30 Days of RTL Coding

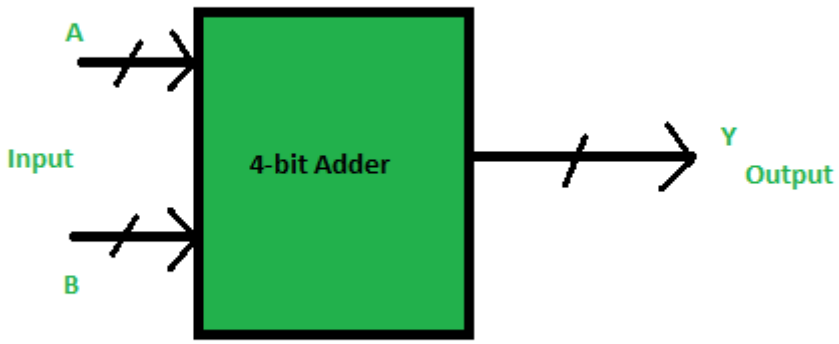
-By T Tushar Shenoy

Day 12

Problem Statement: Implementing a one Digit BCD Adder using structural Style.

Theory:

BCD stands for binary coded decimal. It is used to perform the addition of BCD numbers. A BCD digit can have any of ten possible four-bit representations. Suppose, we have two 4-bit numbers A and B. The value of A and B can vary from 0(0000 in binary) to 9(1001 in binary) because we are considering decimal numbers.



The output will vary from 0 to 18 if we are not considering the carry from the previous sum. But if we are considering the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19). When we are simply adding A and B, then we get the binary sum. Here, to get the output in BCD form, we will use BCD Adder.

One digit BCD adder example

Correct: Result is in BCD.

$$\begin{array}{r} 0110 = 6 \\ +0011 = +3 \\ \hline 1001 = 9 \end{array}$$

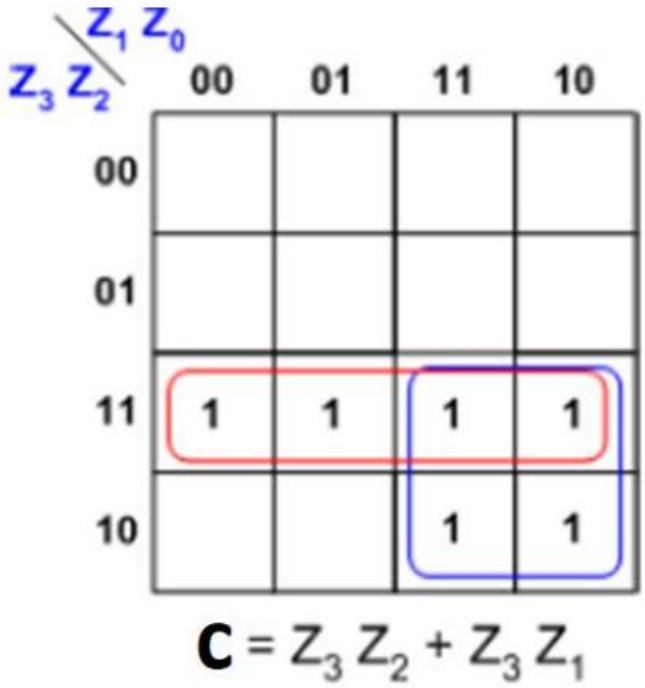
Wrong: Result is not in BCD.

