Improved Architectures for a Floating-Point Fused Dot Product Unit

Jongwook Sohn and Earl E. Swartzlander, Jr.

April 8th, 2013

Outline

- Introduction
- Traditional FP Fused Dot Product Unit
- An Enhanced FP Fused Dot Product Unit
  - A New Alignment Scheme
  - Early Normalization and Fast Rounding Scheme
  - Four-Input LZA
- A Dual-Path FP Fused Dot Product Unit
  - Far Path Logic
  - Close Path Logic
- A Pipelined FP Fused Dot Product Unit
- Results
- Conclusion

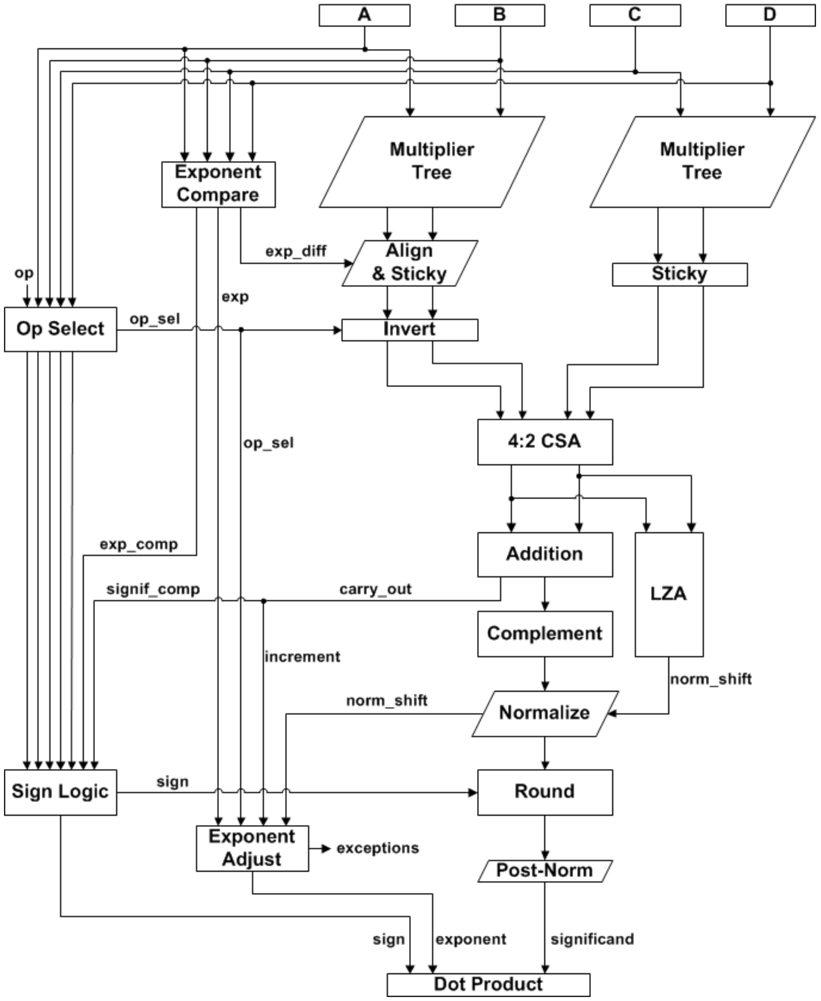
Introduction

- **Problem Statement**
  - **Fact – Floating-point operations are widely used for advanced applications:**
    - 3D graphics, multimedia, signal processing and scientific computations
  - **Problem – Floating-point operations require complex processes:**
    - Alignment, normalization and rounding
  - **Solution –Floating-point fused arithmetic units:**
    - FP Fused Multiply-Add [2] – [4], FP Fused Add-Subtract [5], [6] and FP Fused Two-Term Dot Product [7]
  - **Proposal – Improved floating-point fused two-term dot product unit**

Goto Backup

Traditional FP Fused Dot Product Unit

- **Traditional FP Fused Two-Term Dot Product Unit [7]**
  - P = AB ± CD
  - Useful for FFT butterfly and complex multiplication
  - Reduces area by 20%, Reduces latency by 2%, Improves accuracy

