EE214 FSM Report

Shashwat Shukla Roll Number 150260025

March 21, 2017

1 Overview of the experiment

The aim of this experiment is to design and test a string recogniser, modelled as an ensemble of FSMs that each detects one string. The machine is a clocked device, with a reset button.

If at any time t, the sequences of alphabets seen so far contains one of the words: bomb,gun,knife,terror, then the machine output is true. Otherwise, the output is false.

2 State transition diagrams

The following are the diagrams for the four different string detector FSM is shown below. Note that the outputs are not shown.

The ouput is 1, only on the edge that terminates on the state I (not referring to the self-loop or the reset edge). All other outputs are 0.

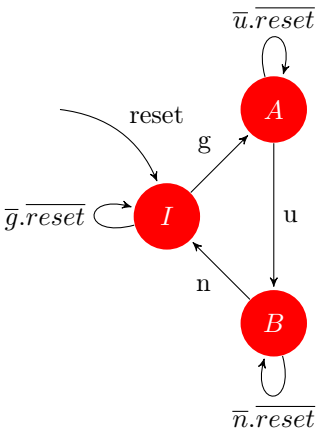

Figure 1: State diagram for gun

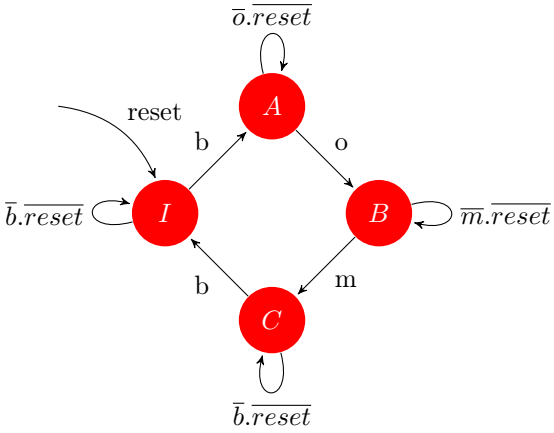

Figure 2: State diagram for bomb

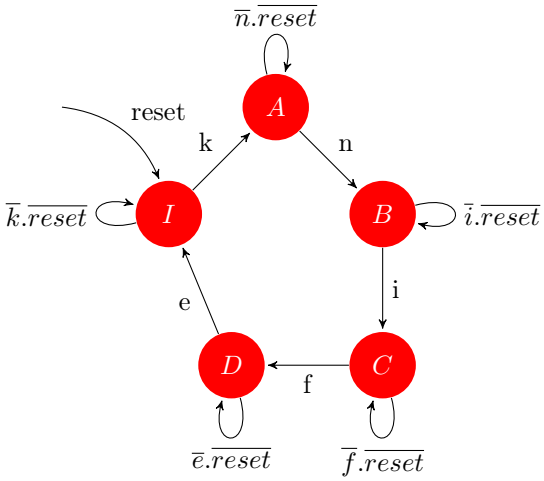

Figure 3: State diagram for knife

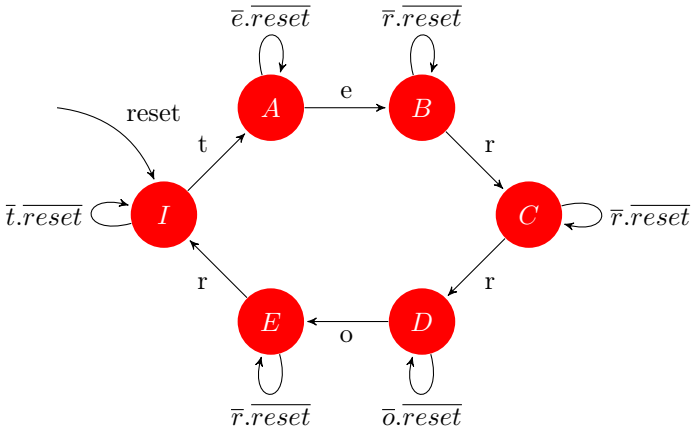

Figure 4: State diagram for terror

3 Code for various components

3.1 stringFSM.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity stringFSM is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end entity;

architecture str of stringFSM is

component FSMbomb is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end component;

component FSMgun is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end component;

component FSMknife is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end component;

component FSMterror is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end component;

