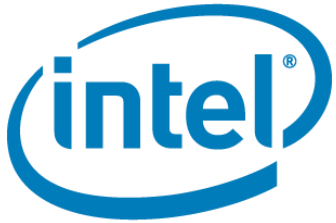


Split-path Fused Floating Point Multiply Accumulate (FPMAC)

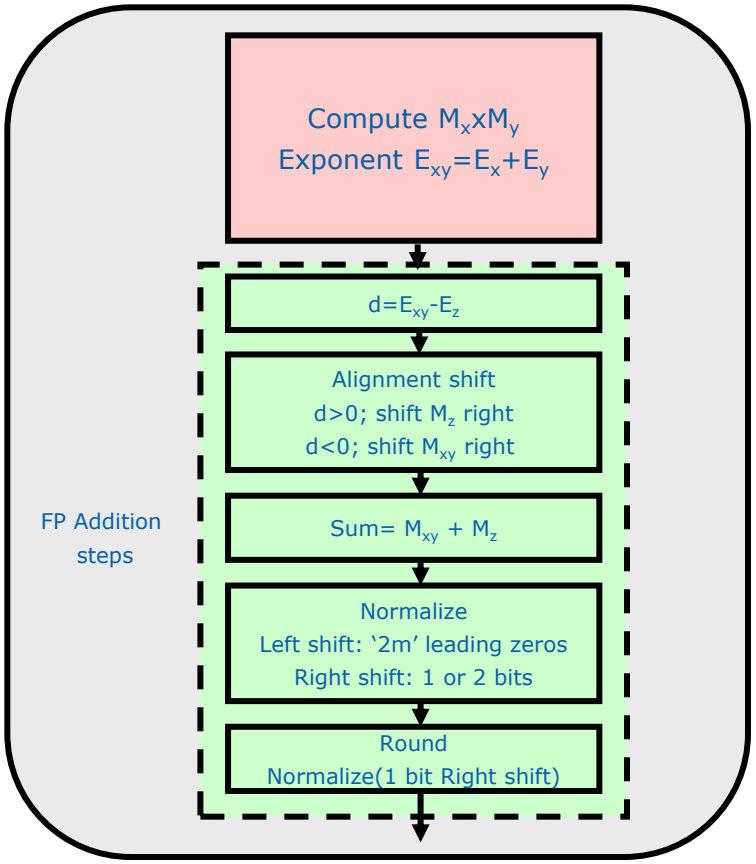
Suresh Srinivasan, Ketan Bhudiya, Rajaraman Ramanarayanan, P Sahit Babu, Tiju Jacob, Sanu. K. Mathew, Ram Krishnamurthy, Vasantha Erraguntla

Outline

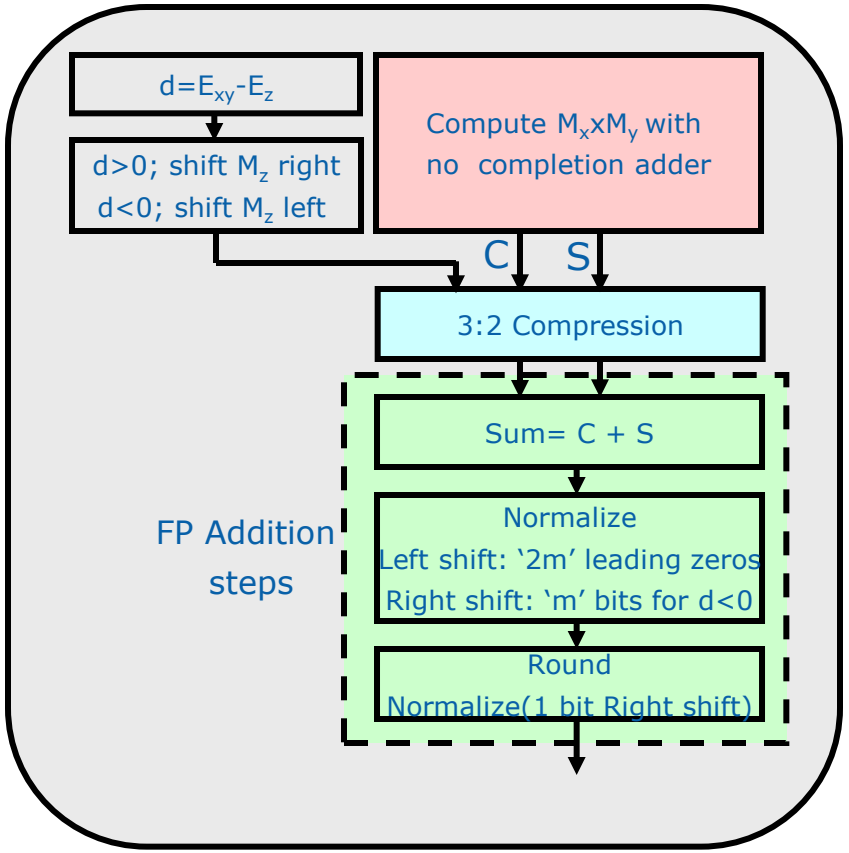
- Introduction to FPMAC algorithms
- Existing implementations
- Proposed Split-path FPMAC design
- Comparison of Logic stages and Area
- Summary

FPMAC Algorithms

Basic FPMAC Algorithm



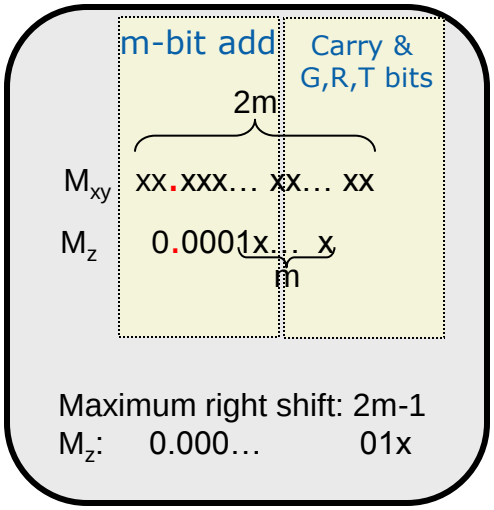
Faster Implementation



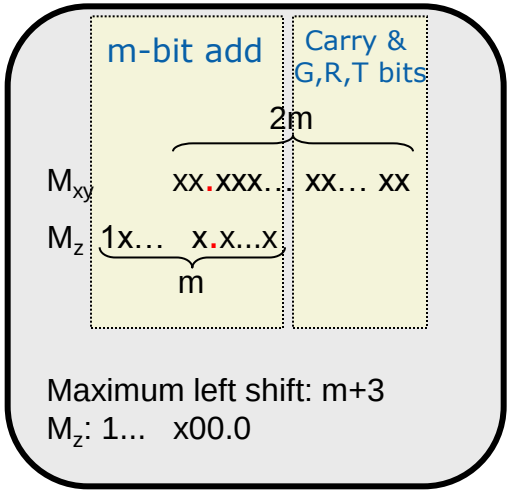
- Operation: M_xxM_y + M_z (each 'm' bit number)
- Faster implemenation
  - Remove alignment shifter from CP by aligning the accumulate (M_z)
  - Remove (M_xxM_y) completion adder

Impact of Exponent Difference (d)

