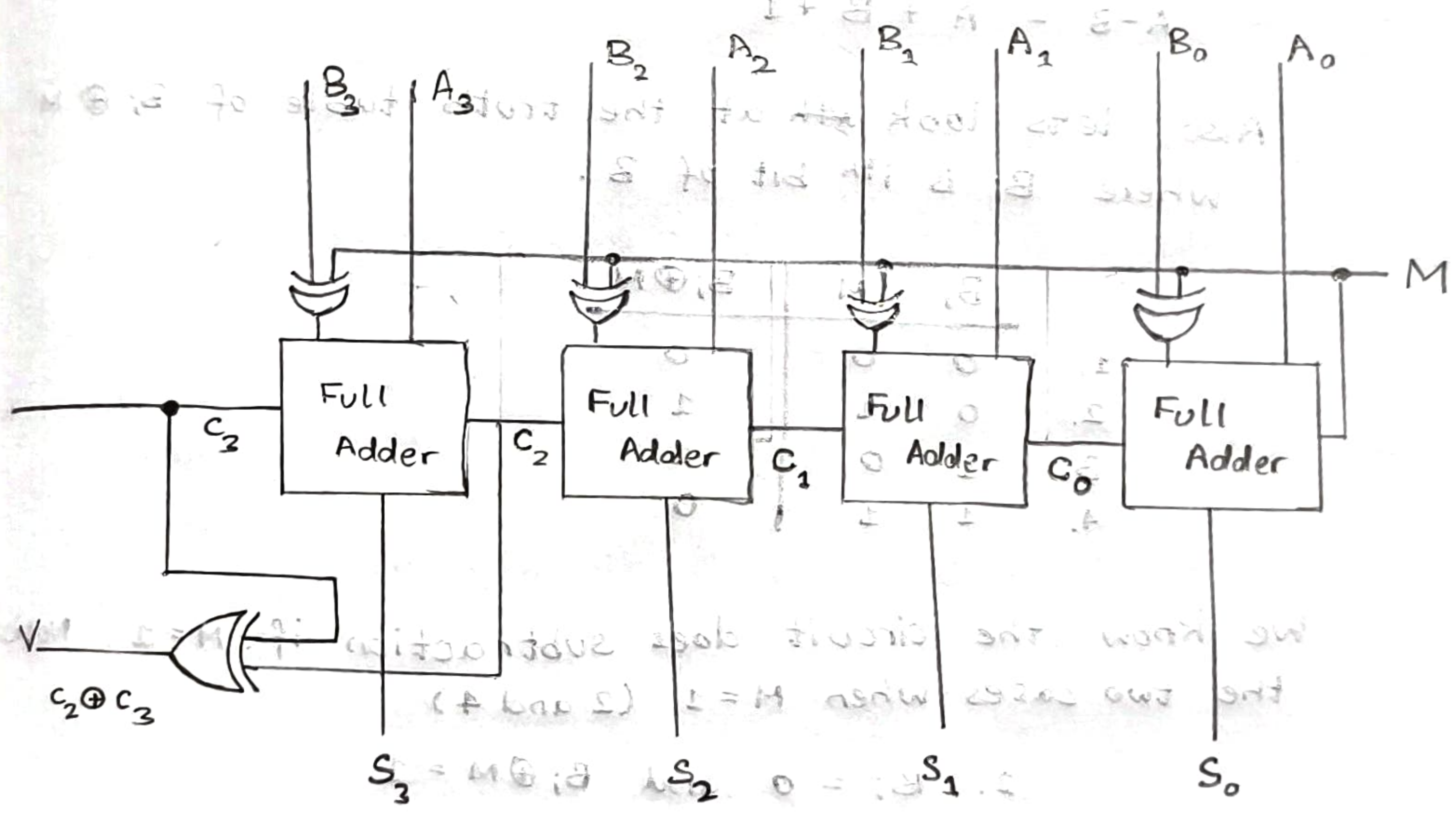
DIGITAL DESIGN

BINARY ADDER / SUBTRACTOR

~ Annamalai A.