signal bomb,gun,knife,terror: std_ulogic;
signal add1,add2: std_ulogic;
```

```
begin

strbomb: FSMbomb port map(x,bomb);
strgun: FSMgun port map(x,gun);
strknife: FSMknife port map(x,knife);
strterror: FSMterror port map(x,terror);

cor1: OR2 port map(bomb,gun,add1);
cor2: OR2 port map(knife,terror,add2);

cz: OR2 port map(add1,add2,z);

end str;
```

3.2 FSMBomb.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity FSMBomb is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end entity;

architecture bomb of FSMbomb is

component AND7 is
        port(x6,x5,x4,x3,x2,x1,x0: in std_ulogic;
                z: out std_ulogic);
end component;

signal invx4, invx3, invx2, invx1, invx0: std_ulogic;
signal notr: std_ulogic;
signal b,o,m: std_ulogic;
signal notb,noto,notm: std_ulogic;

signal s2,s1,invs2,invs1: std_ulogic;
```

```
signal ns2,ns1: std_ulogic;
signal tns2,tns1: std_ulogic;

signal ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8: std_ulogic;

begin

cinvx4: INVERTER port map(x(4), invx4);
cinvx3: INVERTER port map(x(3), invx3);
cinvx2: INVERTER port map(x(2), invx2);
cinvx1: INVERTER port map(x(1), invx1);
cinvx0: INVERTER port map(x(0), invx0);

cinvs2: INVERTER port map(s2, invs2);
cinvs1: INVERTER port map(s1, invs1);

calcb: AND7 port map('1','1',invx4,invx3,invx2,x(1),invx0,b);
calco: AND7 port map('1','1',invx4,x(3),x(2),x(1),x(0),o);
calcm: AND7 port map('1','1',invx4,x(3),x(2),invx1,x(0),m);

cnotb: INVERTER port map(b, notb);
cnoto: INVERTER port map(o, noto);
cnotm: INVERTER port map(m, notm);

cnotr: INVERTER port map(x(6), notr);

-- ns2 = (01)(o) + (11) + (10)(not b)
calcts1: AND7 port map(invs2,s1,o,'1','1','1','1',ts1);
calcts2: AND7 port map(s2,invs1,notb,'1','1','1','1',ts2);
calcts7: AND2 port map(s2,s1,ts7);

calcts8: OR2 port map(ts1,ts2,ts8);

ctns2: OR2 port map(ts7,ts8,tns2);
cns2: AND2 port map(notr,tns2,ns2);

-- ns1 = (00)(b) + (01) + (11)(notm)
calcts3: AND7 port map(invs2,invs1,b,'1','1','1','1',ts3);
calcts4: AND7 port map(s2,s1,notm,'1','1','1','1',ts4);
calcts5: AND2 port map(invs2,s1,ts5);

calcts6: OR2 port map(ts3,ts4,ts6);

ctns1: OR2 port map(ts5,ts6,tns1);
cns1: AND2 port map(notr,tns1,ns1);
```

```
-- z = (10)(b)
cz: AND7 port map(s2,invs1,b,'1','1','1','1',z);

d2: DFF port map (d => ns2, clk => x(5), q => s2);
d1: DFF port map (d => ns1, clk => x(5), q => s1);

end bomb;
```

3.3 FSMGun.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity FSMGun is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end entity;

architecture gun of FSMGun is

component AND7 is
        port(x6,x5,x4,x3,x2,x1,x0: in std_ulogic;
                z: out std_ulogic);
end component;

signal invx4, invx3, invx2, invx1, invx0: std_ulogic;
signal notr: std_ulogic;
signal g,u,n: std_ulogic;
signal notg,notu,notn: std_ulogic;

signal s2,s1,invs2,invs1: std_ulogic;
signal ns2,ns1: std_ulogic;
signal tns2,tns1: std_ulogic;

signal ts1,ts2,ts3,ts4 : std_ulogic;

begin
```

```
cinvx4: INVERTER port map(x(4), invx4);
cinvx3: INVERTER port map(x(3), invx3);
cinvx2: INVERTER port map(x(2), invx2);
cinvx1: INVERTER port map(x(1), invx1);
cinvx0: INVERTER port map(x(0), invx0);

cinvs2: INVERTER port map(s2, invs2);
cinvs1: INVERTER port map(s1, invs1);

calcg: AND7 port map('1','1',invx4,invx3,x(2),x(1),x(0),g);
calcu: AND7 port map('1','1',x(4),invx3,x(2),invx1,x(0),u);
calcn: AND7 port map('1','1',invx4,x(3),x(2),x(1),invx0,n);

cnotg: INVERTER port map(g, notg);
cnotu: INVERTER port map(u, notu);
cnotn: INVERTER port map(n, notn);

cnotr: INVERTER port map(x(6), notr);

