MUTONGA BRIAN NG'ANG'A
I39/34551/2013
30/10/2015

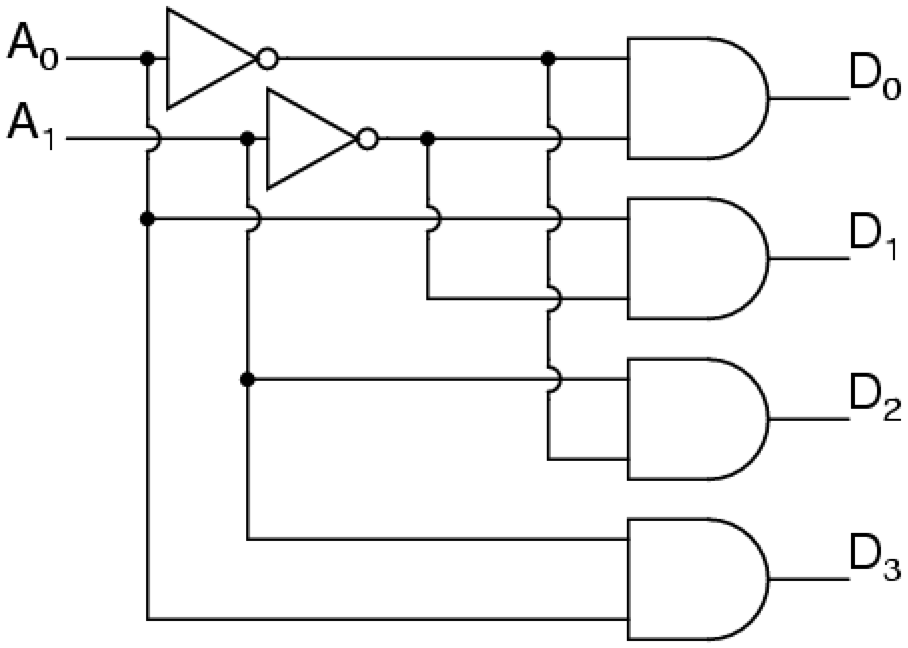
2 BY 4 DECODER

OBJECTIVE.

1.simulating and modelling the 2 by 4 decoder using systemc modelling language.

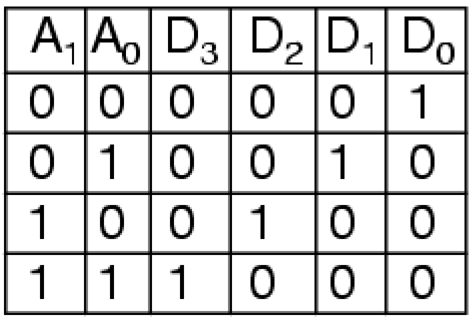
INTTRODUCTION.

A decoder is a circuit that changes a code into a set of signals. It is called a decoder because it does the reverse of encoding, this circuit takes an *n*-bit binary number and produces an output on one of 2^n output lines .In this case this decoder takes two inputs and produces four inputs by using a combination of NOT and AND gates .By using also using truth tables to simulate all this .

