Lab Assignment 2

ES 204: DIGITAL SYSTEMS

16th January, 2024

Indian Institute of Technology, Gandhinagar

Aditya N. Mehta

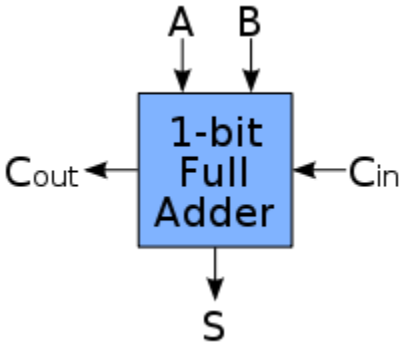
22110017

Hrriday V. Ruparel

22110099

1. Creating a 1-bit full adder

Diagram Representation:



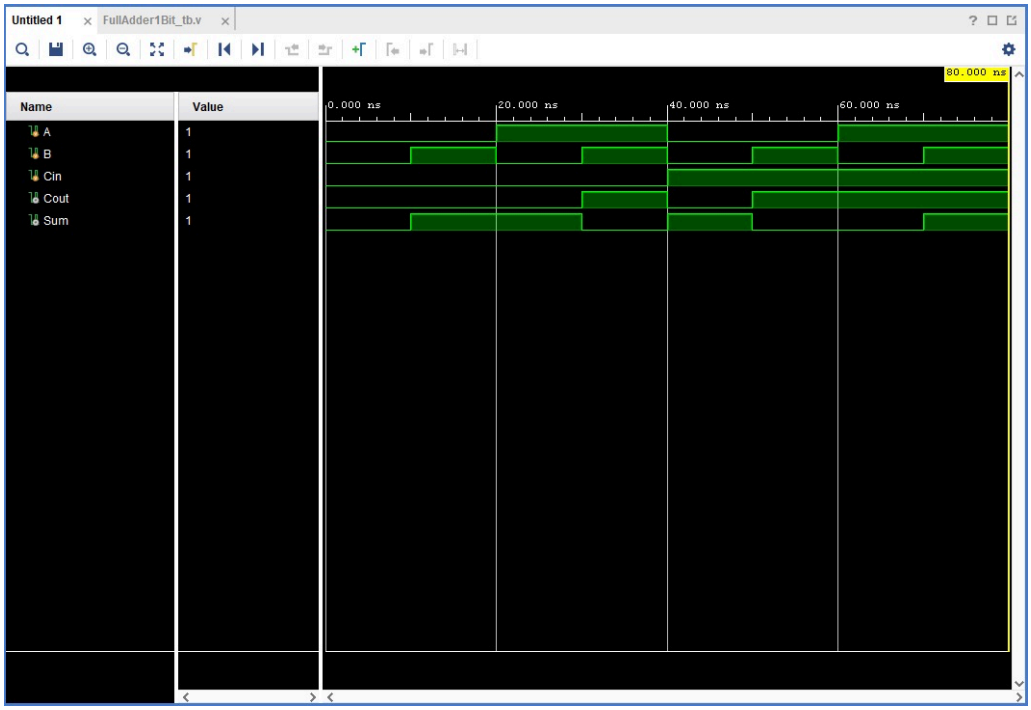
Verilog Code:

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////
// Heading: Lab Assignment 2, ES204: Digital Systems
// Engineer: Aditya Mehta and Hrriday Ruparel
//
// Create Date: 16.01.2024 08:33:09
// Design Name: 1-bit full adder
// Module Name: FullAdder1Bit
// Additional Comments: This module is to be used to implement 4-bit parallel
adders

module FullAdder1Bit(A,B,Cin,Cout,Sum);
input A,B,Cin;
output Cout,Sum;

   and and1(f,A,B);
   and and2(g,B,Cin);
   and and3(h,Cin,A);
   or or1(Cout,f,g,h);

   and and4(i,A,~B,~Cin);
   and and5(j,~A,~B,Cin);
   and and6(k,~A,B,~Cin);
   and and7(m,A,B,Cin);
   or or2(Sum,i,j,k,m);

   // Alternatively, we could have used XOR gate implementation directly.
   // xor xor1(Sum, A, B, Cin);

endmodule
```

Test Beach Code:

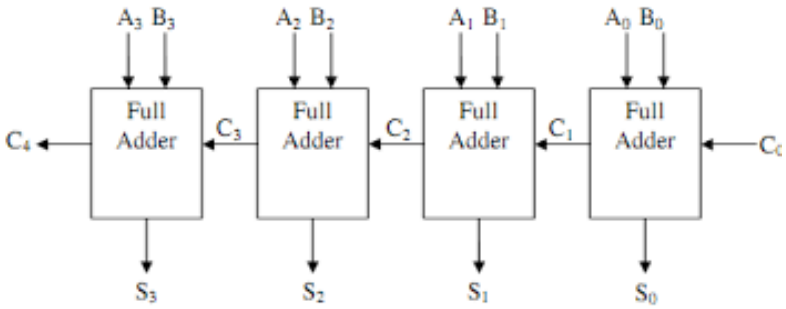
```
`timescale 1ns / 1ps

// Test bench for 1-bit full adder
module FullAdder1Bit_tb();
reg A,B,Cin;
wire Cout,Sum;

   FullAdder1Bit uut(A,B,Cin,Cout,Sum);

   initial
   begin
   A = 0; B = 0; Cin = 0;
   #10;
   A = 0; B = 1; Cin = 0;
   #10;
   A = 1; B = 0; Cin = 0;
   #10;
   A = 1; B = 1; Cin = 0;
   #10;
   A = 0; B = 0; Cin = 1;
   #10;
   A = 0; B = 1; Cin = 1;
   #10;
   A = 1; B = 0; Cin = 1;
   #10;
   A = 1; B = 1; Cin = 1;
   #10;
   $finish();
   end
endmodule
```

Simulation Waveform:



2. Creating a 4-bit Parallel adder

Diagram Representation:



Verilog Code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
///

// Heading: Lab Assignment 2, ES204: Digital Systems
// Engineer: Aditya Mehta and Hrriday Ruparel

// Create Date: 16.01.2024 08:33:09
// Design Name: 4-bit adder using 1-bit full adder module
// Module Name: FullAdder1Bit
// Additional Comments: This module uses 1-bit adder module previously created
to implement 4-bit parallel adder

//////////////////////////////////////////////////////////////////////////////////
///

module FullAdder4Bit(A,B,Cin,Out);

input [3:0]A;
input [3:0]B;
input Cin;
output [4:0]Out;

wire Carry[2:0];

FullAdder1Bit col1(.A(A[0]), .B(B[0]), .Cin(Cin), .Cout(Carry[0]),
.Sum(Out[0]));
FullAdder1Bit col2(.A(A[1]), .B(B[1]), .Cin(Carry[0]), .Cout(Carry[1]),
.Sum(Out[1]));
FullAdder1Bit col3(.A(A[2]), .B(B[2]), .Cin(Carry[1]), .Cout(Carry[2]),
.Sum(Out[2]));
FullAdder1Bit col4(.A(A[3]), .B(B[3]), .Cin(Carry[2]), .Cout(Out[4]),
.Sum(Out[3]));

endmodule
```

Test Beach Code:

```
`timescale 1ns / 1ps

// Test bench for 4-bit full adder
module FullAdder4Bit_tb();
reg [3:0]A;
reg [3:0]B;
reg Cin;
wire [4:0]Out;

   FullAdder4Bit uut(A,B,Cin,Out);
   initial
   begin
   A = 4'd14; B = 4'd7; Cin = 1;
   #10;
   A = 4'd3; B = 4'd8; Cin = 0;
   #10;
   A = 4'd15; B = 4'd15; Cin = 1;
   #10;
   A = 4'd4; B = 4'd2; Cin = 0;
   #10;

   $finish();
   end
endmodule
```

Simulation Waveform: