A Well-Equipped Implementation: Normal/Denormalized Half/Single/Double Precision IEEE 754 Floating-Point Adder/Subtracter

Brett Mathis and James E. Stine
{brett.mathis, james.stine}@okstate.edu
VLSI Computer Architecture Research Laboratory –
Oklahoma State University
Stillwater, OK 74078 USA
http://vlsiarch.ecen.okstate.edu

ASAP
2019

Outline

- Introduction/Motivation
- IEEE 754 Background
- Proposed Architecture Implementation
- Results
- Conclusion

ASAP
2019

IEEE 754 Floating-point Addition: Background and Motivations

- Literature currently spare on FP addition
  - Basic principles can be found – specifics seldom covered
- Intellectual Property access to FP add modules – limited selection and expensive

---

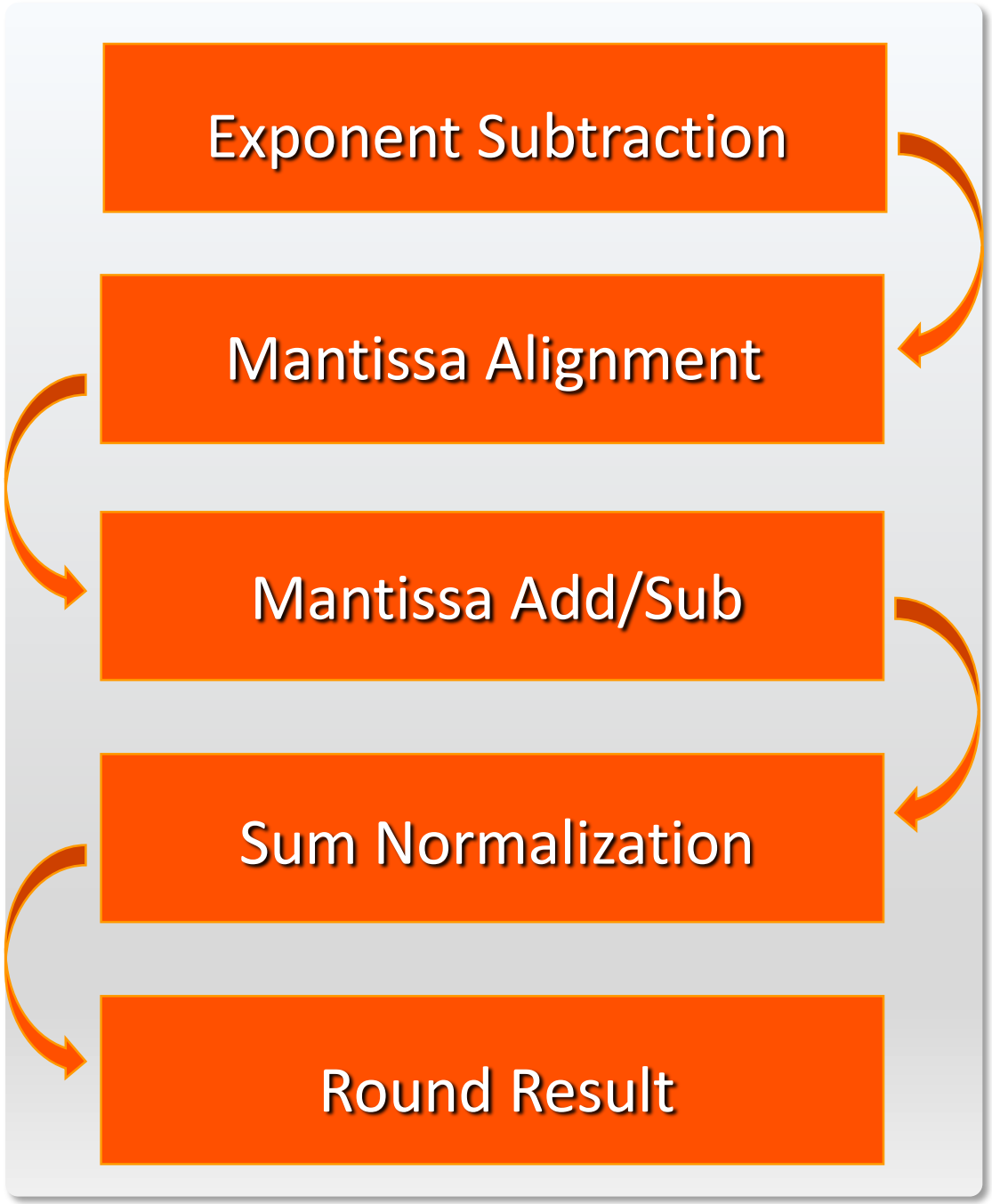
- IEEE 754 Operands use the following format:

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

S | EEEEEEEE | MMMMMMMMMMMMMMMMMMMMMMM

- Single precision binary representation

ASAP 2019



FP Add: Process Overview

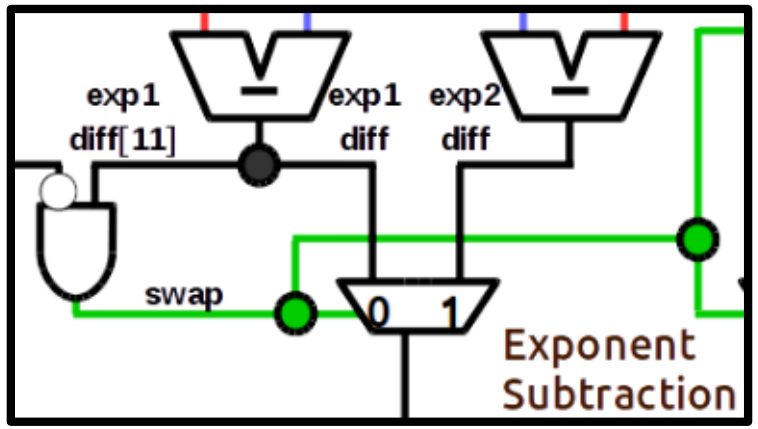
- Subsequent steps in datapath are dependent
- Substantial feature support can be added in parallel

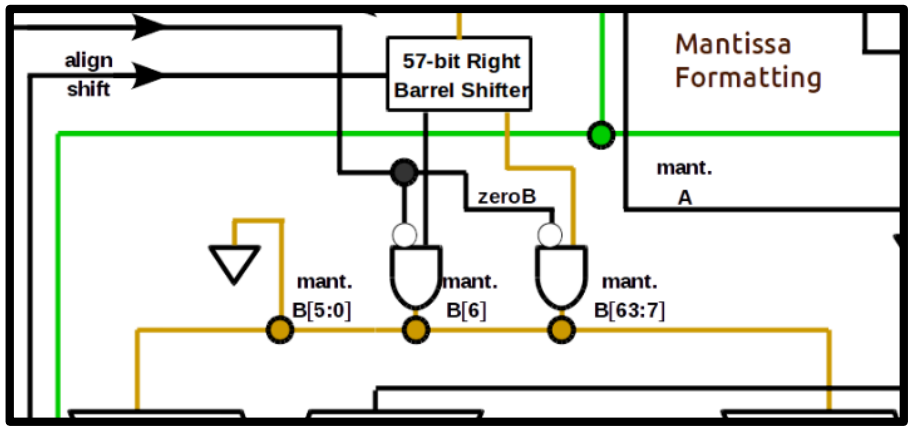
ASAP 2019

Exponent Subtraction & Mantissa Alignment

- IEEE 754 operands are not aligned by default
  - Ex: $1.23 \times 10^4 - 2.34 \times 10^5$ ➡ $0.123 \times 10^5 - 2.34 \times 10^5$
- Only operands with aligned mantissas can be added

- Exponent subtraction quickly determines this
  - Ex: $10^4 - 10^5$ ➡ 10^{-1} ➡ Right shift first mantissa by 1





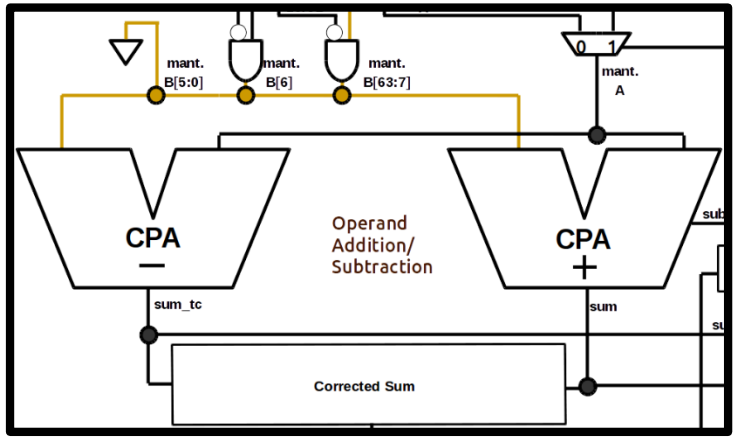
ASAP 2019

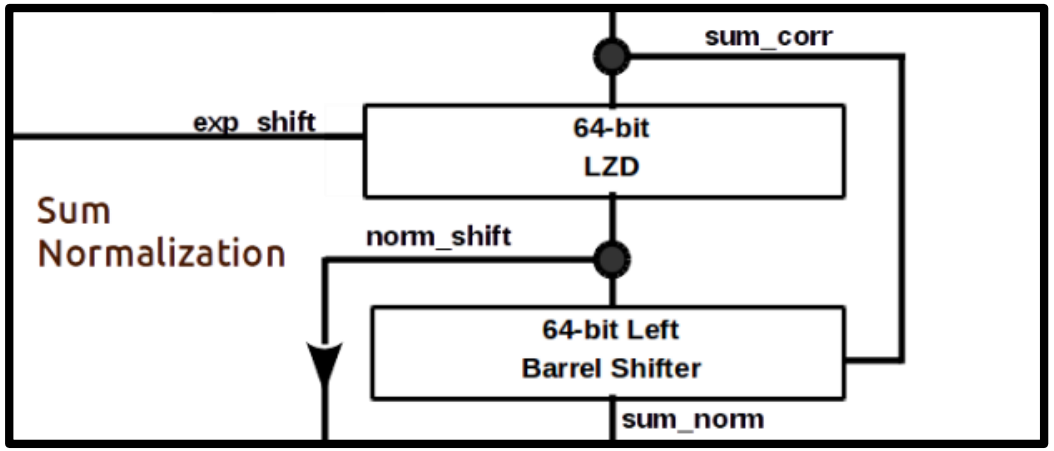
Primary Add / Post-Normalization

- Once mantissas are aligned, the operation can actually take place
- Afterwards, the resulting mantissa has to be re-normalized
  - Shift amount required is determined by the number of leading zeroes in the mantissa

S | EEEEEEEE | 00000 1XXXXXXXXXXXXXXXXXX

  - 5-bit left shift required to renormalize the mantissa





ASAP 2019

Rounding

- IEEE 754 (2008) supports 5 rounding modes:

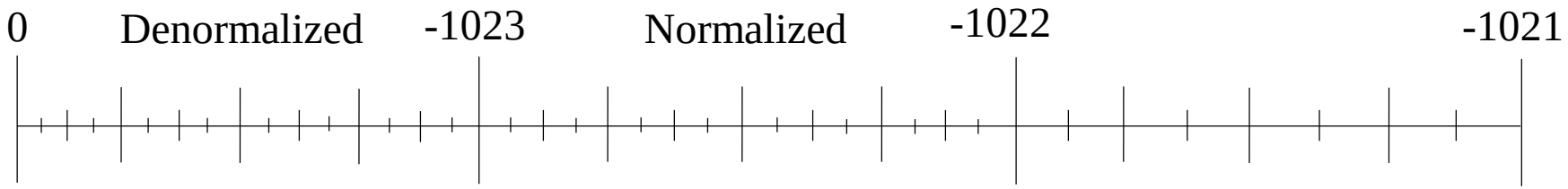
rm[2:0]	Rounding Mode
000	round-to-nearest-even
001	round-towards-zero
010	round-towards positive infinity
011	round-towards minus infinity
100	round-towards away

- Handled by conditionally adding 1 to LSB of post-normalized mantissa, using L (least), R (round), and S (sticky) bits:

$$\begin{aligned} \texttt{add\_one} \quad = \quad & (\overline{rm[1]} \cdot \overline{rm[0]} \cdot R \cdot (L + S)) + \\ & (rm[1] \cdot (Asign \oplus \overline{rm[0]}) \cdot (R + S)) \\ & + \ (\texttt{rm[2]} \cdot R) \ . \end{aligned}$$

Denormalized Values

- Denormalized values exist in the range between the lowest representable normal value (exponent > 0) and zero itself.



- Significantly increased levels of precision for very small numbers.
  - Used to also avoid gradual underflow.

IEEE 754 Single Precision Denormal Example

0x0060_0000 = 0_00000000_11000000000000000000000

$$\underline{0}.11_2 \times 2^{-126} = 1.1_2 \times 2^{-127} \approx 8.8162 \times 10^{-39}$$

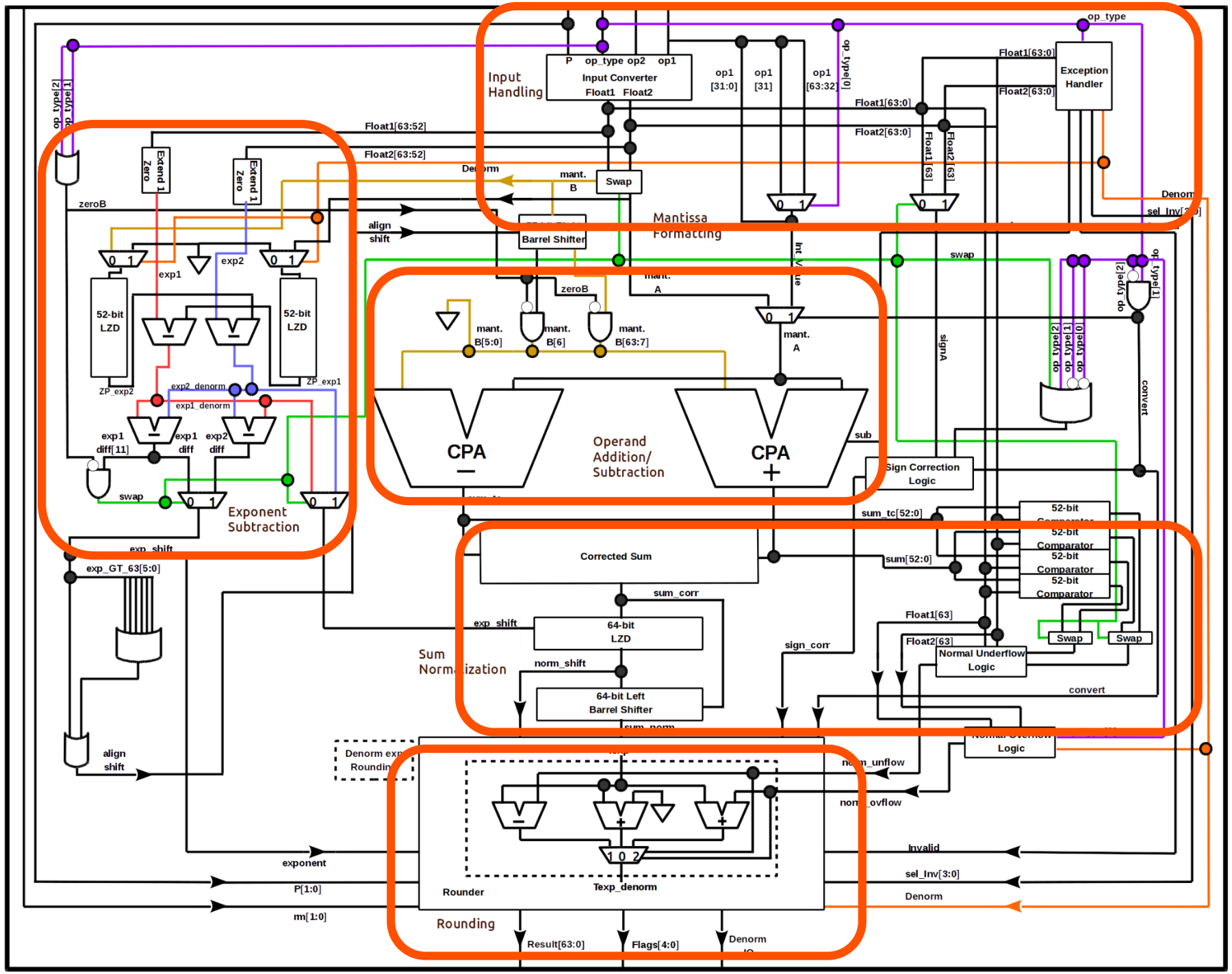
[J. Coonen, "Underflow and the Denormalized Numbers," *Computer*, vol. 14, no. 3, pp. 75–87, 1981, Fig. 6]

EXPONENT	SIGNIFICANT BITS	
−120	1XXXXX	
−121	1XXXXX	
−122	1XXXXX	
−123	1XXXXX	← NORMALIZED NUMBERS
−124	1XXXXX	
−125	1XXXXX	
−126	1XXXXX	
−126	01XXXX	← UNDERFLOW THRESHOLD = λ = 2 − 126
−126	001XXX	
−126	0001XX	← DENORMALIZED NUMBERS
−126	00001X	
−126	000001	
(−126)	000000	← ZERO

6-bit Example

ASAP 2019

Overall Design



ASAP 2019

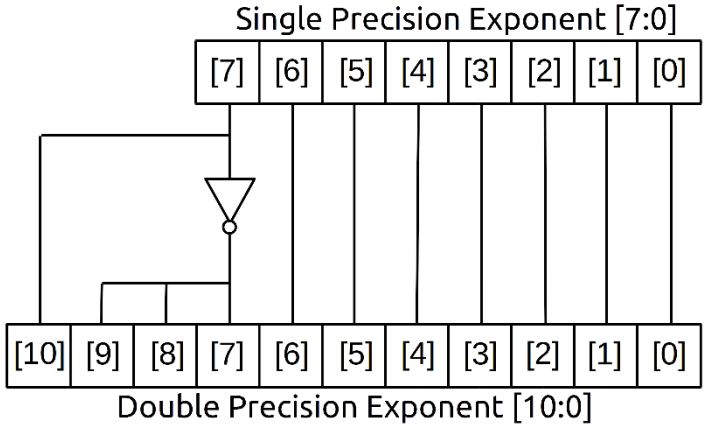
Current Instruction Support

Operation	op_type	P	Description
add.d	0000	00	Add two 754 double precision numbers
add.s	0000	01	Add two 754 single precision numbers
add.h	0000	10	Add two 754 half precision numbers
sub.d	0001	00	Subtract two 754 double precision numbers
sub.s	0001	01	Subtract two 754 single precision numbers
sub.h	0001	10	Subtract two 754 half precision numbers
cvt.w	0010	00	Convert a 64-bit two's complement integer to a 754 double precision number
cvt.w	0010	01	Convert a 64-bit two's complement integer to a 754 single precision number
cvt.w	0010	10	Convert a 64-bit two's complement integer to a 754 single precision number
cvt.b	0011	00	Convert a 32-bit two's complement integer to a 754 double precision number
cvt.b	0011	01	Convert a 32-bit two's complement integer to a 754 single precision number
cvt.b	0011	10	Convert a 32-bit two's complement integer to a 754 half precision number
cvt.h	0110	00	Convert a 16-bit two's complement integer to a 754 double precision number
cvt.h	0110	01	Convert a 16-bit two's complement integer to a 754 single precision number
cvt.h	0110	10	Convert a 16-bit two's complement integer to a 754 half precision number
abs.d	0100	00	Absolute value of a 754 double precision number
abs.s	0100	01	Absolute value of a 754 single precision number
abs.h	0100	10	Absolute value of a 754 half precision number
neg.d	0101	00	Negate a 754 double precision number
neg.s	0101	01	Negate a 754 single precision number
neg.h	0101	10	Negate a 754 half precision number
cvt.s.d	0111	00	Convert from a single precision number to a 754 double-precision number
cvt.d.s	0111	01	Convert from a double precision number to a 754 single-precision number
cvt.h.d	0111	10	Convert from a half precision number to a 754 double-precision number
cvt.d.h	0111	11	Convert from a double precision number to a 754 half-precision number
cvt.s.h	1111	10	Convert from a single precision number to a 754 half-precision number
cvt.h.s	1111	11	Convert from a half precision number to a 754 single-precision number

ASAP
2019

Input Conversion and IEEE 754 Exceptions

- Difficulty in converting between floating-point precisions lies in exponent conversion.
  - IEEE 754 exponents are unsigned with an offset per precision (-1023 for DP).
  - Sign is copied and mantissa is ripped or buffered.
- IEEE 754 exceptions include:
  - Infinity (exponent set to all 1)
  - Zero (all zero – including exponent)
  - qNaN (exponent set to all 1 – mantissa != 0)
  - sNaN (qNaN & mantissa MSB = 1)



$$E_{F64} = E_{F16} - 15 + 1023 = E_{F16} + 1008$$

$$E_{F32} = E_{F16} - 15 + 127 = E_{F16} + 112$$

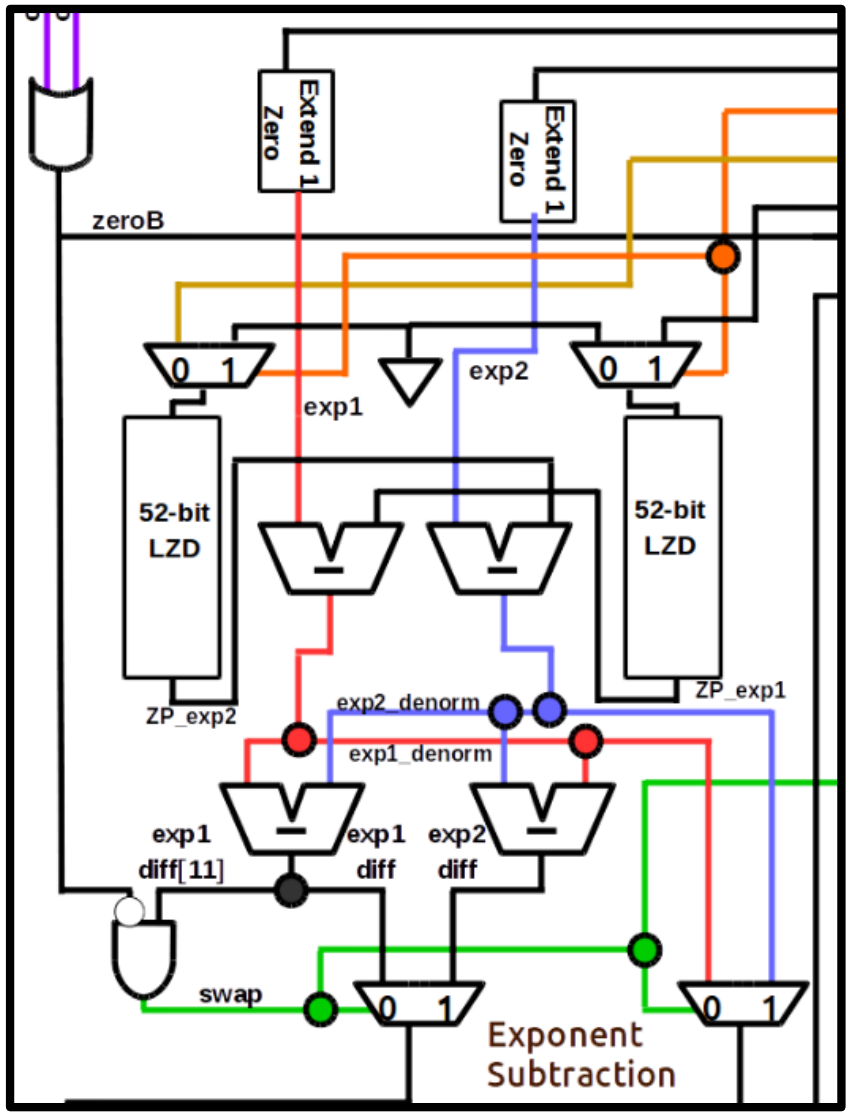
Example Conversions

Operation (+/-)	sNaN	qNaN	Normalized Number	Infinity	Zero
sNaN	qNaN	qNaN	qNaN	qNaN	qNaN
qNaN	qNaN	qNaN	qNaN	qNaN	qNaN
Normalized Number	qNaN	qNaN	IEEE-754 [1]	Infinity	IEEE-754
Infinity	qNaN	qNaN	Infinity	qNaN/Infinity	qNaN
Zero	qNaN	qNaN	IEEE-754	Infinity	Zero

ASAP 2019

Implementation of Exponent Subtraction

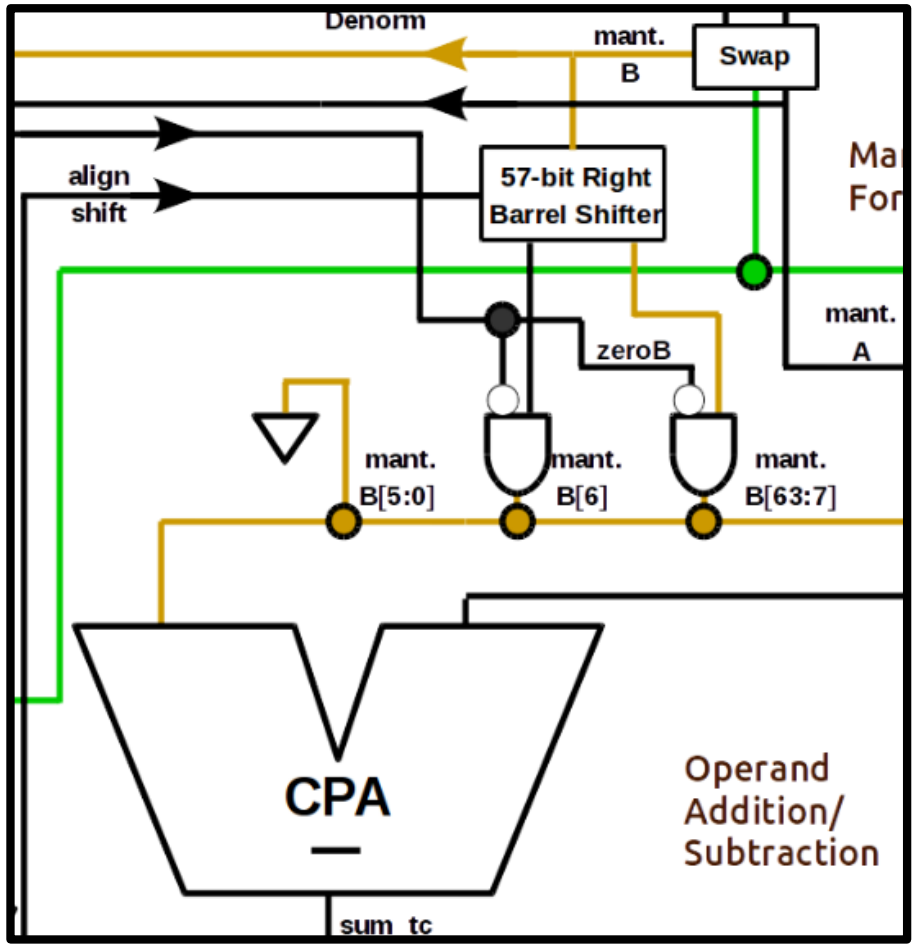
- Denormalized operands (w/ exponents set to all 0) need a LZD to correctly shift mantissa values.
- Precedes primary exponent subtracters – no effect on normalized operands.
- Both difference values are computed in parallel – positive value always used so only on shifter is needed.
  - Determines the ‘swap’ value used throughout architecture.
  - 1-bit buffer to remove effects of overflow.



ASAP 2019

Mantissa Alignment

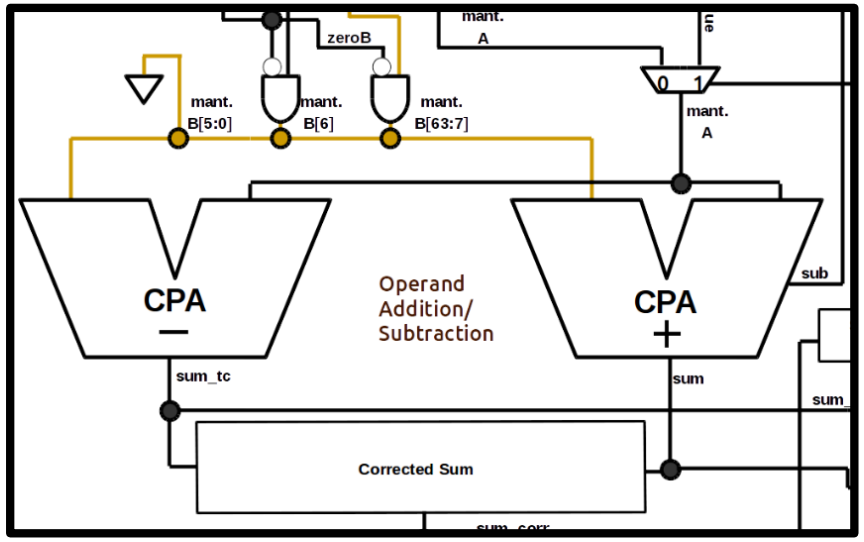
- Shift value gathered from previous stage used to right shift smaller mantissa – determined by ‘swap’ generated earlier.
- 52-bit double precision value is buffered to 57 bits before shift – includes guard and round bits on LSB and MSB of mantissa.
- Shifted values are also sign-extended here if integer conversion is required.



ASAP 2019

Primary Adder/Subtracter Structure

- Parallel carry-prefix adder (CPA) structure utilized in design.
- Mantissa exists on domain of [1,2) – negative values cannot exist.
- Typically corrected by performing two's compliment on result – parallel structure computes both and chooses.
- Logic for choosing the correct sum and sign value of the result:

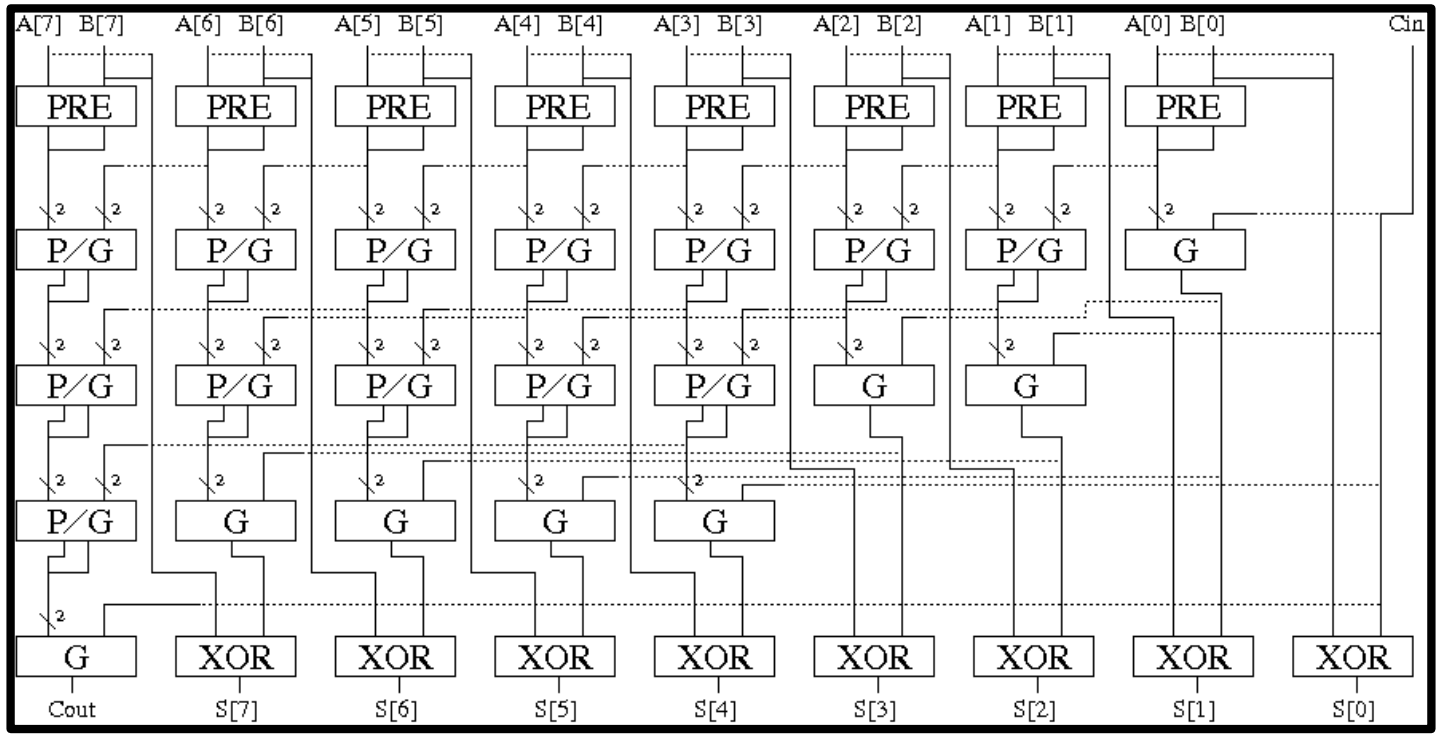


$$sign\_corr = ((corr\_sign \oplus signA) \cdot \overline{convert}) \oplus sum[63] .$$

$$sum\_corr = (DenormIn \cdot (opA\_Norm + opB\_Norm) \cdot (\overline{Float1[63]} \oplus \overline{Float2[63]} \cdot op\_type[0]) \cdot (Float1[63] \oplus Float2[63] \cdot \overline{op\_type[0]}) \oplus sum[63]) \ ? \ sum : sum\_tc .$$

ASAP 2019

Carry-prefix Adder Structure

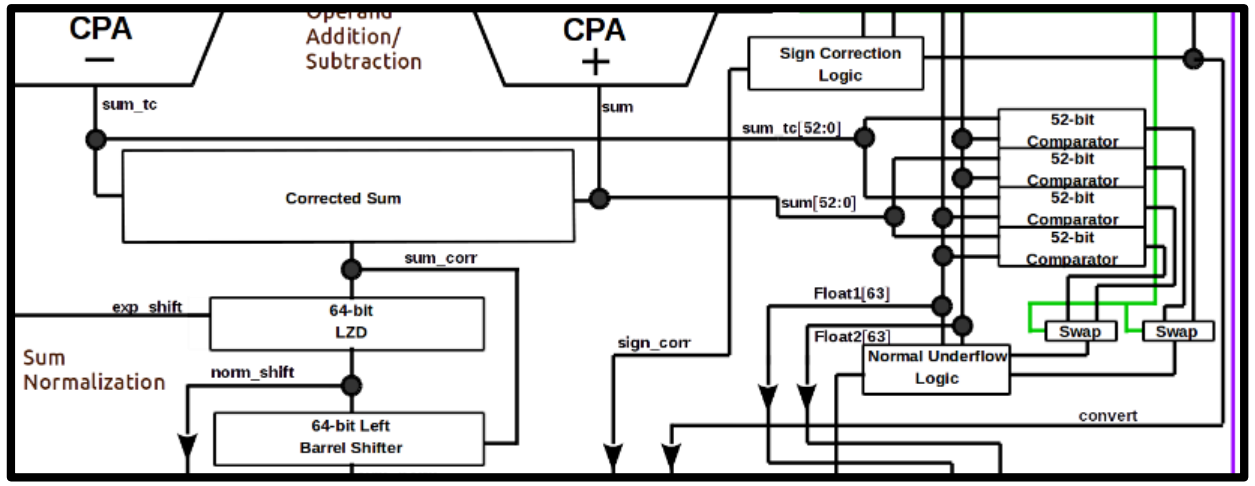


- Generation and propagation signals made as follows:
  - $\mathrm{G[x]} = \sim (A[x]\, B[x])$; $\mathrm{P[x]} = \sim (A[x] \mid B[x])$
- Output of subsequent P & G signals:
  - $\mathrm{G[x]} = \begin{cases} \sim(G[x] \mid (G[x-1]\, P[x])) \\ \sim(G[x]\, (G[x-1] \mid P[x]) \end{cases}$; $\mathrm{P[x]} = \begin{cases} \sim(P[x] \mid P[x-1]) \\ \sim(P[x]\, P[x-1]) \end{cases}$

ASAP 2019

Post-normalization Implementation

- To handle denormalized exponent rounding – ‘normal overflow’ and ‘normal underflow’ must be detected.
- Comparators are used to detect decreases in mantissa value for certain subtraction parameters.
- In parallel, the correct sum is normalized through the shift value generated from a LZD, similar to exponent subtraction.



$$\texttt{norm\_ovflow} = \texttt{sum[63]} \; ? \; \texttt{sum\_tc[52]} : \texttt{sum[52]} \; .$$

$$\texttt{norm\_unflow} = ((\texttt{opA\_Norm} + \texttt{opB\_Norm}) \cdot (\overline{\texttt{Float1[63]}} \oplus \overline{\texttt{Float2[63]}})) \; ? \; \texttt{mantissa\_comp} : 1'b0.$$

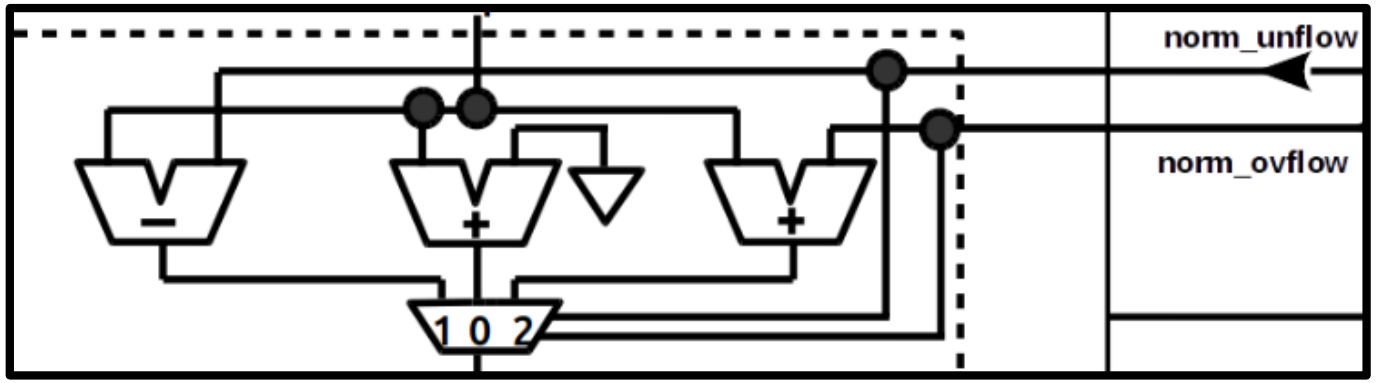
ASAP 2019

Rounding

- Rounding types and logic covered in previous slide
- Generation for L,R, & S bits:

$$(L,R,S) = \begin{cases} (\texttt{A[53]},\texttt{A[52]},S_{HP}) & \text{if } P = 10 \\ (\texttt{A[40]},\texttt{A[39]},S_{SP}) & \text{if } P = 01 \\ (\texttt{A[11]},\texttt{A[10]},S_{DP}) & \text{if } P = 00 \end{cases}.$$

- $S_{HP/SP/DP}$ is the logical or of all bits below the R bit for each precision.
- 'normal underflow/overflow' logic is used to correctly round the exponent at, above, or below zero.
  - Three 12-bit CPA's are used in parallel to compute the modified exponent for denormalized operations [Schwarz, et. al, IEEE TC 2005].



Results

- The proposed design is implemented in RTL-compliant Verilog and designs are then synthesized using an ARM 32nm CMOS library for Global Foundries (GF) cmos32soi technology optimizing for delay.
- Synthesis was optimized for delay utilizing Synopsys® (SNPS) Design Compiler™ in topographical mode using a PVT process at 25°C using TT corners.
- SNPS DW comparison is IP offered by Synopsys®, with synthesis results also generated in Synopsys® (SNPS) Design Compiler™ under the same parameters – SNPS DW used **only** supports non-denomalized addition and subtraction instructions for double precision only.
- Energy numbers generated through a VCD file based on approximately 50,000 vectors in SNPS PrimeTime.

IEEE 754 Adder/Subtracter	# Cells	Area [um^2]	Delay[ps]	Internal Power [mW]	Switching Power [mW]	Leakage Power [mW]	Total Power [mW]
Proposed Architecture (32nm)	8,484	9,671.9	634.81	1.1360	1.4580	5.1630	7.7570
SNPS DW (32nm)	5,289	6,652.6	616.21	5.2347	4.0829	3.1137	12.4313
Adder Architecture (45nm)	-	41,200.0	2,800.0	-	-	-	21.4600

ASAP 2019

Results (Continued)

- The 45nm adder architecture is an another academic implementation of FP add – no 32nm comparisons have been released to our knowledge.
- Our design is 2.94% slower the SNPS DW – this is within two FO4 inverter delays in 32nm.
- Our design is 45.4% larger than SNPS DW – this is expected, considering the small feature support of the SNPS DW module.
- Our design uses only 62.4% of the power of SNPS DW – at a larger area, we believe this may be due to the Synopsys® (SNPS) Design Compiler™ having inability to optimize our RTL Verilog to the same extent as their own modules under delay constraints.

IEEE 754 Adder/Subtracter	# Cells	Area [um^2]	Delay[ps]	Internal Power [mW]	Switching Power [mW]	Leakage Power [mW]	Total Power [mW]
Proposed Architecture (32nm)	8,484	9,671.9	634.81	1.1360	1.4580	5.1630	7.7570
SNPS DW (32nm)	5,289	6,652.6	616.21	5.2347	4.0829	3.1137	12.4313
Adder Architecture (45nm)	-	41,200.0	2,800.0	-	-	-	21.4600

Concluding Remarks

- Our architecture has comparable delay to currently used IP with more feature support.
- Current version verified and taped out in early 2019.
  - Completely verified with John Hauser's TestFloat package.
  - Addition testing done for denormals via a Java-based program.
  - http://www.jhauser.us/arithmetic/TestFloat.html
- Low power results for this design lend it towards mobile FPU's and GPU adders.
- Future work for this architecture:
  - Delay and area decreases via EAC CPA structure on all adders and subtracters in architecture.
  - LZA use for decreased delay in denomalized shift calculation and denomalized exponent logic comparison.

ASAP 2019

Q & A

ASAP
2019

http://vlsiarch.ecen.okstate.edu