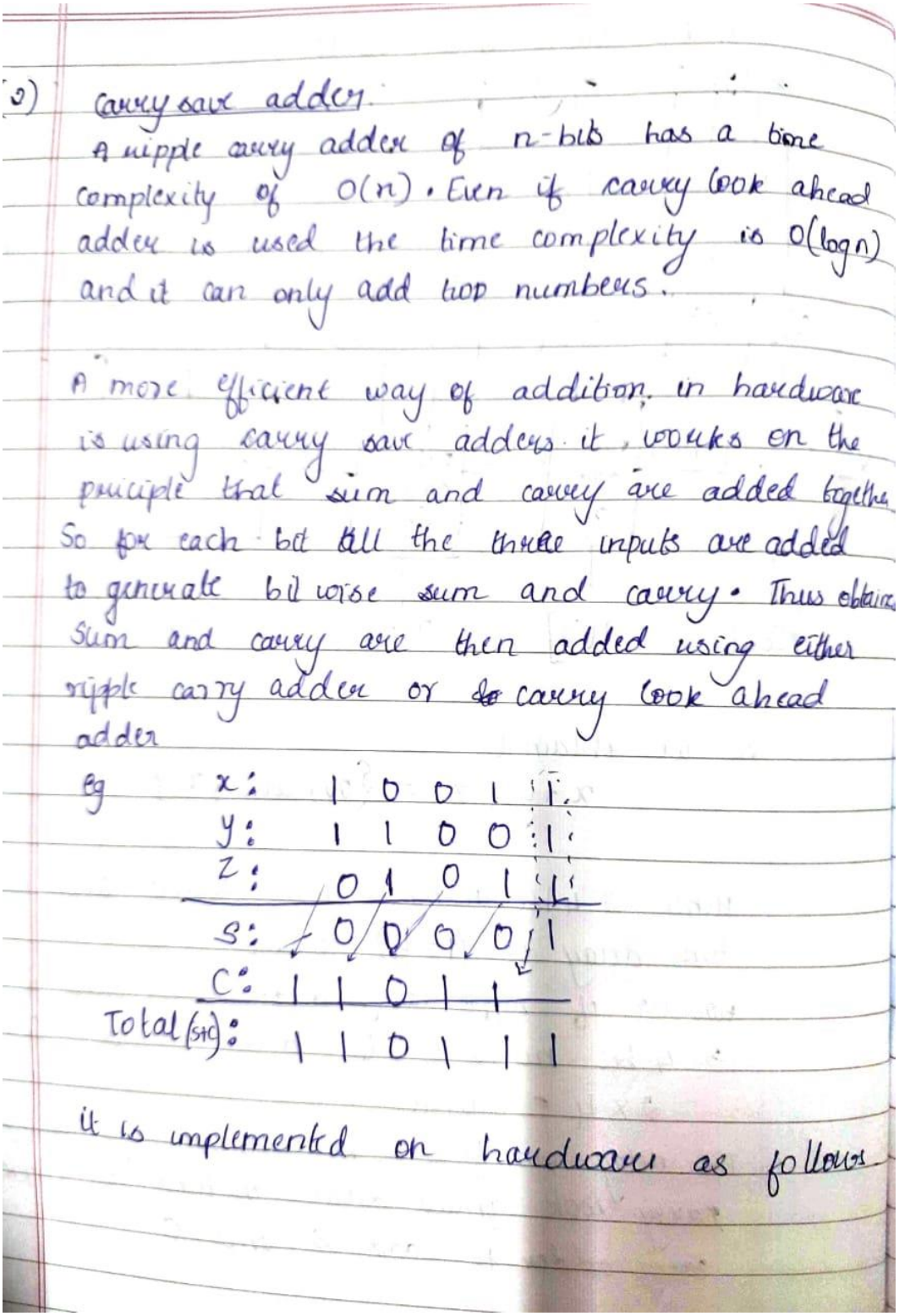
2) Carry Save Adder by instantiating full adder



2) Carry save adder

A ripple carry adder of n-bit has a time complexity of O(n). Even if carry look ahead adder is used the time complexity is O(log n) and it can only add two numbers.

