EE2003 : Computer Organisation

# Assignment 3

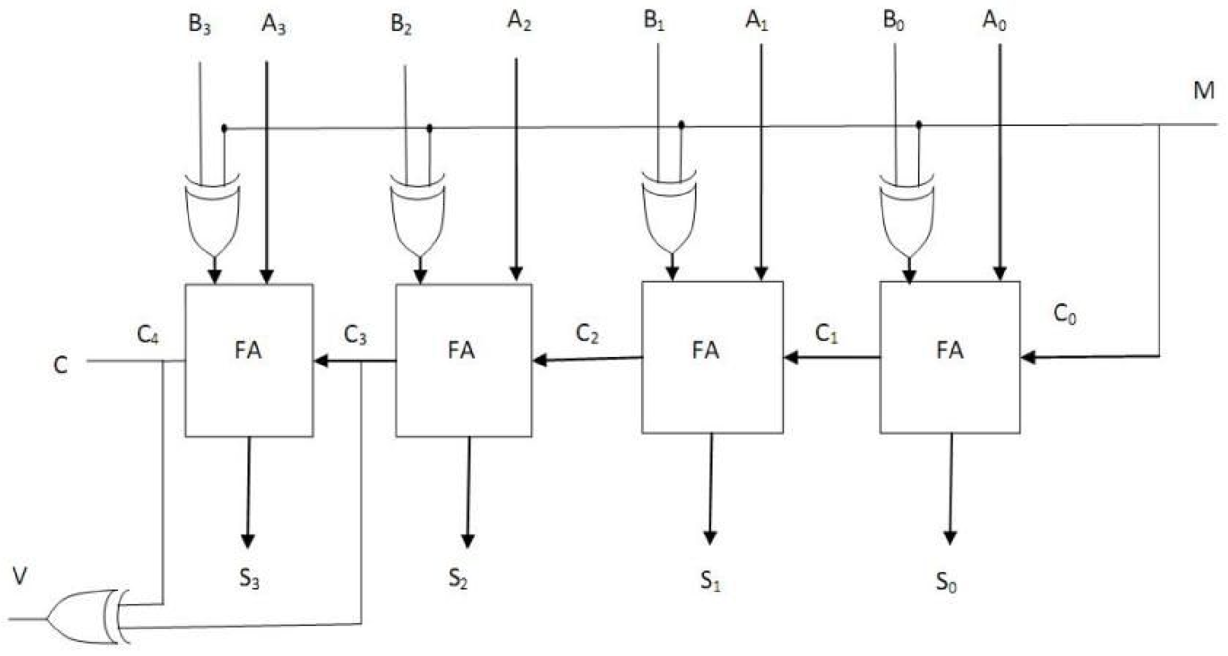
EE15B025 | Ganga Meghanath

## Question

Design a 4-bit Adder/ Subtractor using 2's complement method. The implementation should have a add/subtract control bit. Write a separate implementation (Behavioral) for the Full Adder and instantiate it in the final design as required.

## Answer

In digital circuits, an adder–subtractor is a circuit that is capable of adding or subtracting numbers (in particular, binary). Below is a circuit that does adding or subtracting depending on a control signal.



We use four full-adders with a 4-bit input A, and a 4-bit input B whose bits may be XOR'd based on the mode chosen. The mode will be decided by bit M in the circuit below.

For subtraction M = 1. 1 is chosen because M acts as the carry-in. Therefore, all bits of B will be inverted and 1 will be added to the LSB to find the 2's complement.

This works because when M=1, the A input to the adder is really $\bar{A}$ and the carry in is 1. Adding B to $\bar{A}$ and 1 yields the desired subtraction of B-A.

For addition, M = 0. Therefore, carry-in is set to zero as desired.

If the inputs A and B are unsigned, the answer will give A - B if A >= B OR the 2's complement of (B-A) if A < B.

If the inputs A and B are signed, the range of values I could use are from 0 - 7 and the result will give signed A - B as long as there is no overflow.

