_NS);

        a.write(true);

        b.write(false);

        c.write(true);

        d.write(false);

        s0.write(false);

        s1.write(true);wait(5, SC\_NS);

        a.write(true);

        b.write(false);

        c.write(true);

        d.write(false);

        s0.write(true);

        s1.write(true);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a<<b<<c<<d<<s0<<s1;

        }

};

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a,b,c,d,f,s0,s1;

    void mon()

    {

        cout<<"Inputs: "<<a<<b<<c<<d<<s0<<s1<<"Output: "<<f<<endl;

    }

    SC\_CTOR(monitor)

    {

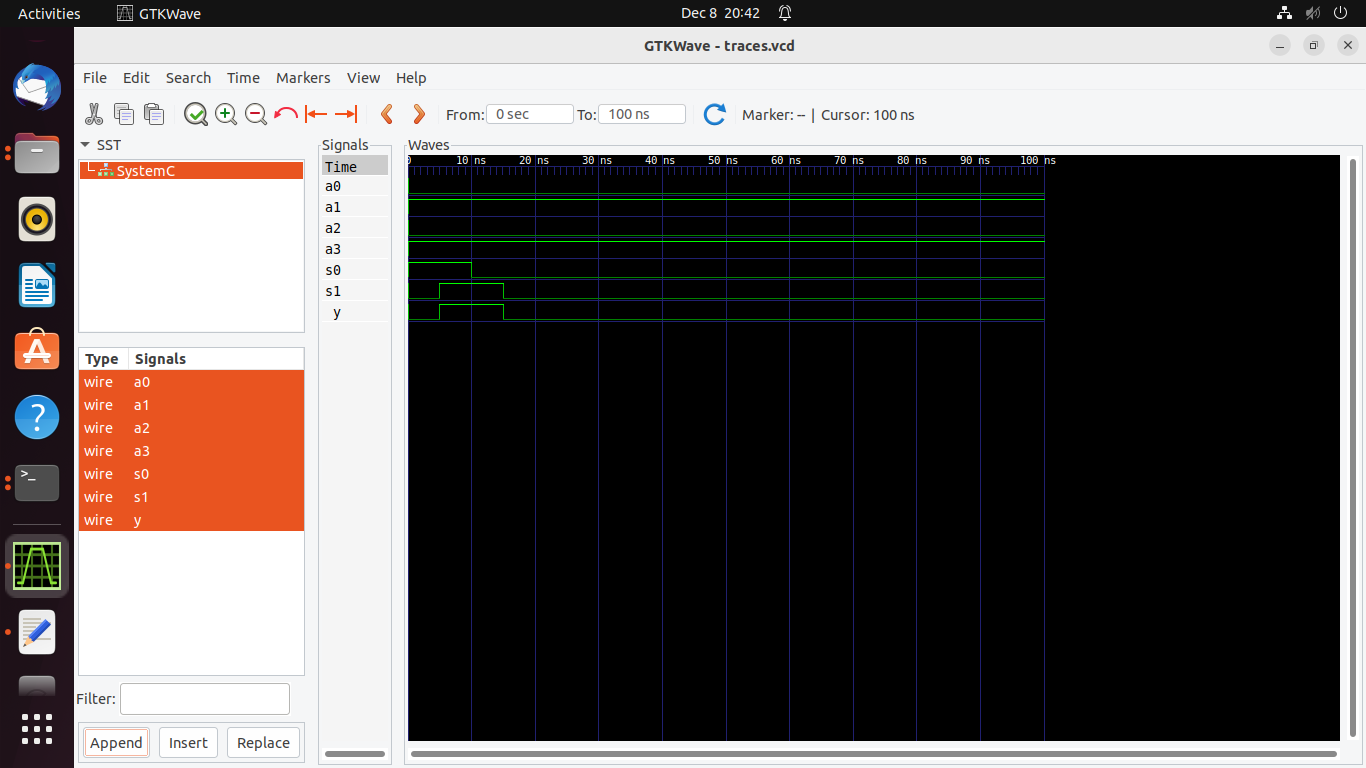
        SC\_METHOD(mon);

        sensitive<<a<<b<<c<<d<<s0<<s1;