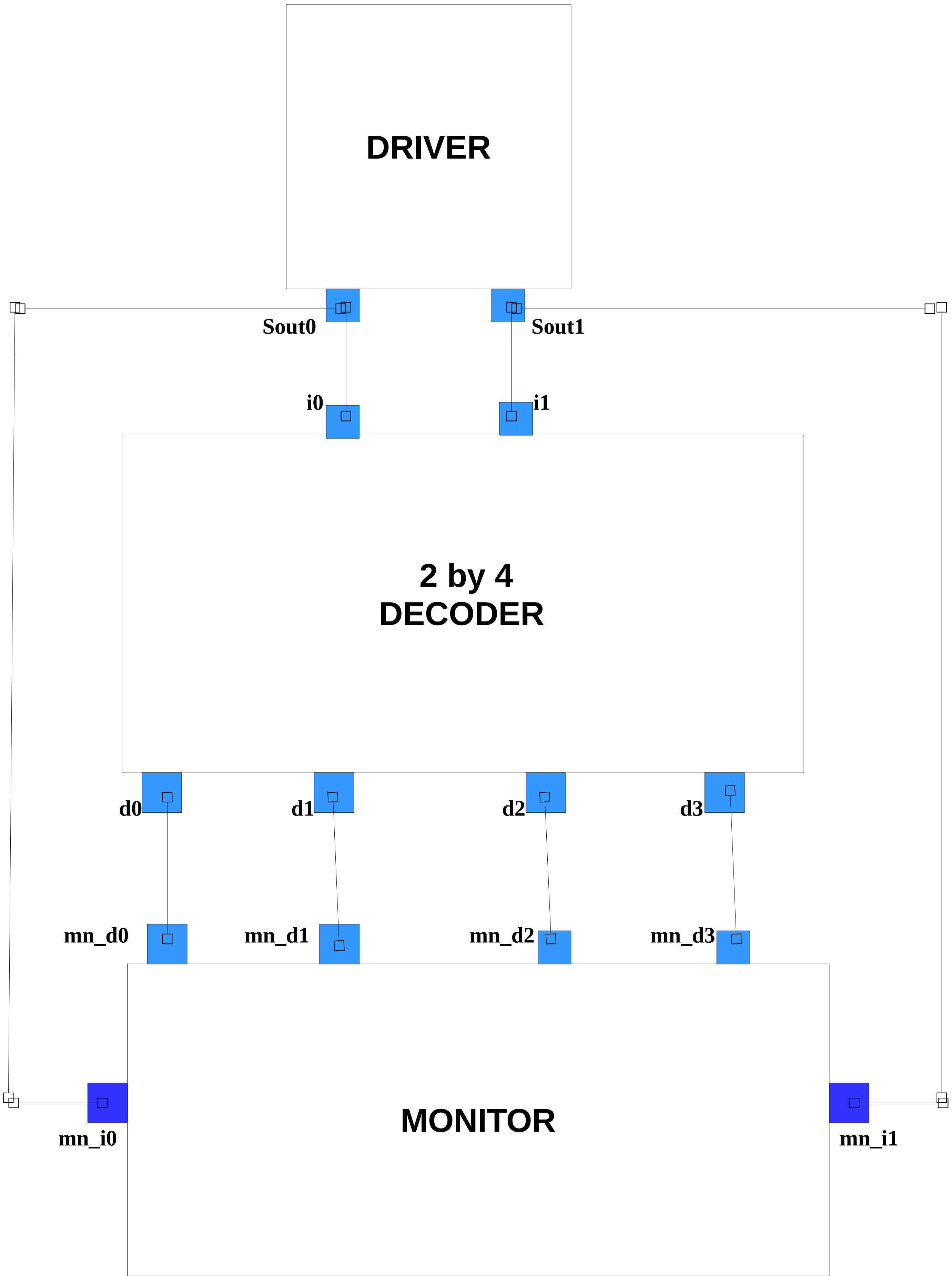


2 BY 4 DECODER SCHEMATICS

2 BY 4 DECODER TRUTH TABLE



A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0


DRIVER
Sout0
Sout1
i0
i1
2 by 4
DECODER
d0
d1
d2
d3
mn_d0
mn_d1
mn_d2
mn_d3
mn_i0
MONITOR
mn_i1

Driver module

```
#ifndef DRIVER_H_
#define DRIVER_H_

#include<systemc>

SC_MODULE(driver){
sc_out<bool> Sout0,Sout1;

SC_CTOR(driver){
	SC_THREAD(drive);
}

void drive(void){
	while(1){
		Sout0=0;
		Sout1=0;
		wait(5,SC_NS);
		Sout1=1;
		wait(5,SC_NS);
		Sout0=1;
		Sout1=0;
		wait(5,SC_NS);
		Sout1=1;
		wait(5,SC_NS);
	}
}
};

#endif
```

