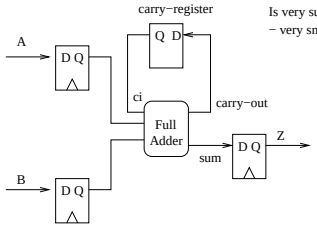
Bit Serial Adder



Is very suitable when the data is coming in/going out serially.
− very small and can deal with data streams at ~ 1Ghz (0.18um)

Carry−Save Adder (CSA) and Carry Save Trees

Regular Way

− add each column and bring carries over to the next column
− the Cn + An + Bn = {Cn+1, SUMn}

```
          carry−bits
            00110
A           10011   19
+ B       + 00110   6
            011001  25
```

This calculation can also be done if we seperately produce the sum and carry bits and add them at the end!

```
                    10011   19
                  + 00110   6
(A xor B) IntSum    10101   21
Carry               00110   2*2
                    011001  25
```

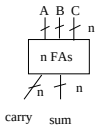
So far this isn't particularly usefull, but if we look at a 3 input adder:

```
A       01100   12
+ B     10011   19
+ C     00110   6
Sum bits      11001   25
Carry bits    00110   6*2
Final Result  100101  37
```

INDEPENDENTLY, for each column produce a sum and carry bit with a normal full−adder

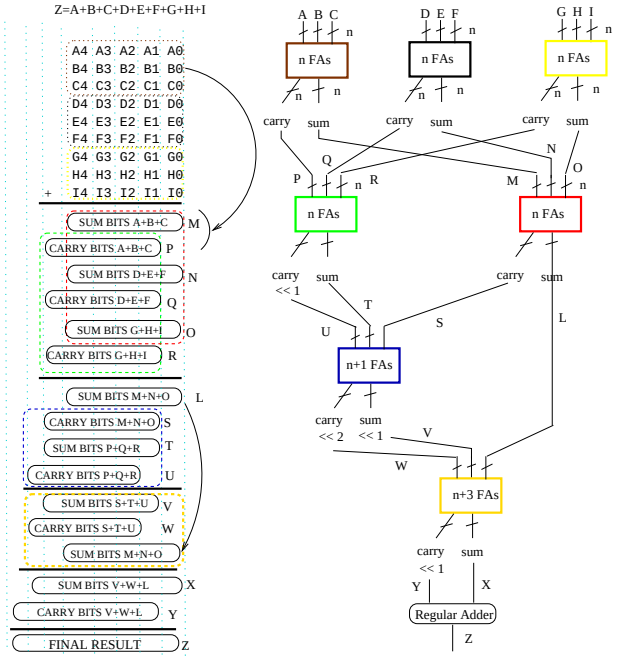
Eventually add them up for the final result.

A CSA adder representation



A 3 to 2 reduction of terms!

We can build a tree and get a logarithmic circuit!



Delay = 4 Full Adders + regular cleanup adder

Multipliers Intro

```
A       1 0 1 1 0   22
* B     0 1 1 0 1   13
carry bits
        1 0 1 1 0
      0 0 0 0 0
    1 0 1 1 0
  1 0 1 1 0
0 0 0 0 0
1 0 0 0 1 1 1 1 0   286
```

Output width is width of A + width of B

If the multiplier is implemented with CSA adders it is a 'Wallace Tree'

Signed Multipliers: Sign extend and remove extra logic

```
A       1 1 1 1 1 1 0 1 1 0   −10
* B     0 0 0 0 0 0 1 1 0 1   13
  1 1 1 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0 0 0
  1 1 1 1 1 1 0 1 1 0
    1 1 1 1 1 1 0 1 1 0
      0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 0 0 0
              0 0 0 0 0 0 0 0 0 0
1 1 0 1 1 1 1 1 1 0   = −130
```

Output width is width of A + width of B