    }

};

Output:



1. SISO

//siso.h

#include <systemc.h>

SC\_MODULE(siso)

{

    sc\_in<sc\_logic> d;

    sc\_out<sc\_logic> r;

    sc\_in<bool> clk;

    sc\_lv<3> temp;

    void compute\_siso()

    {

        r=temp[2];

        temp[2]=temp[1];

        temp[1]=temp[0];

        temp[0]=d.read();

        cout <<sc\_time\_stamp<< " Temp[2]: "<< temp[2]<< " Temp[1]: "<< temp[1] << " Temp[0]: "<< temp[0]<<endl;

    }

    SC\_CTOR(siso)

    {

        SC\_METHOD(compute\_siso);

        sensitive<<clk.pos();

    }

};

#include <systemc.h>

#include "siso.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<sc\_logic> d,r;

    sc\_clock  testclk("testclock",10,SC\_NS,0.5);

    siso siso1("siso");

    driver drive("driver");

    monitor mon("monitor");

    siso1.d(d); siso1.r(r); siso1.clk(testclk);

    drive.d(d); drive.clk(testclk);

    mon.d(d); mon.r(r); mon.clk(testclk);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, d, "d");

    sc\_trace(Tf, r, "r");

    sc\_trace(Tf,testclk,"testclk");

    sc\_start(120, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<sc\_logic> d,r;

    sc\_in<bool> clk;

    void mon()

    {

        cout<<"Input D: "<<d<<" Output Q:= "<<r<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<clk.pos();

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<sc\_logic> d;

    sc\_in<bool> clk;

    void inputs()

    {

        wait();

        d = sc\_logic\_1;

        wait();

        d = sc\_logic\_0;

        wait();

        d = sc\_logic\_1;

        wait();

    }

        SC\_CTOR(driver)

        {

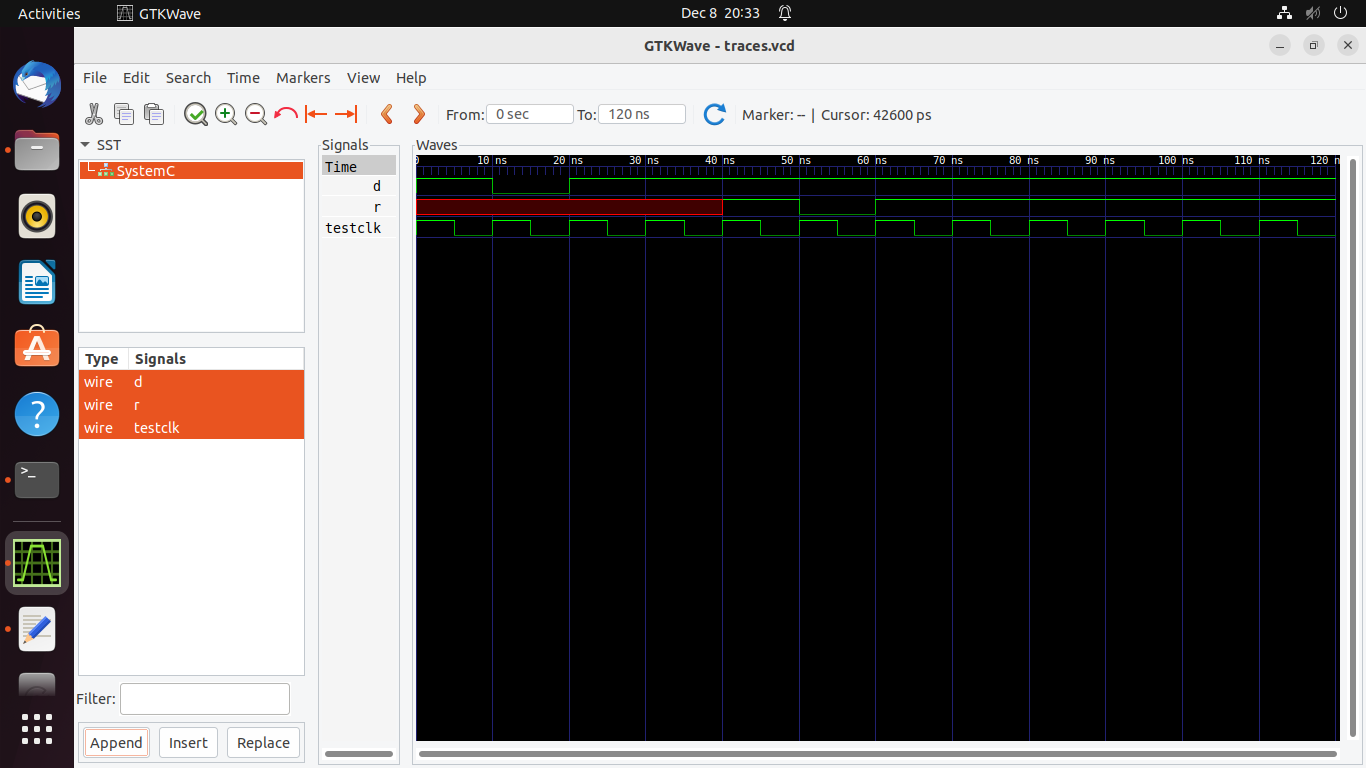
            SC\_THREAD(inputs);