Examples :

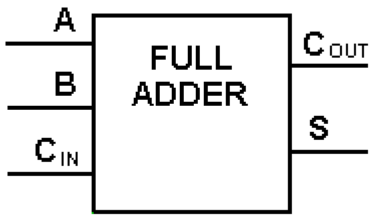
```
  0101 5

+ 1010 -6

  1111 -1
```

```
  0111 7

+ 1101 -3

 10100 -> 0100 : 4
```

So we conclude that the circuit is controlled by the control bit in the following manner :

- **Addition when M = 0**
- **Subtraction when M = 1**

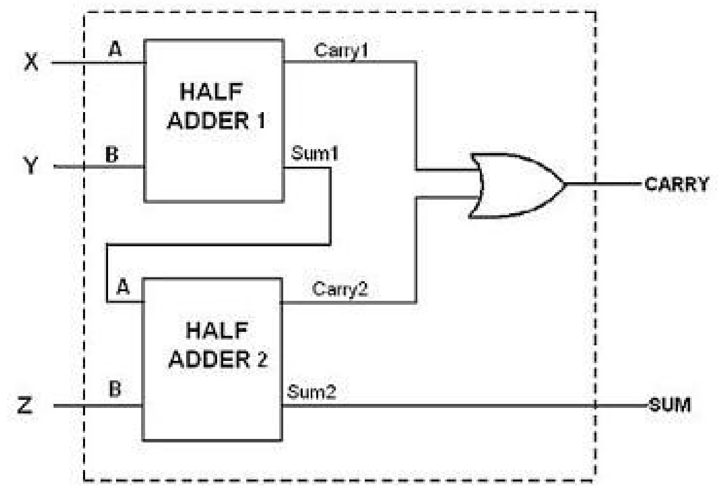
**Full Adder :**



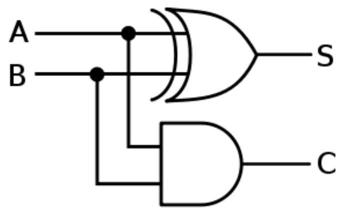
**Full Adder truth table :**

| Cases | $y_{n-1}$ | $x_{n-1}$ | $c_{n-1}$ | $c_n$ | $s_{n-1}$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 1 | 1 | 0 |
| 5 | 1 | 0 | 0 | 0 | 1 |
| 6 | 1 | 0 | 1 | 1 | 0 |
| 7 | 1 | 1 | 0 | 1 | 0 |
| 8 | 1 | 1 | 1 | 1 | 1 |

## Full Adder Implementation using Half Adders :



## Half adder circuit :

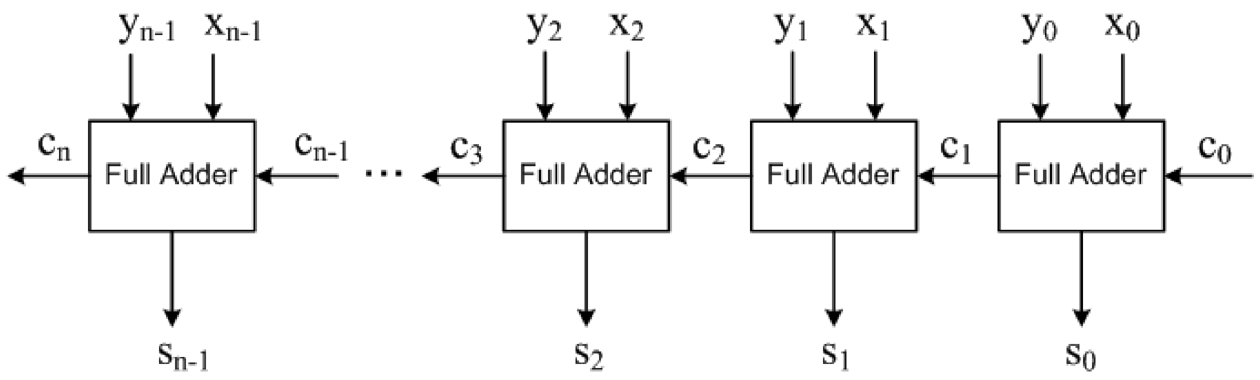


## Half Adder truth table :

| Cases | A (Input) | B (Input) | S (Output) | C (Output) |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 1 | 0 |
| 4 | 1 | 1 | 0 | 1 |

## Overflow :

$c_n \oplus c_{n-1}$ is a correct indicator of overflow in the addition of 2's complement integers.