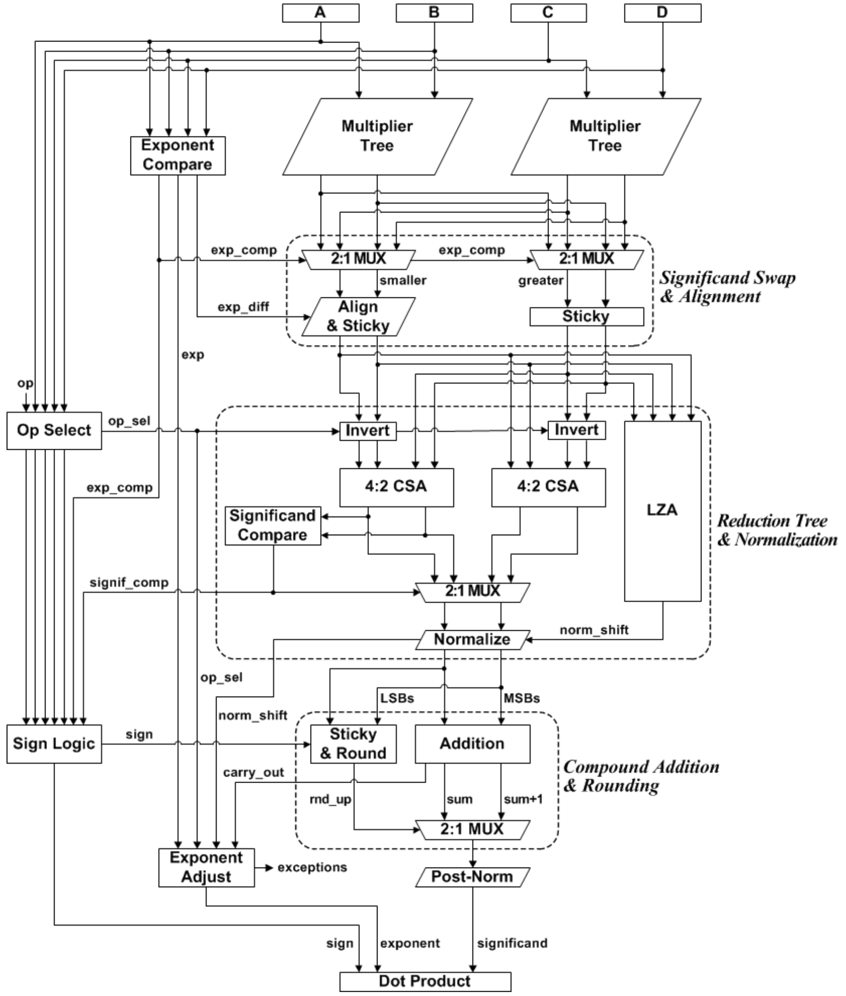


Goto Backup

Enhanced FP Fused Dot Product Unit

▪ Enhanced FP Fused Two-Term Dot Product Unit

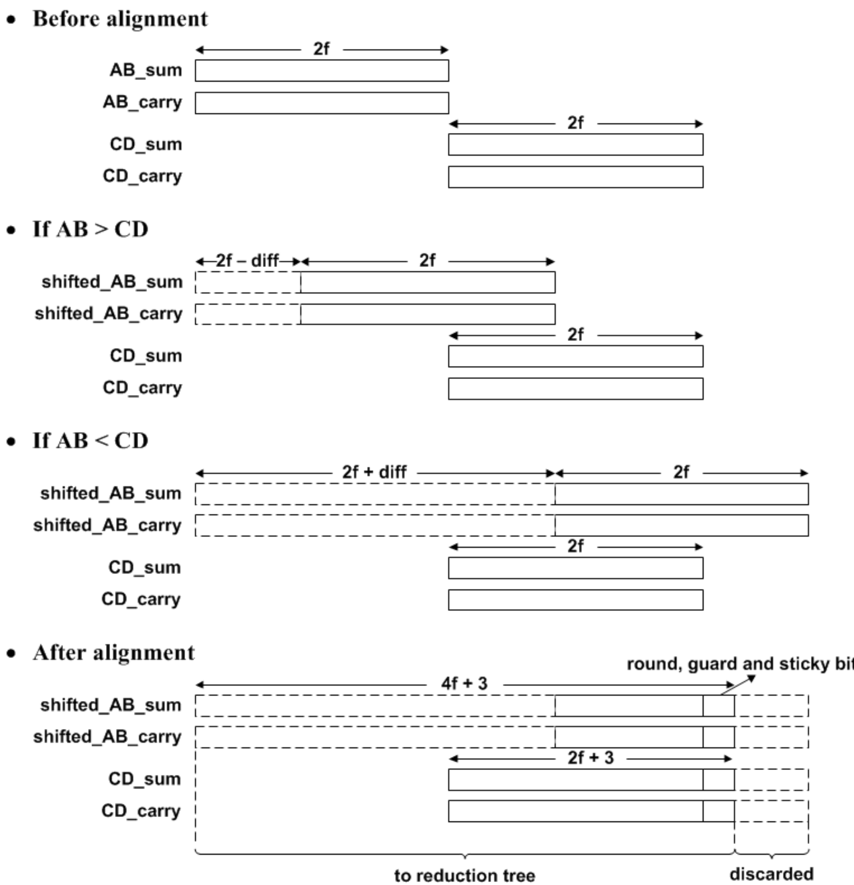
- New alignment scheme
- Early normalization
  & Fast rounding
- Four-input LZA



[Goto Backup](#)

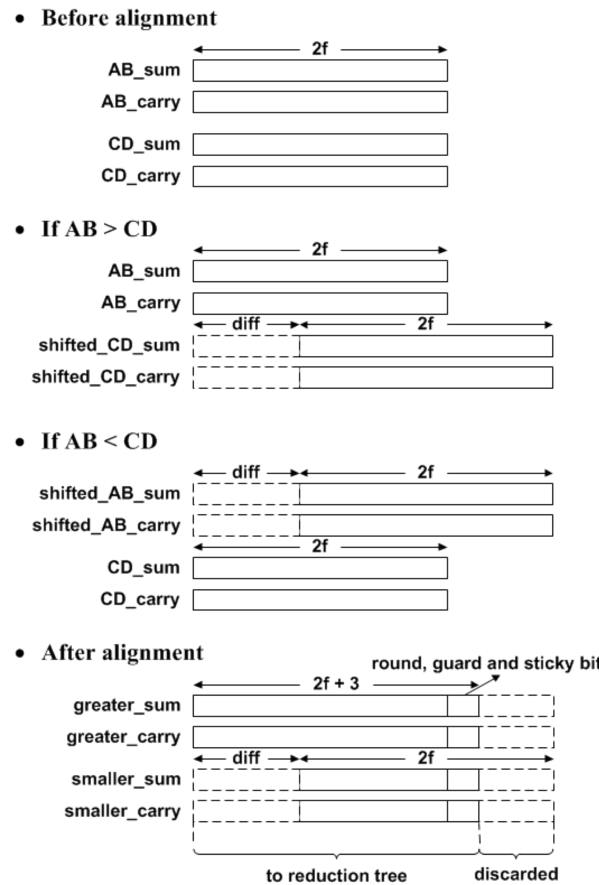
Enhanced FP Fused Dot Product Unit

▪ New Alignment Scheme



* f = number of significand bits
diff = exponent difference

Traditional Alignment

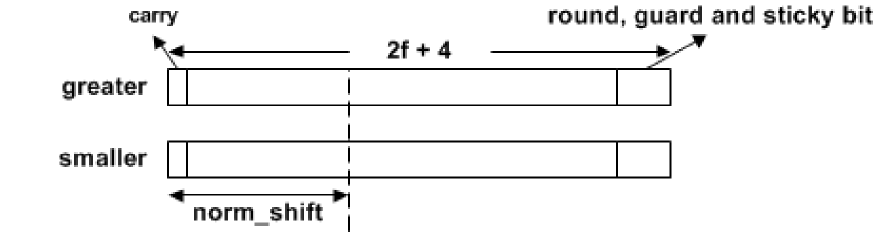


New Alignment

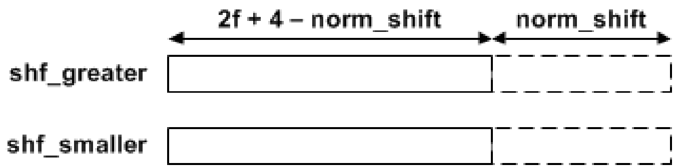
Enhanced FP Fused Dot Product Unit

▪ Early Normalization* and Fast Rounding

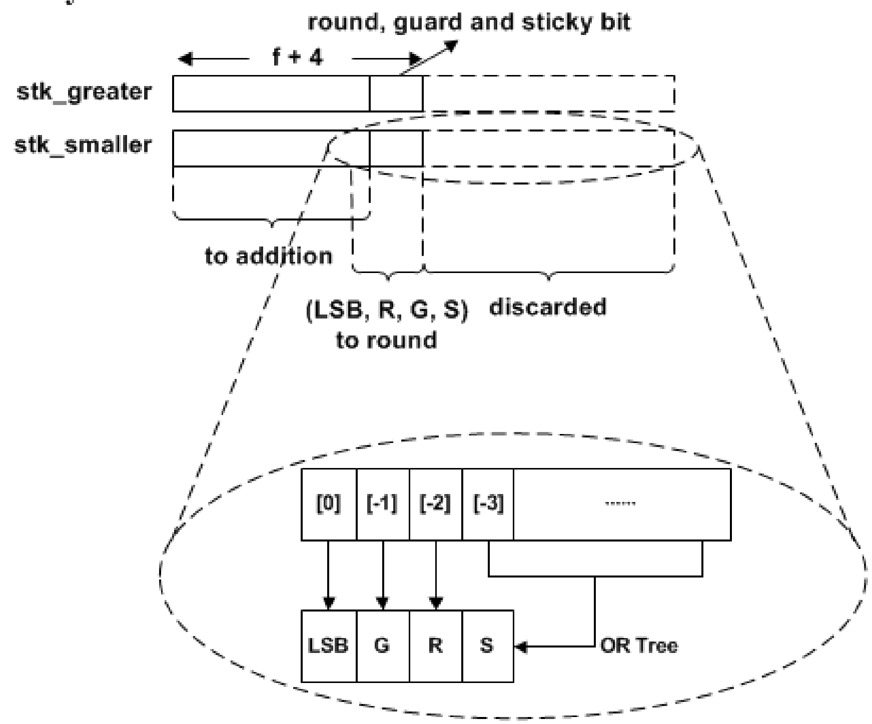
- After reduction tree



- Normalization



- Sticky

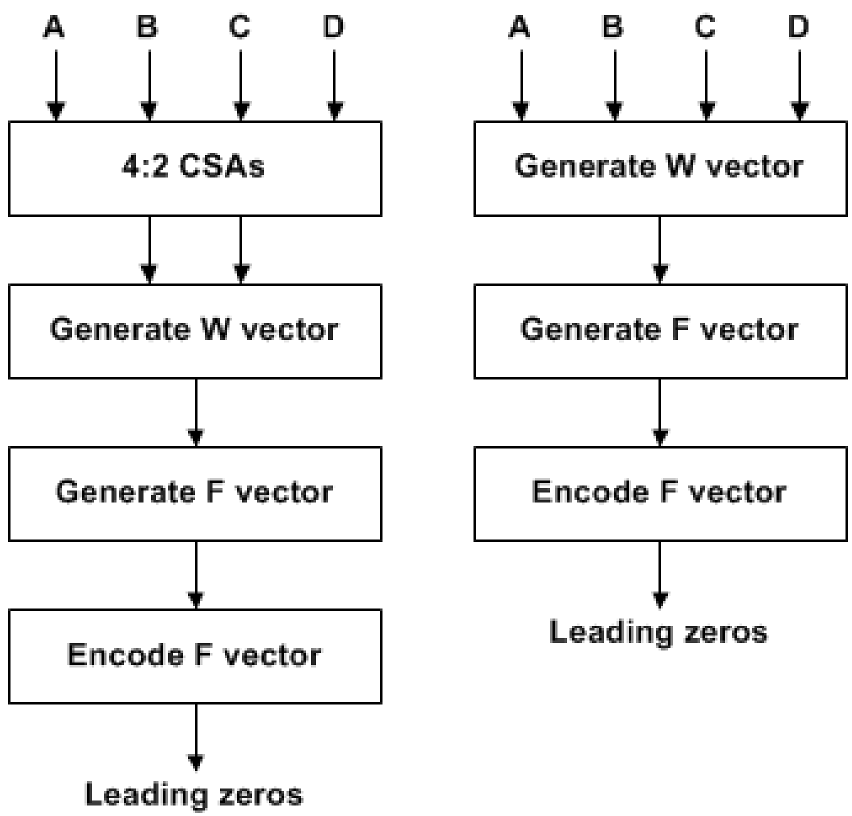


* f = number of significand bits

**** Previously proposed for the fused multiply-add unit with reduced latency [4].***

Enhanced FP Fused Dot Product Unit

- **Four-Input LZA**



Two-input LZA　　Four-input LZA

Goto Backup

Enhanced FP Fused Dot Product Unit

- **Four-Input LZA**
  - **Pre-encoding for Four-input LZA**
    - $W = A + B - C - D$

      $$w_i = a_i + b_i - c_i - d_i, \qquad w_i \in \{-2, -1, 0, 1, 2\}$$

    - $g_i = 1\ if\ w_i = 1, \quad e_i = 1\ if\ w_i = 0, \quad s_i = 1\ if\ w_i = \bar{1}$
    - $g_i = 2_i(2_{i+1} + \bar{2}_{i+1}) + 1_i(1_{i+1} + 0_{i+1} + \bar{1}_{i+1}) + 0_i 2_{i+1}$
    - $e_i = 2_i(1_{i+1} + 0_{i+1} + \bar{1}_{i+1}) + 1_i(2_{i+1} + \bar{2}_{i+1}) + 0_i(1_{i+1} + 0_{i+1} + \bar{1}_{i+1}) + \bar{1}_i(2_{i+1} + \bar{2}_{i+1}) + \bar{2}_i(1_{i+1} + 0_{i+1} + \bar{1}_{i+1})$
    - $s_i = 0_i\bar{2}_{i+1} + \bar{1}_i(1_{i+1} + 0_{i+1} + \bar{1}_{i+1}) + \bar{2}_i(2_{i+1} + \bar{2}_{i+1})$
    - $f_i(pos) = e_{i-1}g_i\bar{s}_{i+1} + \bar{e}_{i-1}s_i\bar{s}_{i+1}$ for $W > 0$
    - $f_i(neg) = e_{i-1}s_i\bar{g}_{i+1} + \bar{e}_{i-1}g_i\bar{g}_{i+1}$ for $W < 0$
    - $f_i = e_{i-1}(g_i\bar{s}_{i+1} + s_i\bar{g}_{i+1}) + \bar{e}_{i-1}(s_i\bar{s}_{i+1} + g_i\bar{g}_{i+1})$

Goto Backup

Enhanced FP Fused Dot Product Unit

- **Four-Input LZA**
  - **Leading Zeros and Pre-encoding Pattern for W > 0**

W vector	Leading Zeros	Pre-encoding Pattern
$0^k11(x)$	k	$e_{i-1}g_ig_{i+1}$
$0^k10(1\ or\ 0)$	k	$e_{i-1}g_ie_{i+1}$
$0^k10^l(\bar{1})$	$k+1$	$e_{i-1}g_ie_{i+1}$*
$0^k1\bar{1}^l1(x)$	$k+l$	$\bar{e}_{i-1}s_ig_{i+1}$
$0^k1\bar{1}^l0(1\ or\ 0)$	$k+l$	$\bar{e}_{i-1}s_ie_{i+1}$
$0^k1\bar{1}^l0^m(\bar{1})$	$k+l+1$	$\bar{e}_{i-1}s_ie_{i+1}$*

* Correction needed

  - **Concurrent correction logic is required [10] – [12]***

**** The error correction logic in [10] is modified by [11] and [12] to improve the accuracy and eliminate the redundancy, respectively.***

Dual-Path FP Fused Dot Product Unit

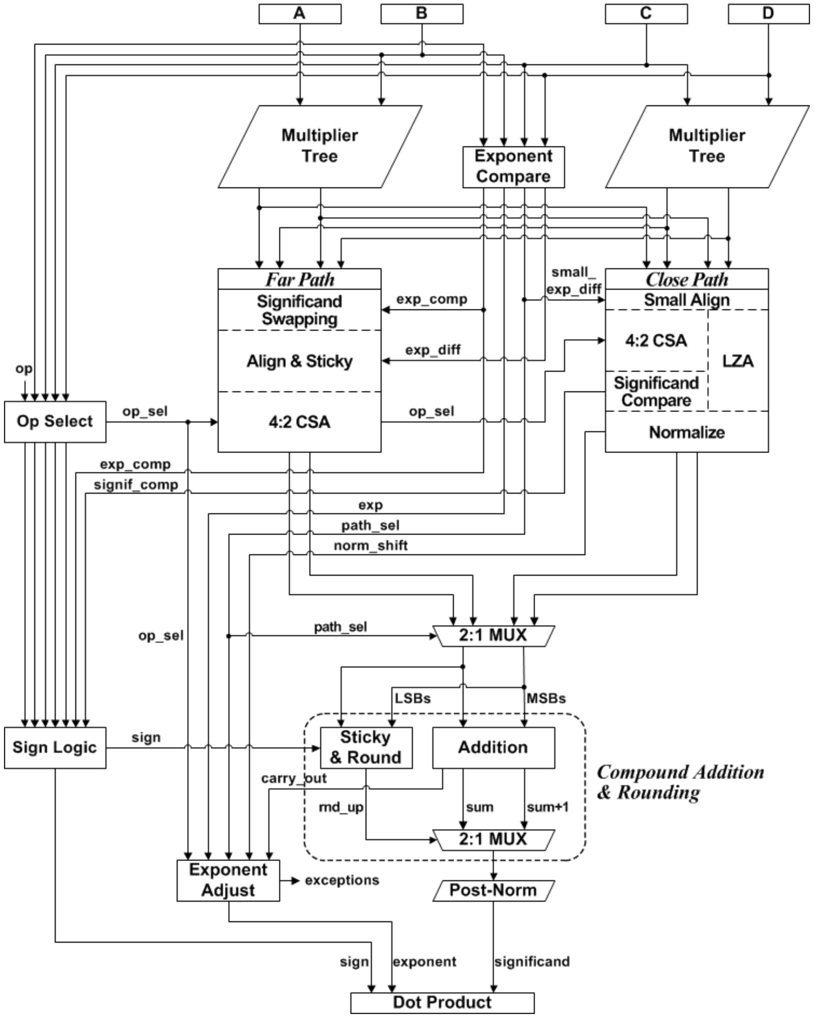
- **Dual-Path FP Fused Two-Term Dot Product Unit**
  - **Dual path algorithm**
    1) **Far path:**

       $|diff_{exp}| > 2$

    2) **Close path:**

       $-2 \leq diff_{exp} \leq 2$

  - **Far path skips normalization**
  - **Close path skips alignment**

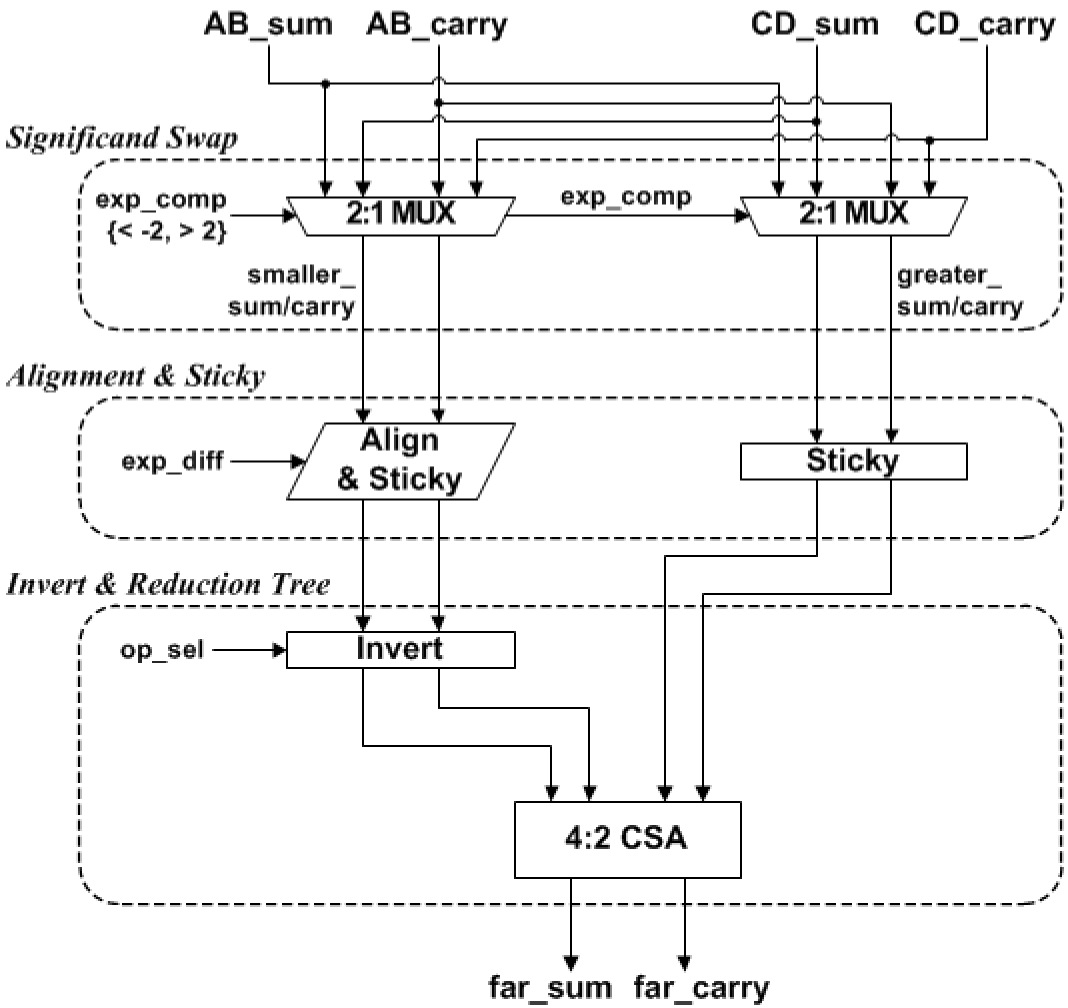


*** $diff_{exp} = A_{exp} + B_{exp} - C_{exp} - D_{exp}$**

Goto Backup

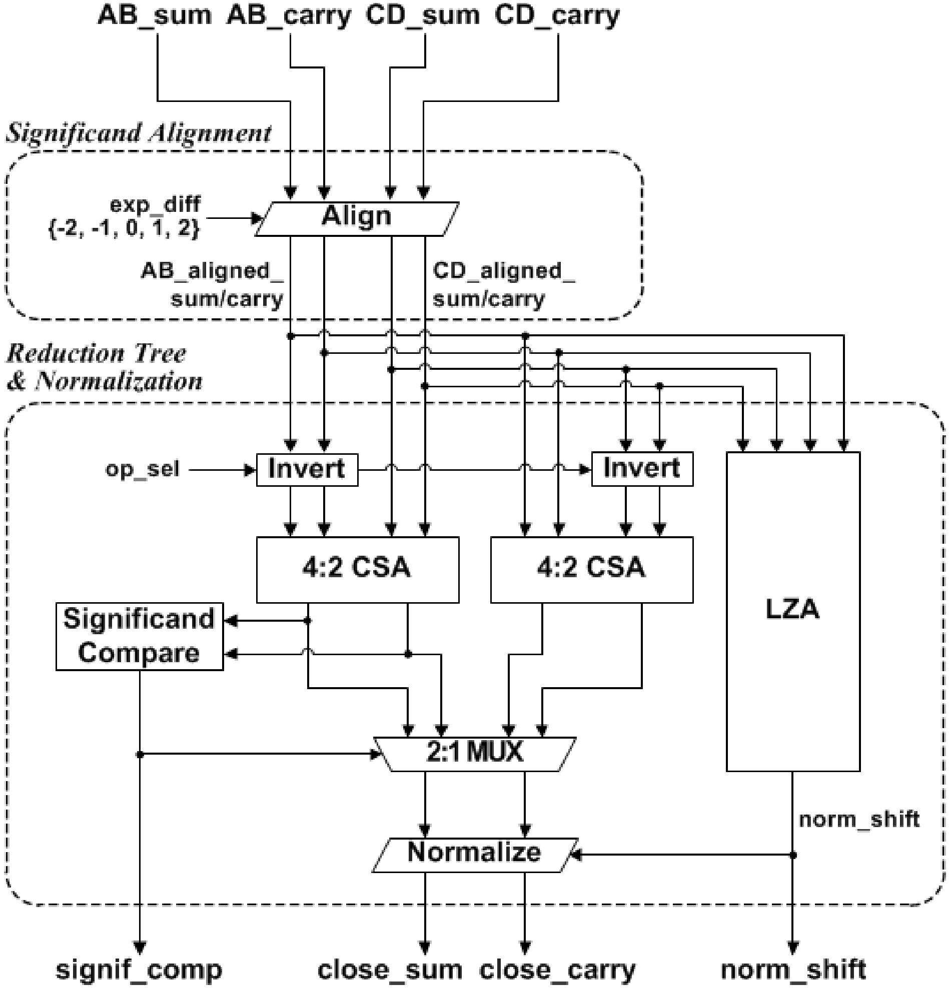
Dual-Path FP Fused Dot Product Unit

- **Far Path Logic**
  - Significand swap
  - Alignment & sticky
  - Reduction Tree



Dual-Path FP Fused Dot Product Unit

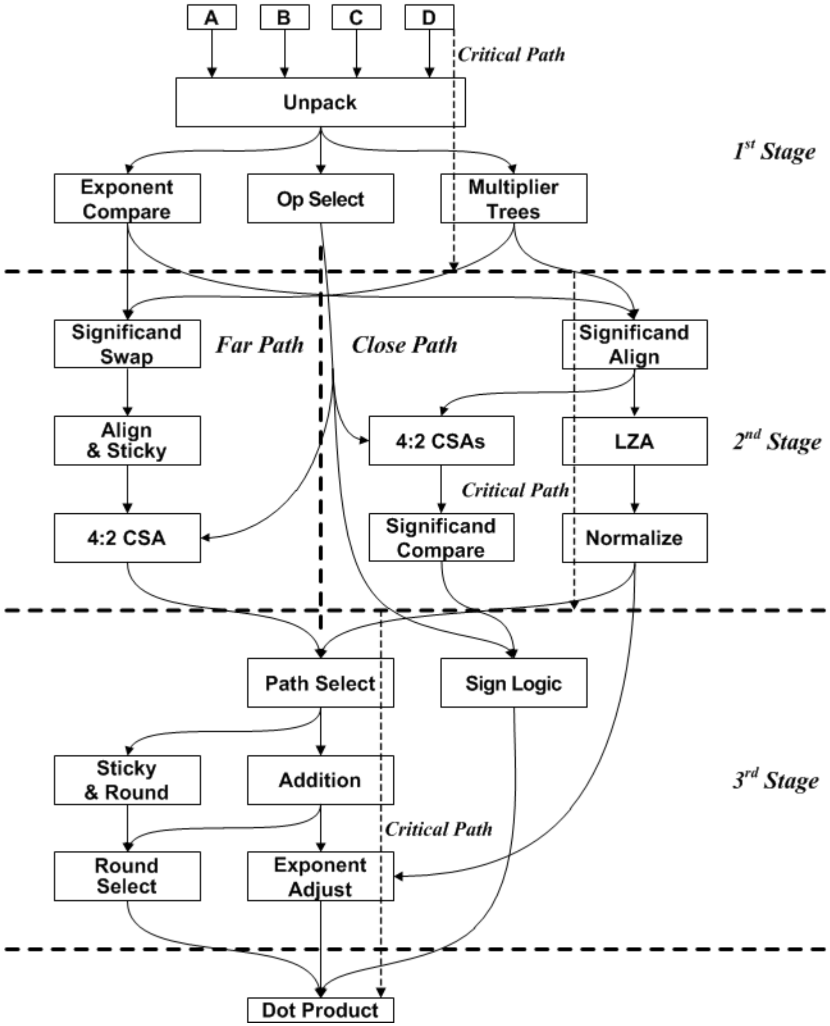
- **Close Path Logic**
  - **Small alignment (≤ 2)**
  - **Reduction Trees**
  - **LZA & Normalization**



Go Back

Pipelined FP Fused Dot Product Unit

- **Pipelined FP Fused Two-Term Dot Product Unit**
  - **First stage *(Critical path)*:**
    - Unpack
    - Multiplier tree
  - **Second stage *(Critical path)*:**
    - Close path significand align
    - LZA
    - Normalization
  - **Third stage *(Critical path)*:**
    - Path Selection
    - Addition
    - Exponent Adjust
  - **Balanced latency for single precision: 0.65ns/stage (= 1.5GHz)**



Results

- **Design Comparison**
  - Single Precision
  - 45nm CMOS Standard Cell Library

	Traditional	Enhanced	Enhanced + Dual Path	Enhanced + Dual-Path + Pipeline
Area (μm^2)	38,654 (100%)	29,159 (75%)	31,472 (81%)	33,228 (86%)
Latency (ns)	2.54 (100%)	2.14 (84%)	1.87 (74%)	2.01 (79%)
Throughput (1/ns)	0.35 (100%)	0.47 (119%)	0.53 (136%)	1.49 (379%)
Power (mW)	20.77 (100%)	15.17 (73%)	16.16 (78%)	16.94 (82%)

Results

- **Pipeline Stages**
  - **Single Precision**
  - **45nm CMOS Standard Cell Library**

	Stage 1	**Stage 2**	**Stage 3**
Area (μm^2)	17,484 (53%)	12,143 (36%)	3,601 (11%)
Latency (ns)	0.65 (33%)	0.67 (35%)	0.63 (32%)
Power (mW)	8.96 (53%)	6.41 (38%)	1.57 (9%)

Conclusion

- **Summary**
  - **Three optimizations for an enhanced FP fused dot product unit**
    - New alignment scheme
    - Early normalization and fast rounding
    - Four-input LZA

    $\Rightarrow$ Reduces the latency by 15%, reduces Area and power by 25%
  - **Dual-path FP fused dot product unit**

    $\Rightarrow$ Reduces the latency by 25%
  - **Pipelined FP fused dot product unit**

    $\Rightarrow$ Increases the throughput by 2.8 times

Conclusion

- **Trade-off**

Category	Optimizations			
	New Alignment	Four-Input LZA	Dual-Path	Pipelining
Area	+	+	–	–
Latency	+	+	++	–
Throughput	+	+	++	+++
Power	+	+	–	–

References

[1] IEEE Standard for Floating-Point Arithmetic, ANSI/IEEE Standard 754-2008, New York: IEEE, Inc., 2008.

[2] R. K. Montoye, E. Hokenek, and S. L. Runyon, "Design of the IBM RISC System/6000 Floating-Point Execution Unit," IBM Journal of Research & Development, Vol. 34. pp. 59 – 70. 1990.

[3] E. Hokenek, R. K. Montoye and P. W. Cook, "Second-Generation RISC Floating Point with Multiply-Add Fused," IEEE Journal of Solid-State Circuits, vol. 25, pp. 1207 – 1213, 1990.

[4] T. Lang and J. D. Bruguera, "Floating-Point Fused Multiply-Add with Reduced Latency," IEEE Transactions on Computers, Vol. 53, pp. 988 – 1003, 2004.

[5] H. H. Saleh and E. E. Swartzlander, Jr., "A Floating-Point Fused Add-Subtract Unit," Proceedings of the 51st IEEE Midwest Symposium on Circuits and Systems, pp. 519 – 522, 2008.

[6] J. Sohn and E. E. Swartzlander, Jr., "Improved Architectures for a Fused Floating-Point Add-Subtract Unit," IEEE Transactions on Circuits and Systems–I, Vol. 59, pp. 2285 – 2291, 2012.

[7] H. H. Saleh and E. E. Swartzlander, Jr., "A Floating-Point Fused Dot- Product Unit," Proceedings of the IEEE International Conference on Computer Design, pp. 427 – 431, 2008.

[8] E. E. Swartzlander, Jr. and H. H. Saleh, "FFT Implementation with Fused Floating-Point Operations," IEEE Transactions on Computers, Vol. 61, pp. 284 – 288, 2010.

References

[9] E. E. Swartzlander, Jr. and H. H. Saleh, "Fused Floating-Point Arithmetic for DSP," Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers, pp. 767 – 771, 2008.

[10] J. D. Bruguera and T. Lang, "Leading-One Prediction with Concurrent Position Correction," IEEE Transactions on Computers, vol. 48, pp. 1083 – 1097, 1999.

[11] R. Ji, Z. Ling, X. Zeng, B. Sui, L. Chen, J. Zhang, Y. Feng, and G. Luo, Comments on "Leading One Prediction with Concurrent Position Correction," IEEE Transactions on Computers, vol. 58, pp. 1726 – 1727, 2009.

[12] P. Kornerup, "Correcting the Normalization Shift of Redundant Binary Representations", IEEE Transactions on Computers, vol. 58, pp. 1435 – 1439, 2009.

Thank you

Backup

- **IEEE-754 Standard for Floating-Point [1]**
  - $\mathbf{fp\_number} = (-1)^{\boldsymbol{sign}} \times \mathbf{2}^{\boldsymbol{exponent}} \times \boldsymbol{significand}$
    - *sign* = 0 or 1
    - *exponent* = *e* – e_{bias} + 1 (*e* = any integer between 0 and 2# of exponent bits)
    - *significand* = $d_{p-1}d_{p-2} \ldots d_2d_1d_0$ (d_i = 0 or 1, *p* = significand precision)

Format	Single Precision	Double Precision
Sign	1	1
Exponent	8	11
Significand	23	52
Total	32	64
Exponent Bias	127	1023
Exponent Range	$2^{-126} - 2^{127}$	$2^{-1022} - 2^{1023}$
Significand Precision	24	53

1	8	23
s	*exponent*	*significand*

Single Precision

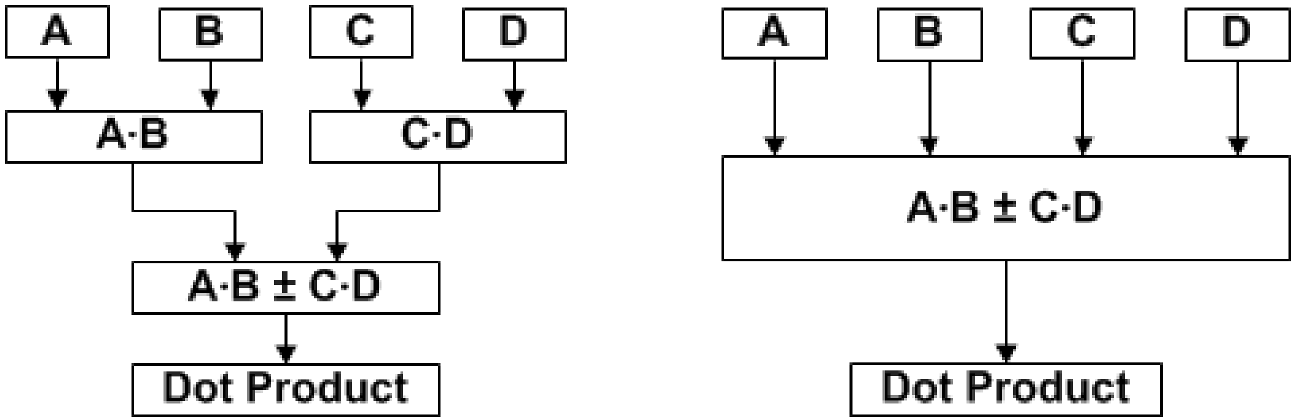
1	11	52
s	*exponent*	*significand*

Double Precision

Go Back

Backup

- **Discrete vs. Fused Two-Term Dot Product**



Go Back

Backup

- **Massive Cancellation**
  - **After the subtraction, MSBs (if it is 0) must be shifted for normalization**

$$
\begin{array}{r}
1.1000000111 \\
-\ 1.0111111000 \\
\hline
0.0000001111 \\
1.1110000000
\end{array}
\quad \ll 7
$$

Go Back

Backup

- **Two-Input LZA for Floating-Point Adder [10]**
  - **Pre-encoding for Two-input LZA**
    - $W = A - B$

$$w_i = a_i - b_i, \qquad w_i \in \{-1, 0, 1\},$$

    - $g_i = 1 \; if \; w_i = 1, \quad e_i = 1 \; if \; w_i = 0, \quad s_i = 1 \; if \; w_i = \bar{1}$
    - $f_i = e_{i-1}(g_i \bar{s}_{i+1} + s_i \bar{g}_{i+1}) + \bar{e}_{i-1}(s_i \bar{s}_{i+1} + g_i \bar{g}_{i+1})$
  - **Leading Zeros and Encoding Pattern for W > 0**

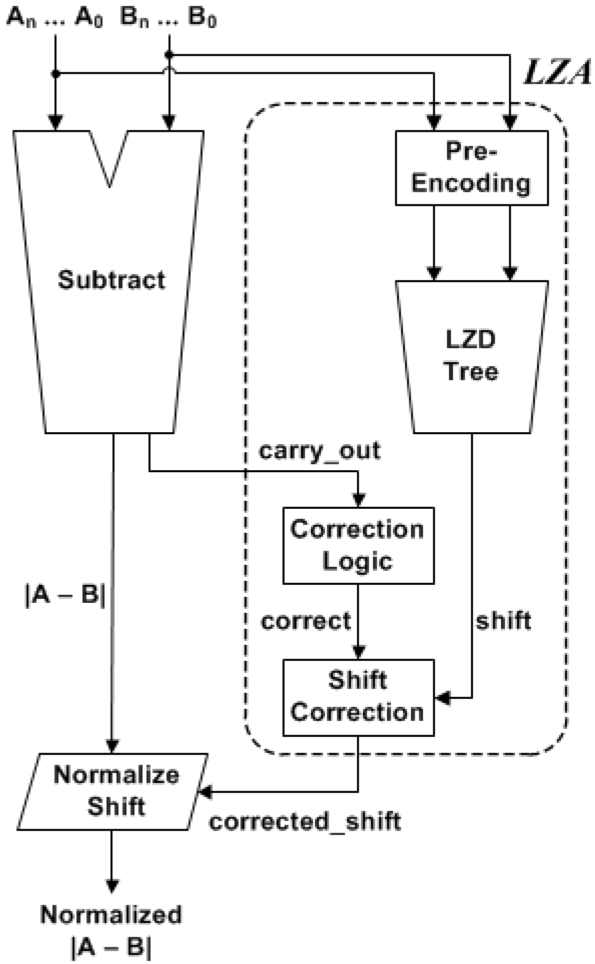
W vector	Leading Zeros	Pre-encoding Pattern
$0^k11(x)$	k	$e_{i-1}g_ig_{i+1}$
$0^k10(1 \; or \; 0)$	k	$e_{i-1}g_ie_{i+1}$
$0^k10^l(\bar{1})$	$k+1$	$e_{i-1}g_ie_{i+1}$*
$0^k1\bar{1}^l1(x)$	$k+l$	$\bar{e}_{i-1}s_ig_{i+1}$
$0^k1\bar{1}^l0(1 \; or \; 0)$	$k+l$	$\bar{e}_{i-1}s_ie_{i+1}$
$0^k1\bar{1}^l0^m(\bar{1})$	$k+l+1$	$\bar{e}_{i-1}s_ie_{i+1}$*