            sensitive <<clk.pos();

        }

};

Output:



1. NAND

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

#include <systemc.h>

#include "nandgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    nandgate nagate("nandgate");

    driver drive("driver");

    monitor mon("monitor");

    nagate.a1(a1); nagate.b1(b1); nagate.c1(c1); nagate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

//nandgate.h

#include <systemc.h>

SC\_MODULE(nandgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_nand()

    {

        d1.write(!(a1.read() && b1.read() && c1.read()));

    }

    SC\_CTOR(nandgate)

    {

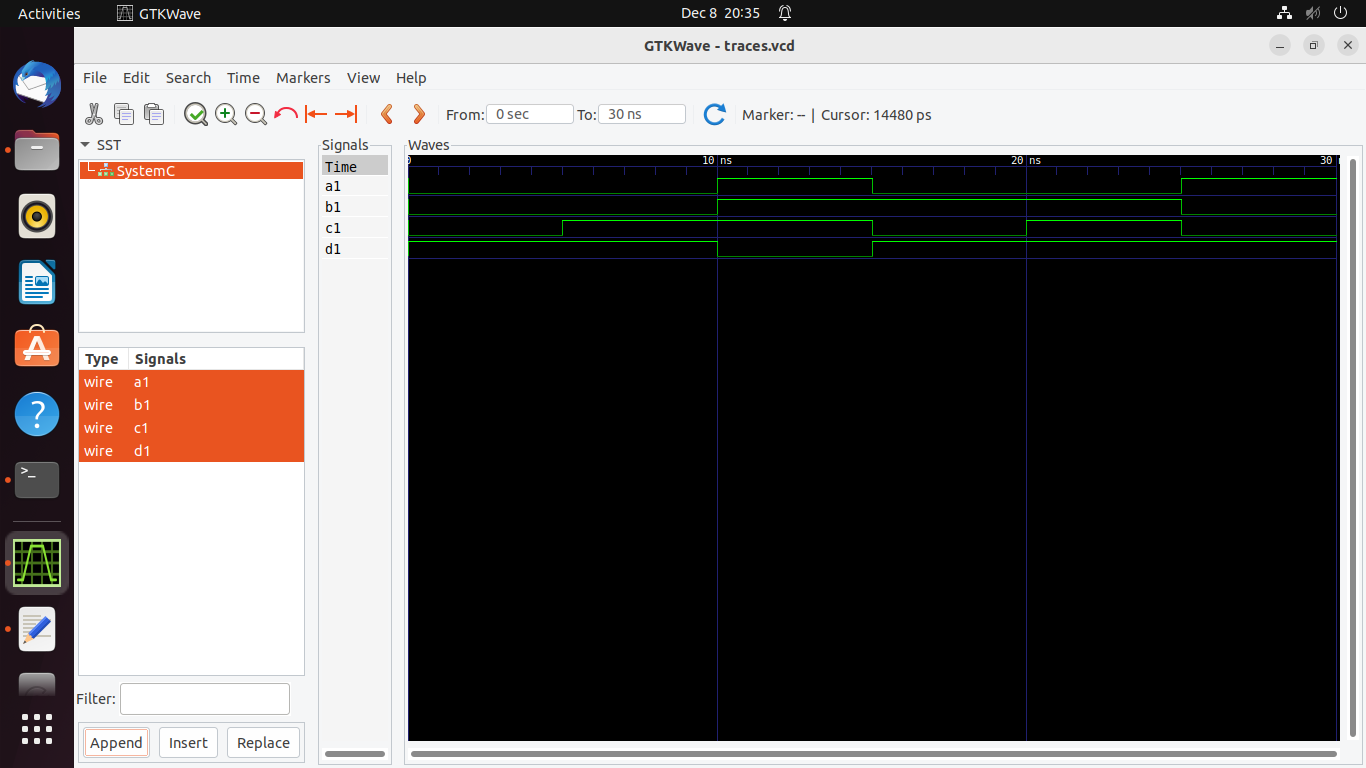
        SC\_METHOD(compute\_nand);