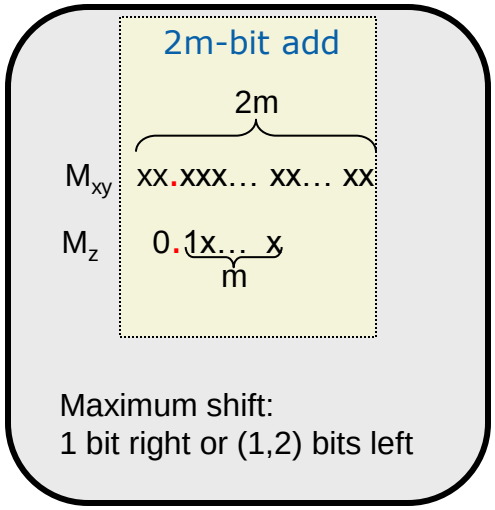
d>1



d<-2



d={0,-1,-2,1}



No possibility of Leading Zeroes

M_{xy}: xx.xxx….xx…xxx
M_z: 0.01.. x

No possibility of Leading Zeroes

M_{xy}: xx.xxx….xx…xxx
M_z: 1xxx.xx.. x

Leading Zeroes Case

M_{xy}: xx.xxx….xx…xxx
M_z: 1xx.x..x (d=-2)
Mz: 1x.x.. x (d=-1)
Mz: 1.x.. x (d=0)
Mz: 0.1x.. x (d=1)

- Operations differ significantly for different ‘d’
- Near path (d={0,-1,-2,1})
  - Possibility of leading zeros during subtraction
  - Requires all the ‘2m’ bit sum
- Far path (d>1;d<-2)
  - Only requires m-bit sum
  - Requires big alignment shifters

Three Cases: Case 1 [d>1]

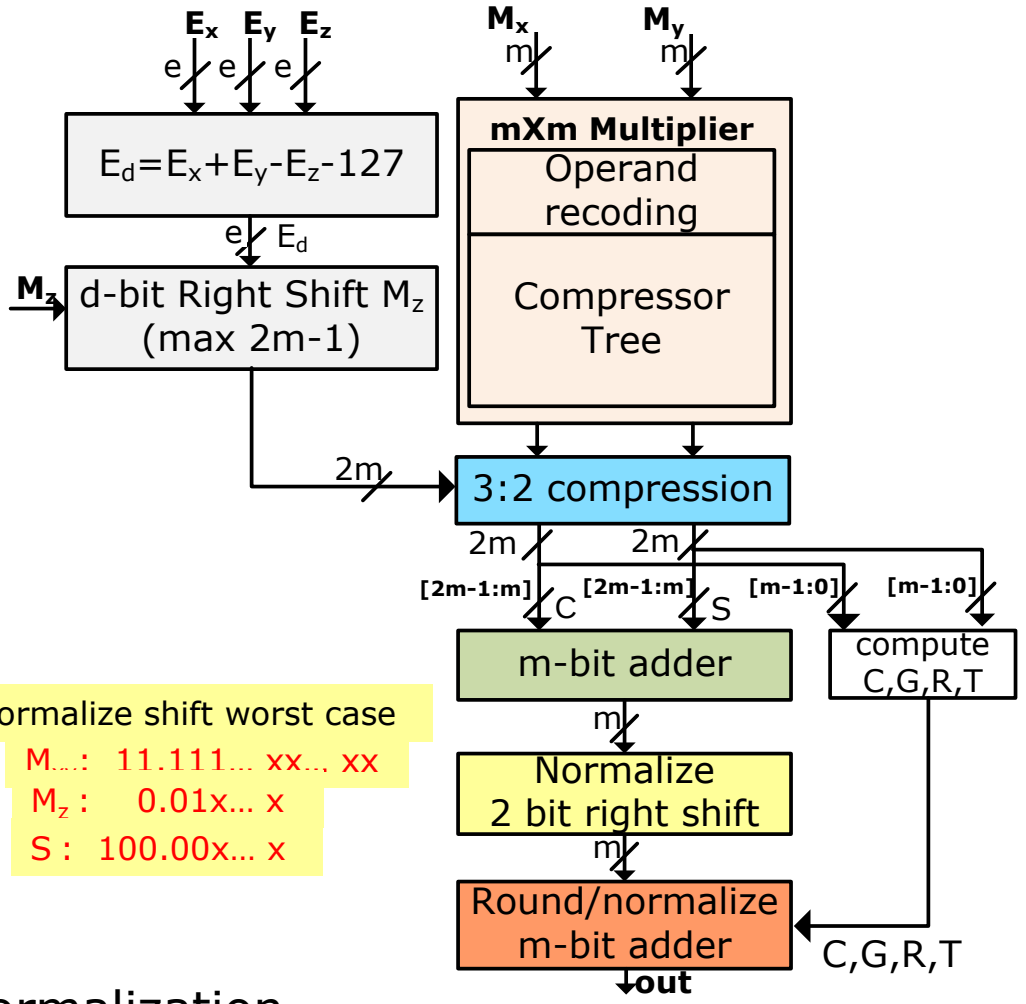
Right shift M_Z to Align Mantissa:

M_{xy}: xx.xxx… xx… xx (2m)

M_z : 0.01x… x (d=2) (m)

Maximum shift case

M_z: 0.00… 01x (2m-1)



Normalize shift worst case

M_{xy}: 11.111… xx… xx

M_z : 0.01x… x

S : 100.00x… x

- Far Path with E_{xy} >> E_z
  - –Big shifts NOT required for normalization
  - –Simplest case, fastest and smallest

Three case: Case 2 [d<-2]

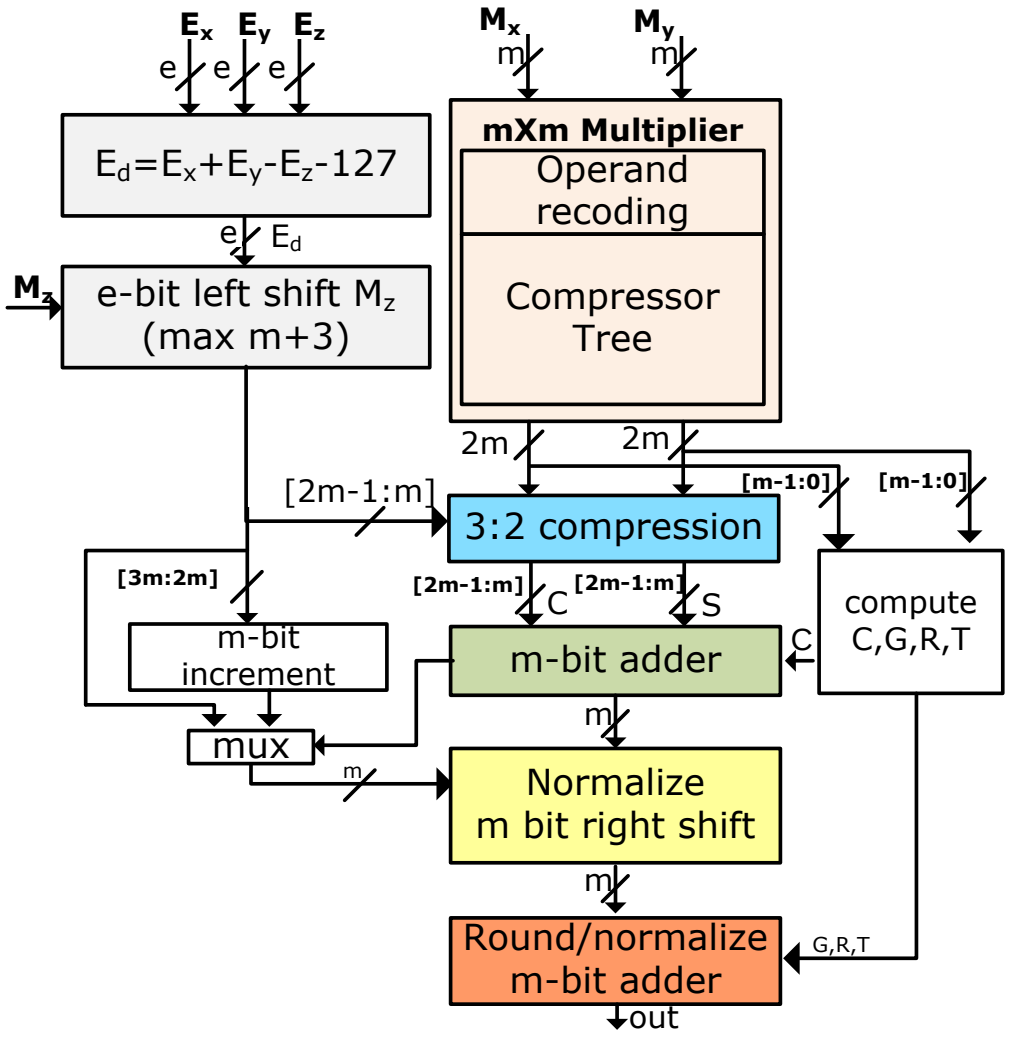
Left shift M_z to Align Mantissa:

M_{xy}: xx.xx… xx… xx (2m)

M_z: 1xxx.xx...xx (d=-3) (m)

Maximum shift case

M_z: 1..x000.0 (d=-(m+3)) (m+2)



- Far Path with $E_z >> E_{xy}$
  - –Additional m-bit increment
  - –M-bit normalize shift in critical path

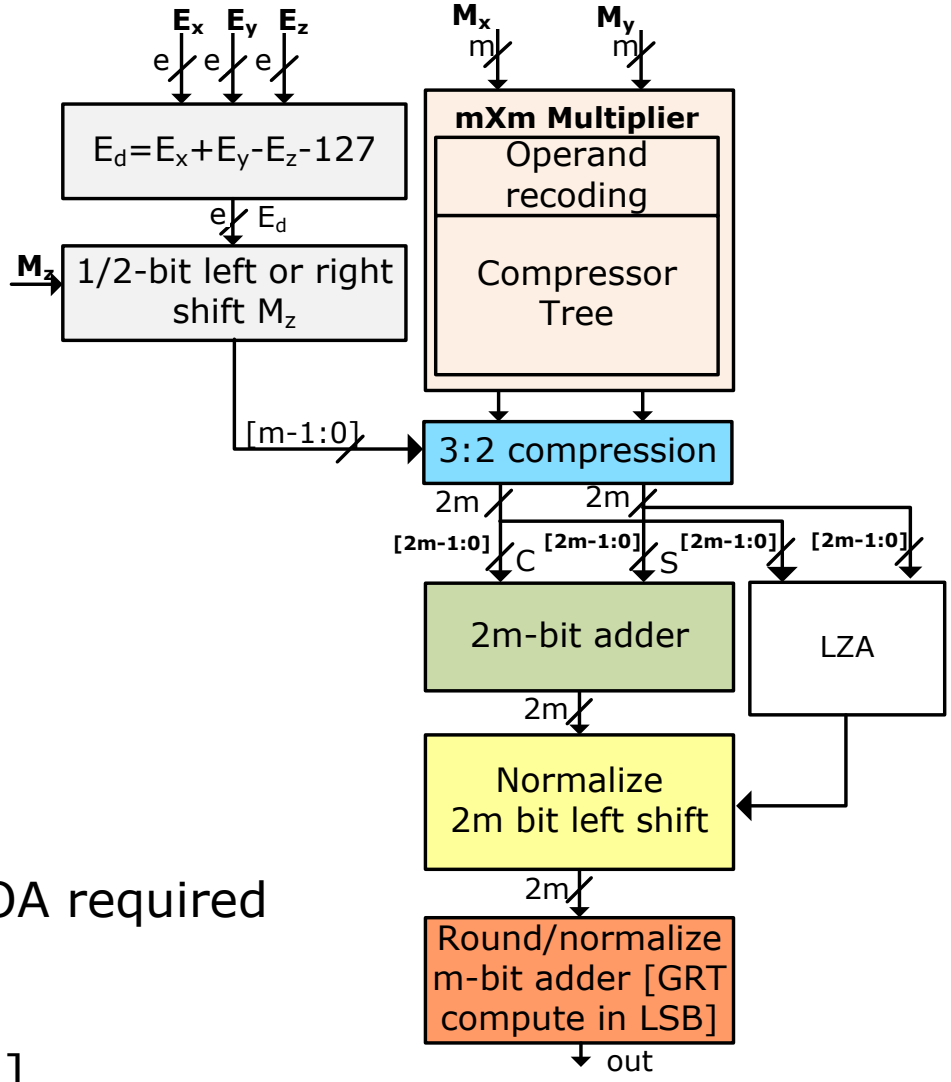
Three Cases: Case 3 [d=-2,-1,0,1]

Shift 1/2 left/right M_Z to Align Mantissa:

M_{xy}: xx.xxx… xx… xx (2m)

M_z: 1.xx… x (m)

M_z: 1xx.xx…x0



- Comprises the critical path
- Needs 2m-bit sum
- 2m bit normalize shift
- Result's sign unknown =>LZA&LOA required
- Post-norm GRT bits computation
- Dominates area [2m bits handled]

Impact of Exponent Difference (d)

Exponent diff. $d = E_{xy}-E_z$	Alignment shift	Accumulate adder	Normalization shifter	Rounding unit
d>1	Worst case 2m-1 right shift M_z	2m-bit 3:2 compress m-bit add	1-bit right shift	m bit add
d<-2	Worst case m+3 left shift M_z	m-bit 3:2 compress m-bit increment m-bit add	m+2 bit right shifter	m-bit add
d={0, 1, -1,-2}	1 bit right or (1,2) bit left shift M_z	2m bit 3:2 compress 2m-bit add	2m-1 bit left shifter Shift amount= Leading Zero Counter	m-bit add

- Exponent difference significantly affects various operations

IBM P6 FPMAC (Unified Datapath)

Handling all cases together

xx.xx ….xx

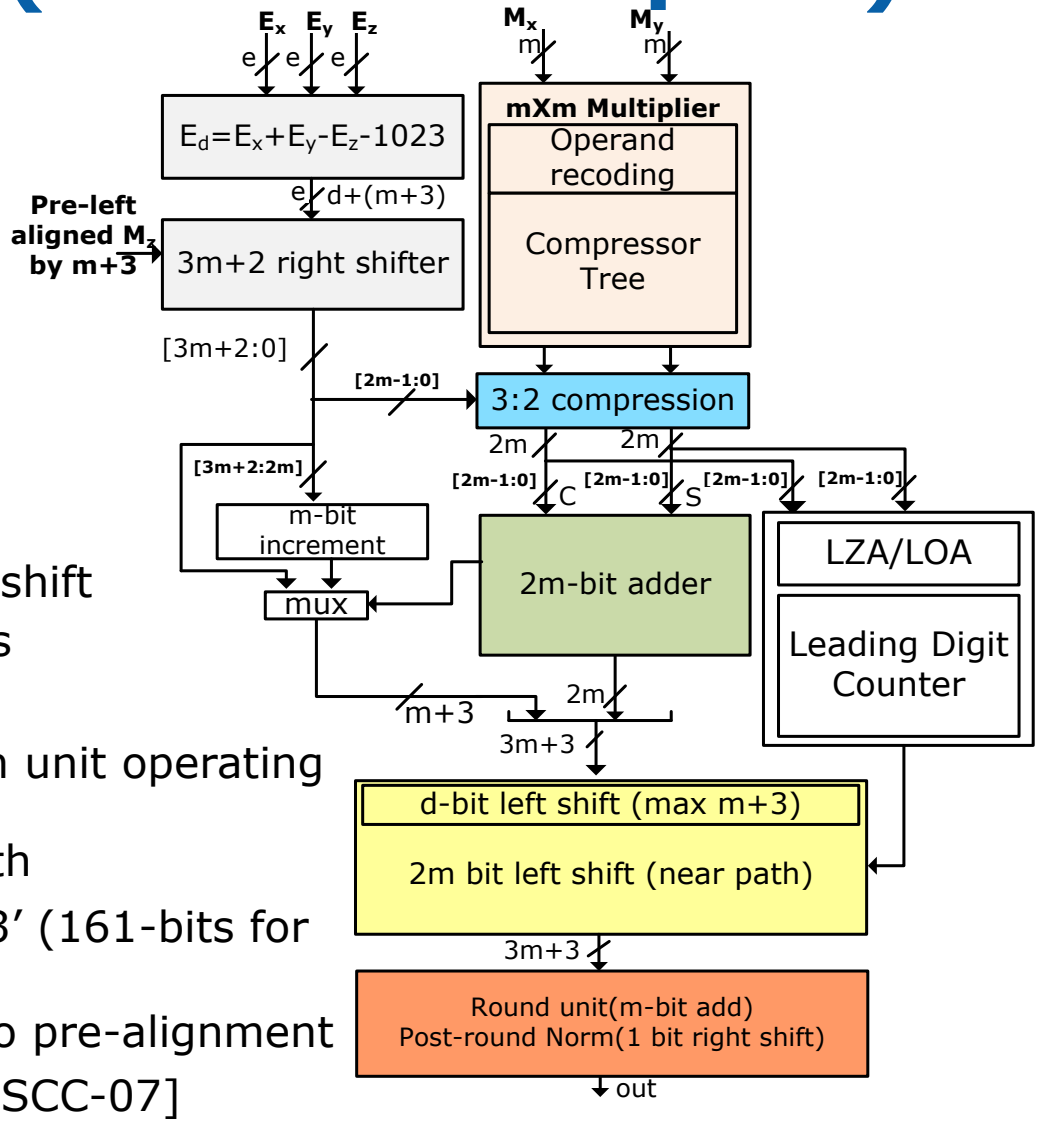
1.xx ….xx0

m

$d = E_{xy} - E_z$

Shift Amt: d+56

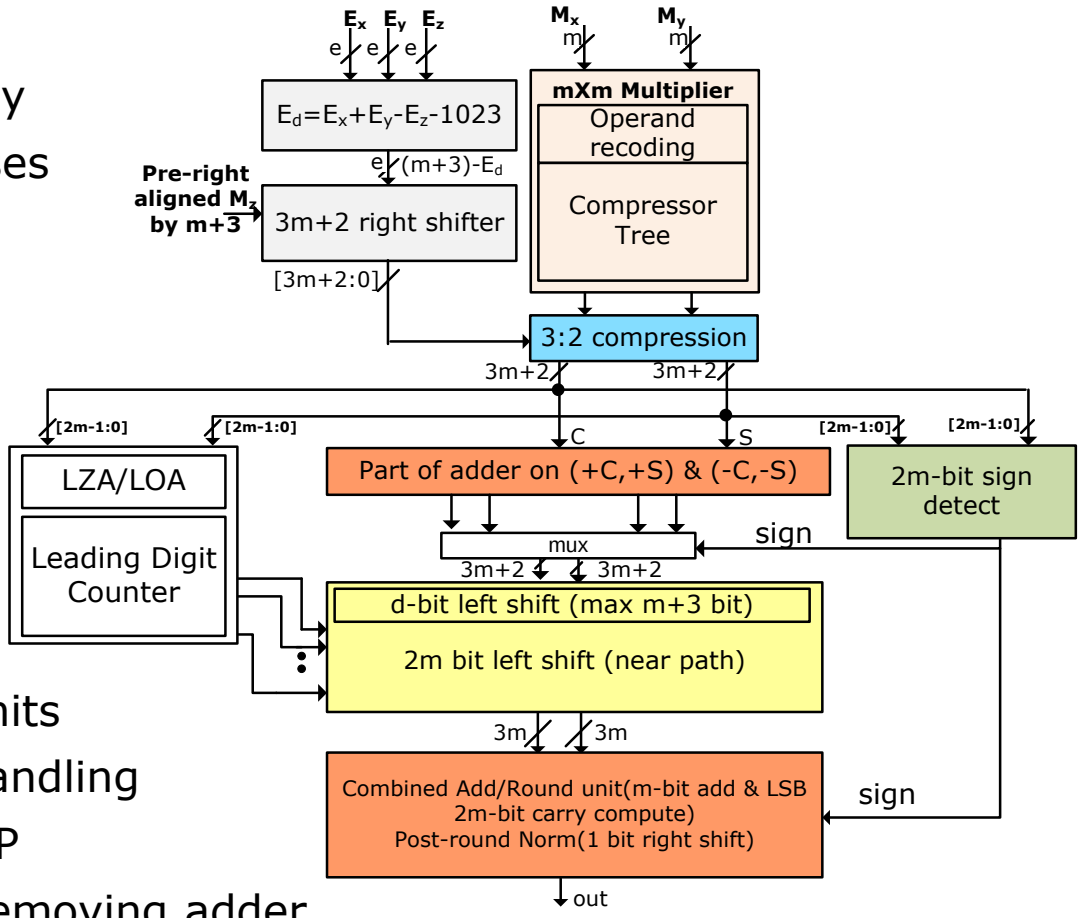
Output Exponent: E_z



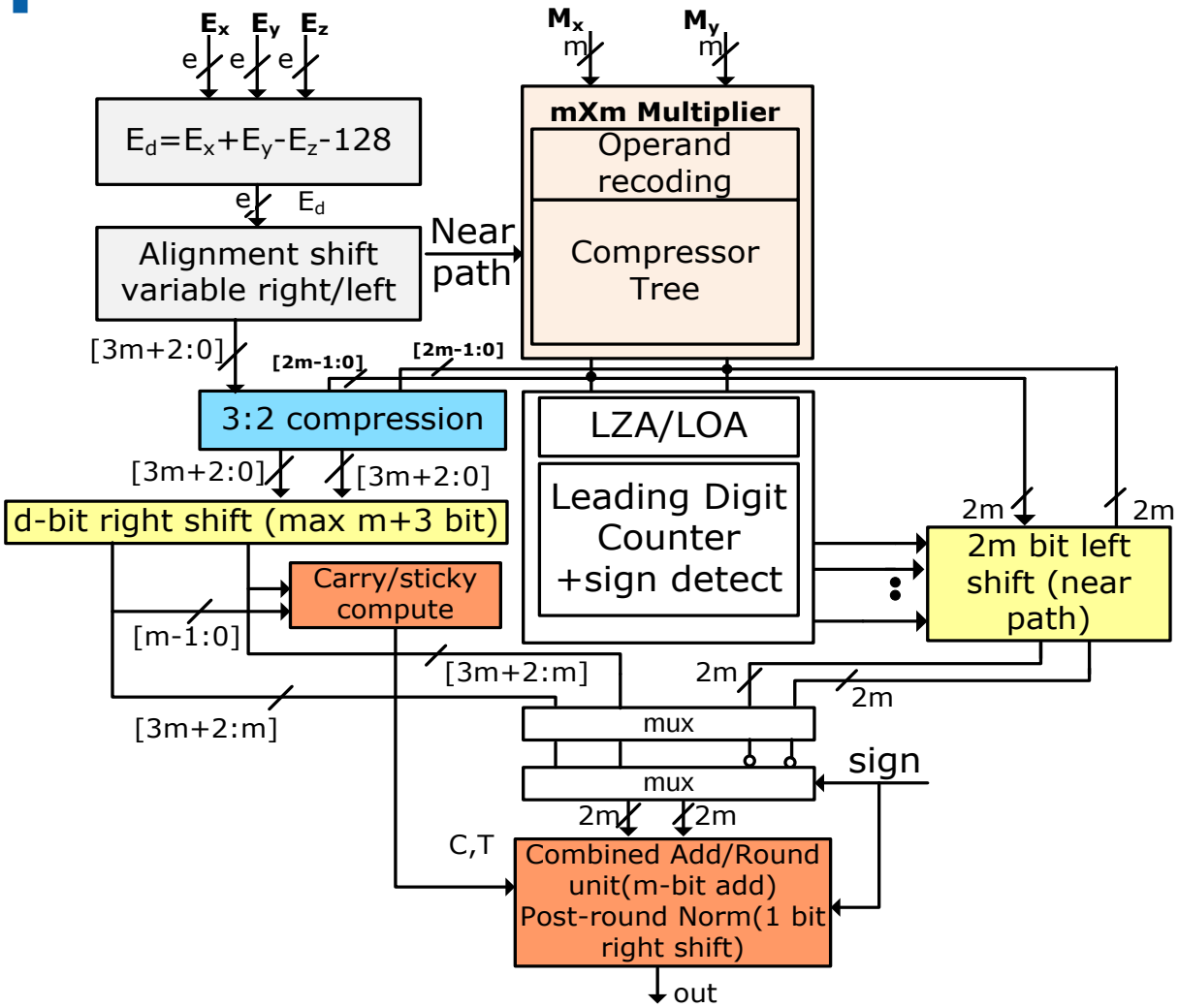
- Pros
  - -Simple alignment shift: Only right shift
  - -Simple unified handling of all cases
- Cons
  - -Increases hardware required: Each unit operating on '3m' bits
  - -Increases logic levels in critical path
    - -Increased size of units: '3m+3' (161-bits for DP!)
    - -Additional norm shifting due to pre-alignment
- Example: IBM POWER 6 [ARITH-07, ISSCC-07]