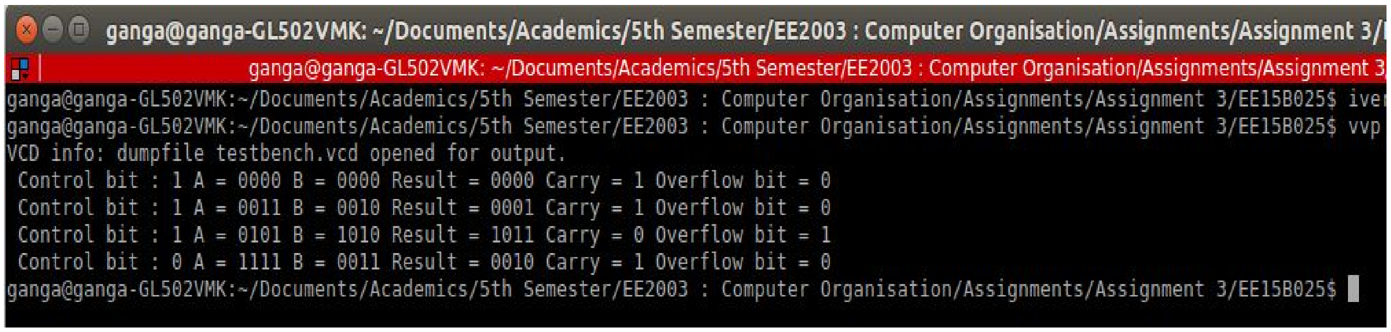


## Truth Table

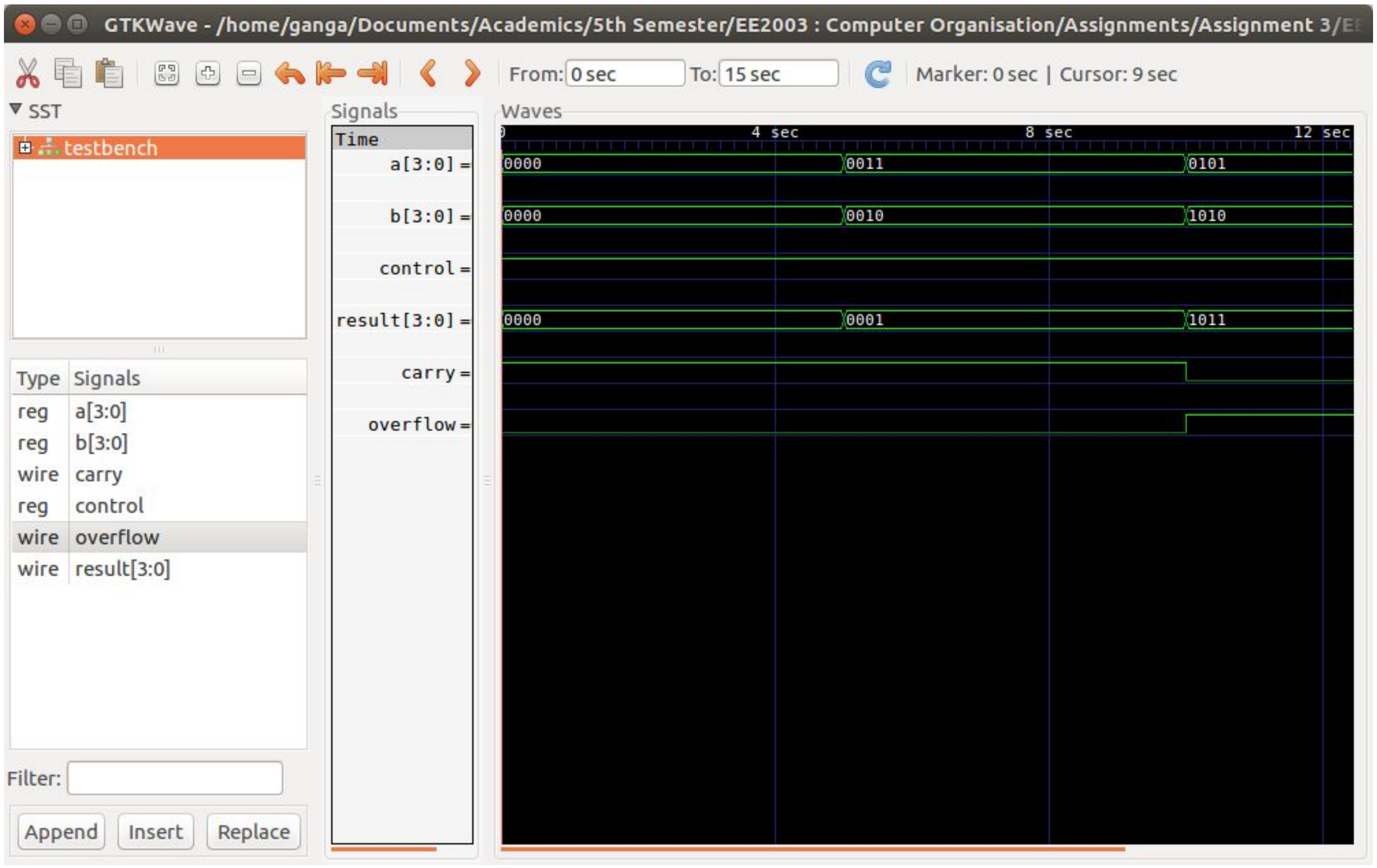
| Cases | $y_{n-1}$ | $x_{n-1}$ | $c_{n-1}$ | $c_n$ | $s_{n-1}$ | $c_n \oplus c_{n-1}$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 0 | 0 |
| 5 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 1 | 1 | 0 | 0 |
| 7 | 1 | 1 | 0 | 1 | 0 | 1 |
| 8 | 1 | 1 | 1 | 1 | 1 | 0 |

As we can see from the truth table, when the output sign bit is different from that of the inputs, $c_n \oplus c_{n-1}$ becomes 1. Hence, we can check for overflow using $c_n \oplus c_{n-1}$.

## Results obtained :



```
ganga@ganga-GL502VMK:~/Documents/Academics/5th Semester/EE2003 : Computer Organisation/Assignments/Assignment 3/EE15B025$ ive
ganga@ganga-GL502VMK:~/Documents/Academics/5th Semester/EE2003 : Computer Organisation/Assignments/Assignment 3/EE15B025$ vvp
VCD info: dumpfile testbench.vcd opened for output.
 Control bit : 1 A = 0000 B = 0000 Result = 0000 Carry = 1 Overflow bit = 0
 Control bit : 1 A = 0011 B = 0010 Result = 0001 Carry = 1 Overflow bit = 0
 Control bit : 1 A = 0101 B = 1010 Result = 1011 Carry = 0 Overflow bit = 1
 Control bit : 0 A = 1111 B = 0011 Result = 0010 Carry = 1 Overflow bit = 0
ganga@ganga-GL502VMK:~/Documents/Academics/5th Semester/EE2003 : Computer Organisation/Assignments/Assignment 3/EE15B025$
```



| Type | Signals |
|---|---|
| reg | a[3:0] |
| reg | b[3:0] |
| wire | carry |
| reg | control |
| wire | overflow |
| wire | result[3:0] |