        sensitive<<a1<<b1<<c1;

    }

};

Output:



1. NOR

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

#include <systemc.h>

#include "norgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    norgate nogate("norgate");

    driver drive("driver");

    monitor mon("monitor");

    nogate.a1(a1); nogate.b1(b1); nogate.c1(c1); nogate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//nogate.h

#include <systemc.h>

SC\_MODULE(norgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_nor()

    {

        d1.write(!(a1.read() || b1.read() || c1.read()));

    }

    SC\_CTOR(norgate)

    {

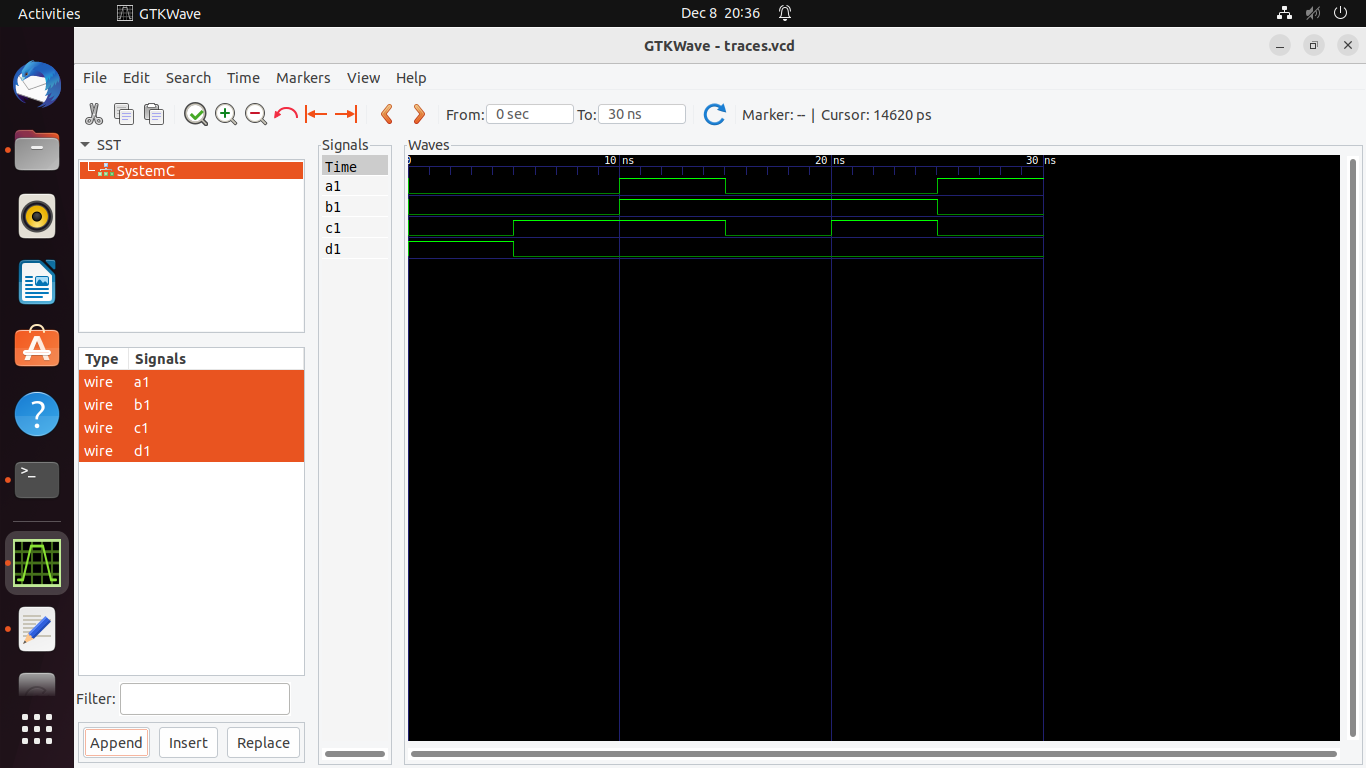
        SC\_METHOD(compute\_nor);