Lang Fastest DP-FPMAC [ARITH'2003]

- Pros
  - -Simple alignment shift: Right only
  - -Simple unified handling of all cases
  - -No completion adder: Post normalization addition
  - -Early LZA outputs used to hide normalization shifting delay
  - -Combined add/round unit
  - -Part of addition performed early
  - -Fastest w.r.t logic levels
- Cons
  - -Increased area due to '3m' bit units
  - -Significant duplication for sign handling
  - -Sign detect unit comprises the CP
    - -No performance benefit of removing adder
    - -Reduces area gains of removing completion adder
  - -Performance penalties of unified handling

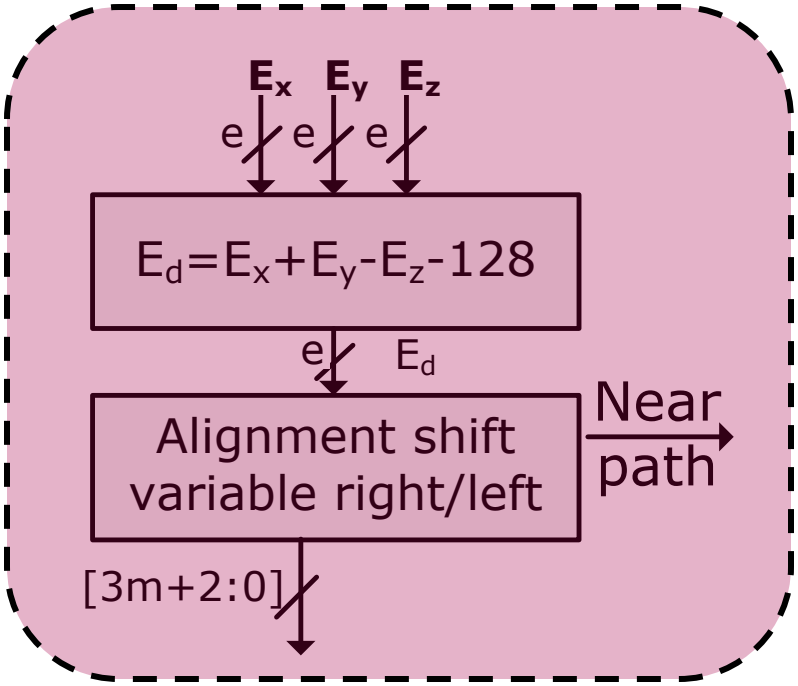


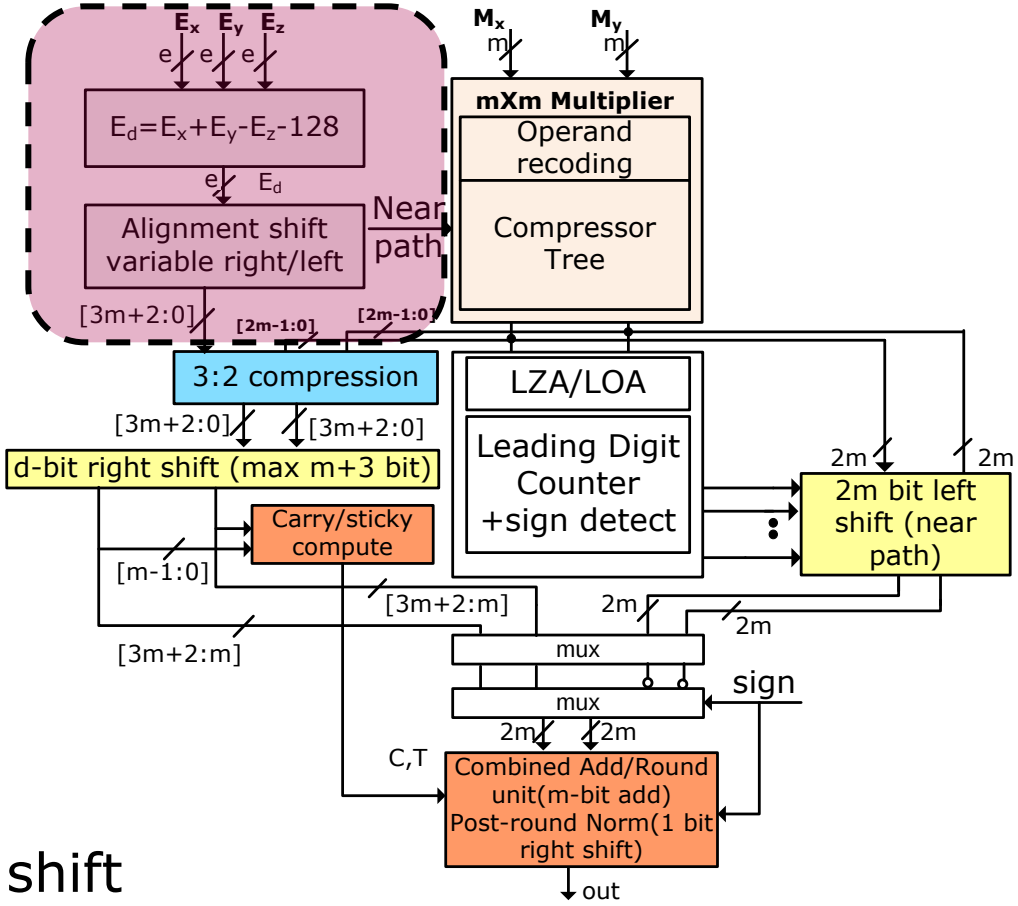
Proposed Dual Precision FPMAC



- Minimal operations/bits-operated for each case=> optimal pipeline w.r.t performance/area respectively

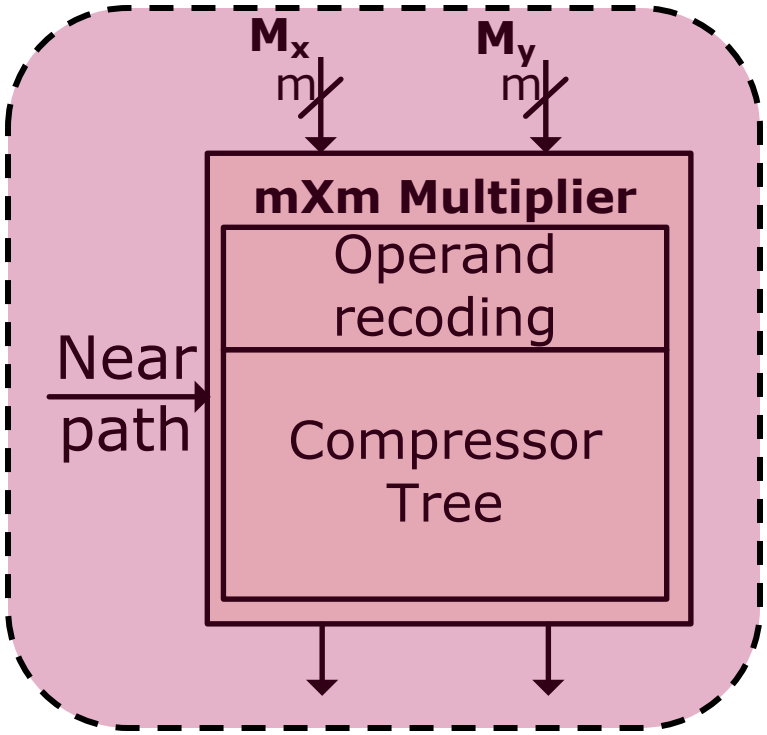
Exponent Compute and Alignment

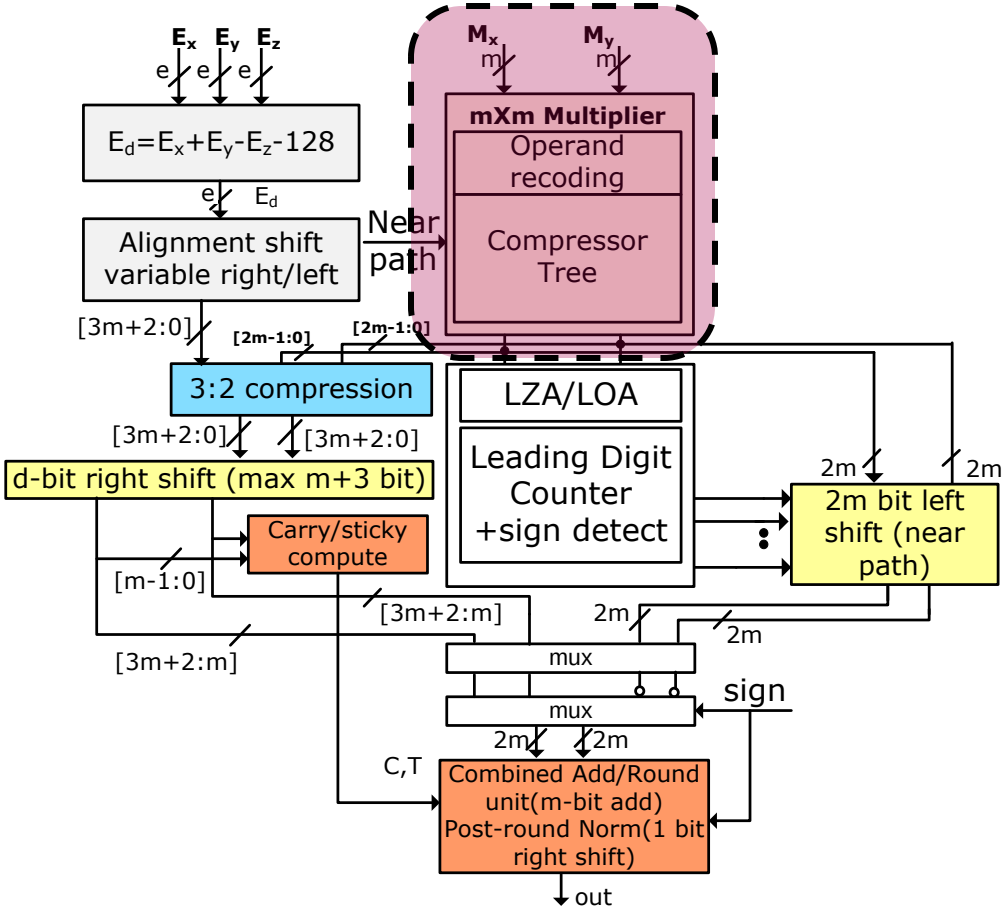




- Variable right/left alignment shift
  - –Early availability of near path accumulate
- Most data path components operate on "2m" bits instead of "3m+2" bits
  - => Significant hardware and performance gains