$$\begin{array}{r} 0101 = 5 \\ +0111 = +7 \\ \hline 1100 = 12 \end{array}$$

One digit BCD Adder

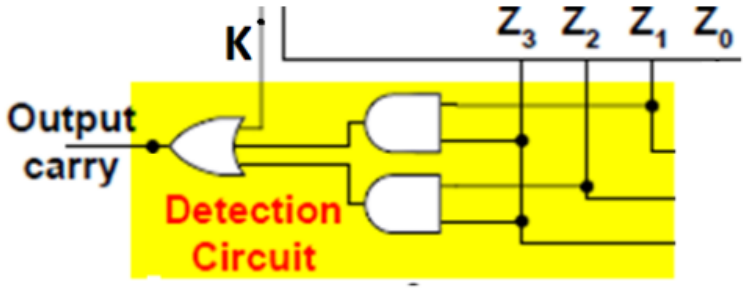
	Binary Sum					BCD Sum					Decimal
	K	Z_3	Z_2	Z_1	Z_0	C	S_3	S_2	S_1	S_0	
0-15	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	1	1
	0	0	0	1	0	0	0	0	1	0	2
	0	0	0	1	1	0	0	0	1	1	3
	0	0	1	0	0	0	0	1	0	0	4
	0	0	1	0	1	0	0	1	0	1	5
	0	0	1	1	0	0	0	1	1	0	6
	0	0	1	1	1	0	0	1	1	1	7
	0	1	0	0	0	0	1	0	0	0	8
	0	1	0	0	1	0	1	0	0	1	9
	0	1	0	1	0	1	0	0	0	0	10
	0	1	0	1	1	1	0	0	0	1	11
	0	1	1	0	0	1	0	0	1	0	12
	0	1	1	0	1	1	0	0	1	1	13
	0	1	1	1	0	1	0	1	0	0	14
	0	1	1	1	1	1	0	1	0	1	15
>15	1	0	0	0	0	1	0	1	1	0	16
	1	0	0	0	1	1	0	1	1	1	17
	1	0	0	1	0	1	1	0	0	0	18
	1	0	0	1	1	1	1	0	0	1	19

Design of Correction Logic



- Solve for C using 4 variable K-Map for (0-15) entries of input variables (Z_3 Z_2 Z_1 Z_0)
- We get C = Z_3 Z_{2+} Z_3 Z_1
- Combining the binary carry out (K) and greater than '9' condition, the BCD correction logic produces the logic

Correction logic = K + (Z_3 Z_2 + Z_3 Z_1)