        sensitive<<a1<<b1<<c1;

    }

};

Output:



1. XOR

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

//xorgate.h

#include <systemc.h>

SC\_MODULE(xorgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_xor()

    {

        d1.write(a1.read() ^ b1.read() ^ c1.read());

    }

    SC\_CTOR(xorgate)

    {

        SC\_METHOD(compute\_xor);

        sensitive<<a1<<b1<<c1;

    }

};

#include <systemc.h>

#include "xorgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    xorgate xogate("xorgate");

    driver drive("driver");

    monitor mon("monitor");

    xogate.a1(a1); xogate.b1(b1); xogate.c1(c1); xogate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

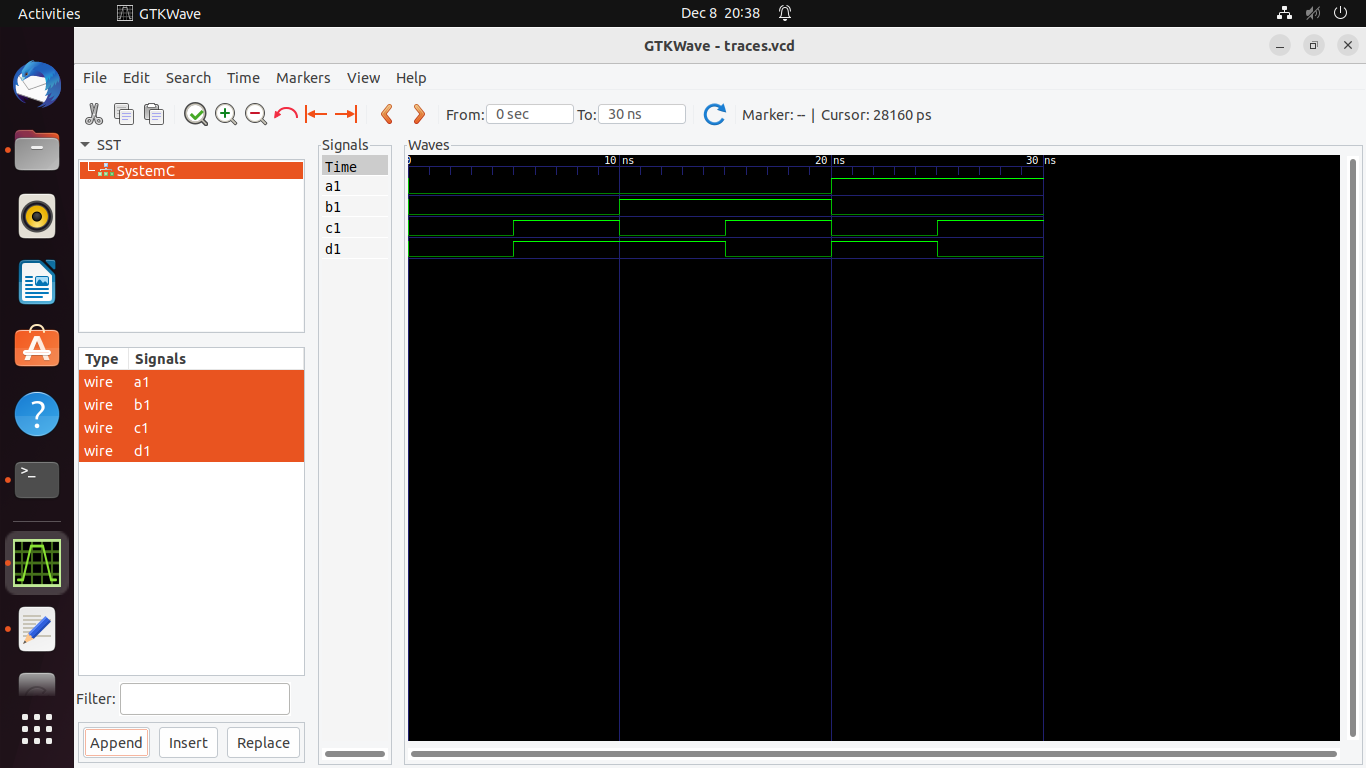
    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

