VLSI project

Implementation team

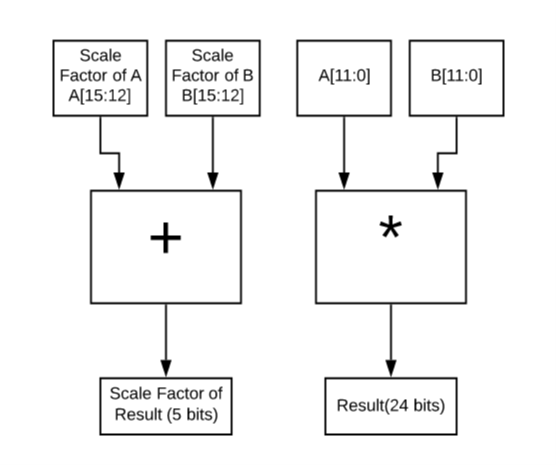
Design:

Multiplication:

Input: two 16-bit numbers (4 bits for scale factor & 12 bits for magnitude)

Output: 29-bit number (5 bits for scale factor & 24 bit for magnitude)

Limitations: can only express numbers with 12 bits

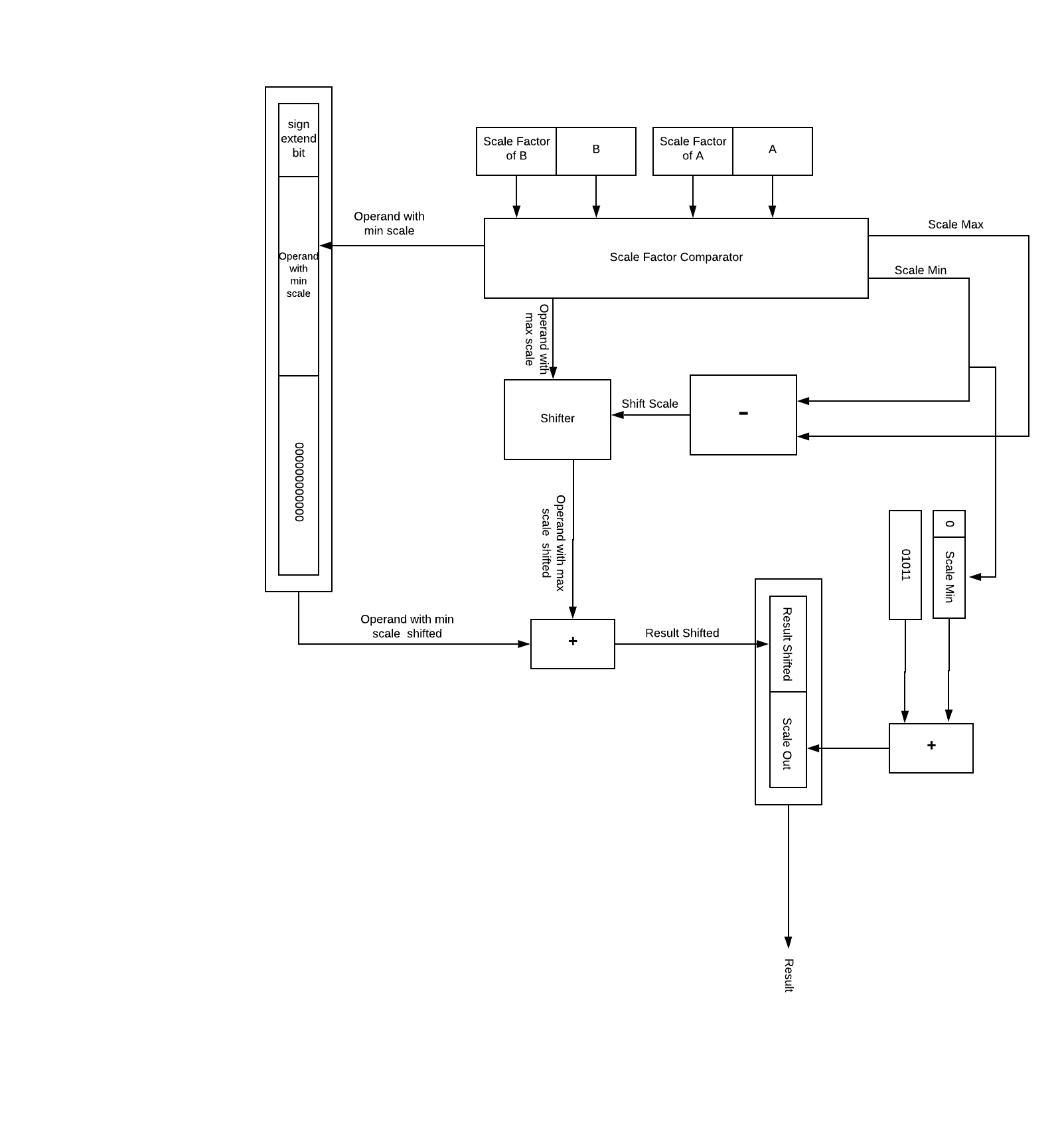


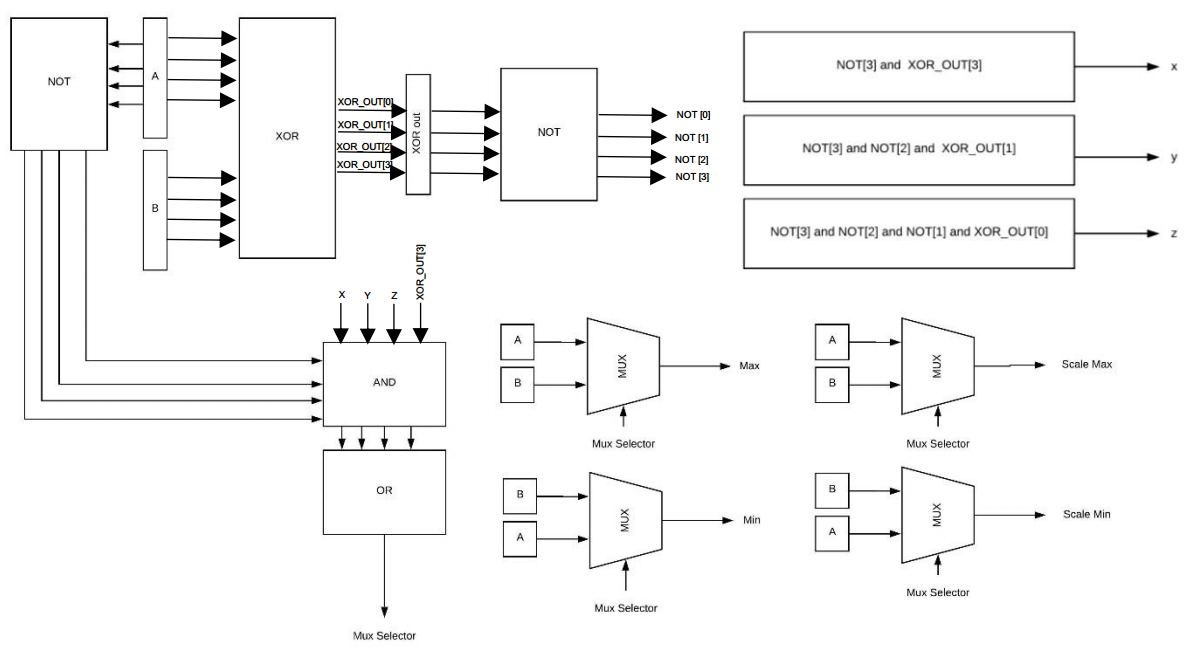
Addition:

Input: two 16-bit numbers (4 bits for scale factor & 12 bits for magnitude)

Output: 29-bit number (5 bits for scale factor & 24 bit for magnitude)

Limitations: can only express numbers with 12 bits





Adder:

Area:

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Module | Cells | Cell Area |
| 1 | top | 206 | 258 |

Timing:

|  |  |  |
| --- | --- | --- |
| delay | Data arrival time | Slack |
| 18000.0 ns | 903.9 ns | 41096.1 ns |

Power:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instance | Internal Power(uw) | Switching Power(uw) | Leakage Power(uw) | Total Power(uw) |
| \*i\_0\_0\_0 | 1.656095 | 3.800332 | 0.013429 | 5.469854 |
| \*i\_0\_0\_1 | 2.101796 | 3.969229 | 0.017742 | 6.088767 |
| \*i\_0\_0\_2 | 2.193800 | 4.028895 | 0.017742 | 6.240437 |
| \*i\_0\_0\_3 | 2.234306| | 4.059590 | 0.017742 | 6.311637 |
| … | … | … | … | … |
| 206 | 0.685771 | 3.479996 | 0.005333 | 4.171101 |
| \*TOTAL | 132.220764 | 602.471924 | 1.448580 | 736.141296 |

Multiplier:

Area:

|  |  |  |  |
| --- | --- | --- | --- |
| Instance | Module | Cells | Cell Area |
| 1 | top | 457 | 731 |

Timing:

|  |  |  |
| --- | --- | --- |
| delay | Data arrival time | Slack |
| 18000.0 ns | 909.0 ns | 41091.0 ns |

Power:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instance | Internal Power(uw) | Switching Power(uw) | Leakage Power(uw) | Total Power(uw) |
| \*i\_0\_0 | 368.966522 | 683.641418 | 3.439007 | 1056.046875 |
| \*i\_0\_1 | 8.826293 | 18.002272 | 0.065275 | 26.893839 |
