# CS666 Assignment 1

**Vissa Surya Sai Sandeep**
22111084
vsssandeep22@iitk.ac.in

**Abhinandan Singh Baghel**
231110002
asbaghel23@iitk.ac.in

**Himanshu Karnatak**
231110017
himanshuk23@iitk.ac.in

**HRUSHIKESH NAKKA**
231110018
hnakka23@iitk.ac.in

## 1 Question 1

Design a Verilog module for the 1-bit full adder that should have three inputs: A, B, and Cin (carry-in), and two outputs: Sum and Cout (carry-out). Now, using instanciation of the 1-bit full adder module, design a 8-bit full adder that should have three inputs: A (8-bit input), B (8-bit input), and Cin (1-bit carry-in), and two outputs: Sum (8-bit output) and Cout (1-bit carry-out).

**Ans:**

Full adder circuit:

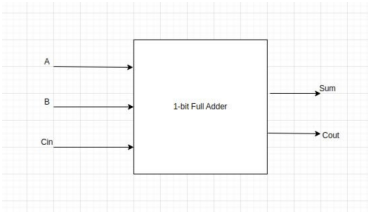


Figure 1: 1-bit full adder

Truth table:

| A | B | Cin | Sum | 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 |

**Cout = A.B + Cin . (A $\oplus$ B)**
**Sum = A $\oplus$ B $\oplus$ Cin**

The above full adder circuit is implemented in **one_bit_full_adder.v**. Now series of such full adders are connected to get n-bit full adder circuit as shown below:

Preprint. Under review.

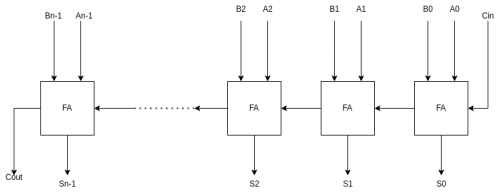


Figure 2: n-bit full adder

## 2 Question 2

Write a Verilog module for 4-bit multiplication? The module should have two 4-bit inputs, A and B, and produce an 8-bit product, P. The computational workflow is shown in Figure 1. Firstly, generate a partial product matrix with dimensions of 4x8. Then, compress the matrix to a 2x8 matrix using a 1-bit full adder module that was designed earlier. Finally, generate the final result using an 8-bit adder module that was also designed earlier.

**Ans:**

The multiplication two 4-bit values A [$A_3A_2A_1A_0$] and B[$B_3B_2B_1B_0$] results in 4 partial products which can be represented in 4 x 8 matrix as shown below:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $A_0B_0$ | $A_1B_0$ | $A_2B_0$ | $A_3B_0$ | 0 | 0 | 0 | 0 |
| 0 | $A_0B_1$ | $A_1B_1$ | $A_2B_1$ | $A_3B_1$ | 0 | 0 | 0 |
| 0 | 0 | $A_0B_2$ | $A_1B_2$ | $A_2B_2$ | $A_3B_2$ | 0 | 0 |
| 0 | 0 | 0 | $A_0B_3$ | $A_1B_3$ | $A_2B_3$ | $A_3B_3$ | 0 |

Now, the first two rows and last two rows are added using 8-bit full adder realised as in question 1 to generate 2 X 8 matrix. Finally, these two rows are added using 8-bit full adder to get the final product.

## 3 Question 3

Write a Verilog code for a 4-bit forward counting synchronous Johnson counter with a one-bit reset signal. The Johnson counter is a type of shift register that cycles through a sequence of 2n states. In this case, the 4-bit counter will produce 16 states, each represented by a unique 4-bit binary pattern. Additionally, the counter should be able to reset to its initial state when a one-bit reset signal is asserted. The input consists of two signals 1. clock; 2. reset signal; and the output is a 4-bit value representing the state of the counter at a particular state.

**Ans:**

Jhonson counter is synchronous sequential circuit that uses same clock to all the D- flip flops used in the circuit. For n-flipflop Johnson counter we have a MOD-2n counter. That means the counter has 2n different states. Total number of used and unused states in n-bit Johnson counter:

1. number of used states= $2 * n$
2. number of unused states= $2^n - 2 * n$

4-bit Jhonson counter circuit: Truth table:

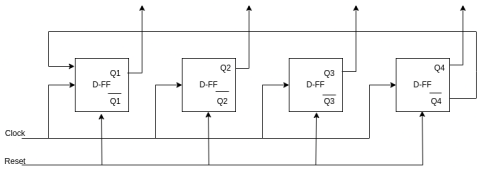


Figure 3: 4-bit Jhonson Counter

| clock | Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 0 | 0 | 0 | 0 |