Output:



1. OR

//orgate.h

#include <systemc.h>

SC\_MODULE(orgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_or()

    {

        d1.write(a1.read() | b1.read() | c1.read());

    }

    SC\_CTOR(orgate)

    {

        SC\_METHOD(compute\_or);

        sensitive<<a1<<b1<<c1;

    }

};

#include <systemc.h>

#include "orgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    orgate ogate("orgate");

    driver drive("driver");

    monitor mon("monitor");

    ogate.a1(a1); ogate.b1(b1); ogate.c1(c1); ogate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

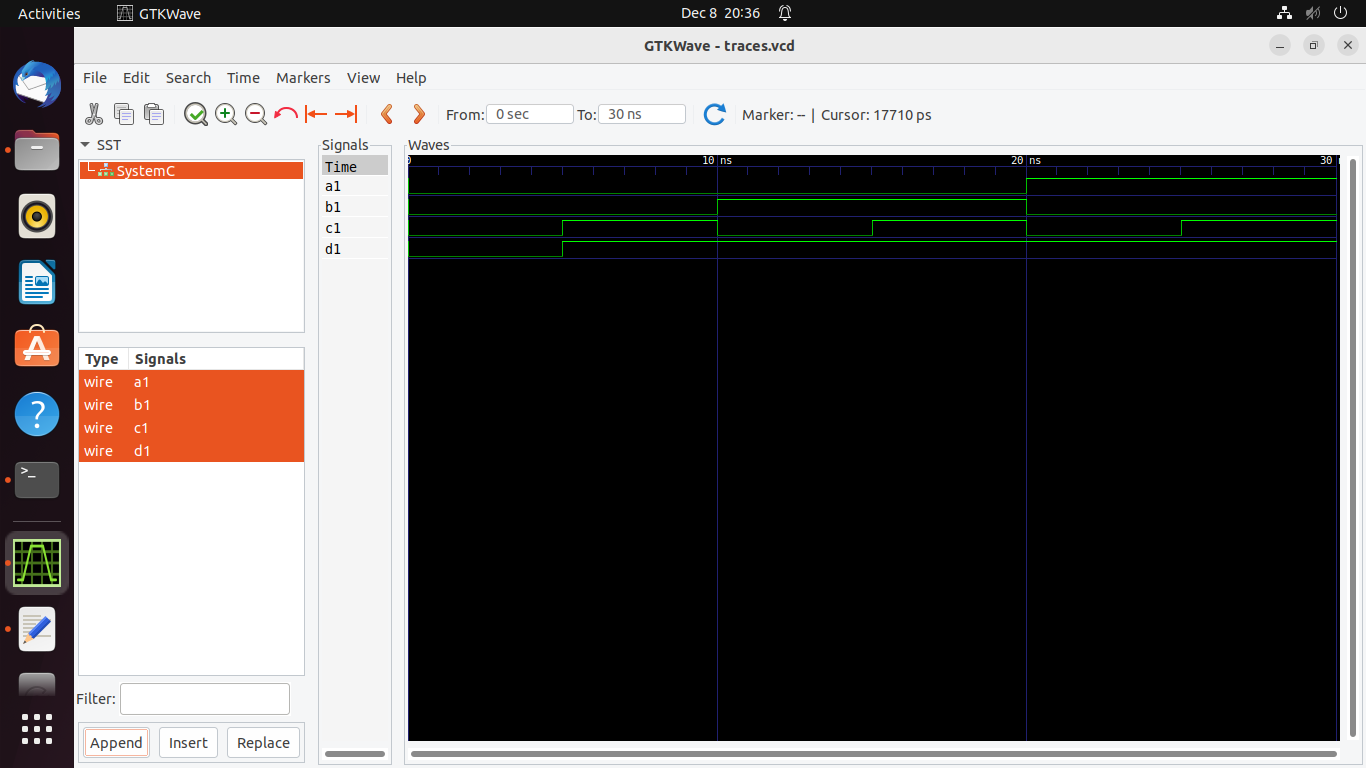
        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