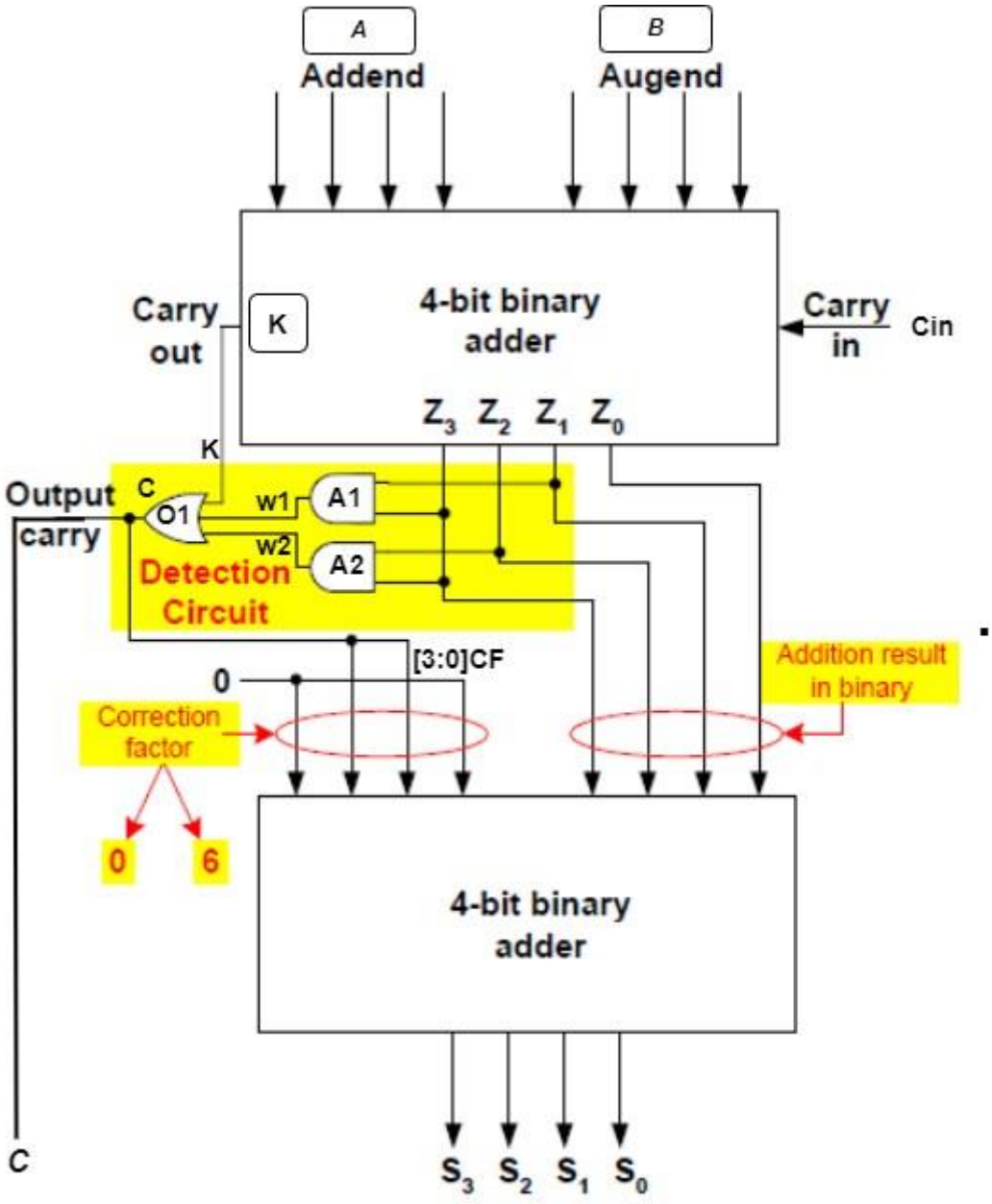


FIG: Block Diagram of one Digit BCD Adder

Verilog Code:

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

assign sum=a^b^cin;
assign carry=(a&b)|(b&cin)|(a&cin);

endmodule

//Verilog Code for Parallel Adder/4 Bit Adder
module parallel_adder(A,B,Cin,S,Cout);
input [3:0]A,B;
input Cin;
output [3:0]S;
output Cout;

wire c0,c1,c2;
full_adder_1bit  F0(A[0],B[0],Cin,S[0],c0);
full_adder_1bit  F1(A[1],B[1],c0,S[1],c1);
full_adder_1bit  F2(A[2],B[2],c1,S[2],c2);
full_adder_1bit  F3(A[3],B[3],c2,S[3],Cout);

endmodule
```

```
//Verilog Code for One Digit BCD Adder
module one_digit_BCD_adder(A,B,Cin,S,C);
input [3:0]A,B;
input Cin;

output [3:0]S;
output C;

wire [3:0]Z;
wire K;

parallel_adder P1(.A(A),.B(B),.Cin(Cin),.S(Z),.Cout(K));
// Correction Logic for BCD Addition
wire w1,w2;
and A1(w1,Z[1],Z[3]);
and A2(w2,Z[2],Z[3]);

or O1(C,w1,w2,K);

wire [3:0]CF;
assign CF={1'b0,C,C,1'b0};

parallel_adder P2(.A(CF),.B(Z),.Cin(Cin),.S(S));

endmodule
```

Testbench Code:

```
//Verilog Testbench Code for one digit BCD Adder
module one_digit_BCD_adder_tb();

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

one_digit_BCD_adder dut(A,B,Cin,S,C);

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;
// Can Add More Cases
#5 $finish;
end