Decoder module

```
				#include<systemc.h>

SC_MODULE(decoder){
//input and output ports
sc_in<bool> i0,i1;
sc_out<bool> d0,d1,d2,d3;
//constructor: where the processes are bound to simulation kernel
SC_CTOR(decoder){
	SC_METHOD(decode);
	sensitive<<i0<<i1; //sensitive to i0 and i1 which is the input
	//dont_initialize();
}

~decoder(){
//delete stuff :P
}

void decode(void){
	if (i0==0 && i1==0){
  d0=1;
```

```
   d1=0;
   d2=0;
   d3=0; }
    else if (i0==0 && i1==1){
     d1=0;
     d2=1;
     d3=0;
     d0=0;
        }
    else  if (i0==1 && i1==0){
           d3=0;
        d1=1;
        d2=0;
           d0=0;

     }
      else  if (i0==1 && i1==1){
         d3=1;
         d2=0;
            d1=0;
            d0=0;
}
};
```

monitor.h

```
#ifndef MONITOR_H_
#define MONITOR_H_

#include<iostream>
#include<systemc>

using namespace std;

SC_MODULE(monitor){
sc_in<bool> mn_i0, mn_i1,mn_d0,mn_d1,mn_d2,mn_d3;

SC_CTOR(monitor){
         SC_METHOD(monita);
         sensitive<<mn_d0<<mn_d1<<mn_d2<<mn_d3;
         dont_initialize();
}

void monita(void){
cout<<"at "<<sc_time_stamp()<<" input 0 is: "<<mn_i0<<"input 1 is "<<mn_i1<<" outputs are: "<<mn_d0<<"
and "<<mn_d1<<"and"<<mn_d2<<"and"<<mn_d3<<endl;
}
};

#endif
```

Decoder.cc

```
#include"decoder.h"
#include"driver.h"
#include"monitor.h"
#include<systemc>

int sc_main(int argc, char *argv[]){
//some signals for interconnections
sc_signal<bool> in_0,in_1,out0, out1,out2,out3;
//module instances
decoder dec("decoder_instance");
driver dr("driver_instance");
monitor mn("monitor_instance");
//interconnections between modules instance.port()
dr.Sout0(in_0);
dr.Sout1(in_1);
dec.i0(in_0);
dec.i1(in_1);
mn.mn_i0(in_0);
mn.mn_i1(in_1);

dec.d0(out0);
dec.d1(out1);
dec.d2(out2);
dec.d3(out3);
//monitor outputs
mn.mn_d0(out0);
mn.mn_d1(out1);
mn.mn_d2(out2);
mn.mn_d3(out3);
//create a trace file with nanosecond resolution
sc_trace_file *tf;
tf = sc_create_vcd_trace_file("timing_diagram");
tf->set_time_unit(1, SC_NS);
//trace the signals interconnecting modules
sc_trace(tf, in_0, "binary_input1"); // signals to be traced
sc_trace(tf,in_1,  "binary_input2");

sc_trace(tf, out0, "input_is_zero");
sc_trace(tf, out1, "input_is_one");
sc_trace(tf, out2, "input_is_two");
sc_trace(tf, out3, "input_is_three");


//run a simulation for 50 systemc nano-seconds
if( !sc_pending_activity() )
sc_start(50,SC_NS);
//close the trace file
sc_close_vcd_trace_file(tf);
return 0;
}
```

Conclusion

The modelling and simulation was successful.

References

1.A SystemC Primer
J. BHASKER
Published by
Star Galaxy Publishing
015 Treeline Drive, Allentown,PA 18103
2.SystemC:
From the Ground Up
D.C. Black, J. Donovan, B. Bunton, A. Keist