Output:



1. XNOR

//xnorgate.h

#include <systemc.h>

SC\_MODULE(xnorgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_xnor()

    {

        d1.write(!(a1.read() ^ b1.read() ^ c1.read()));

    }

    SC\_CTOR(xnorgate)

    {

        SC\_METHOD(compute\_xnor);

        sensitive<<a1<<b1<<c1;

    }

};

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

#include <systemc.h>

#include "xnorgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    xnorgate xngate("xnorgate");

    driver drive("driver");

    monitor mon("monitor");

    xngate.a1(a1); xngate.b1(b1); xngate.c1(c1); xngate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

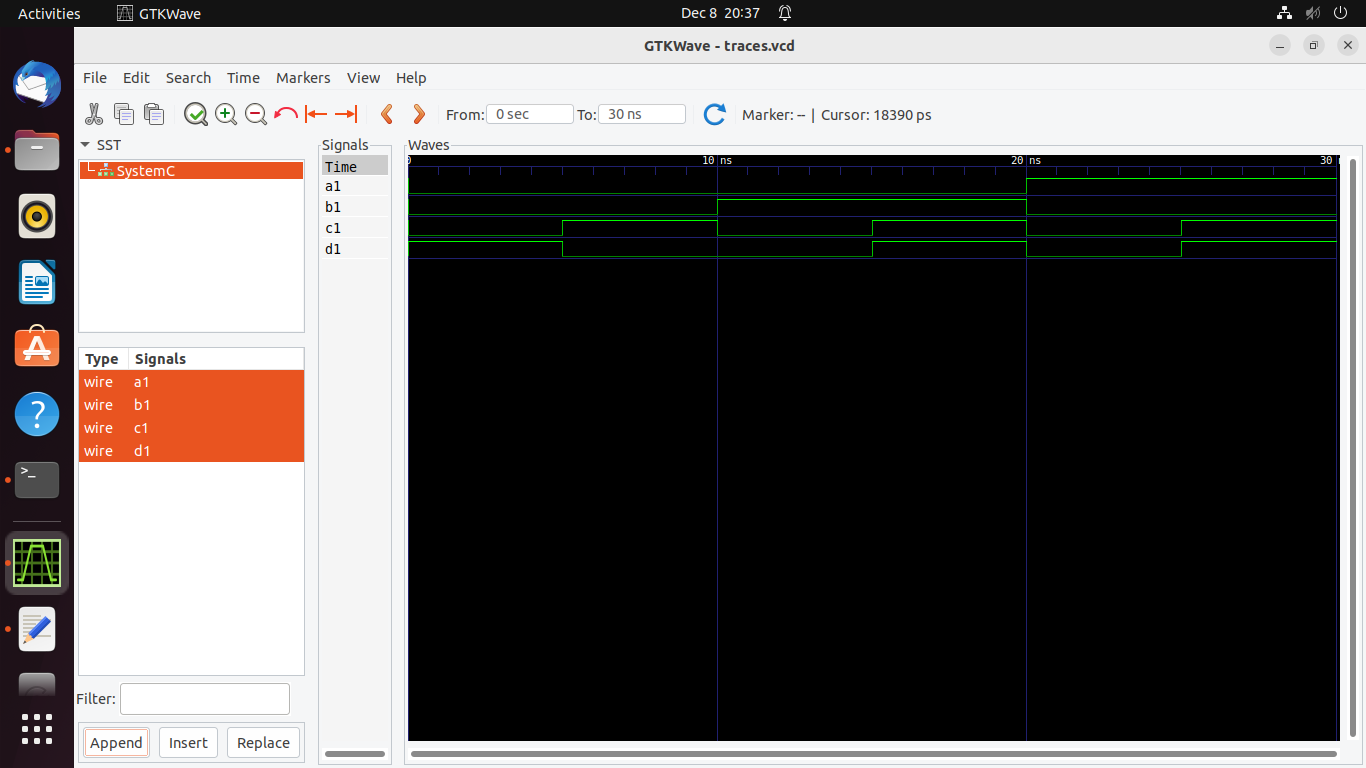
    sc\_start(30, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

