DAY-3

FULL ADDER

EXPLANATION:

Full adder is a digital circuit that performs addition of three binary digits: two inputs, typically referred to as A and B, and a carry input, often denoted as Cin. It produces a **sum output (S) and a carry-out (Cout)** based on the input values.

The logic equations for a full adder are as follows:

Sum (S) = A XOR B XOR Cin

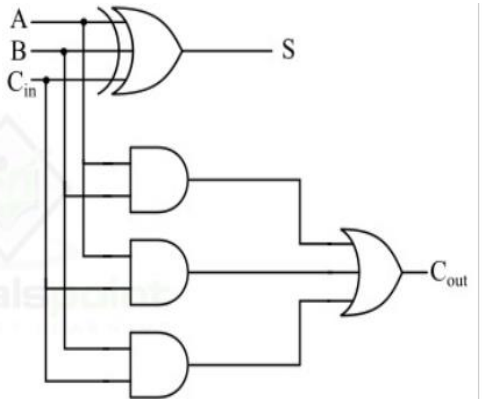
Carry-out (Cout) = (A AND B) OR (B AND Cin) OR (Cin AND A)

A full adder circuit adds three binary digits, where two are the inputs and one is the carry forwarded from the previous addition

Block diagram

Truth Table

Inputs			Outputs	
A	B	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



ADVANTAGES:

- Full adder provides facility to add the carry from the previous stage.
- The power consumed by the full adder is relatively less as compared to half Adder

- Full adder can be easily converted into a half subtractor just by adding a NOT gate in the circuit.
- Full adder produces higher output that half adder.
- Full adder is one of the essential part of critic digital circuits like multiplexers.
- Full adder performs operation at higher speed.

APPLICATIONS:

- Full adders are used in ALUs (arithmetic logic units) of CPUs of computers.
- Full adders are used in calculators.
- Full adders also help in carrying out multiplication of binary numbers.
- Full adders are also used to realize critic digital circuits like multiplexers.
- Full adders are used to generate memory addresses.
- Full adders are also used in generation of program counterpoints.
- Full adders are also used in GPU (Graphical Processing Unit).

RTL CODE

DATAFLOW METHOD:

```
module full_adder(A,B,C,sum,carry);
  input A,B,C;
  output sum,carry;
  assign {sum,carry} = A+B+C;
endmodule
```

STRUCTURAL METHOD:

```
module full_adder(A,B,C,sum,carry);
  input A,B,C;
  output sum,carry;
```

```
  wire w1,w2,w3;
  xor b1(sum,A^B^C);
  and g1(w1,A,B);
  and g2(w1,C,B);
  and g3(w1,A,C);
  or g4 (carry,w1,w2,w3);

endmodule
```

BEHAVIORAL METHOD:

```
module full_adder(A,B,C,sum,carry);
  input A,B,C;
  output sum,carry;
  always@(*)
    begin
      sum=A^B^C;
      carry=A&B|B&C|C&A;
    end
endmodule
```

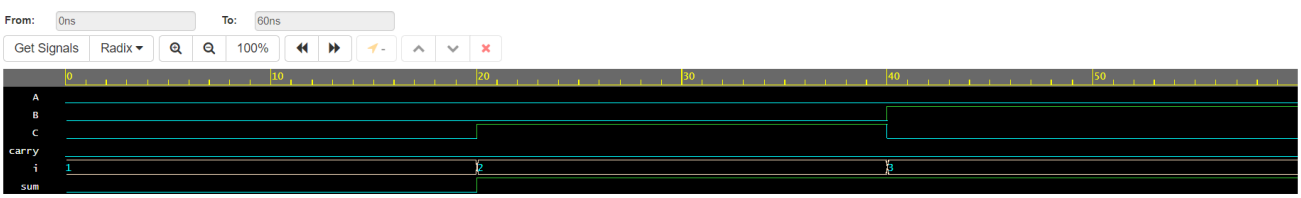
TEST BENCH:

```
module testbench;

  reg A,B,C;
  wire sum,carry;
  integer i;
```

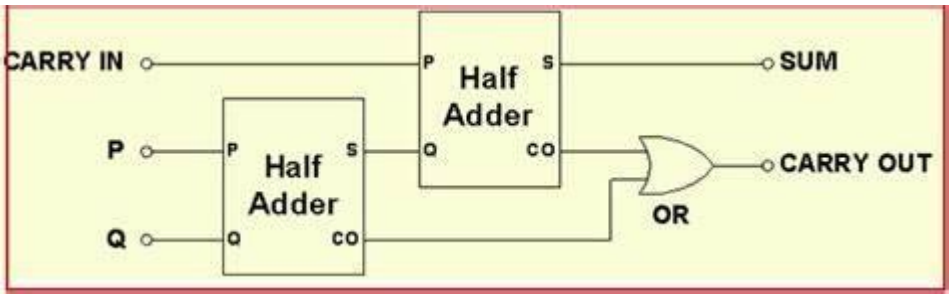
```
  full_adder a1(A,B,C,sum,carry);
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end

  initial
    begin
      A=0;B=0;C=0;
    end
  initial
    begin
      for(i=1;i<8;i=i+1)
        begin
          #20 {A,B,C}=i;
        end
    end
  initial
    begin
      #60 $finish();
    end
endmodule
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

FULL ADDER USIN TWO HALF ADDERS



From the logic diagram of the full adder using half adders, it is clear that we require two XOR gates, two AND gates and one OR gate for the implementation of a full adder circuit using half-adders.

However, the implementation of full adder using half adder has a major disadvantage that is the increased propagation delay. That means, the input bits must propagate through several gates in succession that increases the total propagation delay of the full adder circuit.

RTL CODE:

```
module full_adder(A,B,C,sum,carry);
  input A,B,C;
  output sum,carry;
  wire w1,w2,w3;
  assign w1=A^B;
  assign w2=A&B;
  assign sum =w1^C;
  assign w3 =w1&C;
  assign carry=w3|w2;
endmodule
```

TEST BENCH:

```
module testbench;

 reg A,B,C;
 wire sum,carry;
 integer i;
 full_adder a1(A,B,C,sum,carry);
 initial
  begin
   $dumpfile("dump.vcd");
   $dumpvars(1);
  end

 initial
  begin
   A=0;B=0;C=0;
  end
 initial
  begin
   for(i=1;i<8;i=i+1)
    begin
     #20 {A,B,C}=i;
    end
  end
 initial
  begin
   #60 $finish();
```

```
    end
endmodule
```