-- ns2 = (01)(u) + (10)(not n)
calcts1: AND7 port map(invs2,s1,u,'1','1','1','1',ts1);
calcts2: AND7 port map(s2,invs1,notn,'1','1','1','1',ts2);

ctns2: OR2 port map(ts1,ts2,tns2);
cns2: AND2 port map(notr,tns2,ns2);

-- ns1 = (00)(g) + (01)(not u)
calcts3: AND7 port map(invs2,invs1,g,'1','1','1','1',ts3);
calcts4: AND7 port map(invs2,s1,notu,'1','1','1','1',ts4);

ctns1: OR2 port map(ts3,ts4,tns1);
cns1: AND2 port map(notr,tns1,ns1);

-- z = (10)(n)
cz: AND7 port map(s2,invs1,n,'1','1','1','1',z);

d2: DFF port map (d => ns2, clk => x(5), q => s2);
d1: DFF port map (d => ns1, clk => x(5), q => s1);

end gun;
```

3.4 FSMKnife.vhd

```
library std;
use std.textio.all;
```

```
library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity FSMKnife is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end entity;

architecture knife of FSMknife is

component AND7 is
        port(x6,x5,x4,x3,x2,x1,x0: in std_ulogic;
                z: out std_ulogic);
end component;

signal invx4, invx3, invx2, invx1, invx0: std_ulogic;
signal notr: std_ulogic;
signal k,n,i,f,e: std_ulogic;
signal notk,notn,noti,notf,notee: std_ulogic;

signal s3,s2,s1,invs3,invs2,invs1: std_ulogic;
signal ns3,ns2,ns1: std_ulogic;
signal tns3,tns2,tns1: std_ulogic;

signal ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10,ts11,ts12,ts13,ts14: std_ulogic;

begin

cinvx4: INVERTER port map(x(4), invx4);
cinvx3: INVERTER port map(x(3), invx3);
cinvx2: INVERTER port map(x(2), invx2);
cinvx1: INVERTER port map(x(1), invx1);
cinvx0: INVERTER port map(x(0), invx0);

cinvs3: INVERTER port map(s3, invs3);
cinvs2: INVERTER port map(s2, invs2);
cinvs1: INVERTER port map(s1, invs1);

calck: AND7 port map('1','1',invx4,x(3),invx2,x(1),x(0),k);
calcn: AND7 port map('1','1',invx4,x(3),x(2),x(1),invx0,n);
```

```
calci: AND7 port map('1','1',invx4,x(3),invx2,invx1,x(0),i);
calcf: AND7 port map('1','1',invx4,invx3,x(2),x(1),invx0,f);
calce: AND7 port map('1','1',invx4,invx3,x(2),invx1,x(0),e);

cnotk: INVERTER port map(k, notk);
cnotn: INVERTER port map(n, notn);
cnoti: INVERTER port map(i, noti);
cnotf: INVERTER port map(f, notf);
cnote: INVERTER port map(e, notee);

cnotr: INVERTER port map(x(6), notr);

-- ns3 = (011)(i) + (111) + (101)(not e)
calcts1: AND7 port map(invs3,s2,s1,i,'1','1','1',ts1);
calcts2: AND7 port map(s3,s2,s1,'1','1','1','1',ts2);
calcts3: AND7 port map(s3,invs2,s1,notee,'1','1','1',ts3);

calcts4: OR2 port map(ts1,ts2,ts4);

ctns3: OR2 port map(ts3,ts4,tns3);
cns3: AND2 port map(notr,tns3,ns3);

-- ns2 = (001)(n) + (011) + (111)(not f)
calcts5: AND7 port map(invs3,invs2,s1,n,'1','1','1',ts5);
calcts6: AND7 port map(invs3,s2,s1,'1','1','1','1',ts6);
calcts7: AND7 port map(s3,s2,s1,notf,'1','1','1',ts7);

calcts8: OR2 port map(ts5,ts6,ts8);

ctns2: OR2 port map(ts7,ts8,tns2);
cns2: AND2 port map(notr,tns2,ns2);