| \*TOTAL | 377.792847 | 701.643677 | 3.504282 | 1082.940796 |

Design limitations and assumptions

We use last 4 bits in the array as the scale factor so the actual number can't exceed 12 bits and the scale factor can't exceed "1100".

Both multiplication and addition outputs are 29 bits Last 5 bits are the new scale factor and the other 24 bits are the result without the fixed point.

Adder algorithms

1. Fixed point:
   1. Carry-Skip Adder:
      1. Reasons: best area & delay values.
   2. Carry-Select:
      1. Reasons: better delay but more area

Reference: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.5931&rep=rep1&type=pdf>

1. Floating point:
   1. A Floating-Point Fused Add-Subtract Unit
      1. Reference: <https://www.researchgate.net/publication/4367369_A_Floating-Point_Fused_Add-Subtract_Unit>
   2. A 64-bit Decimal Floating-Point Adder:
      1. Reference: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.5931&rep=rep1&type=pdf>

Multiplier algorithms

1. Fixed point:
   1. Booth Multiplication
      1. Less hardware and less additions
   2. Binary Multiplication
      1. Faster but much bigger hardware
2. Floating point:
   1. Dual precision IEEE floating-point multiplier
      1. Less hardware and capable of performing either a double-precision multiplication or a single-precision multiplication
   2. An Efficient Implementation of Floating-Point Multiplier
      1. Higher Speed and more precision through not rounding

Task Distribution:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Task | Hours spent | problems faced |
| Ahmed elkrashily | Multiplier design + multiplier algos | 4 | Bad Internet + Unclear Document |
| Abdullah Ezzat | Testbench | 3 | Unclear Document |
| Omar salah | Synthesizing+reports +adder algos | 5 | None |
| Mohamed ahmed Mohamed | Multiplier & adder implementation +  Software test generator + Adder design | 12+ | None |