EE 709 Testing and Verification of VLSI Circuits

Devesh Kumar, 16D070044

1. ASSIGNMENT 2: BDD

1.1 Specification

The given specification of the 8 bit adder resembles an ripple carry adder.
Ripple carry Adder
In a ripple carry adder the previous carry bit is used to find the next sum bit.

$$
\begin{aligned}
s0 &= a0 \oplus b0 \\
c0 &= a0.b0 \\
s1 &= a1 \oplus b1 \oplus c0 \\
c1 &= a1.b1 + a1.c0 + b1.c0 \\
s2 &= a2 \oplus b2 \oplus c1 \\
c1 &= a2.b2 + a2.c1 + b2.c1 \\
\ldots & \\
s7 &= a7 \oplus b7 \oplus c6
\end{aligned}
$$

Figure 1.

1.2 Implementation

To implement the adder we used carry look ahead adder. In this i computed propagate(a xor b) and generate bit(a and b). Then used these bits to calculate all the carry bits in one step(simultaneously).

$$
\begin{aligned}
C_1 &= G_0 + P_0 C_{in} \\
C_2 &= G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in} \\
C_3 &= G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in} \\
C_4 &= G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}
\end{aligned}
$$

Figure 2.

1.3 Equivalence

To prove equivalence I used bdd on every bit. If bdd from both the typeare same then they will be equivalent