endmodule
```

Schematic:

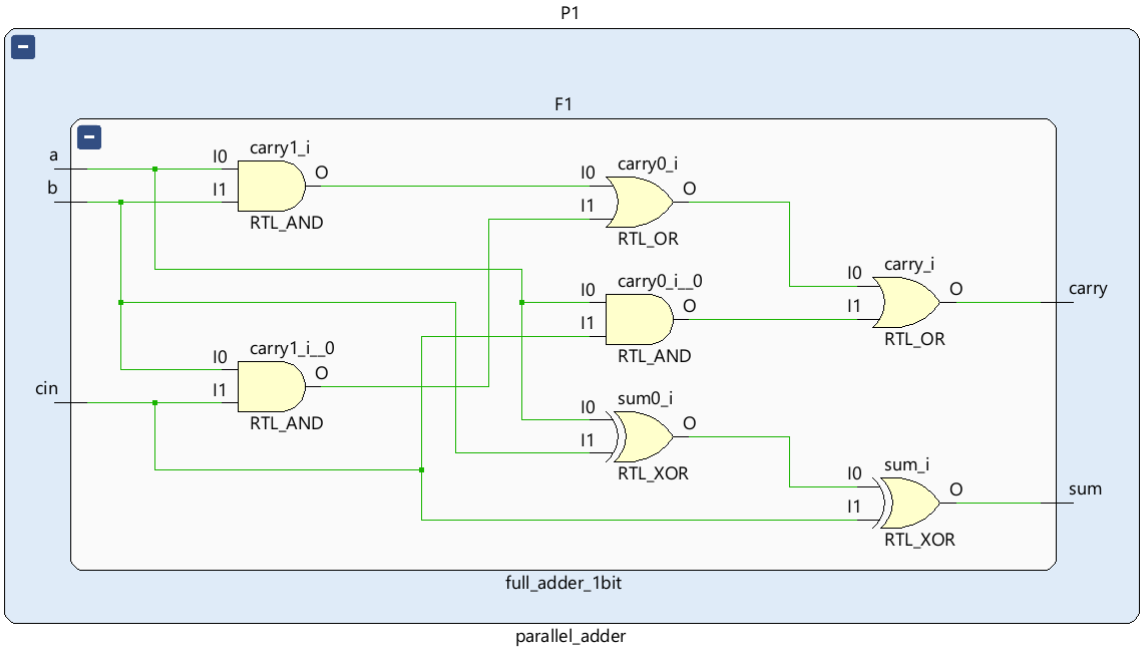


FIG: Full adder Schematic

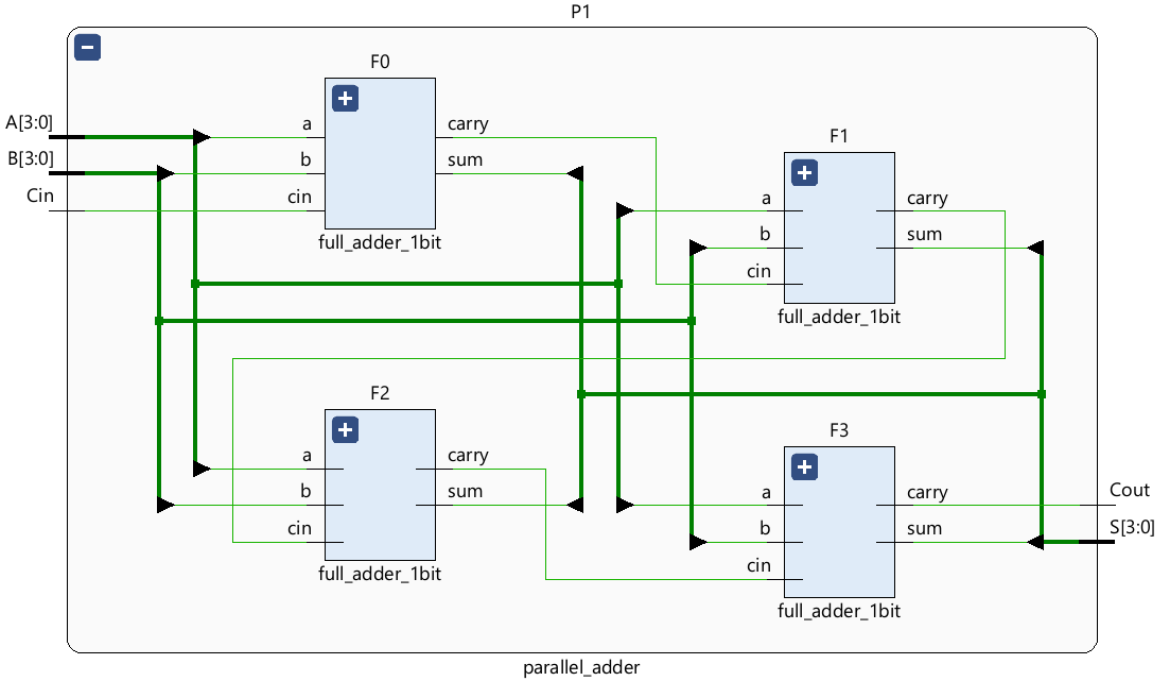


FIG: Parallel adder/Four Bit Adder Schematic