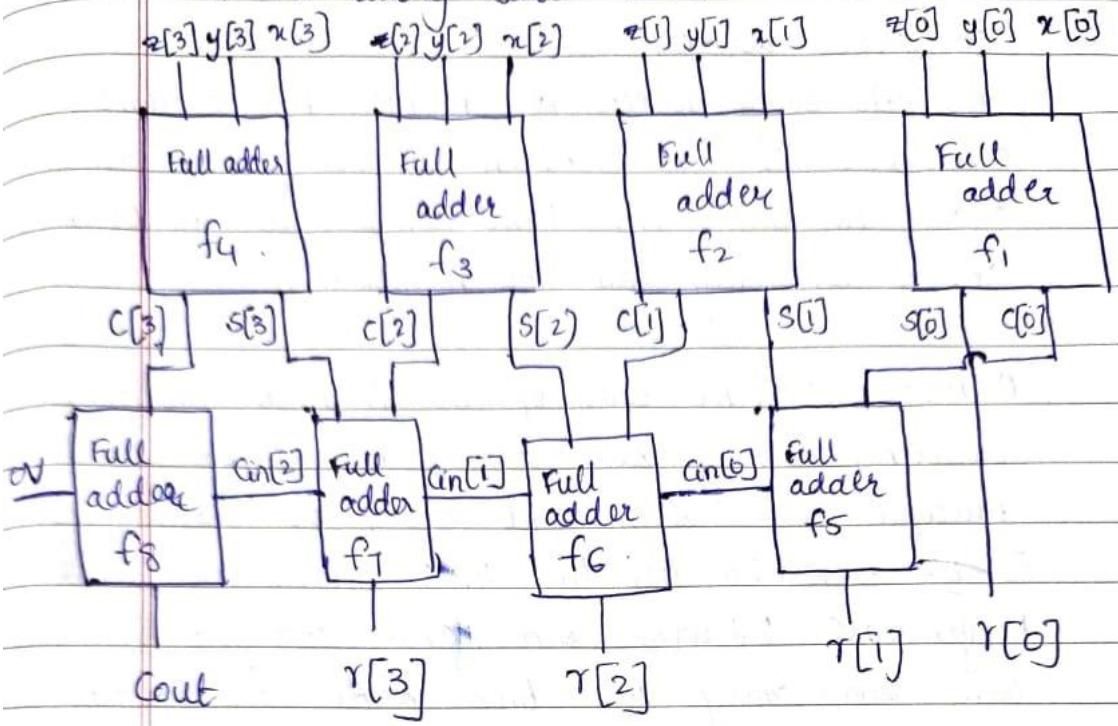
A more efficient way of addition in hardware is using carry save adders it works on the principle that sum and carry are added together. So for each bit all the three inputs are added to generate bit wise sum and carry. Thus obtained sum and carry are then added using either ripple carry adder or carry look ahead adder

eg

x:	1	0	0	1	1
y:	1	1	0	0	1
z:	0	1	0	1	1
s:	0	0	0	0	1
c:	1	1	0	1	1
Total (sum):	1	1	0	1	1

It is implemented on hardware as follows

4-bit carry save adder



so we obtained

$$x + y + z = \{ov, cout, r\}.$$

Hence 3 4-bit numbers were added and the time delay is $(1+4) = 5$ units

However if ripple carry adder was used to add 3 4-bit numbers, the delay would be

$$2 \times 4 = 8 \text{ units.}$$

The delay can further be reduced by using a carry look ahead adder instead of ripple carry adder to add S and C

Module 1: full_adder

```
`timescale 1ns / 1ps
module full_adder(  input x,  input y,   input cin,output r,output cout );
assign r=(x^y)^cin;
assign cout=(x&y)|(cin&(x^y));
endmodule
```