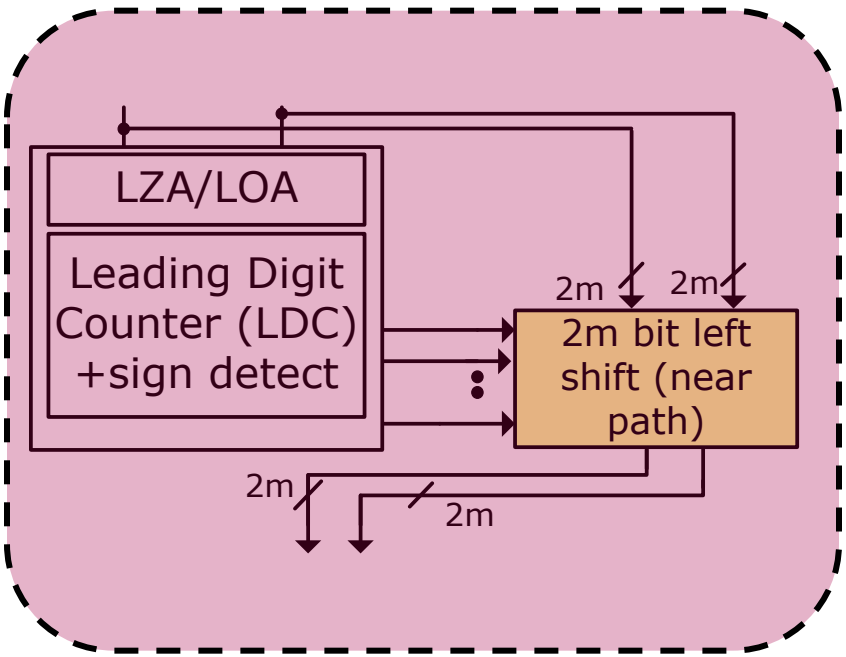
Optimizations in the CS Tree

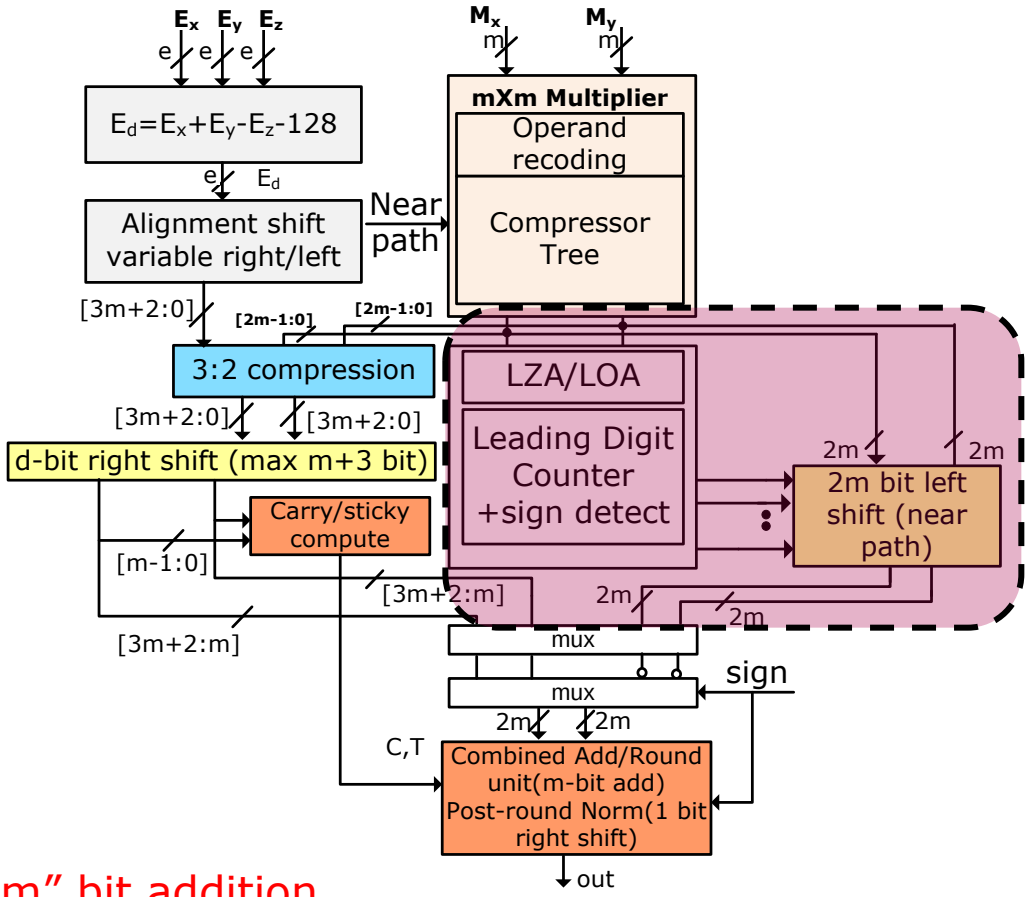




- Early injection of near path aligned mantissa into CS tree
  - No impact on stages needed for compression
- 3:2 compression stage removed from critical near path

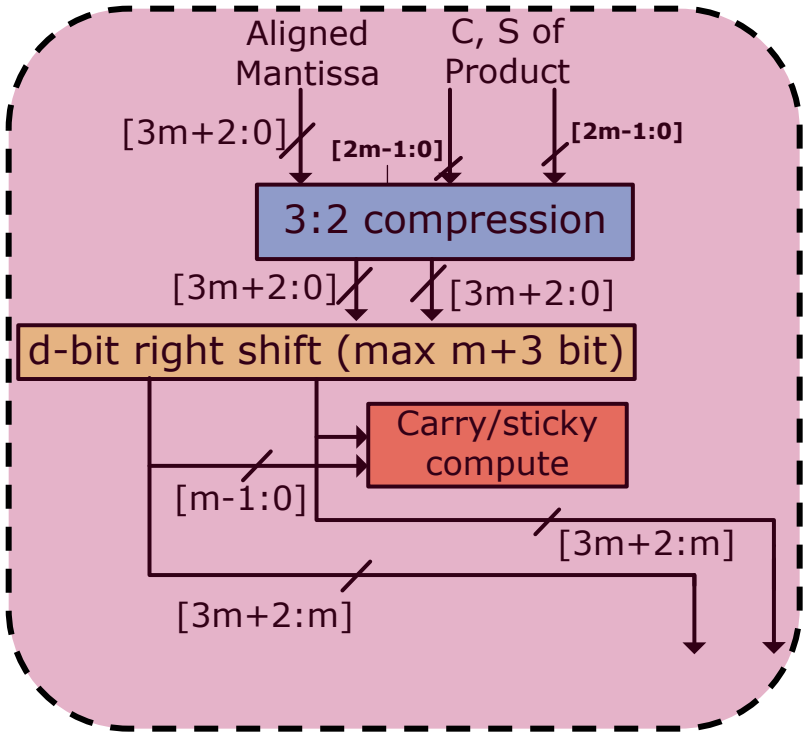
Near Path Normalization

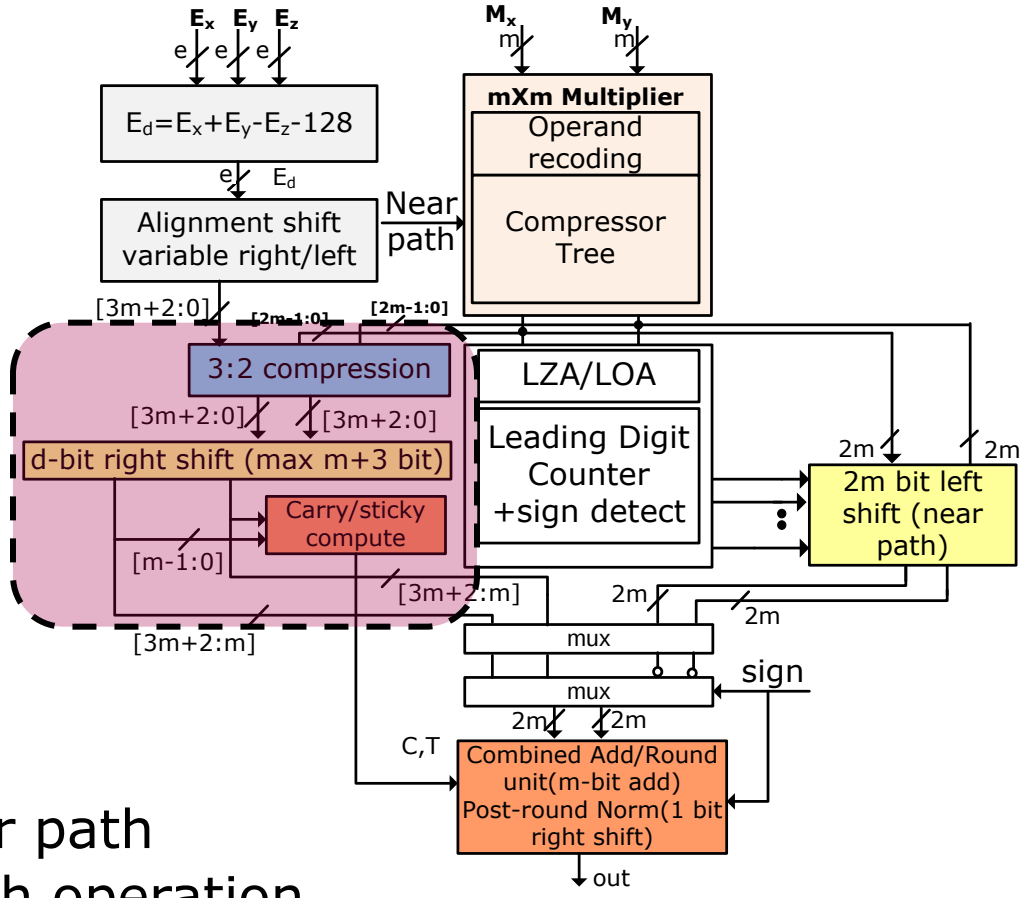




- LZA used for sign detect eliminating "2m" bit addition
- Post-normalization negation for sign handling
  - -LZA outputs immediately used to shift (C,S) hiding normalization shifting delay
  - => Shifter delay almost completely eliminated from critical path