Output:



1. AND

//driver.h

#include <systemc.h>

SC\_MODULE(driver)

{

    sc\_out<bool> a1, b1, c1;

    void inputs()

    {

        a1.write(false);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(false);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(false);

        c1.write(true);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(false);wait(5, SC\_NS);

        a1.write(true);

        b1.write(true);

        c1.write(true);wait(5, SC\_NS);

    }

        SC\_CTOR(driver)

        {

            SC\_THREAD(inputs);

            sensitive <<a1 <<b1 <<c1;

        }

};

#include <systemc.h>

#include "andgate.h"

#include "driver.h"

#include "monitor.h"

int sc\_main(int argc, char\* argv[])

{

    sc\_signal<bool> a1, b1,c1, d1;

    andgate agate("andgate");

    driver drive("driver");

    monitor mon("monitor");

    agate.a1(a1); agate.b1(b1); agate.c1(c1); agate.d1(d1);

    drive.a1(a1); drive.b1(b1); drive.c1(c1);

    mon.a1(a1); mon.b1(b1); mon.c1(c1); mon.d1(d1);

    // waveform

    sc\_trace\_file \*Tf;

    Tf = sc\_create\_vcd\_trace\_file("traces");

    sc\_trace(Tf, a1, "a1");

    sc\_trace(Tf, b1, "b1");

    sc\_trace(Tf, c1, "c1");

    sc\_trace(Tf, d1, "d1");

    sc\_start(120, SC\_NS);

    sc\_close\_vcd\_trace\_file(Tf);

    return 0;

}

//monitor.h

#include<systemc.h>

SC\_MODULE(monitor)

{

    sc\_in<bool> a1,b1,c1,d1;

    void mon()

    {

        cout<<"Inputs: "<<a1<<b1<<c1<<"Output: "<<d1<<endl;

    }

    SC\_CTOR(monitor)

    {

        SC\_METHOD(mon);

        sensitive<<a1<<b1<<c1;

    }

};

//andgate.h

#include <systemc.h>

SC\_MODULE(andgate)

{

    sc\_in<bool> a1;

    sc\_in<bool> b1;

    sc\_in<bool> c1;

    sc\_out<bool> d1;

    void compute\_and()

    {

        d1.write(a1.read() && b1.read() && c1.read());

    }

    SC\_CTOR(andgate)

    {

        SC\_METHOD(compute\_and);

        sensitive<<a1<<b1<<c1;

    }

};

Output:

