30 Days of RTL Coding

-By T Tushar Shenoy

Day 29

Problem Statement: Implementing Carry Look Ahead Adder.

Theory:

A **carry-look ahead adder** (CLA) or **fast adder** is a type of electronics adder used in digital logic. A carry-look ahead adder improves speed by reducing the amount of time required to determine carry bits. It can be contrasted with the simpler, but usually slower, ripple-carry adder (RCA), for which the carry bit is calculated alongside the sum bit, and each stage must wait until the previous carry bit has been calculated to begin calculating its own sum bit and carry bit. The carry-look ahead adder calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of the larger-value bits of the adder.

Advantages of Carry Look-ahead Adder

In this adder, the propagation delay is reduced. The carry output at any stage is dependent only on the initial carry bit of the beginning stage. Using this adder it is possible to calculate the intermediate results. This adder is the fastest adder used for computation.

Applications

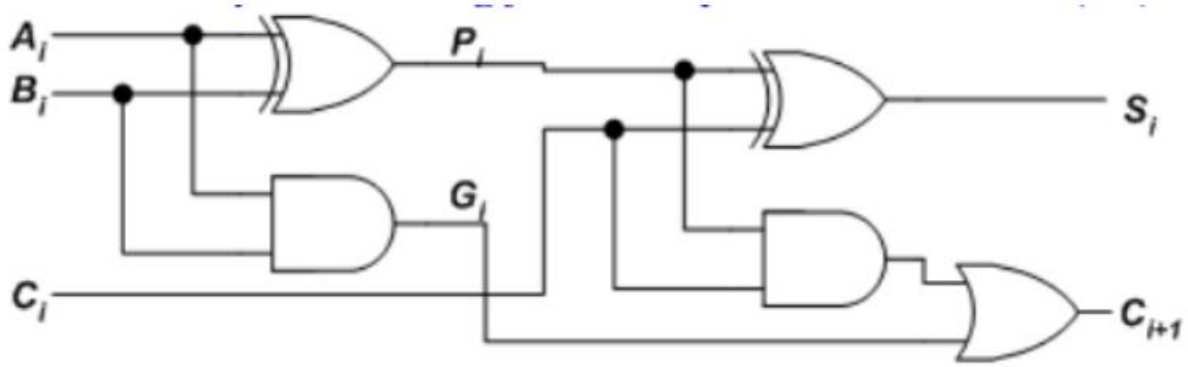
High-speed Carry Look-ahead Adders are used as implemented as IC's. Hence, it is easy to embed the adder in circuits. By combining two or more adders calculations of higher bit Boolean functions can be done easily. Here the increase in the number of gates is also moderate when used for higher bits.

For this Adder there is a trade-off between area and speed. When used for higher bit calculations, it provides high speed but the complexity of the circuit is also increased thereby increasing the area occupied by the circuit. This adder is usually implemented as 4-bit modules which are cascaded together when used for higher calculations. This adder is costlier compared to other adders.

For Boolean computation in computers, adders are being used regularly. Charles Babbage implemented a mechanism for anticipating the carry bit in computers, to reduce the delay caused by the ripple carry adders. While designing a system, the speed of computation is the highest deciding factor for

a designer. In 1957, Gerald B. Rosenberger patented the modern Binary Carry Look-ahead Adder. Based on the analysis of gate delay and simulation, experiments are being conducted to modify the circuit of this adder to make it even faster.

Carry Look Ahead logic



In this circuit, the 2 internal signals P_i and G_i are given by:

$$P_i = A_i \oplus B_i \quad \text{.....(1)}$$

$$G_i = A_i B_i \quad \text{.....(2)}$$

The output sum and carry can be defined as :

$$S_i = P_i \oplus C_i \quad \text{.....(3)}$$

$$C_{i+1} = G_i + P_i C_i \quad \text{.....(4)}$$

G_i is known as the ***carry Generate*** signal since a carry (C_{i+1}) is generated whenever G_i =1, regardless of the input carry (C_i).

P_i is known as the ***carry propagate*** signal since whenever P_i =1, the input carry is propagated to the output carry, i.e., C_{i+1}. = C_i (note that whenever P_i =1, G_i =0).

$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1 (G_0 + P_0C_0)$$

$$= G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$$

$$C_4 = G_3 + P_3C_3$$

$$= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$$

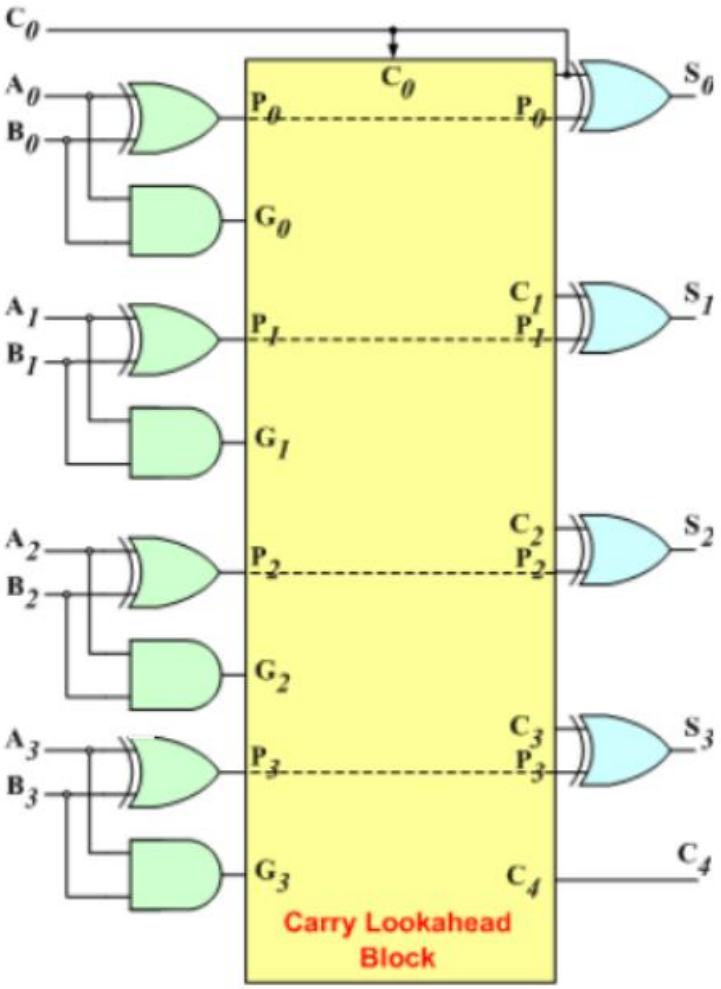


FIG: Carry Look ahead Block

Verilog Code:

```
//Verilog Code for 1 Bit Full-adder
module full_adder_1bit(a,b,cin,sum);
input a,b,cin;
output sum;

assign sum=a^b^cin;

endmodule



//Verilog Code for Carry Look Ahead Adder
module carry_look_ahead_adder(A,B,Cin,S,Cout);
input [3:0]A,B;
input Cin;
output [3:0]S;
output Cout;

wire C1,C2,C3;

full_adder_1bit  F0(A[0],B[0],Cin,S[0]);
full_adder_1bit  F1(A[1],B[1],C1,S[1]);
full_adder_1bit  F2(A[2],B[2],C2,S[2]);
full_adder_1bit  F3(A[3],B[3],C3,S[3]);

wire [3:0]P,G;

assign P[0]=A[0]^B[0];
```

```
assign P[1]=A[1]^B[1];
assign P[2]=A[2]^B[2];
assign P[3]=A[3]^B[3];

assign G[0]=A[0]&B[0];
assign G[1]=A[1]&B[1];
assign G[2]=A[2]&B[2];
assign G[3]=A[3]&B[3];

assign C1=G[0]|(P[0]&Cin);
assign C2=G[1]|(P[1]&G[0])|(P[1]&P[0]&Cin);
assign C3=G[2]|(P[2]&G[1])|(P[2]&P[1]&G[0])|(P[2]&P[1]&P[0]&Cin);
assign
Cout=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]&Cin);

endmodule
```

Testbench Code:

```
//Verilog Testbench Code for Carry Look Ahead Adder
module carry_look_ahead_adder_tb();

reg [3:0]A,B;
reg Cin;
wire [3:0]S;
wire Cout;

carry_look_ahead_adder dut(.A(A),.B(B),.Cin(Cin),.S(S),.Cout(Cout));

initial begin
   A=4'b0000;B=4'b0000;Cin=0;
#5 A=4'b0001;B=4'b0001;
#5 A=4'b0010;B=4'b0010;
#5 A=4'b0011;B=4'b0011;
#5 A=4'b0100;B=4'b0100;
#5 A=4'b0101;B=4'b0101;
#5 A=4'b0110;B=4'b0110;
#5 A=4'b0111;B=4'b0111;
#5 A=4'b1000;B=4'b1000;
#5 A=4'b1001;B=4'b1001;
#5 A=4'b1010;B=4'b1010;
#5 A=4'b1011;B=4'b1011;
#5 A=4'b1100;B=4'b1100;
#5 A=4'b1101;B=4'b1101;
#5 A=4'b1110;B=4'b1110;
#5 A=4'b1111;B=4'b1111;
// Can Add More Cases
```

```
#5 $finish;
end

endmodule
```

Schematic:

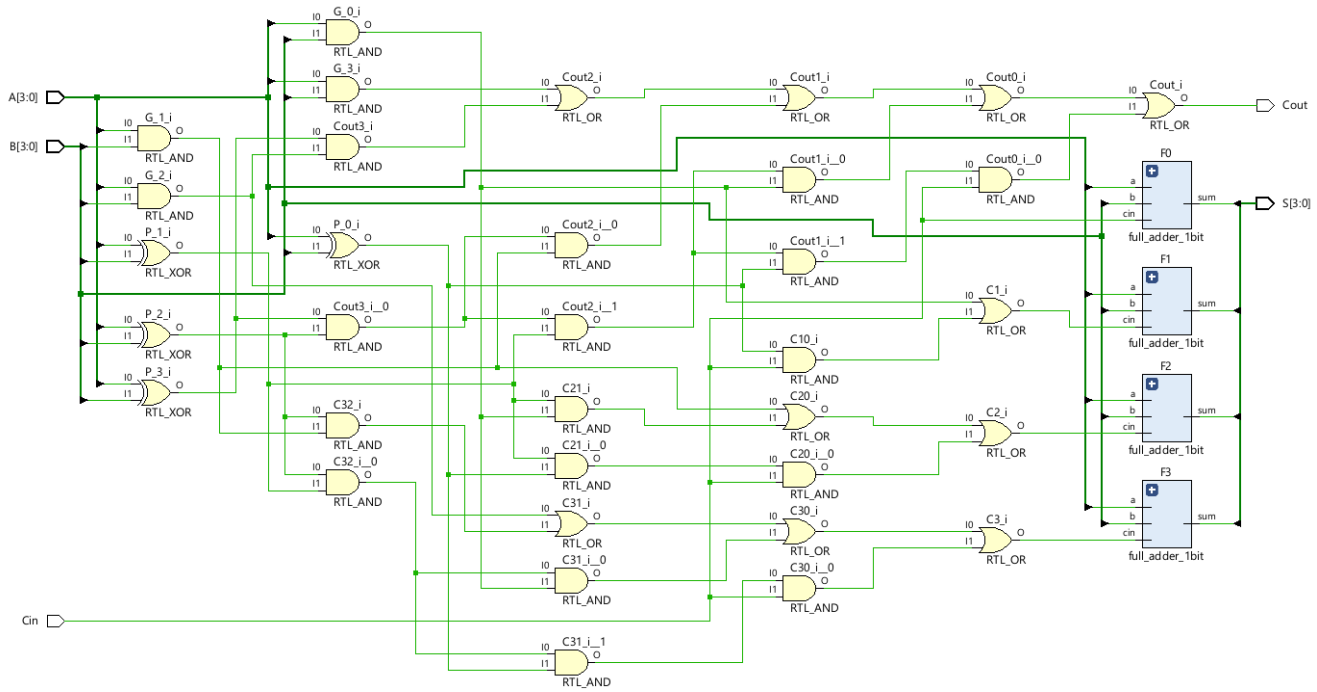
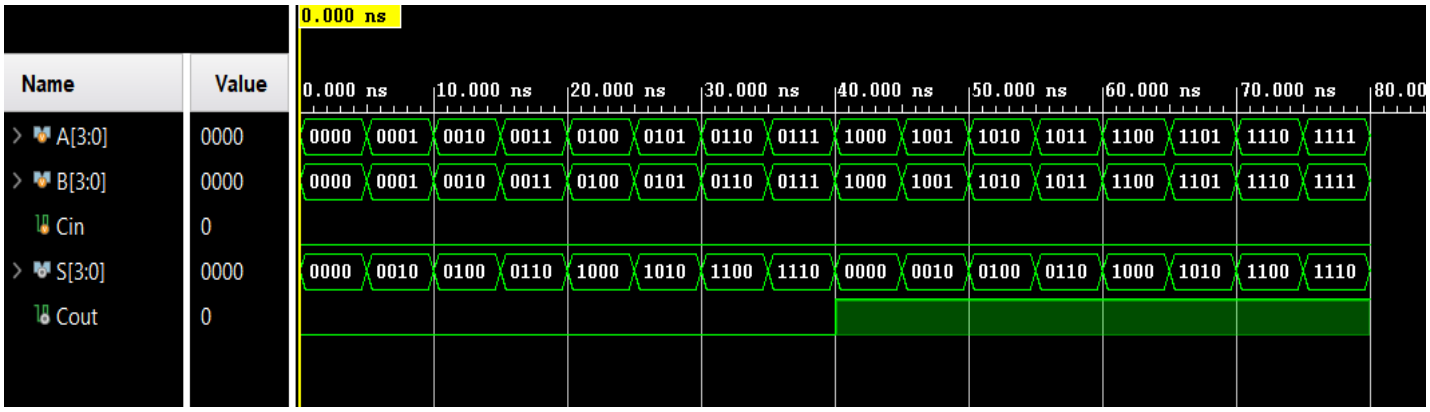


FIG Carry Look Ahead Adder

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-29-Carry-Look-Ahead-Adder