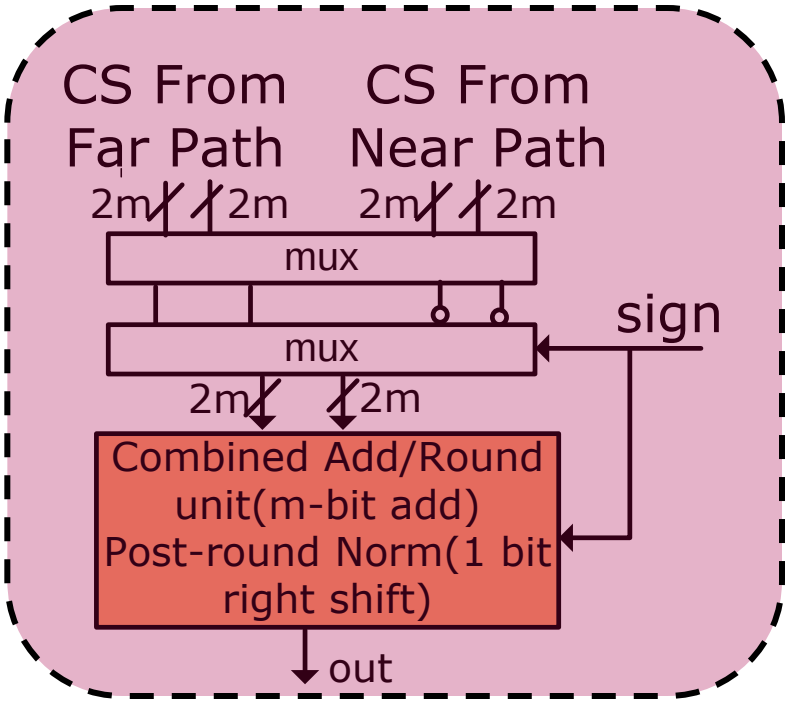
Far Path Normalization

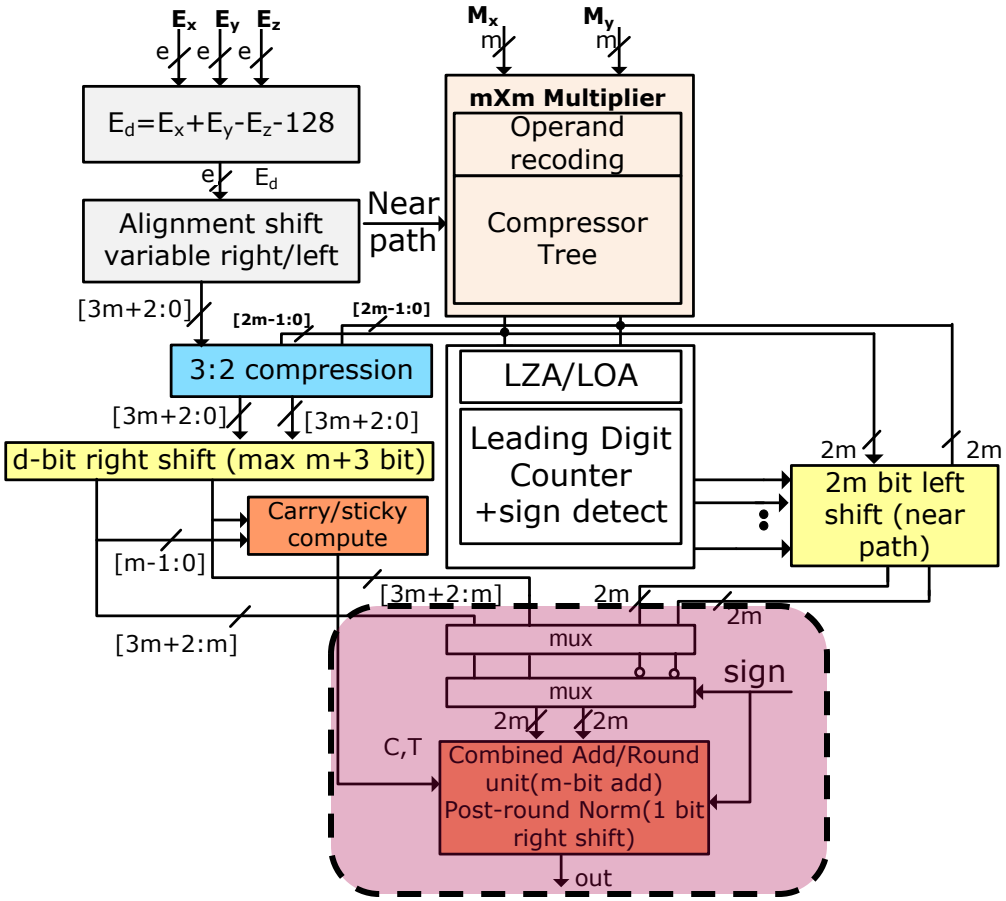




- Split handling of near and far path
  - -Minimal hardware for each operation
  - -Non-critical far path components do not penalize critical near path
  - -Easy clock gating controls for significant power savings
- Eliminated m-bit add and m-bit increment from far path

Add/round unit





- Final addition and rounding is common for both paths
- Operates on "2m" bits unlike earlier "3m+2" bits => area/delay gains

Logic Level Comparison

FMA Design	IBM Power6 FPMAC	Lang FPMAC	Proposed
Multiplier (logic levels)	53-bit multiplier (33)	53 bit multiplier with injected near path (33)	53 bit multiplier with injected near path (33)
Accumulate adder (logic levels)	3:2 compressor (4)	(4)	3:2 compressor 106-bit accumulate-add (0)
	106-bit accumulate add (19)	(0)	(0)
Normalization shifter (logic levels)	LZA (0)	(9)	LZA (9)
	106-bit left shift (8)	161 bit shifter (8)	106-bit left shift (8)
Rounding unit (logic levels)	54 bit round add+GRT on lower 54 bits (18)	(20)	52-bit dual add+GRTC on lower 56 bits (20)
Post rounding norm (logic levels)	1/2-bit right shift (2)	1/2-bit right shift (2)	1/2-bit right shift (2)
Total Logic Levels	**84**	**76**	**72**

Area Comparison

FMA Design	IBM Power6 FPMAC	Lang FPMAC	Proposed
Multiplier (area)	53bit mulltiplier (400)	53bit multiplier (400)	53-bit mul with near path Mz inserted (400)
Alignment shift (area)	56-bit LS on 53 bits, 106-bit RS on 53 bits (100)	161 bit RS on 53 bits (100)	56-bit LS on 53 bits, 106-bit RS on 53 bits (100)
Accumulate adder (area)	106-bit add and 3:2 compress (87) LZA on 106-bit (113)	106 bit 3:2 compress 106-bit (LZA) + 106 bit sign detect + HA (251)	106 bit 3:2 compress 106-bit (LZA + sign detect) (169)
Normalization shifter (area)	106 bit left shifter on 106 bits (100)	106-bit left shift S 106-bit left shift C 53-bit right shift S 53-bit right shift C (200)	106-bit left shift S 106-bit left shift C 53-bit right shift S 53-bit right shift C (200)
Rounding Unit (area)	53-bit add (63)	53-bit dual add (70)	53-bit dual add (70)
Total Norm Area	**863**	**1021**	**939**

Summary

- FPMAC design proposed with following optimizations:
  - Split near and far path during exponent computation stage
  - Optimizes the most critical near path computations
  - Minimal area cost with power saving opportunities
- Comparison with existing implementations showed
  - 14% lesser logic level than fastest silicon implementation
  - 15-20% less switching of gates resulting in power savings
- Implementation of the proposed FPMAC in silicon is ongoing