0. What is the purpose of this circuit ?

The circuit is capable of doing both addition and subtraction of two numbers. It returns the result of the desired operation and an overflow bit.



1. How can it do two operations in a single circuit ?

The input M is of 1 bit which tells whether to do ~~a~~ addition or subtraction.

The XOR gate present before each full adder $(B \oplus M)$ tells the full adder whether to do addition or subtraction.

If $M=0$, the circuit does ~~subtr~~ addition, and if $M=1$, the circuit does subtraction.

2. How can a full adder do subtraction?

Consider two inputs A and B of 4 bits each.

A full adder can do addition by default (A + B). ~~But~~,

Now notice that the subtraction is:

$$A - B = A + (-B) = A + (2\text{'s complement of } B)$$

Further,

2's complement of B = $\bar{B} + 1$ (Add 1 to $\bar{B}$)

$\bar{B} = \bar{B}_3 \bar{B}_2 \bar{B}_1 \bar{B}_0$ here.

So, basically

$$A - B = A + \bar{B} + 1$$

Also, lets look ~~ath~~ at the truth table of $B_i \oplus M$

where B_i is ith bit of B.

	B_i	M	$B_i \oplus M$
1.	0	0	0
2.	0	1	1
3.	1	0	1
4.	1	1	0

We know the circuit does subtraction if M = 1. Notice the two cases when M = 1 (2 and 4)

2. $B_i = 0$ and $B_i \oplus M = 1$

4. $B_i = 1$ and $B_i \oplus M = 0$

So we have $\bar{B}$ when M = 1, and hence achieved a method to use M (mode) to do various operations.

Now, we need to add 1 if we are subtracting, i.e. M = 1. So we just pass M as C_{in} to the first full adder and it adds 1 to $\bar{B}$ if M = 1 (It adds if M = 0 too. But that makes no difference).

So the full adder does:

$$A + \underbrace{(B \oplus M)}_{B \text{ or } \bar{B}} + \underbrace{M}_{\substack{1 \text{ if sub.} \\ 0 \text{ if add.}}}$$

3. What is overflow ?

Overflow (V) returns the operation done was correct or wrong.

Since we are concerned about both addition and subtraction, the signed bit matters. For an n bit number the range is ~~-8~~, $-(2^{n-1})$ to $2^{n-1}-1$. So for 4 bits, it is -8 to +7.

Consider these examples:

$$\begin{array}{rl} {}^{0\,0} & \\ 0010 & (+2) \\ +\ 1100 & (-4) \\ \hline 1110 & (-2) \end{array} \qquad \begin{array}{rl} {}^{1\,1} & \\ 1101 & (-3) \\ +\ 1110 & (-2) \\ \hline 1011 & (-5) \end{array} \qquad \begin{array}{rl} {}^{0\,1} & \\ 0111 & (+7) \\ +\ 0110 & (+6) \\ \hline 1101 & (-3) \end{array} \qquad \begin{array}{rl} {}^{1\,0} & \\ 1001 & (-7) \\ +\ 1010 & (-6) \\ \hline 1011 & (+3) \end{array}$$

Correct	Correct	Wrong	Wrong
$C_2 \oplus C_3 = 0$	$C_2 \oplus C_3 = 0$	$C_2 \oplus C_3 = 1$	$C_2 \oplus C_3 = 1$
$V = 0$	$V = 0$	$V = 1$	$V = 1$

Due to the sign bit, change in MSB causes big difference. ~~Although +7+6~~ Although $(+7)+(+6) = +13$, we get -3 because of the signed bit. So we need a way to tell if the result was wrong (due to overflow), and that is this.

A common pattern that can be observed is that if $C_2 \neq C_3$, or $C_2 \oplus C_3 = 1$, then the operation is wrong. So

$$V = C_2 \oplus C_3$$