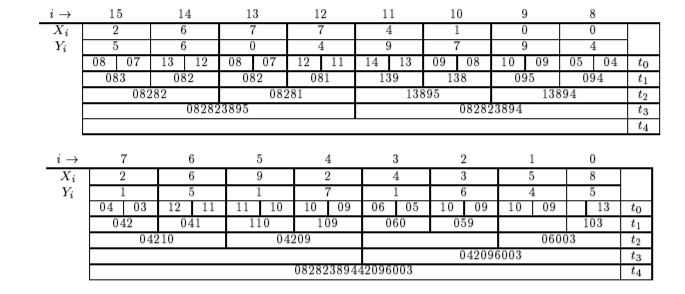
**Addition algorithms**

1. Fixed point
   1. Conditional sum:
      1. sum every corresponding two bits together independently two times, first one as if the carry was 0 and second one as if the carry was 1
      2. each summation of the two bits we get the carry bit as a selector for the next two results to concatenate with,
      3. then we repeat this until the total number is resulted

example for the sum of the two numbers

2677410026724358

5604979415171645

Advantages:

Very fast algorithm and it undependable on the previous stages for the carry

* 1. Carry look ahead:

In this algorithm it simply makes the addition combinational, for each two corresponding bits calculate the carry and sum the three bits then output the result,

As explained in the equations stated:

S(i) = A(i) XOR B(i) XOR C(i) in general

C(i+1) = A(i)B(i) + C(i)(A(i) + B(i)) where i>=1

Assuming G(i) = A(i)B(i) and P(i) = A(i) + B(i))

Then

C1 = G0 + P0C0

C2 = G1 + P1C1

By substitute C1 in C2

C2 = G1 + P1G0 + P1P0C0

In general

C(i+1) = G(i) + P(i)G(i-1) + P(i)P(i-1)G(i-2) + ….. + P(i)P(i-1)P(i-2)….P0C0

So the two equation of the addition becomes

S(i) = A(i) XOR B(i) XOR C(i)

C(i+1) = G(i) + P(i)G(i-1) + P(i)P(i-1)G(i-2) + ….. + P(i)P(i-1)P(i-2)….P0C0

Advantages:

Easy to implement and module in the integrated circuits.

Reference:

<https://web.stanford.edu/class/ee486/doc/chap3.pdf>

1. Floating point
   1. FP\_ADD1:

Assuming the two numbers A ,B are represented in IEEE-754 representation

|  |  |  |
| --- | --- | --- |
| Sign | Exponent | Mantissa |
| 1 | 10 | 5 |

A = S1E1M1 B = S2E2M2

1. Compare the two exponent and find the larger
2. If E1>E2: M2` = shift M2by E1-E2 and E = E1

Else: M1` = shift M1 by E2-E1 and E = E2

1. Sum = M1` + M2` and Esum = E1 or E2

Advantages:

Reduces the latency and the power consumption with out affecting the correctness also increases the performance of the addition unit

Reference:

<http://www.arpnjournals.org/jeas/research_papers/rp_2017/jeas_0117_5606.pdf>

* 1. FP\_ADD2:

<https://web.stanford.edu/class/ee486/doc/smith1999.pdf>

explained in details in this link.

**Multiplication**

1. Fixed point:
   1. Algorithm1:

The multiplication equation: Multiplicand\*Multiplier = result

Steps:

1. Start from the 0 bit
2. Test the current bit equals to zero
3. If true do nothing

If false add the multiplicand to the result in the Product result

1. Shift the Multiplicand register left 1 bit
2. Test if there is the last bit
3. If false go to the next one

If true output the result

* 1. Booth algorithm:

Assuming A is the accumulator initialized with zero, M is the multiplicand, Q is the multiplier, q0 is additional bit initialized with zero,q1 is the first bit of the multiplicand, n is the number of bits

Steps:

* + 1. Check for q1q0
    2. If q1q0 = 00 or 11 then do nothing
    3. If q1q0 = 10 then A = A – M
    4. If q1q0 = 01 then A = A + M
    5. Arithmetic shift right for AQq0
    6. n = n-1
    7. check for n
    8. if n != 0 then repeat
    9. if n=0 then output the result (the result will be in AQ concatenated together)

Reference:

1. <https://www.allaboutcircuits.com/technical-articles/multiplication-examples-using-the-fixed-point-representation/>
2. <https://www.researchgate.net/figure/Fig-2-Flow-Chat-of-Booth-Multiplier-Booths-algorithm-can-be-implemented-by-repeatedly_fig2_324819984>
3. <https://www.studytonight.com/computer-architecture/booth-multiplication-algorithm>
4. Floating point:
   1. FP Multiplication algorithm1:

Assuming the numbers are represented in IEEE-754 representation

* + 1. Exponents of the two numbers are added directly, extra bias is subtracted from the exponent result.
    2. Significands multiplication of the two numbers using fixed point algorithm
    3. Finding the sign of the result
    4. result is normalized such that there should be 1 in the MSB of the result
  1. FP Multiplication algorithm2:

The main advantages of this algorithm using the pipelining

Assuming the numbers are represented in IEEE-754 representation and the two numbers are A & B

* + 1. First multiply the mantissa of A with the first half of B mantissa and add the two exponents
    2. Second multiply A with the second half of the B mantissa and add the carry form the first step, while doing that round the result of the first step
    3. Round the result from the previous step and then add it to the total result

Advantages:

This will allow the multiplier to work in higher clock rate than making it in one stage

References:

1. <https://www.slideshare.net/grintokyro/chapter-03-arithmetic-for-computers>
2. <http://wineyard.in/Abstract/mtech/VLSI/bp/WYV54.pdf>
3. <http://www.eng.tau.ac.il/~guy/Papers/dual_precision_FP_MUL.pdf>