-- ns1 = (000)(k) + (001) + (011) + (111) + (101)(not e)
calcts9: AND7 port map(invs3,invs2,invs1,k,'1','1','1',ts9);
calcts10: AND7 port map(invs3,'1',s1,'1','1','1','1',ts10);
calcts11: AND7 port map(s3,s2,s1,'1','1','1','1',ts11);
calcts12: AND7 port map(s3,invs2,s1,notee,'1','1','1',ts12);

calcts13: OR2 port map(ts9,ts10,ts13);
calcts14: OR2 port map(ts11,ts12,ts14);

ctns1: OR2 port map(ts13,ts14,tns1);
cns1: AND2 port map(notr,tns1,ns1);

-- z = (10)(b)
cz: AND7 port map(s3,invs2,s1,e,'1','1','1',z);
```

```
d3: DFF port map (d => ns3, clk => x(5), q => s3);
d2: DFF port map (d => ns2, clk => x(5), q => s2);
d1: DFF port map (d => ns1, clk => x(5), q => s1);

end knife;
```

3.5 FSMTerror.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity FSMTerror is
        port(x: in std_ulogic_vector(6 downto 0);
                z: out std_ulogic);
end entity;

architecture terror of FSMterror is

component AND7 is
        port(x6,x5,x4,x3,x2,x1,x0: in std_ulogic;
                z: out std_ulogic);
end component;

signal invx4, invx3, invx2, invx1, invx0: std_ulogic;
signal notrr: std_ulogic;
signal t,e,r,o: std_ulogic;
signal nott,notee,notr,noto: std_ulogic;

signal s3,s2,s1,invs3,invs2,invs1: std_ulogic;
signal ns3,ns2,ns1: std_ulogic;
signal tns3,tns2,tns1: std_ulogic;

signal ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10,ts11,ts12: std_ulogic;

begin
```

```
cinvx4: INVERTER port map(x(4), invx4);
cinvx3: INVERTER port map(x(3), invx3);
cinvx2: INVERTER port map(x(2), invx2);
cinvx1: INVERTER port map(x(1), invx1);
cinvx0: INVERTER port map(x(0), invx0);

cinvs3: INVERTER port map(s3, invs3);
cinvs2: INVERTER port map(s2, invs2);
cinvs1: INVERTER port map(s1, invs1);

calct: AND7 port map('1','1',x(4),invx3,x(2),invx1,invx0,t);
calce: AND7 port map('1','1',invx4,invx3,x(2),invx1,x(0),e);
calcr: AND7 port map('1','1',x(4),invx3,invx2,x(1),invx0,r);
calco: AND7 port map('1','1',invx4,x(3),x(2),x(1),x(0),o);

cnott: INVERTER port map(t, nott);
cnote: INVERTER port map(e, notee);
cnotr: INVERTER port map(r, notr);
cnoto: INVERTER port map(o, noto);

cnotrr: INVERTER port map(x(6), notrr);

-- ns3 = (010)(r) + (110) + (100)(not r)
calcts1: AND7 port map(invs3,s2,invs1,r,'1','1','1',ts1);
calcts2: AND7 port map(s3,s2,invs1,'1','1','1','1',ts2);
calcts3: AND7 port map(s3,invs2,invs1,notr,'1','1','1',ts3);

calcts4: OR2 port map(ts1,ts2,ts4);

ctns3: OR2 port map(ts3,ts4,tns3);
cns3: AND2 port map(notrr,tns3,ns3);

-- ns2 = (001)(e) + (011) + (010) + (110)(not o)
calcts5: AND7 port map(invs3,invs2,s1,e,'1','1','1',ts5);
calcts6: AND7 port map(invs3,s2,'1','1','1','1','1',ts6);
calcts7: AND7 port map(s3,s2,invs1,noto,'1','1','1',ts7);

calcts8: OR2 port map(ts5,ts6,ts8);

ctns2: OR2 port map(ts7,ts8,tns2);
cns2: AND2 port map(notrr,tns2,ns2);

-- ns1 = (000)(t) + (001) + (011)(not r)
calcts9: AND7 port map(invs3,invs2,invs1,t,'1','1','1',ts9);
calcts10: AND7 port map(invs3,invs2,s1,'1','1','1','1',ts10);
```

```
calcts11: AND7 port map(invs3,s2,s1,notr,'1','1','1',ts11);

calcts12: OR2 port map(ts9,ts10,ts12);

ctns1: OR2 port map(ts11,ts12,tns1);
cns1: AND2 port map(notrr,tns1,ns1);

-- z = (100)(r)
cz: AND7 port map(s3,invs2,invs1,r,'1','1','1',z);


d3: DFF port map (d => ns3, clk => x(5), q => s3);
d2: DFF port map (d => ns2, clk => x(5), q => s2);
d1: DFF port map (d => ns1, clk => x(5), q => s1);

end terror;
```