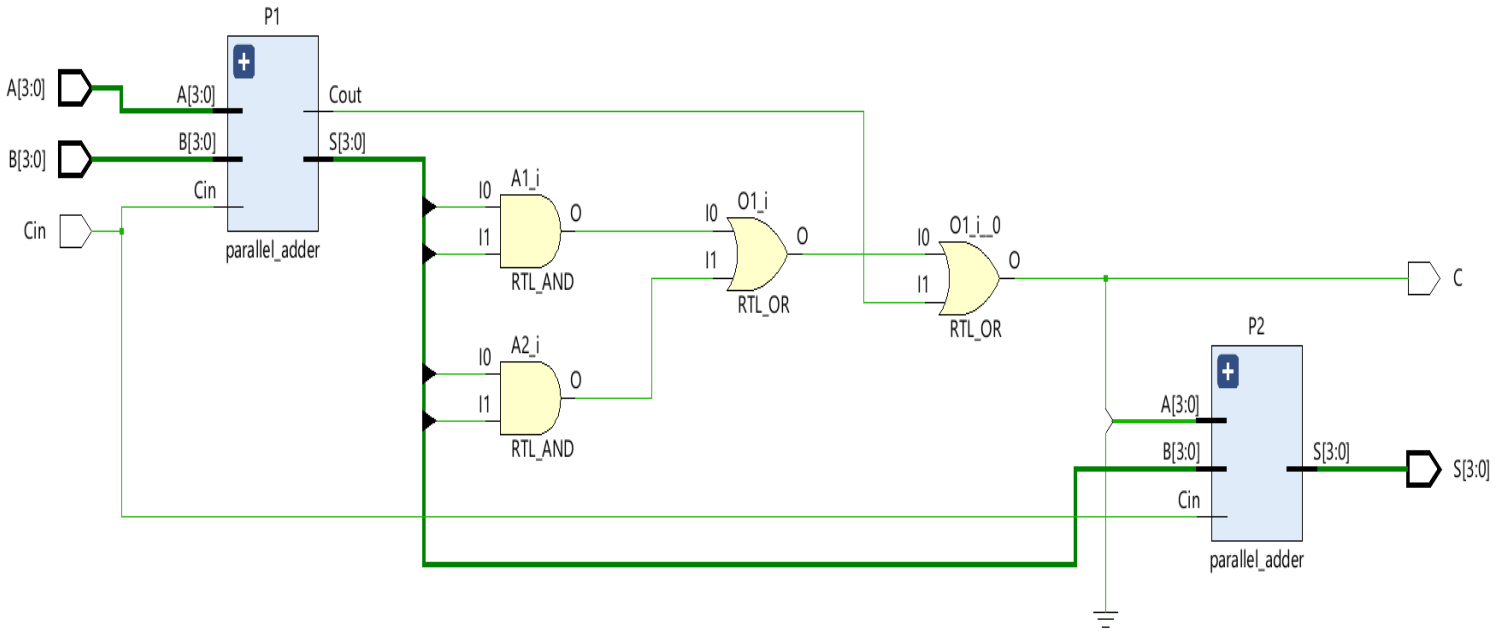
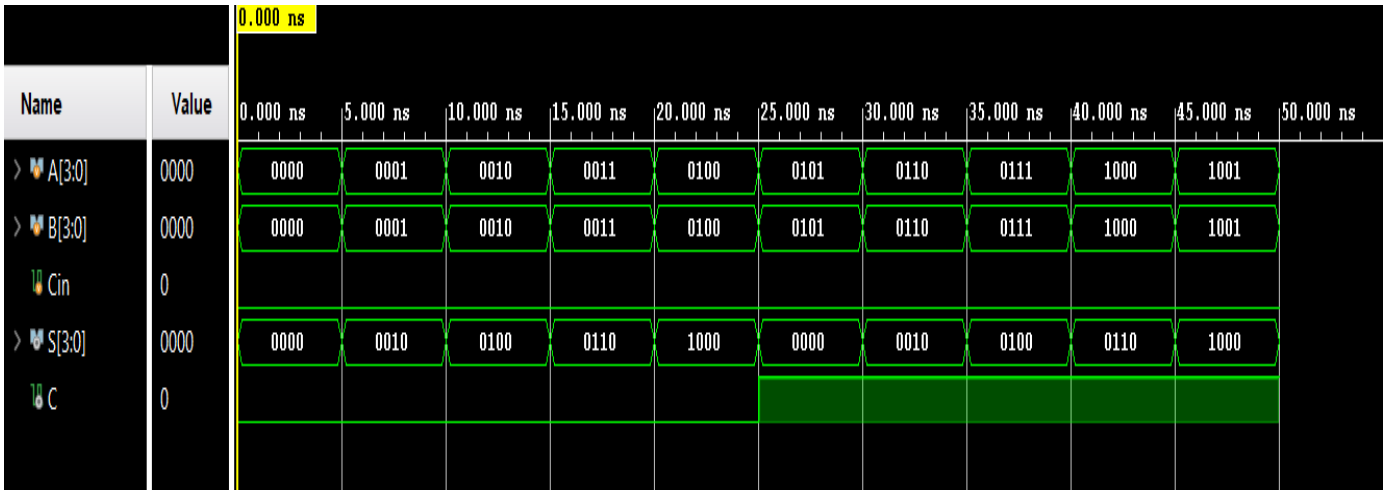


FIG: One Digit BCD Adder Schematic

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-12-One-Digit-BCD-Adder