1.4 Code

```
    #include <stdlib.h>
#include <stdio.h>
#include <bdduser.h>

int main (int argc, char* argv[])
{
bdd_manager bddm = bdd_init();
//////////////////////////////////////Input//////////////////////////////////////////////////
bdd a0 = bdd_new_var_last(bddm);
bdd a1 = bdd_new_var_last(bddm);
bdd a2 = bdd_new_var_last(bddm);
```

```
bdd a3 = bdd_new_var_last(bddm);
bdd a4 = bdd_new_var_last(bddm);
bdd a5 = bdd_new_var_last(bddm);
bdd a6 = bdd_new_var_last(bddm);
bdd a7 = bdd_new_var_last(bddm);

bdd b0 = bdd_new_var_last(bddm);
bdd b1 = bdd_new_var_last(bddm);
bdd b2 = bdd_new_var_last(bddm);
bdd b3 = bdd_new_var_last(bddm);
bdd b4 = bdd_new_var_last(bddm);
bdd b5 = bdd_new_var_last(bddm);
bdd b6 = bdd_new_var_last(bddm);
bdd b7 = bdd_new_var_last(bddm);
//////////////////////////////////////Specification : ripple carry adder//////////////////////////////
bdd s0 = bdd_xor(bddm, a0, b0);
bdd c0 = bdd_and(bddm, a0, b0);

bdd s1 = bdd_xor(bddm, bdd_xor(bddm,a1,b1), c0);
bdd c1 = bdd_or(bddm, bdd_and(bddm,a1,b1),bdd_and(bddm,bdd_or(bddm,a1,b1),c0) );

bdd s2 = bdd_xor(bddm, bdd_xor(bddm,a2,b2), c1);
bdd c2 = bdd_or(bddm, bdd_and(bddm,a2,b2),bdd_and(bddm,bdd_or(bddm,a2,b2),c1) );

bdd s3 = bdd_xor(bddm, bdd_xor(bddm,a3,b3), c2);
bdd c3 = bdd_or(bddm, bdd_and(bddm,a3,b3),bdd_and(bddm,bdd_or(bddm,a3,b3),c2) );

bdd s4 = bdd_xor(bddm, bdd_xor(bddm,a4,b4), c3);
bdd c4 = bdd_or(bddm, bdd_and(bddm,a4,b4),bdd_and(bddm,bdd_or(bddm,a4,b4), ));

bdd s5 = bdd_xor(bddm, bdd_xor(bddm,a5,b5), c4);
bdd c5 = bdd_or(bddm, bdd_and(bddm,a5,b5),bdd_and(bddm,bdd_or(bddm,a5,b5),c4) );

bdd s6 = bdd_xor(bddm, bdd_xor(bddm,a6,b6), c5);
bdd c6 = bdd_or(bddm, bdd_and(bddm,a6,b6),bdd_and(bddm,bdd_or(bddm,a6,b6),c5) );

bdd s7 = bdd_xor(bddm, bdd_xor(bddm,a7,b7), c6);
//////////////////////////////////////Carry Look-Ahead Adder//////////////////////////////////////////
bdd p0 = bdd_xor(bddm,a0,b0);
bdd g0 = bdd_and(bddm,a0,b0);

bdd p1 = bdd_xor(bddm,a1,b1);
bdd g1 = bdd_and(bddm,a1,b1);

bdd p2 = bdd_xor(bddm,a2,b2);
bdd g2 = bdd_and(bddm,a2,b2);

bdd p3 = bdd_xor(bddm,a3,b3);
bdd g3 = bdd_and(bddm,a3,b3);

bdd p4 = bdd_xor(bddm,a4,b4);
bdd g4 = bdd_and(bddm,a4,b4);
```

```
bdd p5 = bdd_xor(bddm,a5,b5);
bdd g5 = bdd_and(bddm,a5,b5);

bdd p6 = bdd_xor(bddm,a6,b6);
bdd g6 = bdd_and(bddm,a6,b6);


bdd p7 = bdd_xor(bddm,a7,b7);
bdd g7 = bdd_and(bddm,a7,b7);

bdd cla_s0 = bdd_xor(bddm, a0, b0);
//c0 is same as go
bdd cla_s1 = bdd_xor(bddm, p1, g0);
bdd cla_c1 = bdd_or(bddm,g1,bdd_and(bddm,p1,g0));

bdd cla_s2 = bdd_xor(bddm, p2, cla_c1);
bdd cla_c2 = bdd_or(bddm,g2,bdd_and(bddm,p2,..
...bdd_or(bddm,g1,bdd_and(bddm,p1,g0))));

bdd cla_s3 = bdd_xor(bddm, p3, cla_c2);
bdd cla_c3 = bdd_or(bddm,g3,bdd_and(bddm,p3,..
..bdd_or(bddm,g2,bdd_and(bddm,p2,bdd_or(bddm,g1,bdd_and(bddm,p1,g0))))));

bdd cla_s4 = bdd_xor(bddm, p4, cla_c3);
bdd cla_c4 = bdd_or(bddm,g4,bdd_and(bddm,p4,..
..bdd_or(bddm,g3,bdd_and(bddm,p3,bdd_or(bddm,g2,bdd_and(bddm,p2,bdd_or(bddm,g1,bdd_and(bddm,p1,g0))))))))

bdd cla_s5 = bdd_xor(bddm, p5, cla_c4);
bdd cla_c5 = bdd_or(bddm,g5,bdd_and(bddm,p5,bdd_or(bddm,g4,bdd_and(bddm,p4,bdd_or(bddm,g3,...
...bdd_and(bddm,p3,bdd_or(bddm,g2,bdd_and(bddm,p2,bdd_or(bddm,g1,bdd_and(bddm,p1,g0)))))))));

bdd cla_s6 = bdd_xor(bddm, p6, cla_c5);
bdd cla_c6 = bdd_or(bddm,g6,bdd_and(bddm,p6,bdd_or(bddm,g5,bdd_and(bddm,p5,....
....bdd_or(bddm,g4,bdd_and(bddm,p4,bdd_or(bddm,g3,bdd_and(bddm,p3,bdd_or(bddm,g2,bdd_and(bddm,p2,....
....bdd_or(bddm,g1,bdd_and(bddm,p1,g0)))))))))));

bdd cla_s7 = bdd_xor(bddm, p7, cla_c6);

//bdd_print_bdd(bddm,cla_s1,NULL, NULL,NULL, stdout);
//bdd_print_bdd(bddm,s1,NULL, NULL,NULL, stdout);


if (s0 == cla_s0)
{
printf("s0 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
```

```
}
if (s1 == cla_s1)
{
printf("s1 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}
if (s2 == cla_s2)
{
printf("s2 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}
if (s3 == cla_s3)
{
printf("s3 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}
if (s4 == cla_s4)
{
printf("s4 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}
if (s5 == cla_s5)
{
printf("s5 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}
if (s6 == cla_s6)
{
printf("s6 Equal\n");
}
else
{
```

```
printf("Not Equal\n");
// print z
}
if (s7 == cla_s7)
{
printf("s7 Equal\n");
}
else
{
printf("Not Equal\n");
// print z
}

return(0);
}
```

