Day5-4bit Carry Look Ahead Adder #75daysRTL

A 4-bit Carry Look Ahead Adder is an adder that adds 4 bits of input a and b to produce sum and carry as output. The most basic type of adder is RCA(Ripple Carry Adder) where we cascade each output of an adder to the next adder. Using RCA there is a carry delay at each step of the process.

CLA(Carry Look Ahead Adder) is used to reduce the delay in carry generation. In CLA the carry at each stage is generated using CLA Logic Block where A&B are given as inputs. CLA is used to produce faster outputs with an increase in hardware complexity.

Generation of 4bit CLA Adder- To implement 4bit CLA Adder we require 4 full adders & Carry Logic (Propagate &Generate).

Carry Lookahead Logic-

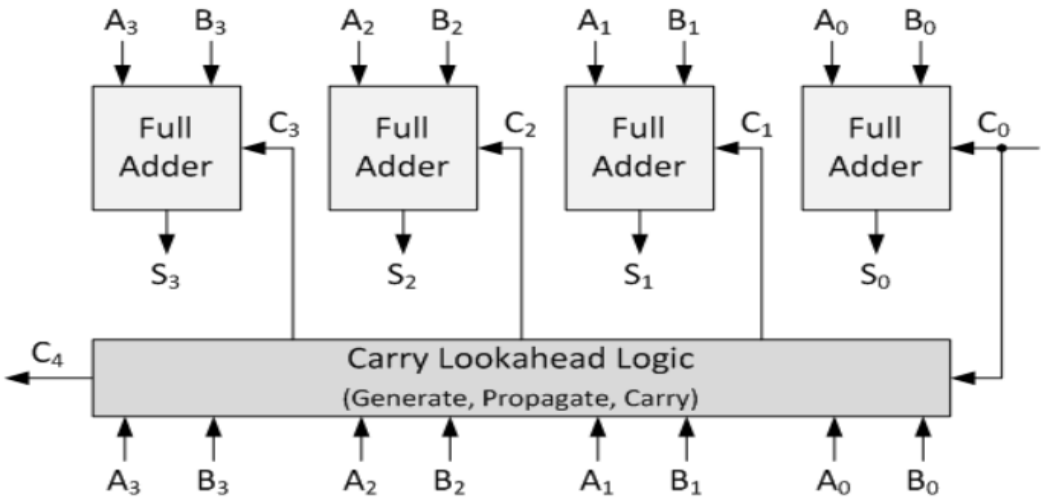
- It consists of 2 functions **Carry Generate (G)** & **Carry Propagate (P)** to increase the speed
- **Carry Generate (G):** This function denotes how the carry is generated for single-bit two inputs regardless of any input carry.
  G = A.B
- **Carry Propagate (P):** This function denotes when the carry is propagated to the next stage with an addition whenever there is an input carry.
  P=A^B^Cin

Input - A	Input - B	Output - Carry-In	Description
0	0	1	not propagated (0)
0	1	1	propagated (1)
1	0	1	propagated (1)
1	1	1	propagated (1)

- **Carry computation function for next stage**

  Cj+1 = Gj+(Pj·Cj) = Aj·Bj + (Aj + Bj)·Cj

Block Diagram –



Verilog Code –

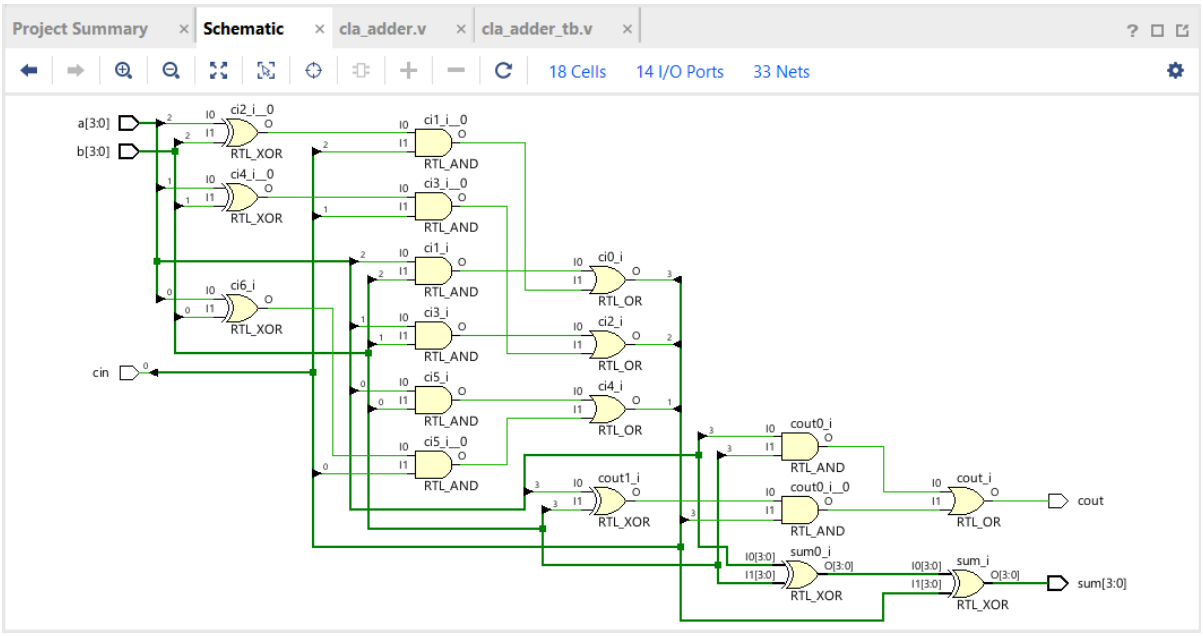
```
module cla_adder(
input [3:0]a,b, input cin,
output [3:0]sum, output cout   );
 wire [3:0] ci;
 assign ci[0] = cin;
  assign ci[1] = (a[0] & b[0]) | ((a[0]^b[0]) & ci[0]);
  assign ci[2] = (a[1] & b[1]) | ((a[1]^b[1]) & ((a[0] & b[0]) | ((a[0]^b[0]) & ci[0])));
  assign ci[3] = (a[2] & b[2]) | ((a[2]^b[2]) & ((a[1] & b[1]) | ((a[1]^b[1]) & ((a[0] & b[0]) | ((a[0]^b[0]) & ci[0])))));
  assign cout  = (a[3] & b[3]) | ((a[3]^b[3]) & ((a[2] & b[2]) | ((a[2]^b[2]) & ((a[1] & b[1]) | ((a[1]^b[1]) & ((a[0] & b[0]) | ((a[0]^b[0]) & ci[0])))))));
  assign sum = a^b^ci;
endmodule
```

TestBench Code-

```
module cla_adder_tb();
reg [3:0]a,b;
reg cin;
wire [3:0] sum;
wire cout;
cla_adder dut(a,b,cin,sum,cout);
initial
begin
a=2;b=3;
cin=0;
#10;
a=1;cin=8;
cin=1;
#10;
a=9;b=4;
cin=0;
#10;
```

```
a=5;b=9;
cin=1;
#10;
a=3;b=11;
cin=1;
$finish;
end
endmodule
```

Schematic View of 4bit CLA Adder-



Simulation Result of 4bit CLA Adder-