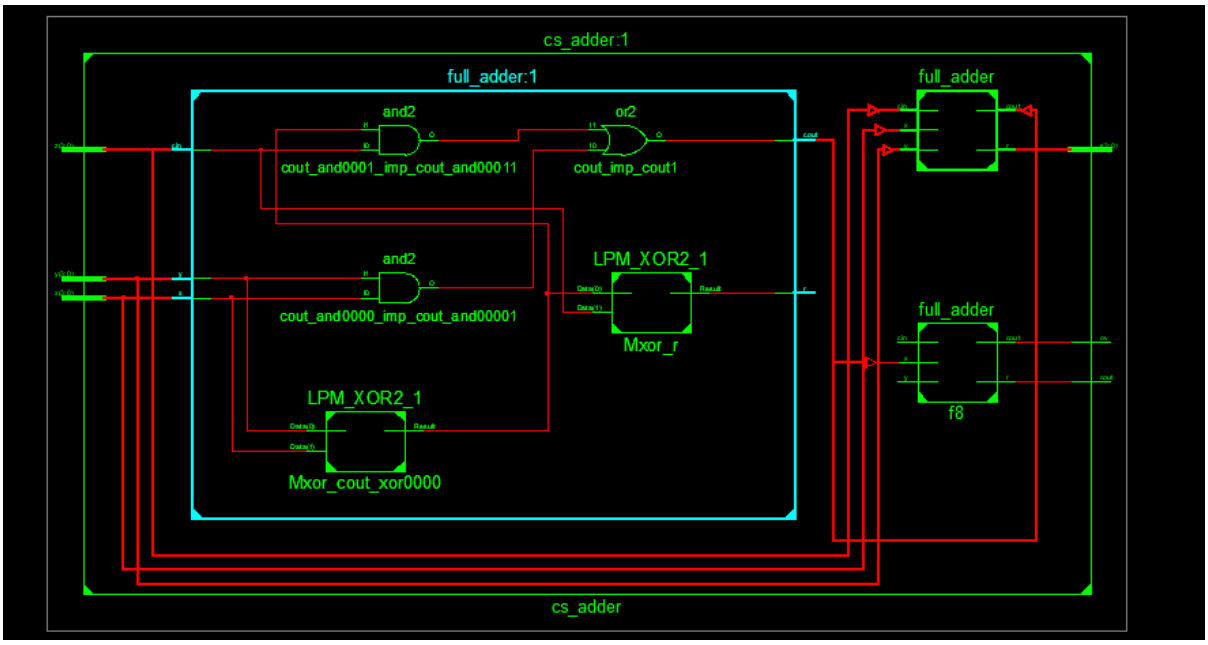
Module 2: 4-bit carry save adder

```
`timescale 1ns / 1ps
module cs_adder(input [3:0] x,input [3:0] y,input [3:0] z,output [3:0] r,output cout,output ov);
wire [3:0] s;
wire [3:0] c;
wire [2:0] cin;
full_adder f1 (x[0],y[0],z[0],s[0],c[0]);
full_adder f2 (x[1],y[1],z[1],s[1],c[1]);
full_adder f3 (x[2],y[2],z[2],s[2],c[2]);
full_adder f4 (x[3],y[3],z[3],s[3],c[3]);

//adding the s and c components to get z and cout

full_adder f5 (.x(s[1]),.y(c[0]),.cin(0),.r(r[1]),.cout(cin[0]));
full_adder f6 (s[2],c[1],cin[0],r[2],cin[1]);
full_adder f7 (s[3],c[2],cin[1],r[3],cin[2]);
full_adder f8 (c[3],0,cin[2],cout,ov);
assign r[0]=s[0];
endmodule
```

Module 2: test bench for carry save adder

```
`timescale 1ns / 1ps
module tb_carrysave;
        // Inputs
        reg [3:0] x;reg [3:0] y;reg [3:0] z;

// Outputs
        wire [3:0] r;wire cout;wire ov;
// Instantiate the Unit Under Test (UUT)
        cs_adder uut (.x(x),.y(y),.z(z),.r(r),.cout(cout),.ov(ov));
        initial begin
                // Initialize Inputs
                x = 0;y = 0;z = 0;
                #15 x = 1;y = 5;z = 5;
                #15 x = 15;y = 5;z = 2;
                #15 x = 15;y = 15;z = 15;
                #15 x = 7;y = 6;z = 8;
                #15 x = 15;y = 5;z = 2;
                #15 x = 12;y = 6;z = 11;
                end
        initial begin
        $monitor("x=%d,y=%d,z=%d,r=%d,cout=%b,ov=%b",x,y,z,r,cout,ov);
        end
        initial begin
        #100 $finish;
        end
    endmodule
```

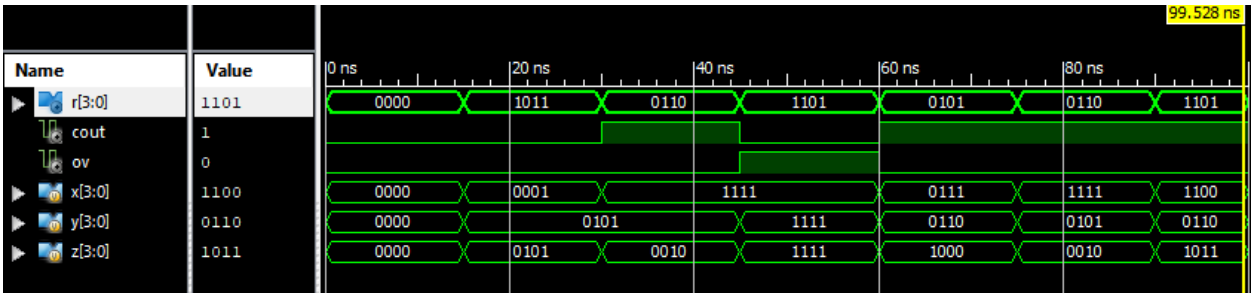
RTL Schematic:



RESULTS:

Console

```
ISim P.58f (signature 0x7708f090)
This is a Full version of ISim.
WARNING:  For instance uut/f5/, width 1 of formal port cin is not equal to width 32 of actual constant.
WARNING:  For instance uut/f8/, width 1 of formal port y is not equal to width 32 of actual constant.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
x= 0,y= 0,z= 0,r= 0,cout=0,ov=0
x= 1,y= 5,z= 5,r=11,cout=0,ov=0
x=15,y= 5,z= 2,r= 6,cout=1,ov=0
x=15,y=15,z=15,r=13,cout=0,ov=1
x= 7,y= 6,z= 8,r= 5,cout=1,ov=0
x=15,y= 5,z= 2,r= 6,cout=1,ov=0
x=12,y= 6,z=11,r=13,cout=1,ov=0
Stopped at time : 100 ns : File "I:/xilinxfiles/carry_save_adder/tb_carrysave.v" Line 65
ISim>
```



Discussion: It has been observed that when an input port is left unconnected in the simulation, the results becomes x or unknown. Hence while instantiating f5 and f8, input is given as 0. Since the input is constant which is allotted 16 bits in Verilog, simulator generates a warning, which however doesn't affect the output as all 32 bits are zero and hence the input bit is still zero.

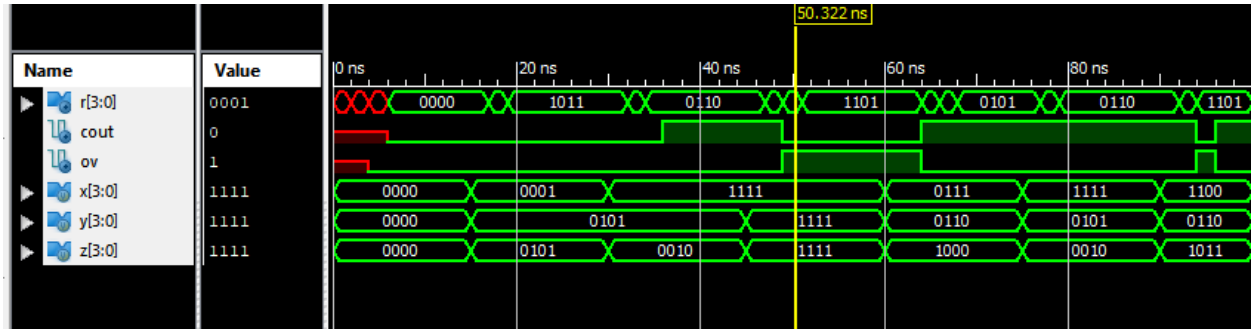
For the above module delay is calculated as follows:

It is assumed that each full adder generates 1unit delay, then

The intermediate s[3:0] and c[3:0] are generated after 1 unit of delay O(1)

To add s and c we use a ripple carry adder whose time complexity is O(n) so the total delay is 5 units.

If sum and carry have been assigned 2 units delay in the full adder module, a 10 units delay can be observed in the resulting waveform attached below.



Conclusion: A 3:8 decoder and carry save adder had been instantiated using 2:4 decoder and full adder modules respectively. The delay advantage of the carry save adder has been discussed. If a carry look ahead adder was used to add s and c of carry save adder then the delay can further be reduced to log(n)+1 units to add 3 n-bit numbers. The test benches were used to verify the results for both projects.