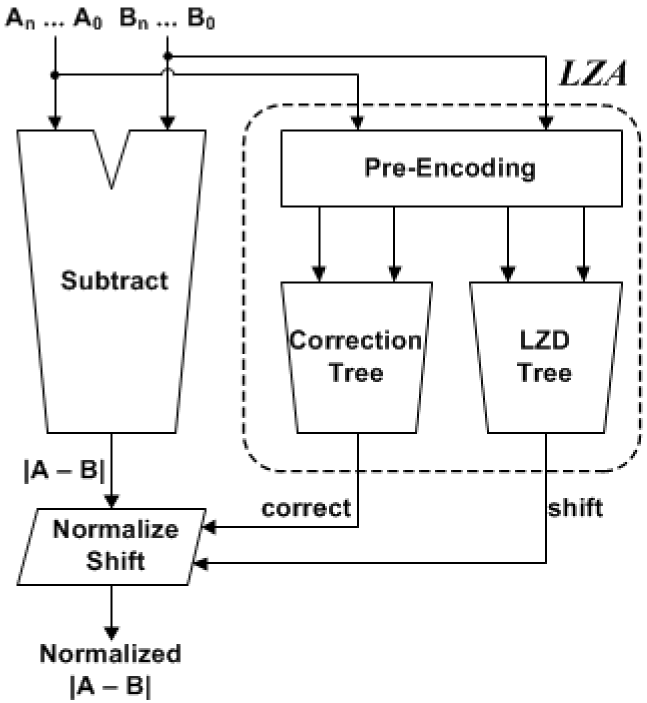
* Correction needed

Go Back

Backup

- **LZA with concurrent correction [10]**

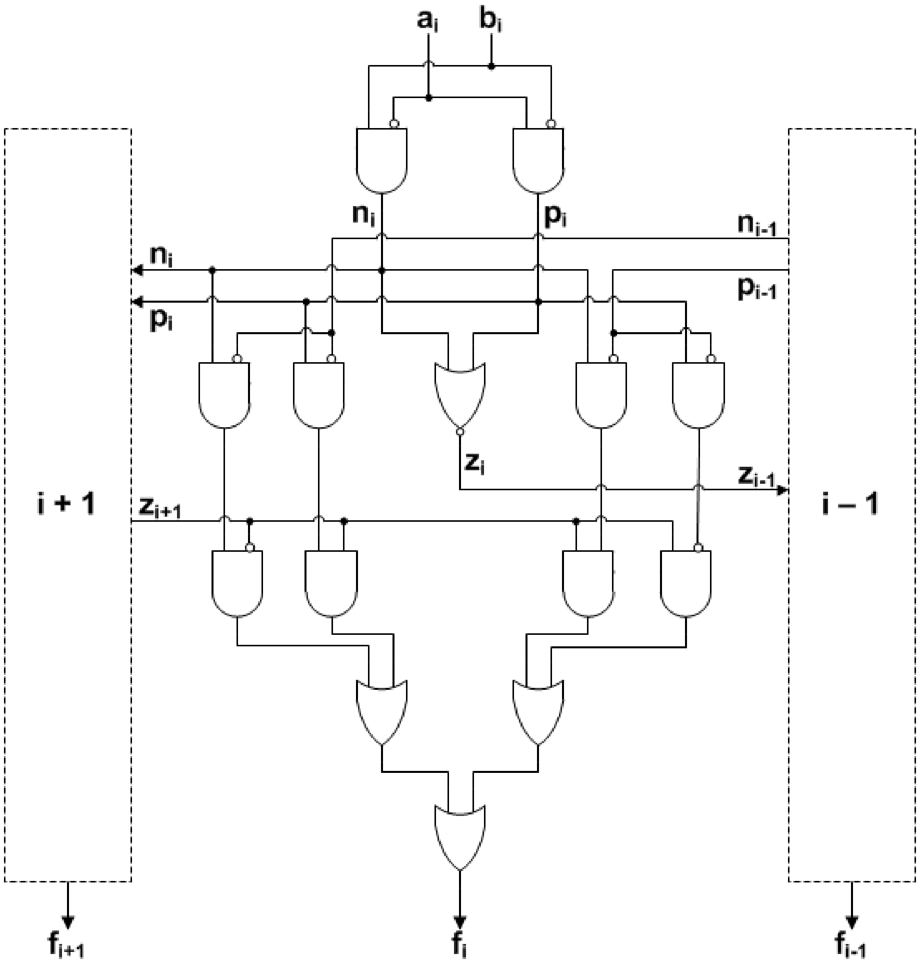




Go Back

Backup

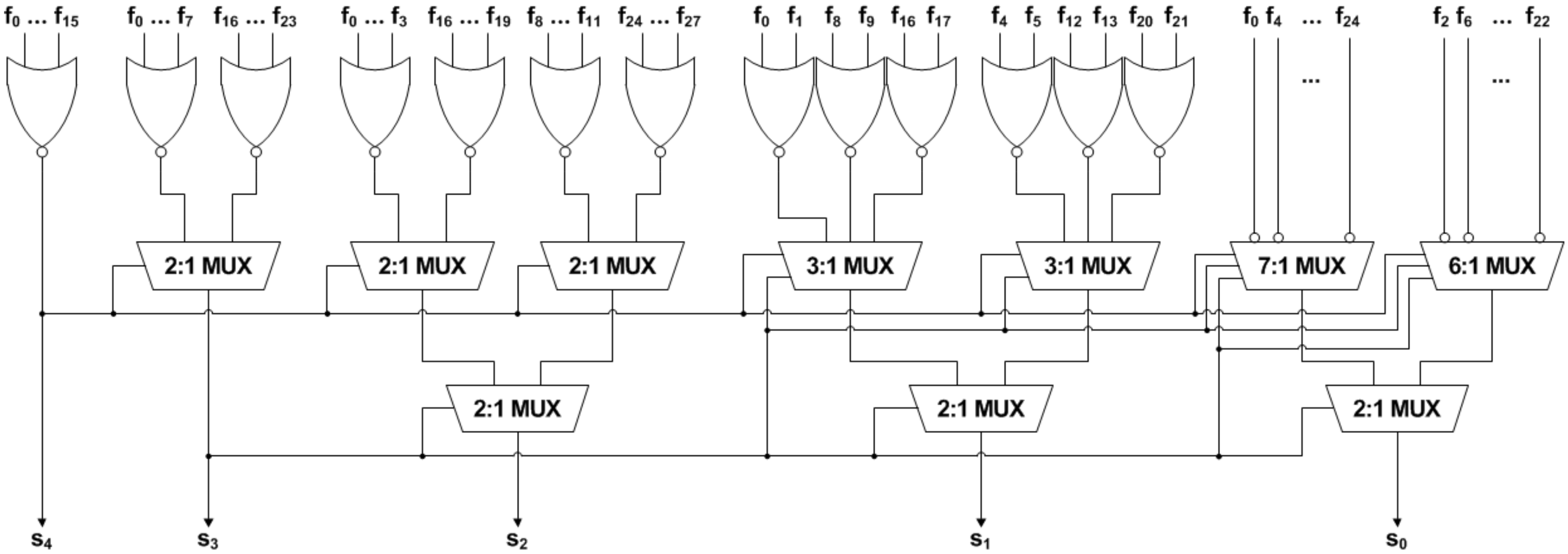
- **Pre-Encoding Logic of LZA [10]**



Go Back

Backup

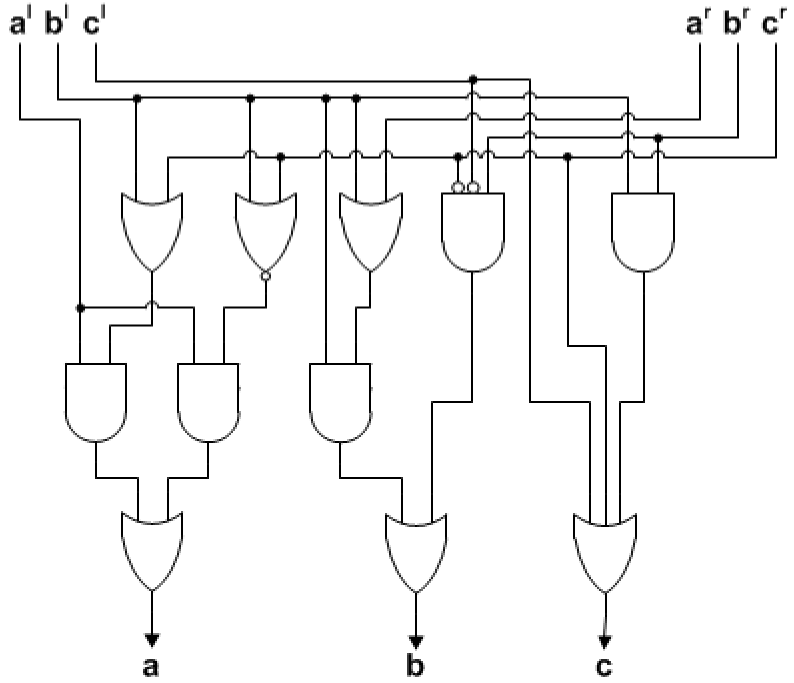
- **25 bit LZD tree [4]**



Go Back

Backup

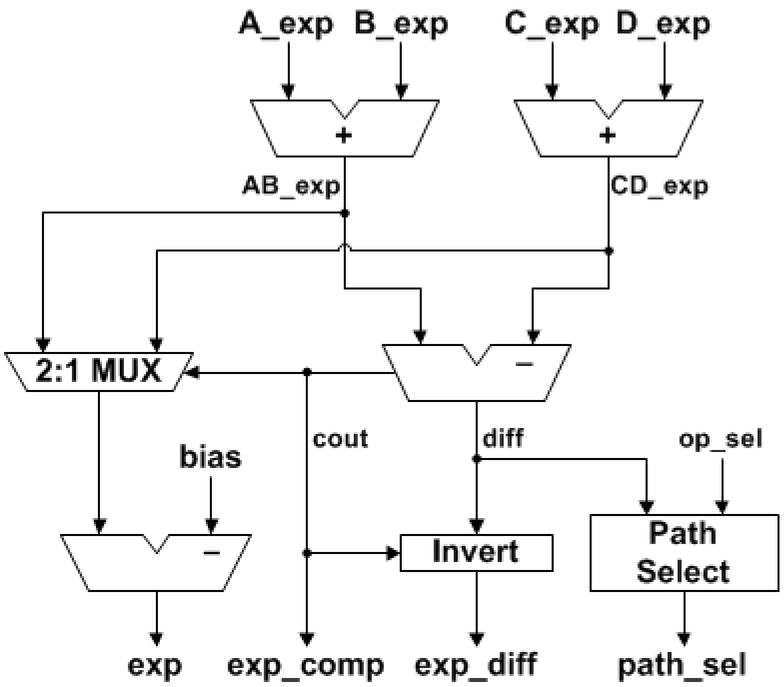
- **Concurrent Correction Tree for LZA [12]**



Go Back

Backup

- **Exponent Compare Logic**



Go Back

Backup

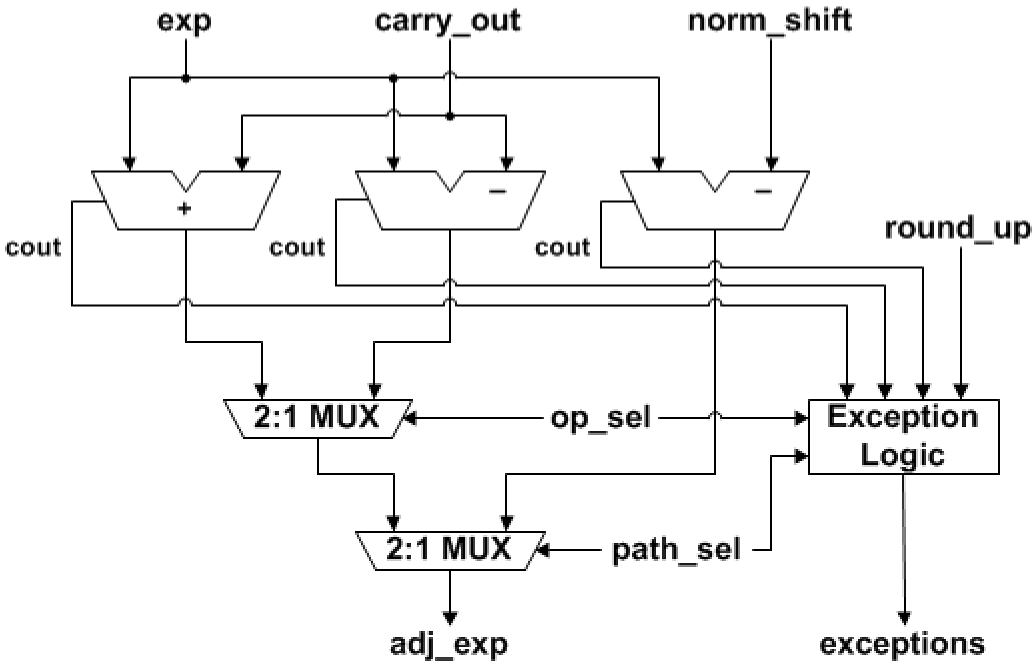
- **Operation Select**

  - $op\_sel = \begin{cases} AB_{sign} \oplus CD_{sign} & if\ op = add \\ \overline{AB_{sign} \oplus CD_{sign}} & if\ op = sub \end{cases}$

Go Back

Backup

- **Exponent Adjust Logic**



Go Back

Backup

- **Path Selection**

  - $path\_sel = \begin{cases} 1 & if\ |AB_{exp} - CD_{exp}| \leq 2\ \ or\ \ op\_sel = 0 \\ 0 & otherwise \end{cases}$

Go Back

Backup

- **Exceptions**

  - $overflow = \begin{cases} 1 & if\ exp \geq max\_exp \\ 0 & otherwise \end{cases}$
  - $underflow = \begin{cases} 1 & if\ exp \leq 0 \\ 0 & otherwise \end{cases}$
  - $inexact = overflow\ ||\ underflow\ ||\ round\_up$

Go Back

Backup

- **Close Path Significand Alignment**

  - $$AB_{aligned} = \begin{cases} (AB_{signif}, 00) & if\ AB_{exp} - CD_{exp} = 0, 1, 2 \\ (0, AB_{signif}, 0) & if\ AB_{exp} - CD_{exp} = -1 \\ (00, AB_{signif}) & if\ AB_{exp} - CD_{exp} = -2 \end{cases}$$

  - $$CD_{aligned} = \begin{cases} (CD_{signif}, 00) & if\ AB_{exp} - CD_{exp} = 2 \\ (0, CD_{signif}, 0) & if\ AB_{exp} - CD_{exp} = 1 \\ (00, CD_{signif}) & if\ AB_{exp} - CD_{exp} = 0, -1, -2 \end{cases}$$

Go Back