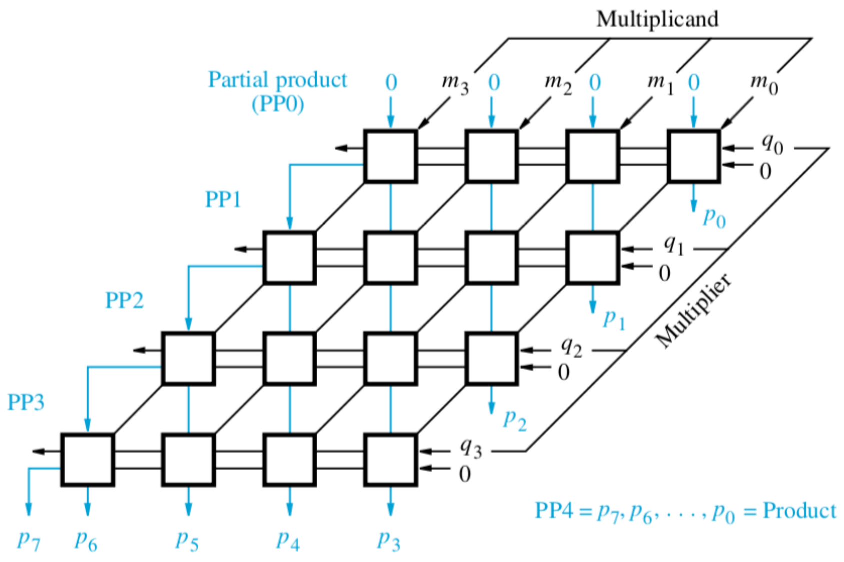
Arithmetic

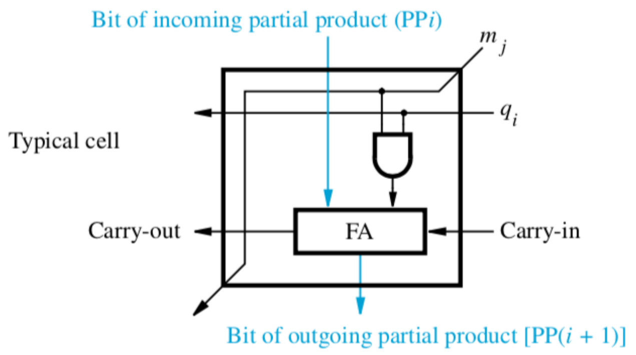
0.1 Full Adders ala ECE 124

$$s_i = x_i \oplus y_i \oplus c_i \quad ; c_{i+1} = y_i c_i + x_i c_i c_i y_i$$

0.2 Multiplication

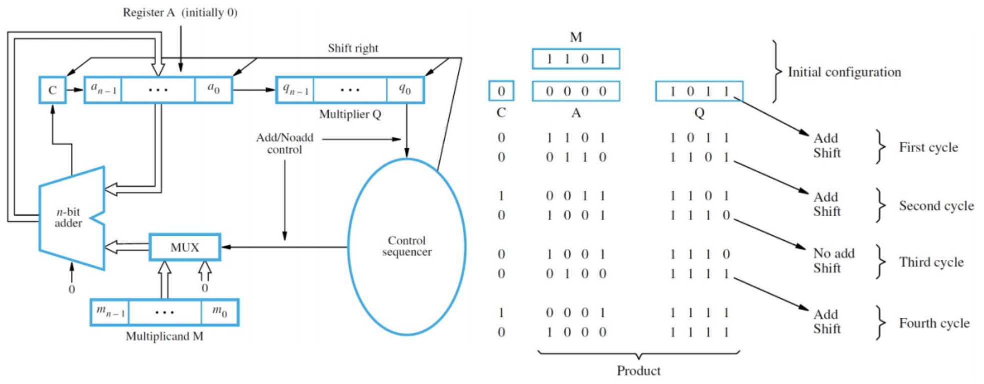
0.2.1 Unsigned - Raw Multiplication




Bit of incoming partial product (PPi)
m_j
q_i
Typical cell
Carry-out
FA
Carry-in
Bit of outgoing partial product [PP(i + 1)]

- Same as decimal multiplication only with ones and zeros
- Parts:
  - Multiplicand: The number at the top
  - Multiplier: The number at the bottom. Preferred to be the one with the fewer number of 1's
  - The digital implementation of the raw multiplication involves a lot of cells, each with an and gate, and a full adder, alongside an input wire carrying the carry-in from the previous step

0.2.2 Unsigned - Sequential Multiplication



- Trade-off between performance and effective resource utilization
- Reduce hardware replication by using a single adder, and data buffers to keep track of partial products and carrier

0.2.3 Signed - Raw Multiplication

Same as unsigned raw multiplication, but only with **sign extension at every stage**

0.2.4 Signed - Booth Multiplication

Iterate from left to right in the multiplier

1. Assume there is zero to the right of the recoded number, and then proceed with normal booth recoding
2. If there is no change in the numbers add another 0
3. If there is a transition from 0 to 1, append -1
4. If there is a transition from 1 to 0, append 1

After recoding the multiplier [$1\ 1\ 0\ 1\ 0 \rightarrow 0\ \ -1\ \ +1\ \ -1\ 0$]proceed with the multiplication as normal, when you get a -1, add the **sign extended 2's complement** of the multiplicand

- If a recoded multiplier has fewer 1s (+ve or -ve) than the original multiplier, Booth multiplication will require fewer operations. [there is no guarantee]
- Worst case: double the operations, Best case: just one operation

0.2.5 Bit Pair Recoding

Same strategy as booth algorithm's this time in groups of three. The least significant block uses only two bits of the multiplier, and assumes a zero for the third bit.
Again starting from the right, taking 2 bits at a time. Notice (+1 -1) is the same as (0 -1). Similarly (-1 0) is the same as (0 -2) Just continue this to get the bit-pair recoded multiplier.
Proceed with multiplication as usual now, only difference being sign extension + 2's complement (for negative digit) and shifting (for the 2's)

0.3 Real Numbers

0.3.1 Fixed Point representation

Decimal at fixed position, same as 204.

0.3.2 Floating Point representation

$$F = M * \beta^E \text{ [M = mantissa, E = exponent, } \beta \text{ = Base]}$$

It is generally assumed that the floating point numbers are normalized, i.e. the most significant digit is assumed to be 1 [saves a bit in the representation]. They can be *normalized* by shifting, and *denormalized* by shifting right.

Extra bits known as *guard bits* are retained by floating-point units to help preserve accuracy during arithmetic operations. The guard bits are used during normalization and rounding to preserve the accuracy of a calculated result

In all operations we have to set NZCV status bits. The Z not only represents zero, but also stands for *inexact and invalid*
Inexact is the name for a result that requires rounding in order to be represented in one of the normal formats.
Invalid exception occurs if operations such as 0/0 or $\sqrt{-1}$ are attempted.

10.3.2.1 Single-precision floating point representation

$$-1^S \times q.M \times 2^{E'-127}$$

- S: 1 Sign bit
- E': 8 bits encoded as excess 127 i.e. E' = actual exponent - 127
- M: 23 bits

10.3.2.1 Double-precision floating point representation

$$-1^S \times 1.M \times 2^{E'-1023}$$

- S: 1 Sign bit
- E': 11 bits encoded as excess 1023 i.e. E' = actual exponent - 1023
- M: 52 bits

10.3.2.2 Addition and Subtraction

1. Get the two numbers to the larger exponent by shifting the one with the smaller mantissa right (this is also the exponent of the result)
2. perform addition/subtraction on the mantissa keeping track of the sign bit
3. Normalize the result if needed

10.3.2.3 Multiplication and division

Multiplication:

1. Add the exponents and subtract 127
2. Multiply the mantissas and determine the sign of the result
3. Normalize the resulting value, if necessary

Division:

- Subtract the exponents and add 127
- Divide the mantissas and determine the sign of the result
- Normalize the resulting value, if necessary

10.3.2.4 Denormalization

When the value of the E field is 0 and M 0, a denormalized floating-point number is represented. Denormalized numbers can be used to allow for gradual underflow and to allow for higher precision on very small values.

- Single Precision Floating Point: F = $-1^S 0.M2^{E-126}$
- Double Precision Floating Point: $F = -1^S 0.M2^{E-1022}$

For denormalized numbers, you don't use $2^{(E-127)}$ any more, but rather 2^{-126}. The reasoning for this is because then there is not a gap between the smallest normal number ($1.00\dots0 \times 2^{-126}$) and the largest denormalized number ($0.111\dots1111 \times 2^{-126}$).

10.3.2.5 Truncation

Unbiased approximations are advantageous if many operands and operations are involved in generating a result, because positive errors tend to offset negative errors as the computation proceeds. Statistically, we can expect the results of a complex computation to be more accurate.

- *Chopping*: Remove guard bits and make no changes to other bits. The result of chopping is a biased approximation because the error range is not symmetrical about 0
- *Von Neumann rounding*: If the bits to be removed are all 0s, they are simply dropped, with no changes to the retained bits. However, if any of the bits to be removed are 1, the least significant bit of the retained bits is set to 1. Although the range of error is larger with this technique than it is with chopping, the maximum magnitude is the same, and the approximation is unbiased because the error range is symmetrical about 0.
- *Rounding*: Round to the nearest number or nearest even number in case of a tie. A 1 is added to the LSB position of the bits to be retained if there is a 1 in the MSB position of the bits being removed. To break the tie in an unbiased way, one possibility is to choose the retained bits to be the nearest even number.

10.3.2.6 Special Values

Special Value	S	E'	M
-0	1	0	0
+0	0	0	0
$-\infty$	1	255	0
$+\infty$	1	255	0
Denormalized value	X	0	M$\neq$0
NaN	X	255	M$\neq$0