3.6 AND7.vhd

```
library std;
use std.textio.all;

library std;
use std.standard.all;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.EE224_Components.all;

entity AND7 is
        port(x6,x5,x4,x3,x2,x1,x0: in std_ulogic;
                z: out std_ulogic);
end entity;

architecture and7 of AND7 is

component AND2 is
        port (a, b: in std_ulogic; c : out std_ulogic);
end component;

signal and1,tand2,and3,and4,and5: std_ulogic;

begin
```

```
cand1: AND2 port map(x6, x5, and1);
cand2: AND2 port map(and1, x4, tand2);
cand3: AND2 port map(tand2, x3, and3);
cand4: AND2 port map(and3, x2, and4);
cand5: AND2 port map(and4, x1, and5);
cand6: AND2 port map(and5, x0, z);

end and7;
```

4 Observations

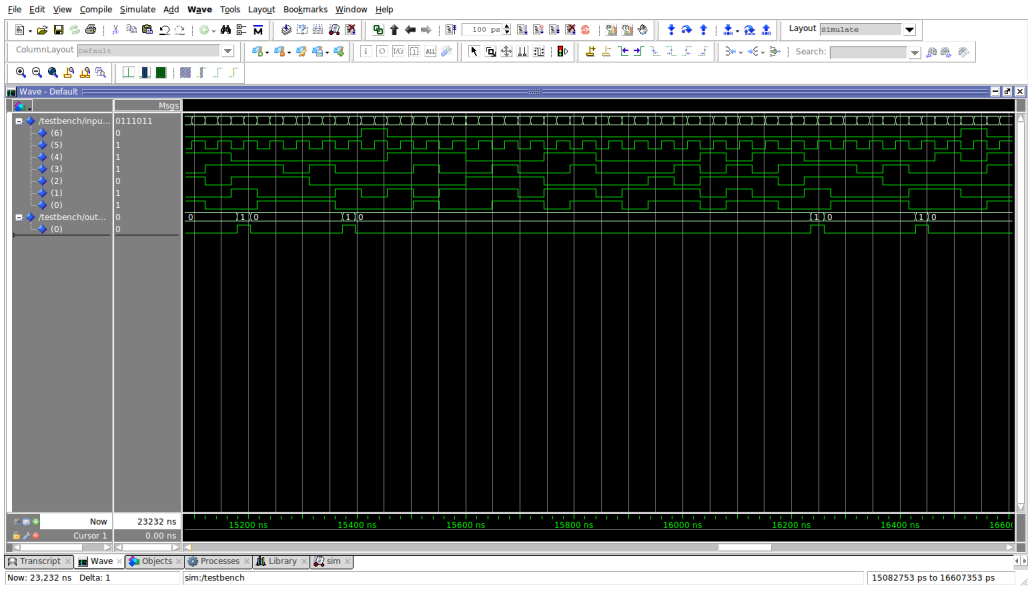

Figure 5: Gate level simulation

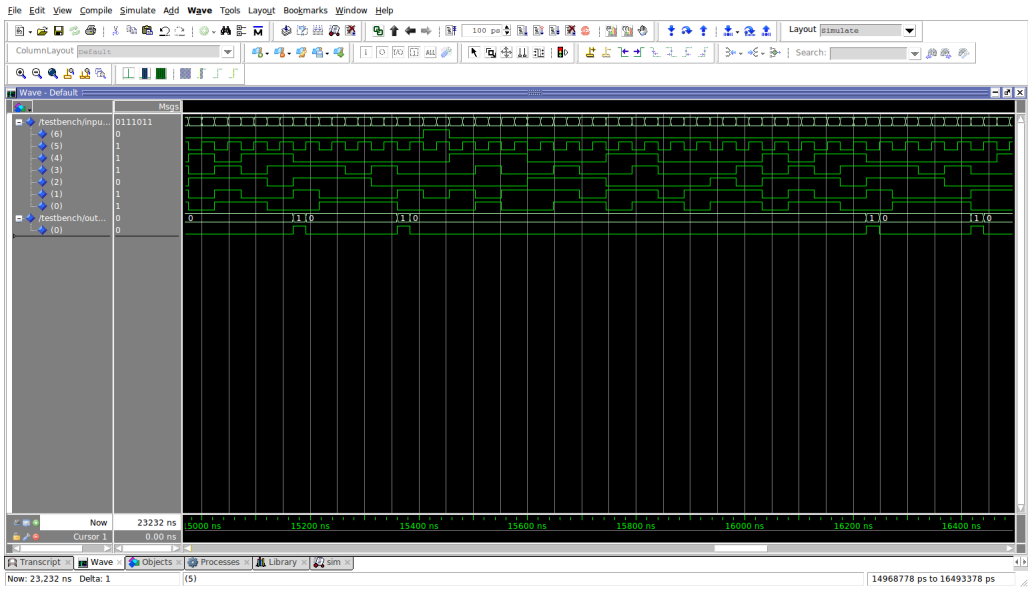

Figure 6: RTL simulation

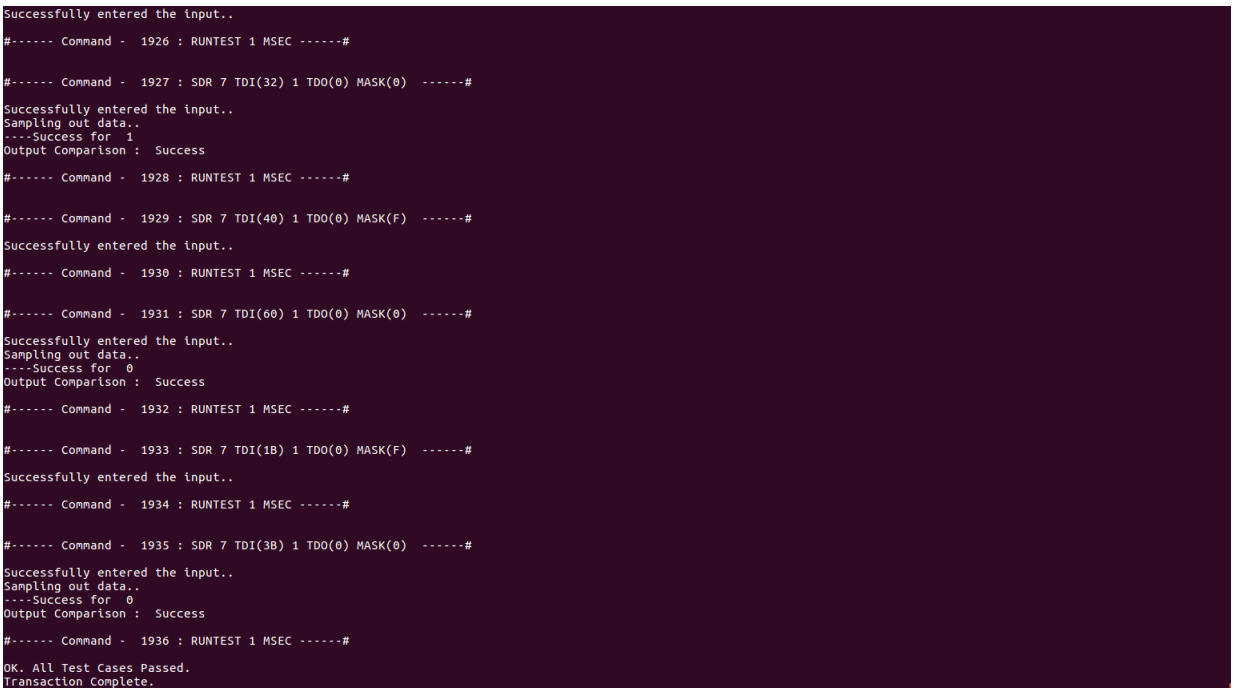

Figure 7: Successful completion of scan chain