# IOortldaysChallenge

#DAY14 DATE: 12/10//2023

## AIM:

**TO DESIGN A 3-BIT BINARY RIPPLE CARRY ADDER**

## DESIGN CODE:

```
// Full Adder Module
module Full_adder(
  // Inputs
  input   A,       // First input A
  input   B,       // Second input B
  input   c_in,    // Carry-in
  // Outputs
  output  S,       // Sum
  output  c_out    // Carry-out
);

// Sum calculation in the Full Adder
assign S = A ^ B ^ c_in;
```

```
// Carry calculation in the Full Adder
assign c_out = (A & B) | (B & c_in) | (A & c_in);
endmodule

// Ripple Carry Adder Module
module ripple_carry_adder(
  // Inputs
  input  [2:0] A,     // 3-bit input A
  input  [2:0] B,     // 3-bit input B
  input        c_in,  // Carry-in signal
  // Outputs
  output       carry_out, // Carry-out signal
  output [2:0] S          // 3-bit sum
);
wire [1:0] carry; // Declare carry[0] and carry[1] as wires

// Instantiate Full Adder 0 for the least significant bit
Full_adder FA0(
.A(A[0]),
.B(B[0]),
.c_in(c_in),
.S(S[0]),
.c_out(carry[0])
);
```

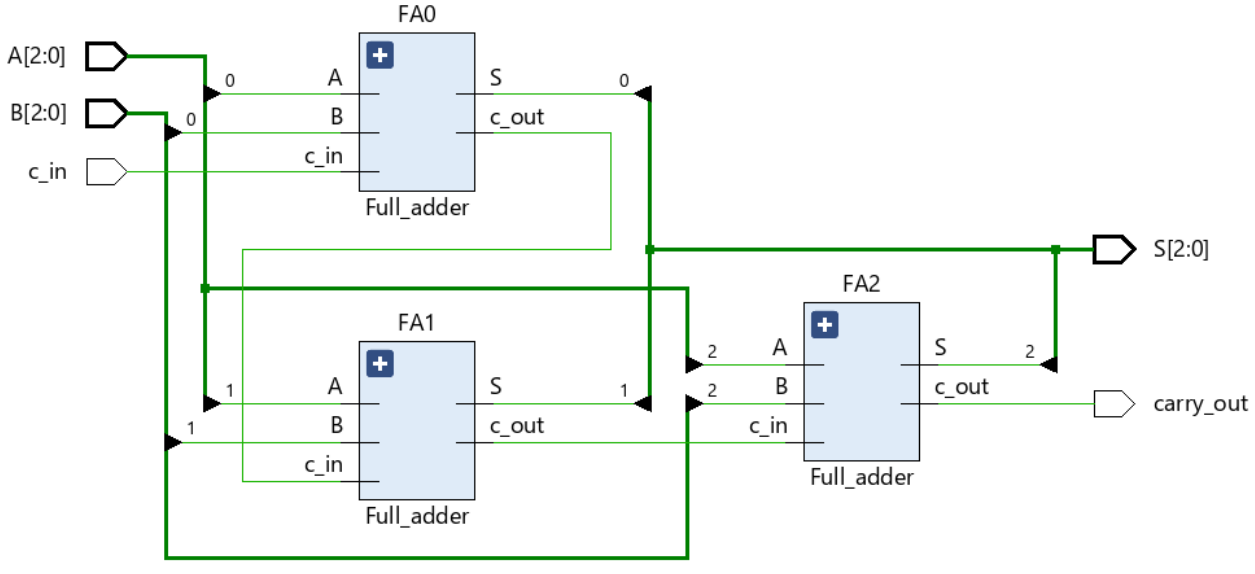
```
// Instantiate Full Adder 1 for the middle bit
Full_adder FA1(
  .A(A[1]),
  .B(B[1]),
  .c_in(carry[0]),
  .S(S[1]),
  .c_out(carry[1])
);

// Instantiate Full Adder 2 for the most significant bit
Full_adder FA2(
  .A(A[2]),
  .B(B[2]),
  .c_in(carry[1]),
  .S(S[2]),
  .c_out(carry_out)
);
endmodule
```

# TEST BENCH CODE:

```
module ripple_carry_adder_tb;

  reg [2:0] A, B;  // Input signals A and B, 3 bits each

  reg c_in;        // Carry-in signal

  wire [2:0] S;    // Output sum, 3 bits

  wire carry_out;  // Carry-out signal


  ripple_carry_adder uut(  // Instantiation of the ripple carry
adder

    .A(A),                 // Connect A to the A port of the
adder

    .B(B),                 // Connect B to the B port of the
adder

    .c_in(c_in),           // Connect c_in to the carry-in port
of the adder

    .S(S),                 // Connect S to the output sum port of
the adder

    .carry_out(carry_out) // Connect carry_out to the carry-out
port of the adder

  );

  // Stimulus generation

  initial begin

    repeat(3) begin  // Generate 3 test cases

      A = $random;   // Assign a random 3-bit value to A

      B = $random;   // Assign a random 3-bit value to B
```

```
      c_in = 0;       // Set the carry-in to 0
      #100;           // Wait for 100 time units before the next
test case
    end
    repeat(3) begin  // Generate 3 test cases
      A = $random;    // Assign a random 3-bit value to A
      B = $random;    // Assign a random 3-bit value to B
      c_in = 1;       // Set the carry-in to 1
      #100;           // Wait for 100 time units before the next
test case
    end
    $finish;           // Finish the simulation
  end
endmodule
```

# SCHEMATICS:




# SIMULATION: