Post Lab Report-04

Course Title: Digital Logic Design

Course Code: CSE345

Section: 04

Experiment 4

Submitted to:

Touhid Ahmed

Lecturer

Department of Computer Science & Engineering

Submitted by:

Name: Adri Saha

ID: 2019-1-60-024

Date of submission: 19 April 2021

Experiment Name: Binary Adder and Subtractor

Abstract: Digital Logic Design is the one of the introductory courses for Electrical and Electronics Engineering students. It can introduce students to circuit design, problem solving, testing, and feature verification. In this experiment, students were asked to design and construct half adder, full adder, half subtractor and full subtractor circuits and verify the truth table using logic gates. And they can also learn behavioral Verilog coding of **adder and subtractor circuits** using procedural model and continuous assign statement. Here, they also learn to implement and test adder & subtractor using discrete gates. The digital logic design lab is a learning experience that most students enjoy because it gives them their first hands-on experience designing and constructing miniature structures.

Introduction:

Adder and subtractor are basically used for performing arithmetical functions like addition, subtraction, multiplication and division in electronic calculators and digital instruments. They are also used in microcontrollers for arithmetic additions, PC (program counter) and timers. It is possible to design adder and subtractor circuit by discrete logic gates. In this experiment, students get to know the connections between the circuit diagram where logic inputs are also given. Then they simulated their adder & subtractor circuit design to observe the output and verify with the truth table. It can be design by many ways. Here applied behavioral Verilog design as per follow the experiment. Most of our design use if-else statements, case statements, for loops, and other procedural statements and continuous assignment statement to create behavioral code.

Adders:

The adder circuit is a digital combinational circuit that is used to add two numbers together. The output of a standard adder circuit is a sum bit (denoted by S) and a carry bit (denoted by C). Adders are commonly used to add binary numbers, but they can also be used to add other formats such as BCD (binary coded decimal), XS3, and others. There are two kinds of adder circuits: half adder and full adder.

Input		**Output**	
A	**B**	**Sum**	**Carry**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 1: Truth Table of Adder

Uses:

- Used for a variety of applications in digital electronics, including address decoding, table index calculation, increment and decrement operators and so on.
- Used in electronic calculators and digital instruments to perform arithmetic functions such as addition, subtraction, multiplication, and division.
- Also used in microcontrollers for arithmetic addition, PC (program counter), and timers.

Subtractors:

By taking the complement of the subtrahend and adding it to the minuend, two binary numbers can be subtracted. By using this process, the subtraction operation is transformed into an addition operation that requires complete adders to implement on a computer. Subtraction can be directly implemented using logic circuits.

Each subtrahend bit of the number is subtracted from its corresponding significant minuend bit to produce a new bit using this process. If the minuend bit is less than the subtrahend bit, a 1 from the next important location is borrowed. The next higher pair of bits must be informed that a 1 has been borrowed.

Truth Table:

Input		**Output**	
A (minuend bit**)**	**B (**subtrahend bit)	**Difference**	**Borrow**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Figure 2: Truth Table of subtractor

Uses:

- Commonly used in computers for ALU (arithmetic logic unit) to deduct as CPU & GPU for graphics applications to reduce circuit complexity.
- Also used in electronic calculators and digital devices to perform arithmetical functions such as subtraction.

<u>Theory:</u>

1-bit Half Adder:

A half adder has two inputs for the two bits to be added and two outputs into the higher adder position, one from the sum ‘S' and the other from the carry ‘c'. From the addition of the less important bits sum from the X-OR Gate to the carry out from the AND gate, the above circuit is known as a carry signal.

Circuit:

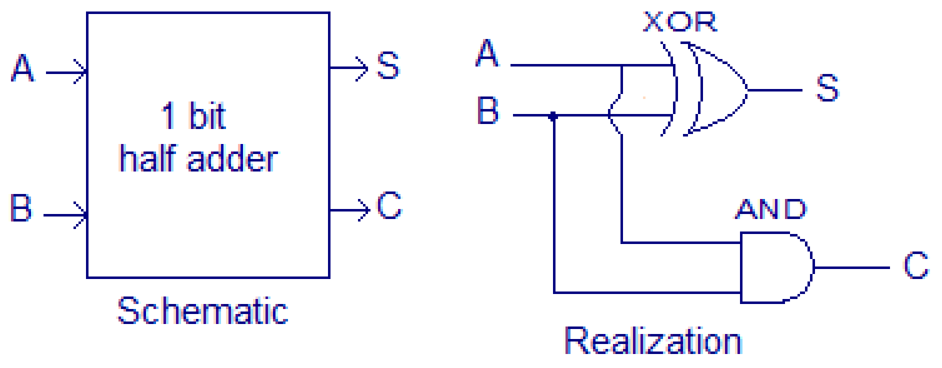


Figure 3: Half Adder Circuit

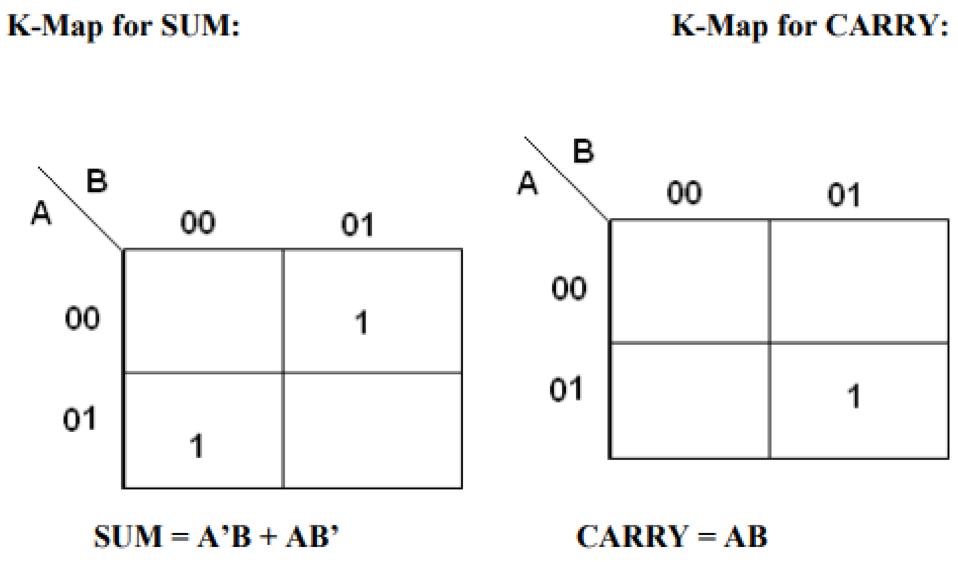


Figure 4: K-Map for half-adder

Uses:

• Calculators, computers, optical measurement instruments, and other electronic devices use it.
• A computer's ALU (arithmetic logic circuitry) computes the binary addition operation on two bits using a half adder.
• A half adder is used to create a full adder since a full adder needs three inputs, one of which is an input carry, allowing us to cascade the carry bit from one adder to the next.

1-bit Full Adder:

A full adder is a combinational circuit with three inputs and two outputs that forms the arithmetic sum of inputs. A full adder can add three bits at once, while a half adder cannot. In a complete adder, the sum output will come from the X-OR Gate, while the carry output will
come from the OR Gate.

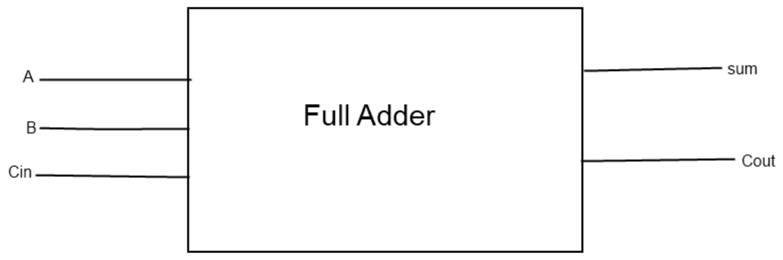
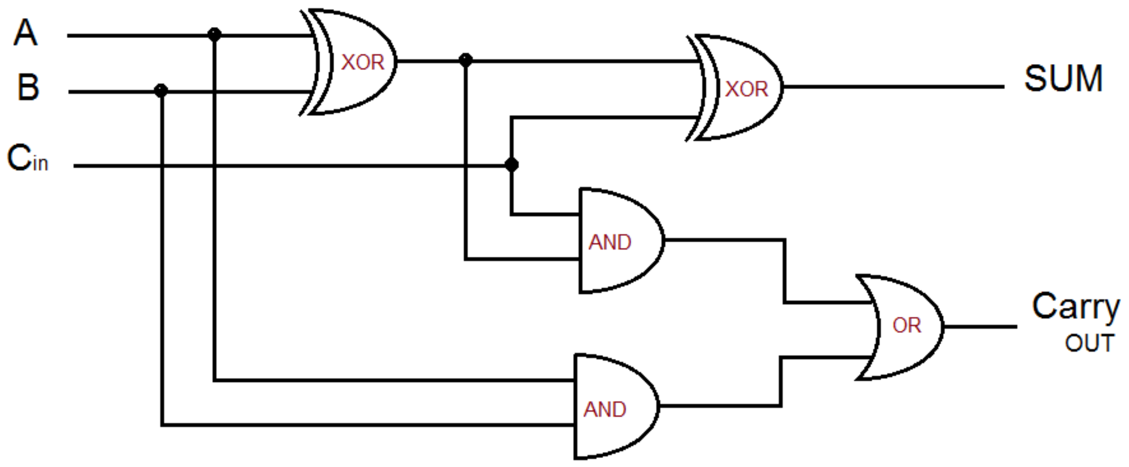
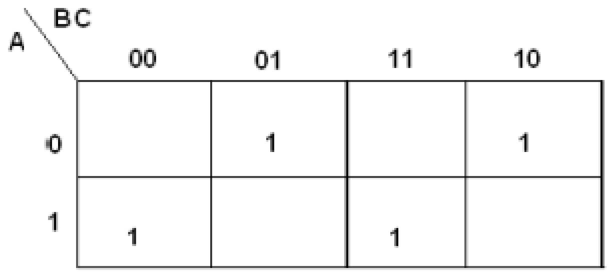


Figure 5: Full Adder Schematic Circuit using 2 half adders



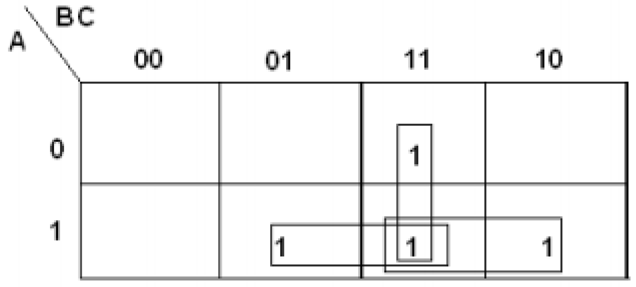
K-Map for SUM:



A \ BC	00	01	11	10
0		1		1
1	1		1	

SUM = A’B’C + A’BC’ + ABC’ + ABC

K-Map for CARRY:



A \ BC	00	01	11	10
0			1	
1		1	1	1

CARRY = AB + BC + AC

Figure 6: K-Map for full-adder

Uses:

- It gives full swing output, low power consumption, high speed and robustness to supply voltage scaling, transistor sizing
- RIPPLE CARRY ADDER is possible to create a logical circuit using multiple full adders to add N-bit numbers.
- ALU- Arithmetic Logic Unit (one of the circuit is a full adder). to generate memory addresses inside a computer and to make the Program Counter point to next instruction, the ALU makes use of this adder.
- For graphics related applications, where there is a very much need of complex computations, the GPU uses optimized ALU which is made up of full adders, other circuits as well.

- The full adder is usually a component in a cascade of adders, which add 8, 16, 32, etc. bit binary numbers.
- Basically, it is used in designing ALU and this ALU is used for wide variety of applications (from designing CPU to GPU).

1-bit Half Subtractor:

Half subtractor is the most essential combinational logic circuit which is **used** in digital electronics. Basically, this is an electronic device or in other terms, we can say it as a logic circuit.

Circuit:

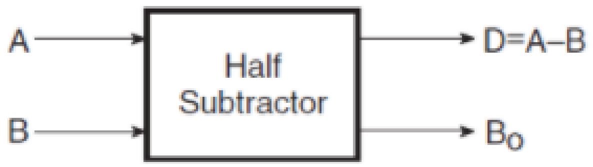
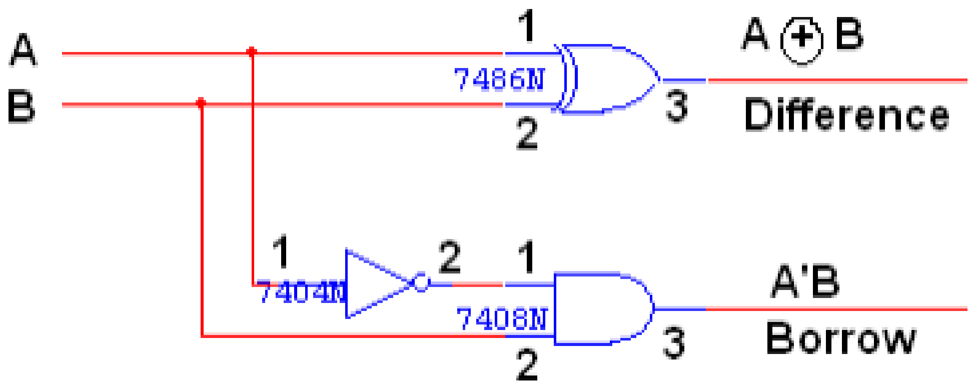


Figure 7: Half Subtractor diagram



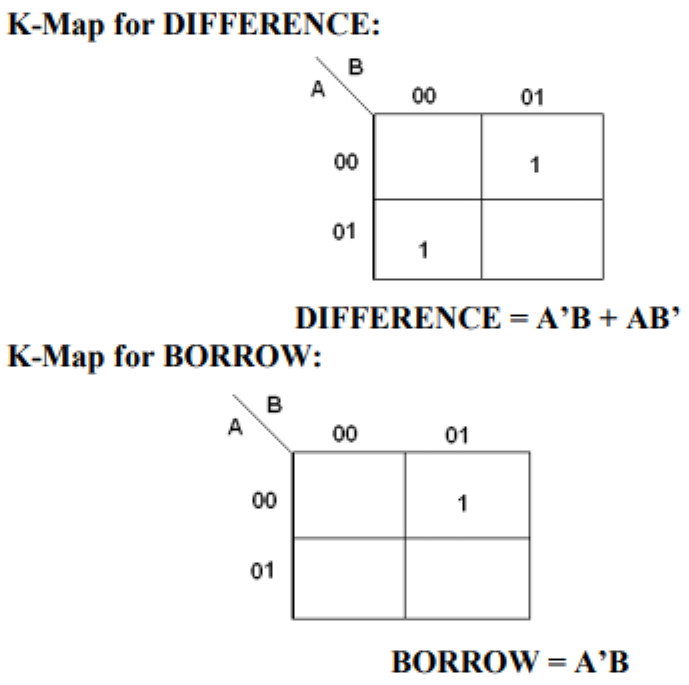


Figure 7: K-Map for half-subtractor

Uses:

This circuit is used to perform two binary digits subtraction.

- Used in amplifiers to reduce sound distortion
- Used in the ALU of processors to reduce the force of audio or radio signals
- To remove the least important column numbers, use this formula. It can be used for multi-digit number subtraction with the LSB.
- used to measure addresses and increase and decrease operators

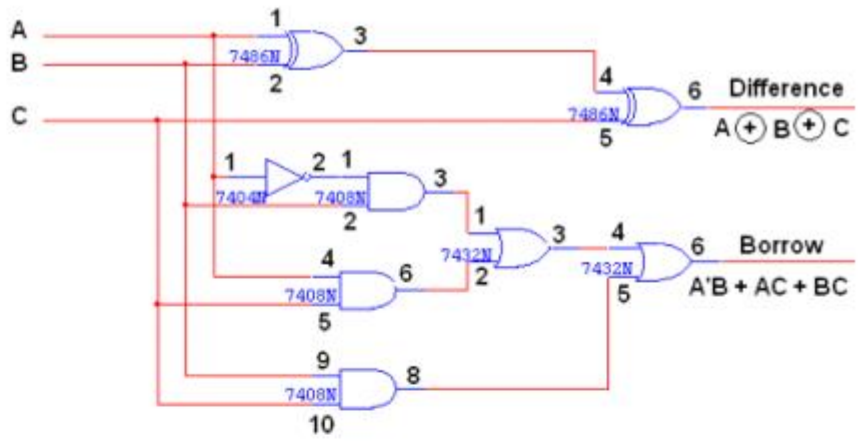
1-bit Full Subtractor:

The full subtractor is used to subtract three 1-bit numbers A, B, and C, which are minuend, subtrahend, and borrow in, respectively. There are three input states and two output states in the complete subtractor: diff and borrow out. An adder circuit will fully replace a subtractor circuit.

A+(-B) = A-B

Same is the case of using subtractor to yield the results of adder.

Circuit:



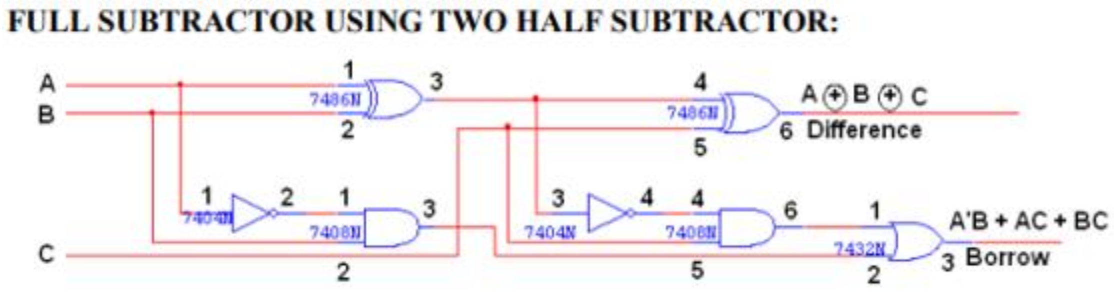
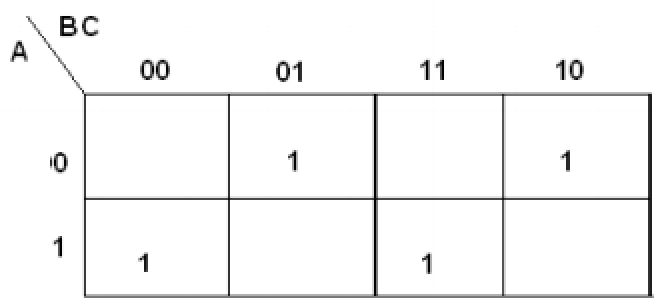


Figure 8: Full Subtractor diagram

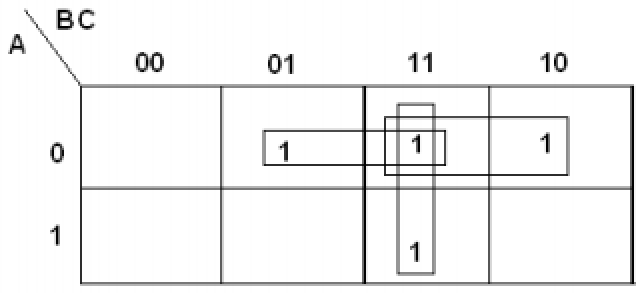
K-Map for Difference:



A \ BC	00	01	11	10
0		1		1
1	1		1	

Difference = A'B'C + A'BC' + AB'C' + ABC

K-Map for Borrow:



A \ BC	00	01	11	10
0		1	1	1
1			1	

Borrow = A'B + BC + A'C

Figure 9: K-Map for full Subtractor

Uses:

• If "multiplication" is "a glorified addition," "division" is "a glorified subtraction." The number of subtractions performed equals the product of the division.

• The complete subtractor is a common and important combinational logic circuit.

N-bit Ripple Carry Adder:

A ripple carry adder is a logic circuit in which each full adder's carry-out is the carry-in of the next most important full adder. Since each carry bit ripples into the next level, it's called a ripple carry adder.

Using multiple complete adders to add N-bit numbers, a logical circuit called a ripple carry adder can be built.

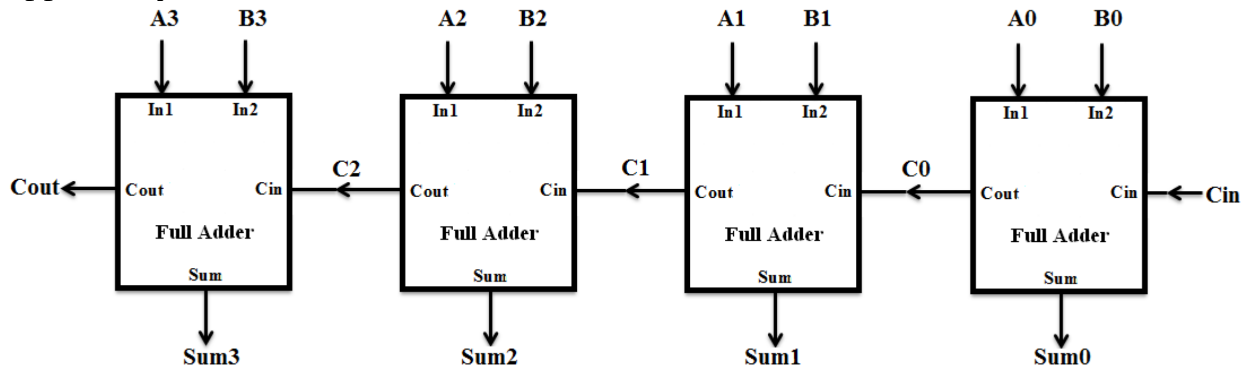


Figure 10: 4-bit ripple Carry Adder Diagram

Truth Table: Here an example of 4-bit RCA truth table

Cin	A				B				Sum				Carry
	A3	**A2**	**A1**	**A0**	**B3**	**B2**	**B1**	**B0**	**S3**	**S2**	**S1**	**S0**	**Cout**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**
0	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**
0	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
0	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**
0	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**0**
0	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
0	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**1**
0	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**1**
0	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
0	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**
0	**1**	**1**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
1(DC)	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

Figure 11: 4-bit ripple carry adder truth table

Uses:

- Ripple carry adder, it adds n-bits at a time.
- These carry adders are usually used in combination with n-bit input sequences.
- These carry adders are used in microprocessors and digital signal processing.
- We can use the addition method to get correct results for n-bit sequences.
- The design of this adder is not a difficult task.
- Alternative for when half adder and full adders do not perform the addition operation when the input bit sequences are large.

XOR Gates: Ex-OR gate will give an output value of logic "1" ONLY when there are an ODD number of 1's on the inputs to the gate, if the two numbers are equal, the output is "0".

Boolean expression for 3 input X-OR is:

$$\mathbf{S = (A \oplus B \oplus C) = A'.B'.C + A'.B.C' + A.B'.C' + A.B.C}$$

Truth Table: The truth table of 3 input XOR gate is:

Here, A, B, C is input & output S.

Input			Output
A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figure 12: Truth table of 3 input XOR Gate

Uses: It is used in simple digital addition circuits which calculate the sum and carry of two (half-adder) or three (full adder) bit numbers.

- XOR gates are also used to determine the parity of a binary number, i.e., if the total number of 1's in the number is odd or even.
- The XOR gate is achieved by combining standard logic gates together to form more complex gate functions that are used extensively in building arithmetic logic circuits, computational logic comparators and error detection circuits.
- The two-input "Exclusive-OR" gate is basically a modulo two adder, since it gives the sum of two binary numbers and as a result are more complex in design than other basic types of logic gate.
- It is used to implement adder and subtractor circuits.

Objectives: Our main purpose is to design binary full adder and subtractor using discrete logic gates. Using Behavioral Verilog Simulation**,** students can implement these circuits and verify with truth table. Here in Verilog codes, students utilize behavioral code by assign statements, if-else statements, case statements, for loops and other procedural statements in adder & subtractor circuits.

- To implement and test a half & full adder using discrete gates.
- To implement and test a half & full subtractor using discrete gates.

- To learn how to implement N bit Ripple Carry Adder/subtractor using (n-bit binary parallel adder)
- Verify the circuit designs with truth table.

Experimental results & discussions:

1. **1 Bit Half Adder using procedural statement :**

   Truth Table:

Input		**Output**	
A	**B**	**S(Sum)**	**C(Carry)**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figure 13: Truth Table for 1-bit half adder

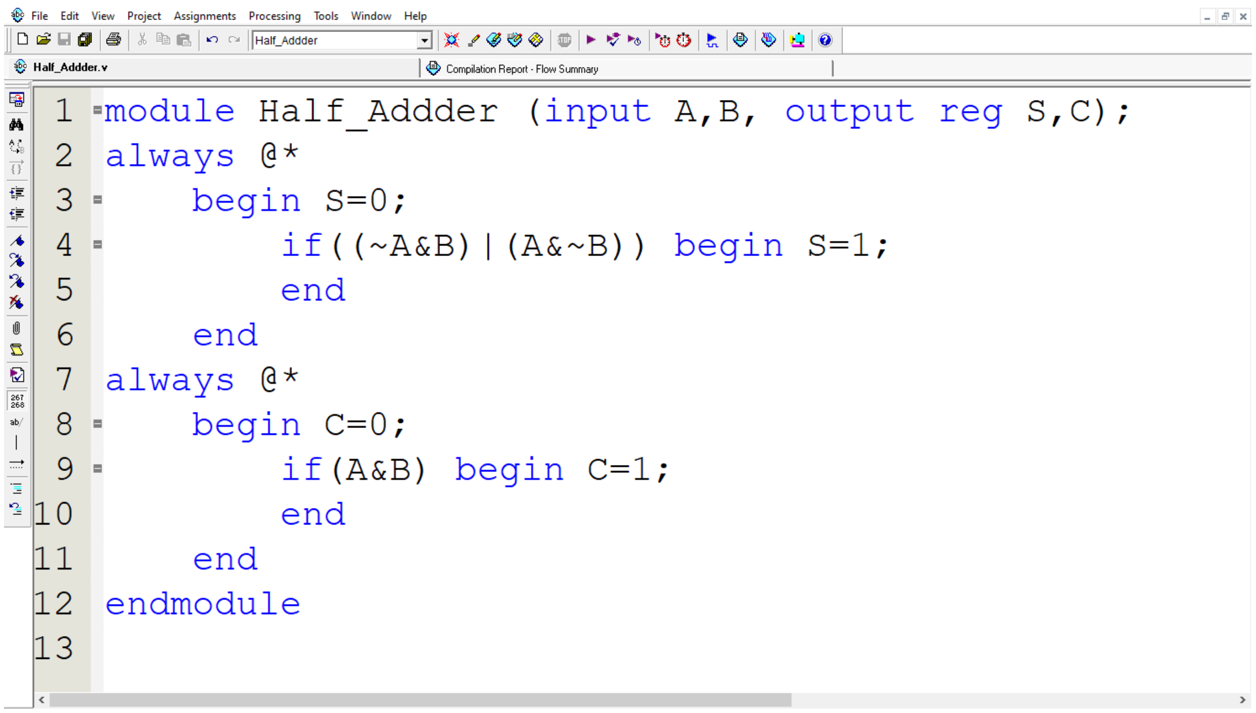
Logical Expression for half adder is: $S=A \oplus B= AB'+A'B$;

$C=A.B$

Verilog Codes: Verilog is a Hardware Description Language; a textual format for describing electronic circuits and systems.

Uses:

- Applied to electronic design.
- Verilog is intended to be used for verification through simulation, for timing analysis, for test analysis (testability analysis and fault grading) and for logic synthesis.



File Edit View Project Assignments Processing Tools Window Help
Half_Adder
Half_Addder.v
Compilation Report - Flow Summary

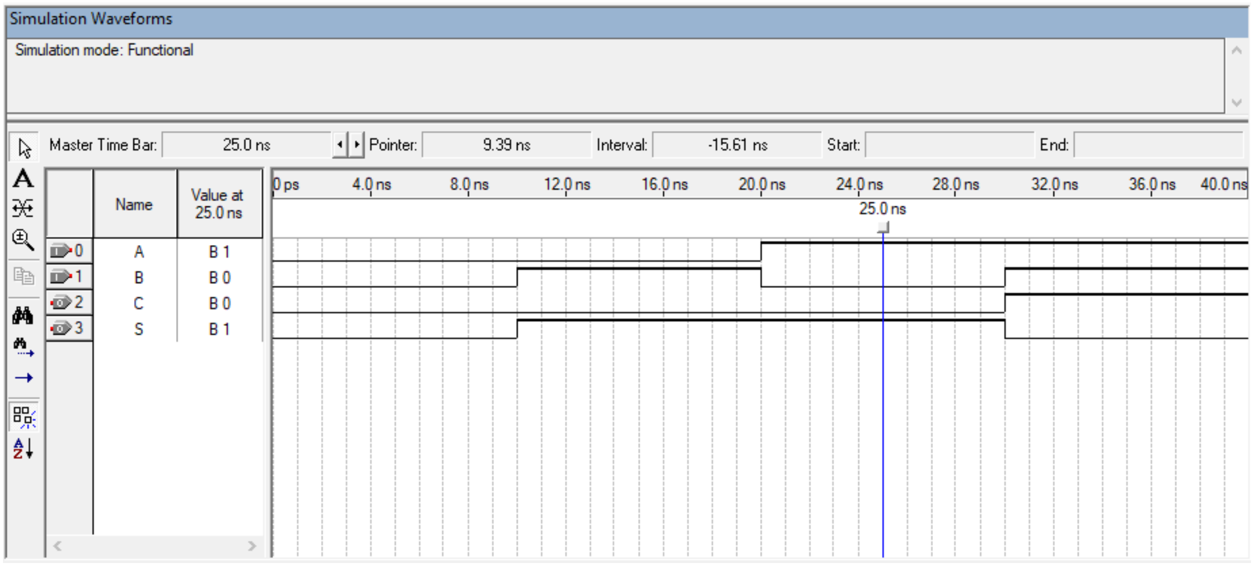
```
module Half_Addder (input A,B, output reg S,C);
always @*
    begin S=0;
        if((~A&B)|(A&~B)) begin S=1;
        end
    end
always @*
    begin C=0;
        if(A&B) begin C=1;
        end
    end
endmodule
```

Figure 14: Verilog code using procedural statement to design 1-bit half adder

Waveform Simulation:

In this simulation, we have taken end time 40 nano seconds for 2 input A, B. So, time period of A & B is 40 & 20 nano seconds in sequent. C is the carry and S is the Sum.

We see, output is “high” or 1 after 10 ns and again low after 30ns. And carry, C is high or 1 after 30 seconds which is 1+1=10, where carry is 1 & sum is 0. So, it is verified with the truth table of 1 bit half adder.



	Name	Value at 25.0 ns
0	A	B 1
1	B	B 0
2	C	B 0
3	S	B 1

Figure 15: Simulation of Verilog code using procedural statement for 1-bit half adder

2. **1 Bit Full Adder using X-OR gates:**
   Truth Table:

Input			**Output**	
A	**B**	**Cin**	**S**	**Cout**
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

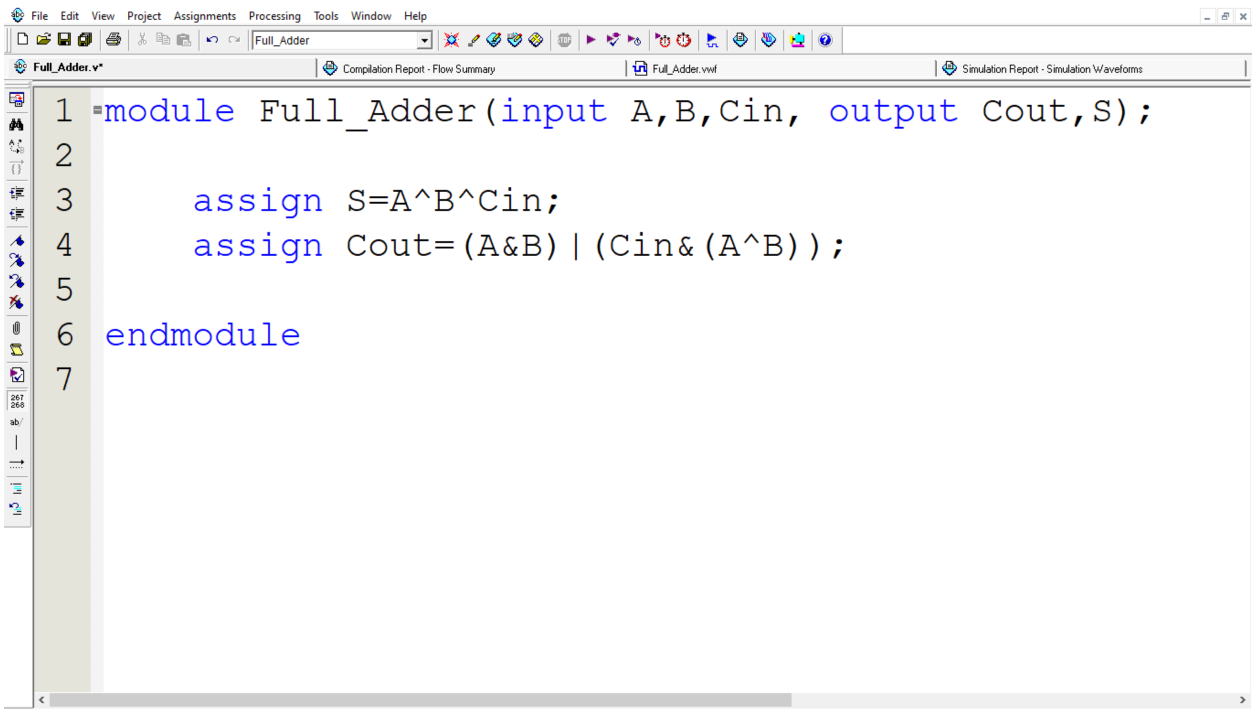
Figure 16: Truth Table for 1-bit full adder

So, Boolean expression for full adder,

S = A'B'Cin + A'BCin' + AB'Cin' + ABCin = A $\oplus$ B $\oplus$ Cin

Cout = ABCin' + AB'Cin+ A'BCin+ ABCin= + A.B + (A $\oplus$ B).Cin

Verilog Codes: Here, inputs are A, B & Cin and Output is Cout & S as follows. We used continuous assignment here to implement full adder using X-OR gates.


File Edit View Project Assignments Processing Tools Window Help
Full_Adder
Full_Adder.v*
Compilation Report - Flow Summary
Full_Adder.vwf
Simulation Report - Simulation Waveforms

```
module Full_Adder(input A,B,Cin, output Cout,S);

    assign S=A^B^Cin;
    assign Cout=(A&B)|(Cin&(A^B));

endmodule

```

Figure 17: Verilog code using continuous assignment to design 1-bit full adder

Verilog Simulation: In this simulation, we have taken end time 80 nano seconds. So, time period of A, B & Cin is 80, 40 & 20 nano seconds in sequent.
We see, output is verified with the truth table of 1bit full adder.

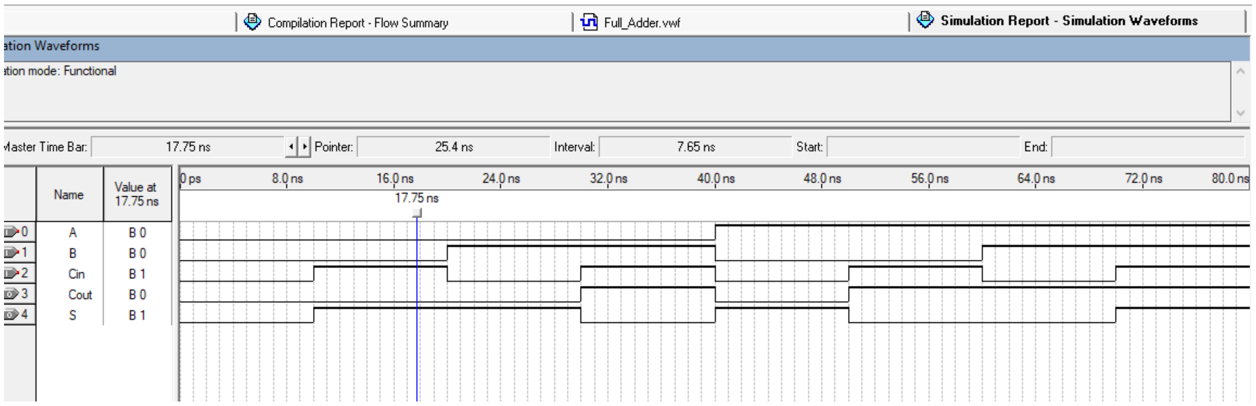


Figure 18: Simulation of 1bit full adder Verilog code using continuous assignment using X-OR gates

3. 1 Bit Half Subtractor

Here, A, B is input & output is D & B.

Truth Table:

Input		Output	
A	**B**	**D(Difference)**	**B(Borrow)**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

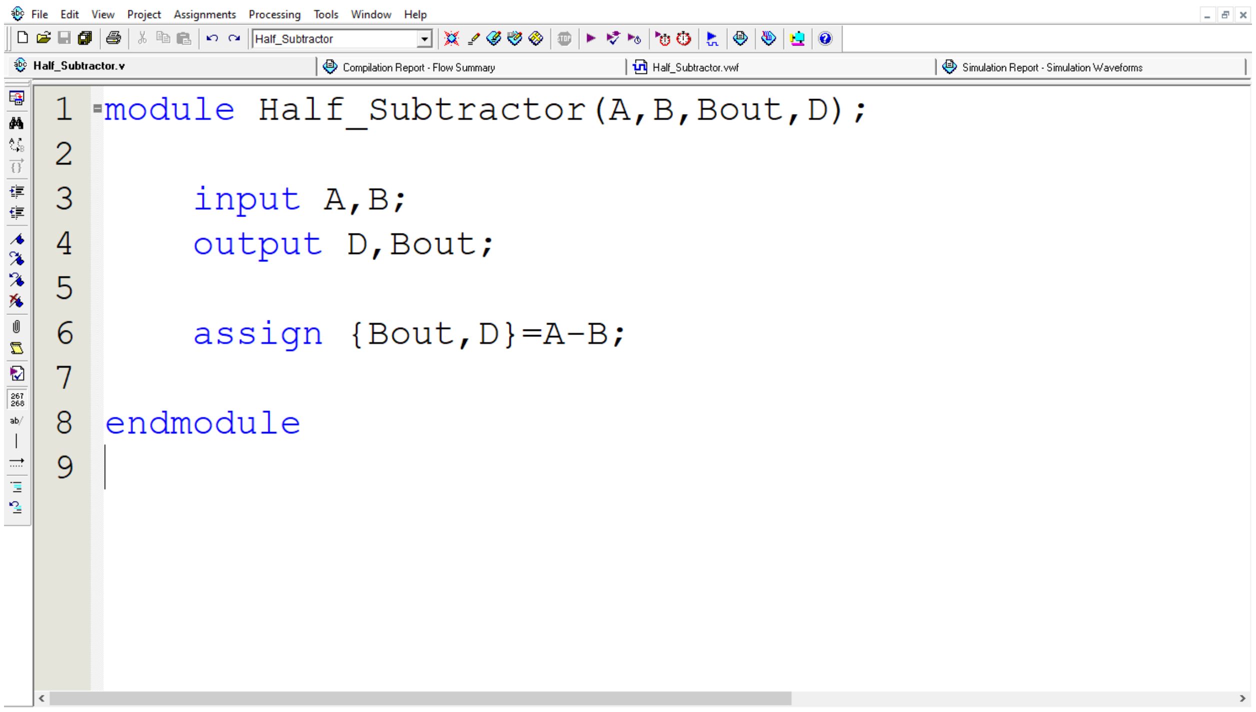
Figure 19: Truth Table for 1-bit half subtractor

The Boolean expression for full adder, D= A'B + AB' = A ⊕ B

Bout= A’B

Verilog Codes:

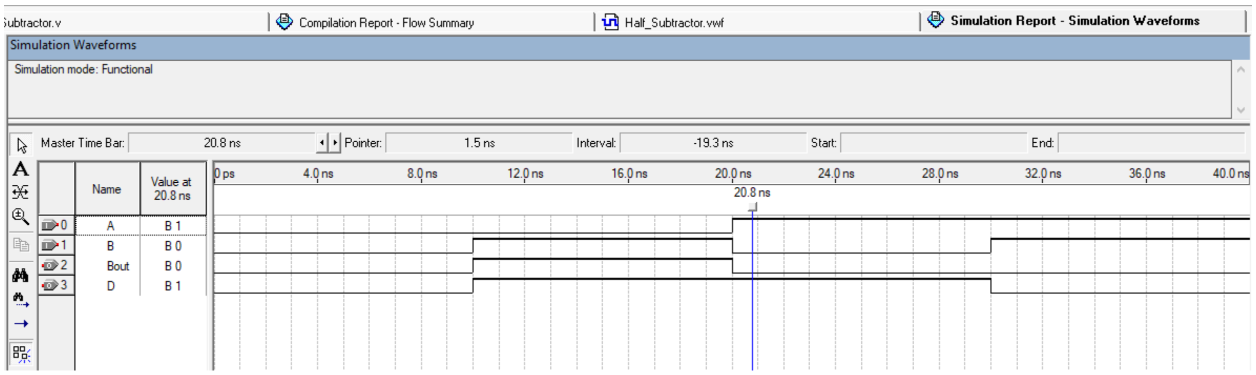
There is total 2^2=4 output for 2 input. Here, the arithmetic assignment is applied for 1bit half subtractor.


File Edit View Project Assignments Processing Tools Window Help
Half_Subtractor
Half_Subtractor.v
Compilation Report - Flow Summary
Half_Subtractor.vwf
Simulation Report - Simulation Waveforms

```
module Half_Subtractor(A,B,Bout,D);

    input A,B;
    output D,Bout;

    assign {Bout,D}=A-B;

endmodule
```

Figure 20: Verilog code using arithmetic assignment to design 1-bit half subtractor

Simulation: In this simulation, we have taken end time 40 nano seconds for 2 input A, B. So, time period of A & B is 40 & 20 nano seconds in sequent. D is the difference and B is the borrow. Here we applied, simple arithmetic operation A-B. And the expected output is same.



Simulation Waveforms

Simulation mode: Functional

Master Time Bar: 20.8 ns Pointer: 1.5 ns Interval: -19.3 ns Start: End:

	Name	Value at 20.8 ns
0	A	B 1
1	B	B 0
2	Bout	B 0
3	D	B 1

Figure 21: Simulation of 1bit half subtractor Verilog code using continuous assignment using arithmetic operator

4. **1 Bit Full Subtractor:**
   The Boolean expression for full adder, D= A'B'C + A'BC' + AB'C' + ABC = A ⊕ B ⊕ C
   Bout= A’B+BC+A’C

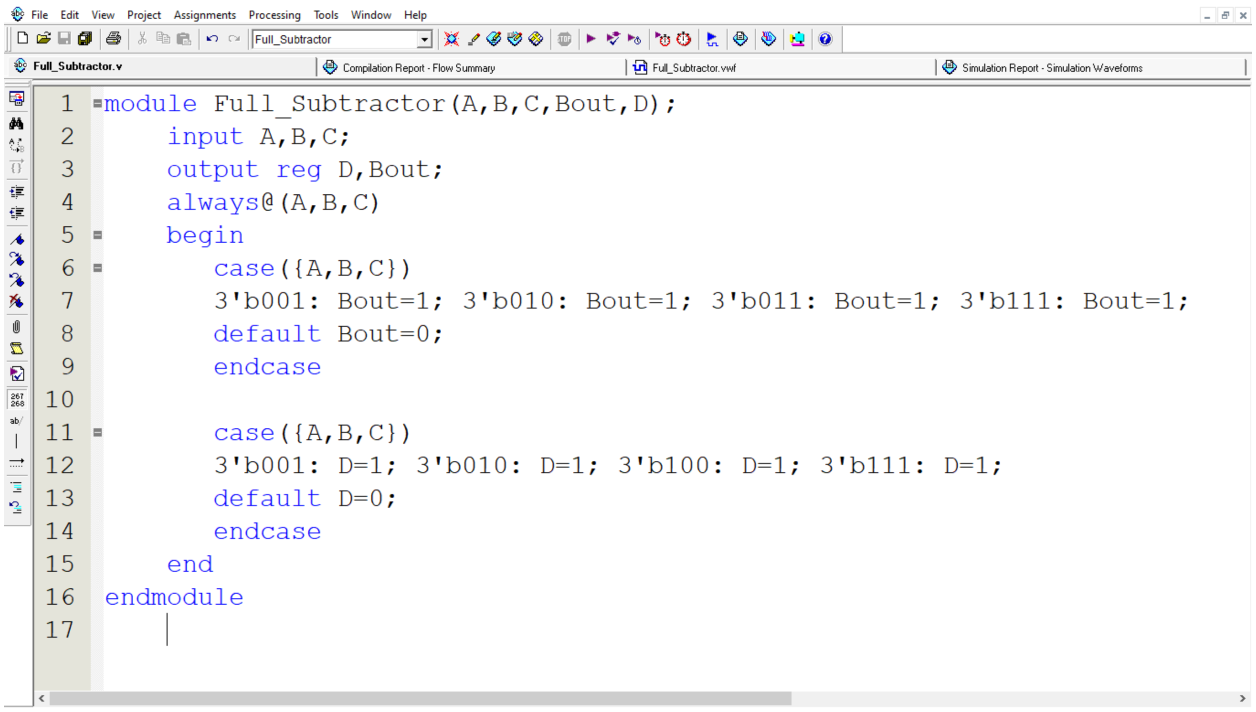
Truth Table:

Input			**Output**	
A	**B**	**C**	**D**	**Bout**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure 22: Truth Table for 1-bit full subtractor

_Here, A, B & Bin is input & output is Bout & D. Here used procedural case statement to implement this design.

Verilog Code:


File Edit View Project Assignments Processing Tools Window Help
Full_Subtractor
Full_Subtractor.v
Compilation Report - Flow Summary
Full_Subtractor.vwf
Simulation Report - Simulation Waveforms

```
module Full_Subtractor(A,B,C,Bout,D);
    input A,B,C;
    output reg D,Bout;
    always@(A,B,C)
    begin
       case({A,B,C})
       3'b001: Bout=1; 3'b010: Bout=1; 3'b011: Bout=1; 3'b111: Bout=1;
       default Bout=0;
       endcase

       case({A,B,C})
       3'b001: D=1; 3'b010: D=1; 3'b100: D=1; 3'b111: D=1;
       default D=0;
       endcase
    end
endmodule
```

Figure 23: Verilog code using procedural case statement to design 1-bit full adder

Verilog Simulation:

In this simulation, we have taken end time 80 nano second. So, time period of A, B & C is 80, 40 & 20 nano seconds in sequent. As there are 3 inputs. All of the output are matched with truth table of half subtractor.

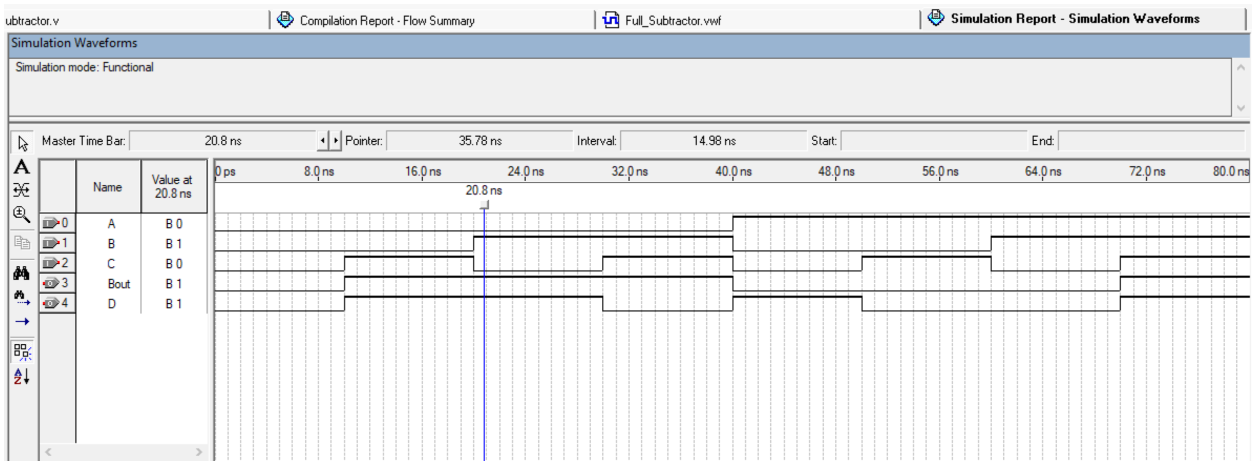


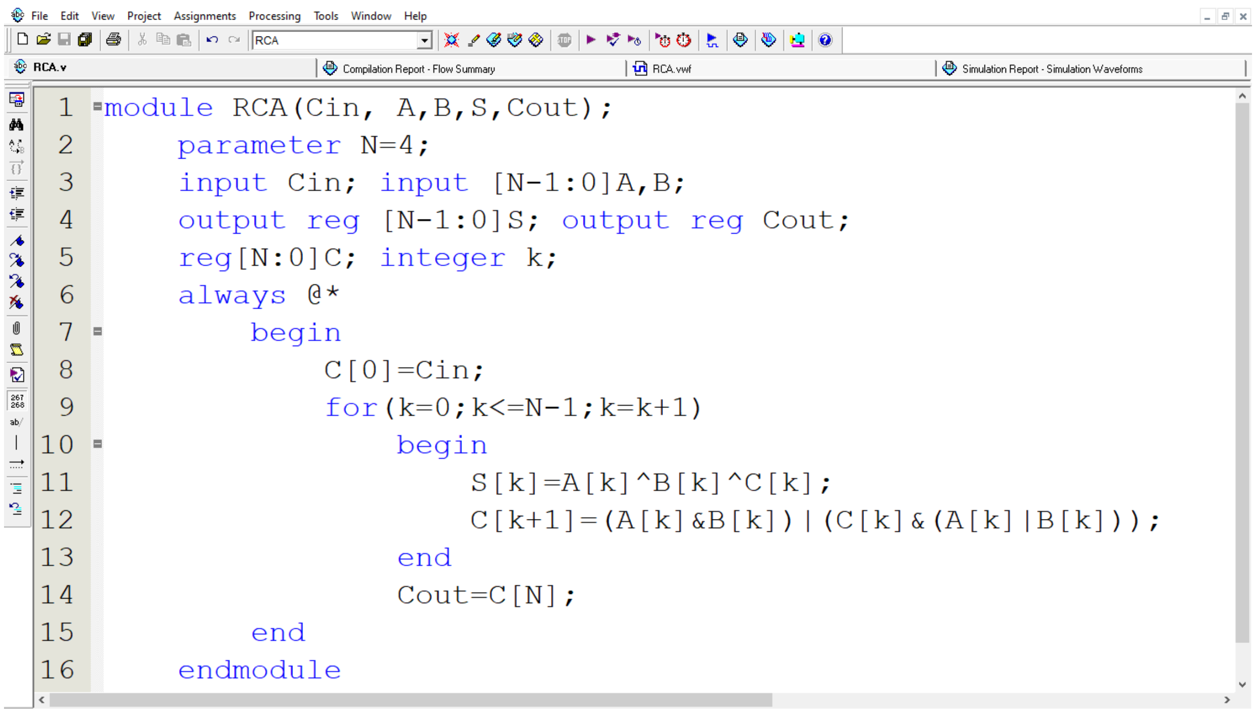
Figure 24: Simulation of 1bit full subtractor Verilog code using procedural statement

5. N bit Ripple Carry Adder:

Here we use, parameter N=4. That means 4-bit ripple carry adder are shown here.

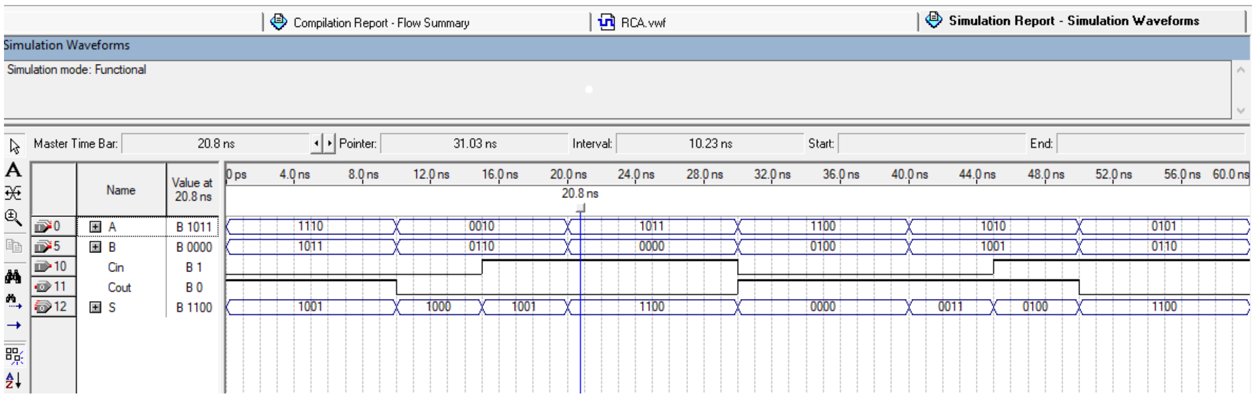
Verilog Code:
In this Verilog code, we use procedural assignment using for loop. We assign the values inside the always block. We use X-OR gate for cascading full adders.

Verilog Simulation: To simulate the Verilog code of 4-bit ripple carry adder, we take 4-bit input A & B. For that, we use random value and fixed 10 ns interval period in each. The other carry in input in 1 bit so no change in that. We just override clock & take time period 30ns where end time is **60ns**. We can also convert inputs in decimal number as well to verify more significantly.


File Edit View Project Assignments Processing Tools Window Help
RCA
RCA.v
Compilation Report - Flow Summary
RCA.vwf
Simulation Report - Simulation Waveforms

```
module RCA(Cin, A,B,S,Cout);
    parameter N=4;
    input Cin; input [N-1:0]A,B;
    output reg [N-1:0]S; output reg Cout;
    reg[N:0]C; integer k;
    always @*
        begin
            C[0]=Cin;
            for(k=0;k<=N-1;k=k+1)
                begin
                    S[k]=A[k]^B[k]^C[k];
                    C[k+1]=(A[k]&B[k])|(C[k]&(A[k]|B[k]));
                end
                Cout=C[N];
        end
    endmodule
```

Figure 25: Verilog code using procedural statement to design n-bit ripple carry adder


Compilation Report - Flow Summary
RCA.vwf
Simulation Report - Simulation Waveforms
Simulation Waveforms

```
Simulation mode: Functional

Master Time Bar: 20.8 ns   Pointer: 31.03 ns   Interval: 10.23 ns   Start:   End:

     Name   Value at 20.8 ns
0    A      B 1011   1110 | 0010 | 1011 | 1100 | 1010 | 0101
5    B      B 0000   1011 | 0110 | 0000 | 0100 | 1001 | 0110
10   Cin    B 1
11   Cout   B 0
12   S      B 1100   1001 | 1000 | 1001 | 1100 | 0000 | 0011 | 0100 | 1100

0 ps  4.0 ns  8.0 ns  12.0 ns  16.0 ns  20.0 ns  24.0 ns  28.0 ns  32.0 ns  36.0 ns  40.0 ns  44.0 ns  48.0 ns  52.0 ns  56.0 ns  60.0 ns
```

Figure 26: Simulation of 4-bit ripple carry adder

We can verify the result by adding A+ B + Cin, we can see every output is matched with the answer. As example: 1110+1011+0=11001, which means Carry out (Cout)=1 & Sum(S)= 1001 which is same as the first result in simulation.

Discussion: So, we see, all the designs implemented by behavioral Verilog codes. all the outputs has given the correct result with truth tables except N-bit ripple carry adder as in simulation inputs are not fixed. But after adding these input numbers we find the same result in output. That is why we can say, there is no error above experiment.

Conclusion: From this experiment, students learned to design both adder & subtractor circuit by using behavioral Verilog code. They also learn to design a full adder by 2 half adders and by discrete logic gates. Here, by cascading adder's implementation of N bit ripple adder is also learned. Finally, students would be able to verify their logic design by truth tables. They also get to know the uses of binary adders & subtractors. The experiment verified his/her hands-on experience by simulating these designs in Quartus Software.

Reference:

1. Book: Fundamentals of Digital Logic with Verilog Design by Stephen Brown, Zvonko Vranesic-McGraw-Hill Science_Engineering_Math (2013)
2. https://www.circuitstoday.com/ripple-carry-adder
3. https://www.philadelphia.edu.jo/academics/qhamarsheh/uploads/Lecture%2012%20Binary%20Adder-Subtractor.pdf
4. http://site.iugaza.edu.ps/wmousa/files/Lab4_Binary-addition-and-subtraction.pdf