1.5 Result

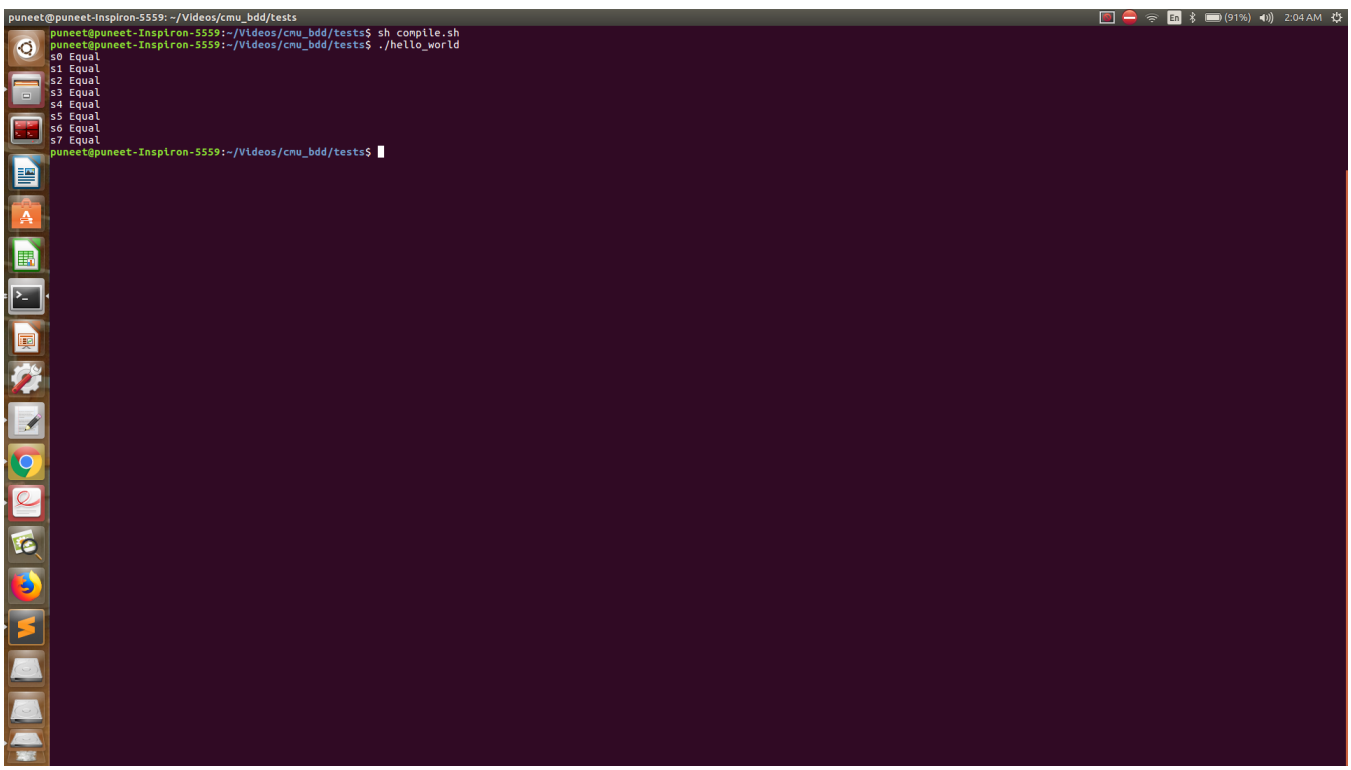

Figure 3. Result