Tunable Floating-Point for Energy Efficient Accelerators

Alberto Nannarelli

DTU Compute, Technical University of Denmark

25th IEEE Symposium on Computer Arithmetic

Outline

- Motivation and Background
- Tunable Floating-Point Multiplication
- Hardware Implementation
- Application Example
- Improvements
- Conclusions and Future Work

Tunable Floating-Point (TFP)

TFP is a floating-point format with adjustable significand and exponent fields bit-width

Features

- significand $m = [4, 24]$ bits (including hidden bit)
- exponent $e = [5, 8]$ bits
- rounding modes
  - RTZ Round toward zero (truncation)
  - RTN Round to the nearest
  - RTNE Round to the nearest (tie to even)
    IEEE 754 *roundTiesToEven* mode
- *subnormals* flushed to zero

TFP includes *binary32* ($m = 24$, $e = 8$), *binary16* ($m = 11$, $e = 5$), Google's Brain-FP ($m = 8$, $e = 8$).

Motivation

- High throughput parallel multiplication required in most computing applications
- Parallel multiplication is a power demanding operation
- A flexible unit, handling a flexible format, can increase the power efficiency

$$\text{Power efficiency} = \frac{\text{throughput}}{\text{watt}}$$

- Multiplication can be approximated by reducing the precision
- Accuracy of reduced precision can be improved by rounding

Example: Machine Learning

Machine Learning applications, such as *Deep Neural Networks (DNN)*, require huge number of multiplications: $\sum^{N} w_i \cdot x_i$

Reducing the precision, when possible, saves energy.

Example DNN

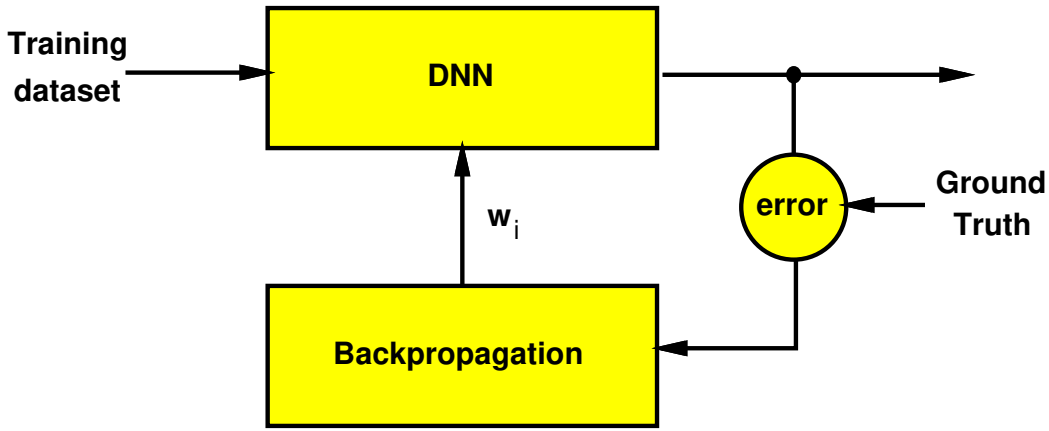
Example: Machine Learning

Machine Learning applications, such as *Deep Neural Networks (DNN)*, require huge number of multiplications: $\sum^{N} w_i \cdot x_i$

Reducing the precision, when possible, saves energy.

Example DNN

- Training



Operations in *binary32* (single-precision)

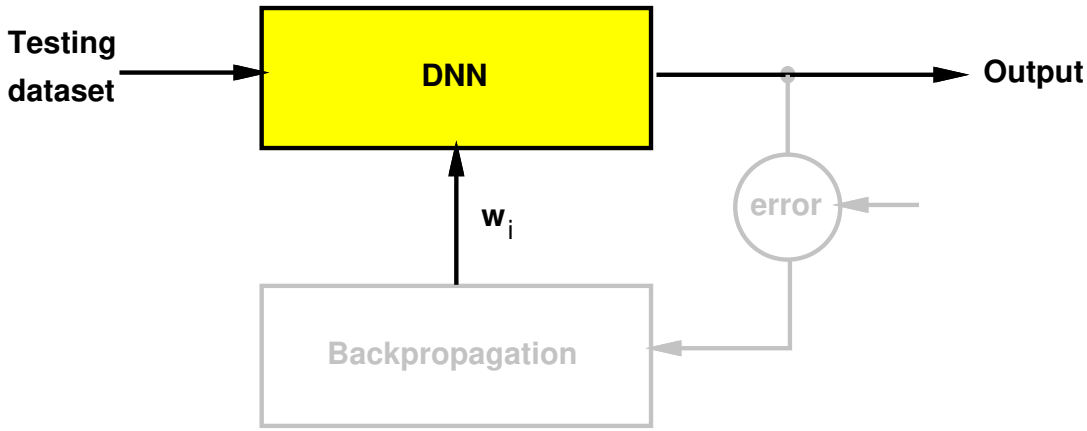
Example: Machine Learning

Machine Learning applications, such as *Deep Neural Networks (DNN)*, require huge number of multiplications: $\sum^{N} w_i \cdot x_i$

Reducing the precision, when possible, saves energy.

Example DNN

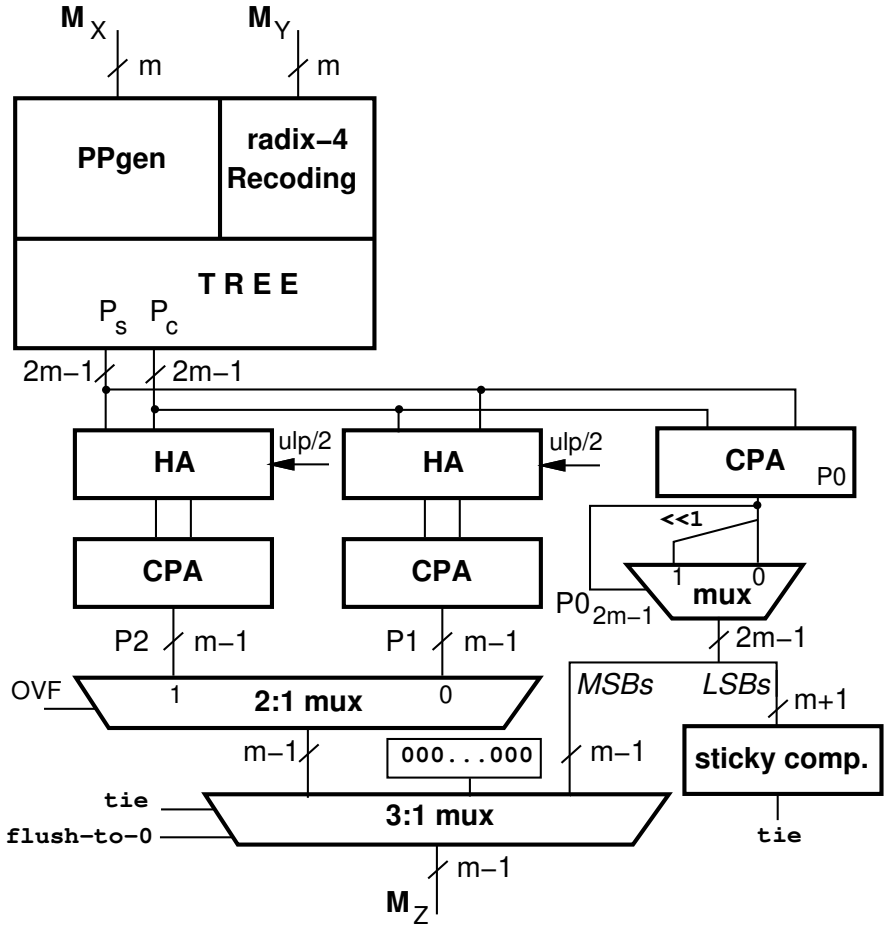
- Inference



Operations in *binary16* (half-precision)

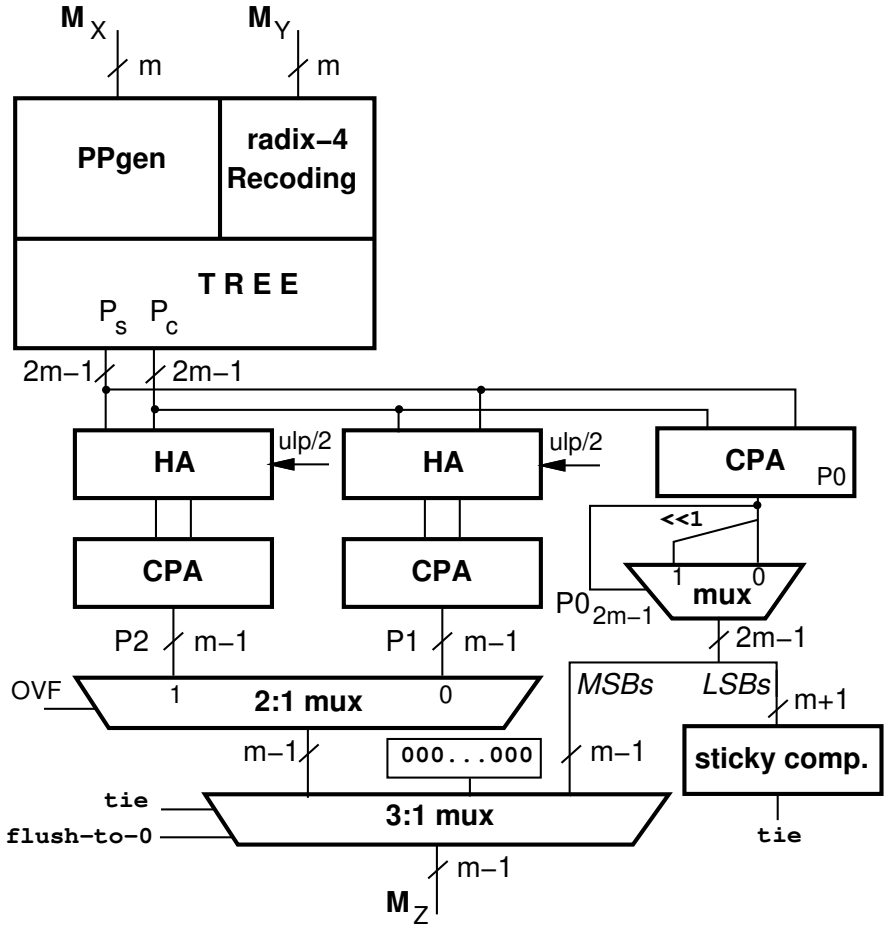
Baseline FP-Multiplier (Significand)

- Radix-4 array multiplier
  - Radix-4 Recoding
  - PPs generation
  - Adder tree (carry-save): P_S, P_C



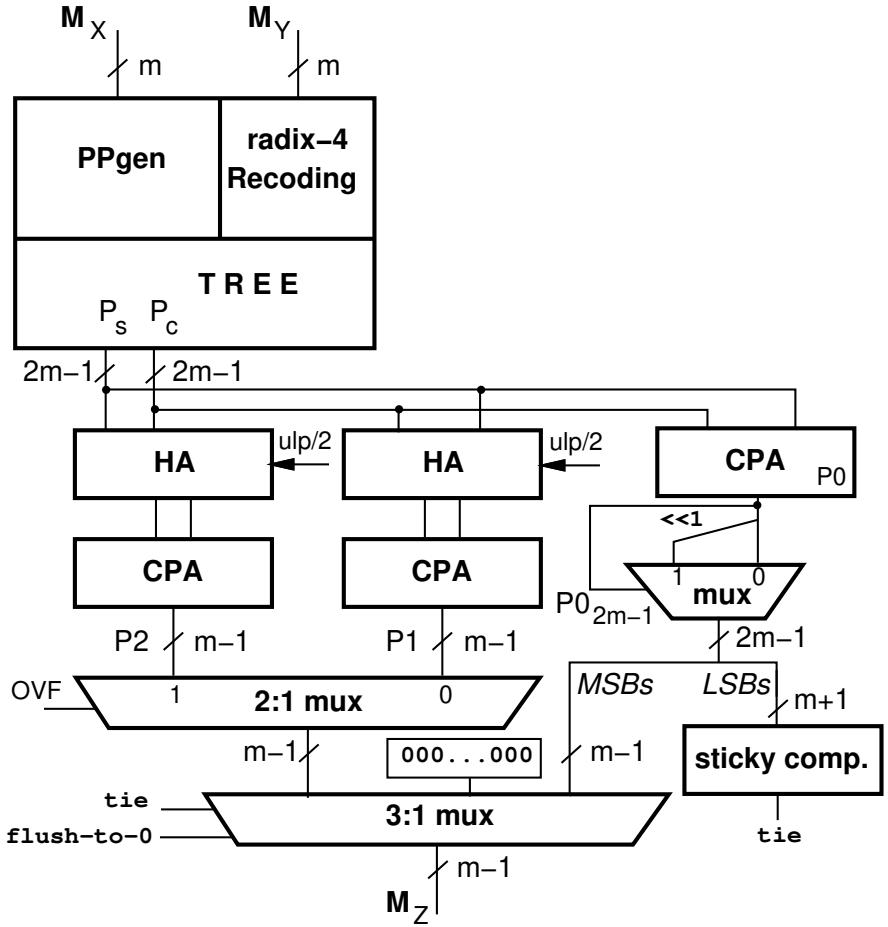
Baseline FP-Multiplier (Significand)

- Radix-4 array multiplier
  - Radix-4 Recoding
  - PPs generation
  - Adder tree (carry-save): P_S, P_C
- Speculative Rounding
  - P2: $2 \leq P < 4$
  - P1: $1 \leq P < 2$ (norm.)
  - P0: no rounding bit injection

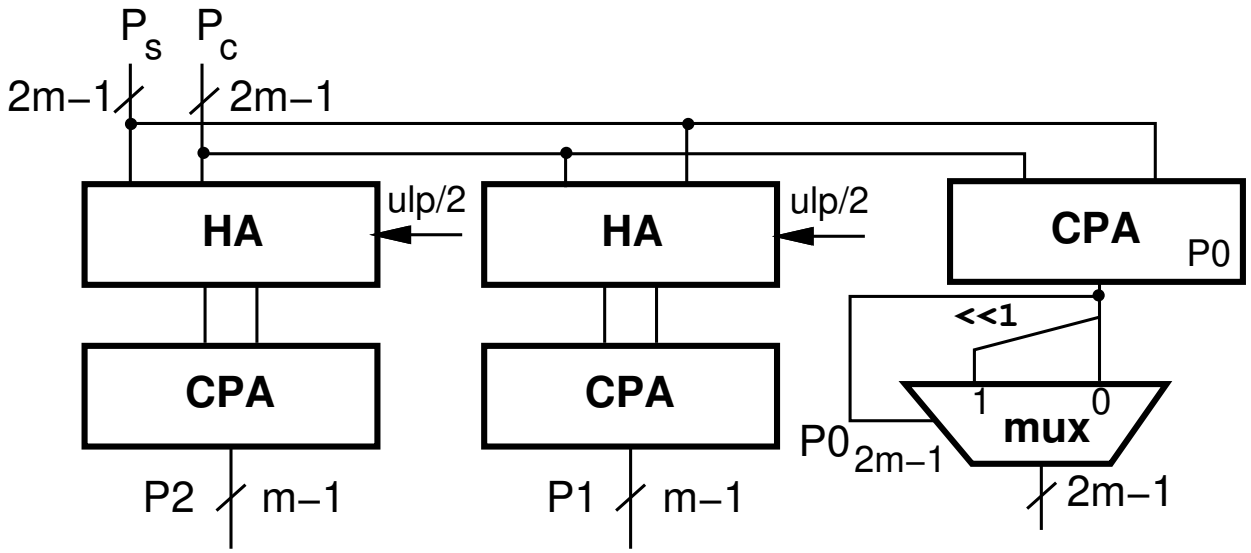


Baseline FP-Multiplier (Significand)

- Radix-4 array multiplier
  - Radix-4 Recoding
  - PPs generation
  - Adder tree (carry-save): P_S, P_C
- Speculative Rounding
  - P2: $2 \leq P < 4$
  - P1: $1 \leq P < 2$ (norm.)
  - P0: no rounding bit injection
- Normalization & Selection
  - OVF overflow $P \geq 2$
  - tie-to-even
  - underflow (exp=0)
    overflow (exp=∞)



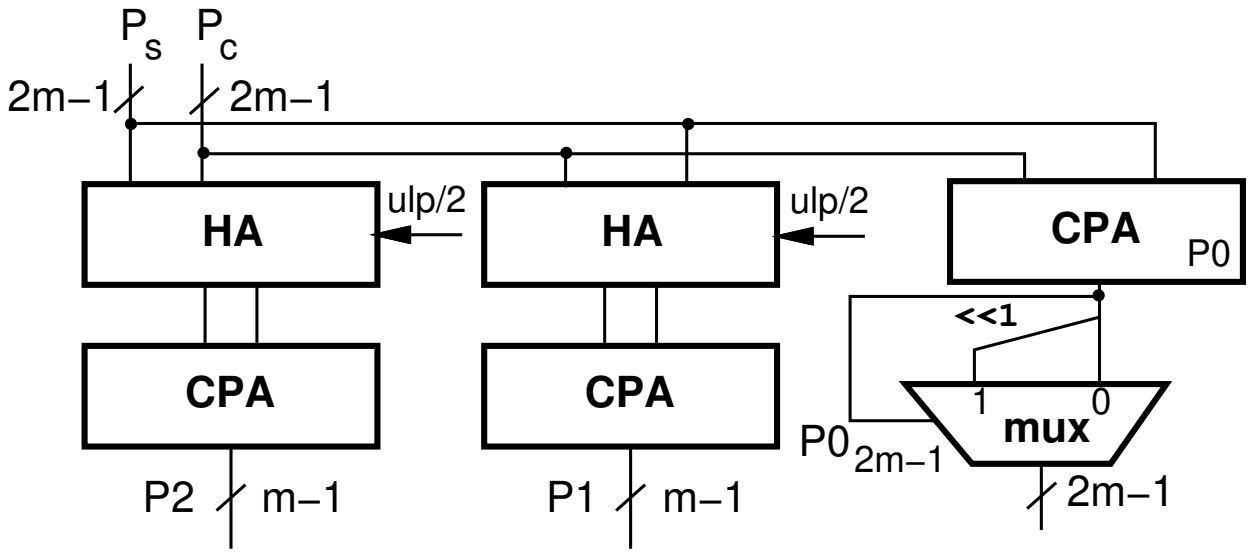
Speculative Rounding



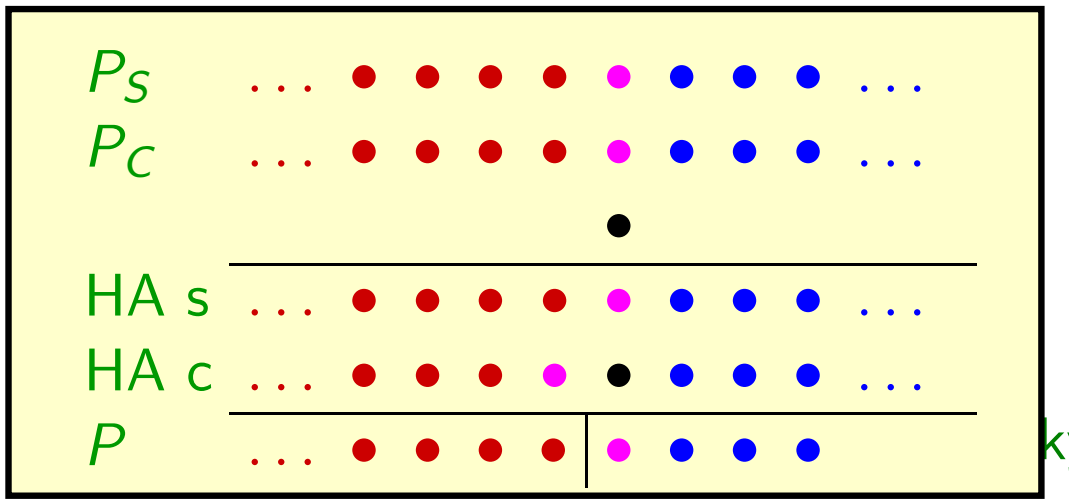
pos.	*2m-1*	*2m-2*	*2m-3*	...	*m+2*	*m+1*	*m*	*m-1*	*m-2*	...	*1*	*0*
P2	1	b	.	b	...	b	L	R	b	b	...	b	b
ulp/2							**1**					
P1	0	1	.	b	...	b	b	L	R	b	...	b	b
ulp/2								**1**				

Sticky bit $T = \mathrm{OR}_{i=0}^{m-1} b_i$

Speculative Rounding



pos. *2m-1 2m-2 2m-3 ... m+2 m+1 m m-1 m-2 ... 1 0*



R b b ... b b

1

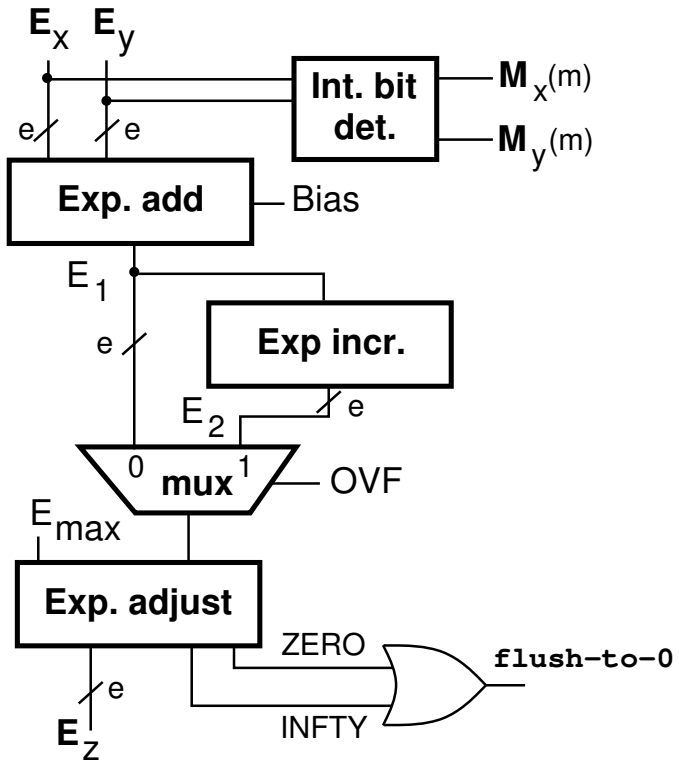
L R b ... b b

1

ky bit $T = \mathrm{OR}_{i=0}^{m-1} b_i$

Baseline FP-Multiplier (Exponent)

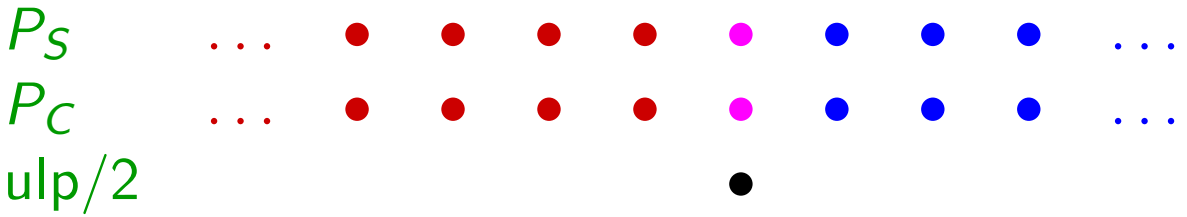
- At beginning of operation and in parallel
  - Detect if operands are zero $\Rightarrow$ set integer bit in M
  - Add exponents $E_1 = E_X + E_Y - \text{Bias}$
- Increment exponent $E_2 = E_1 + 1$
- Select exponent on OVF
- Check conditions
  - underflow (exp=0)
  - overflow (exp=∞)
  - Set `flush-to-0`
- Set exponent E_Z



TFP Rounding

Need to round in variable position according to m

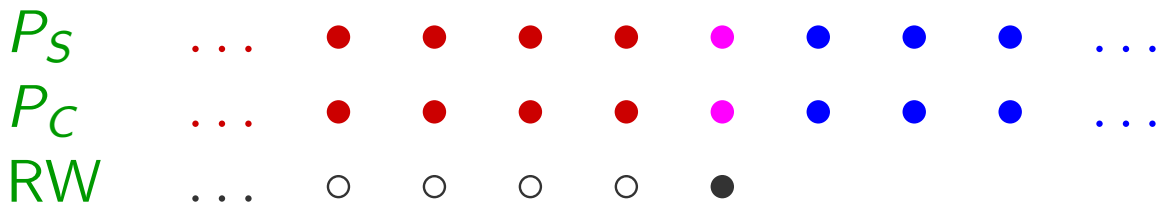
baseline *binary32*



TFP Rounding

Need to round in variable position according to m

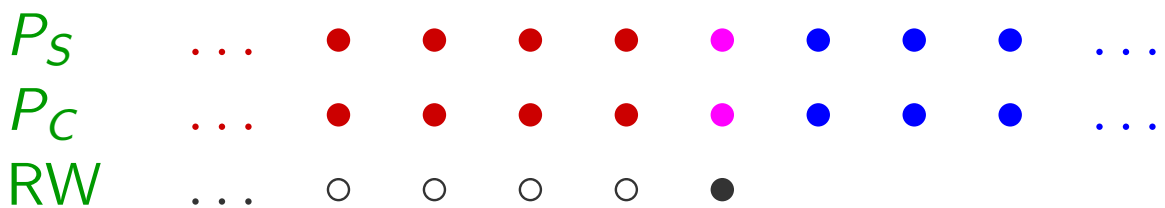
TFP $m = 24$



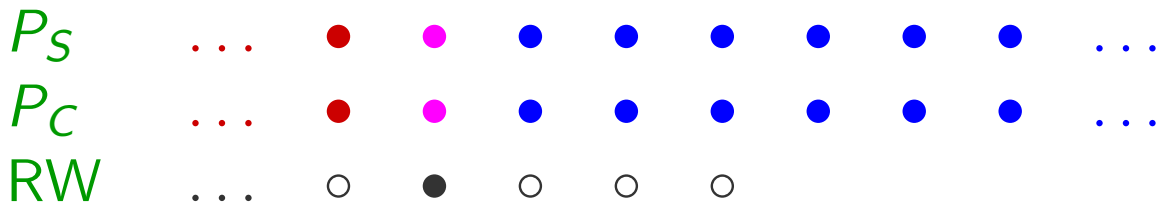
TFP Rounding

Need to round in variable position according to m

TFP $m = 24$



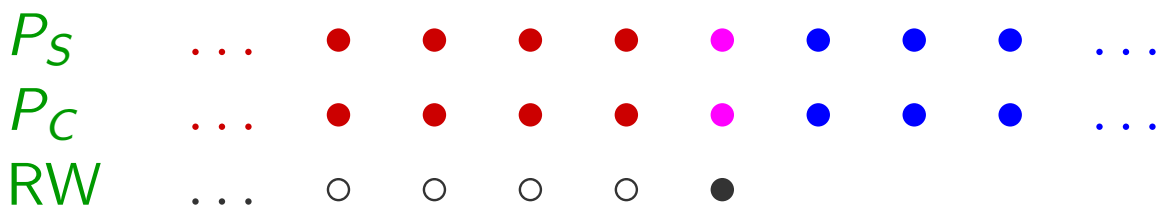
TFP $m = 21$



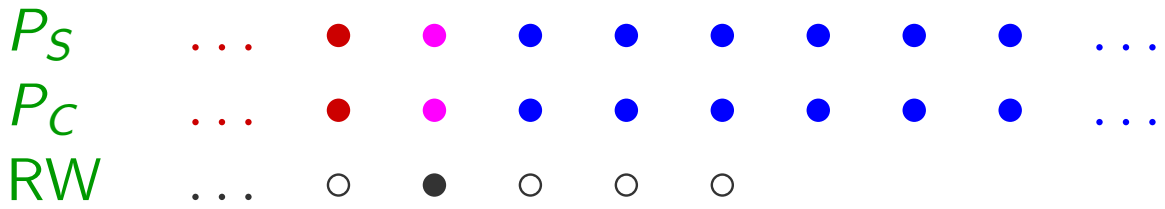
TFP Rounding

Need to round in variable position according to m

TFP $m = 24$



TFP $m = 21$

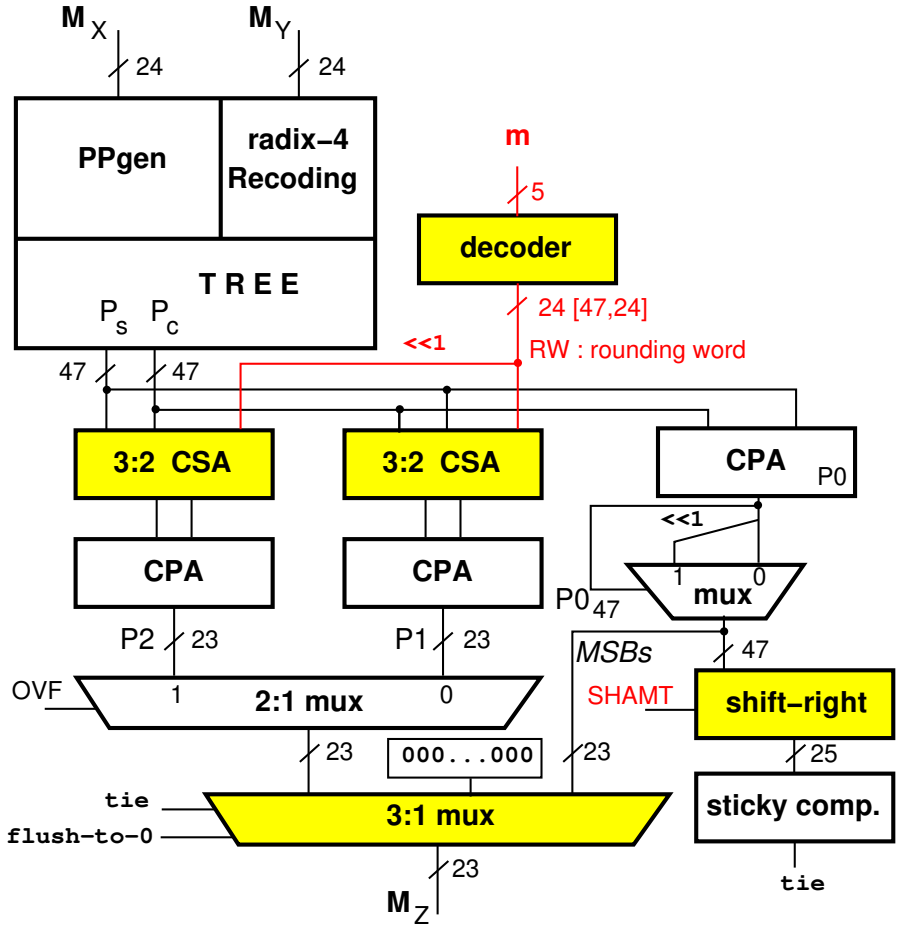


- Need to introduce RW rounding word for rounding
- Need to select *"blue"* bits for sticky bit computation

TFP Multiplier (Significand)

Hardware to support TFP

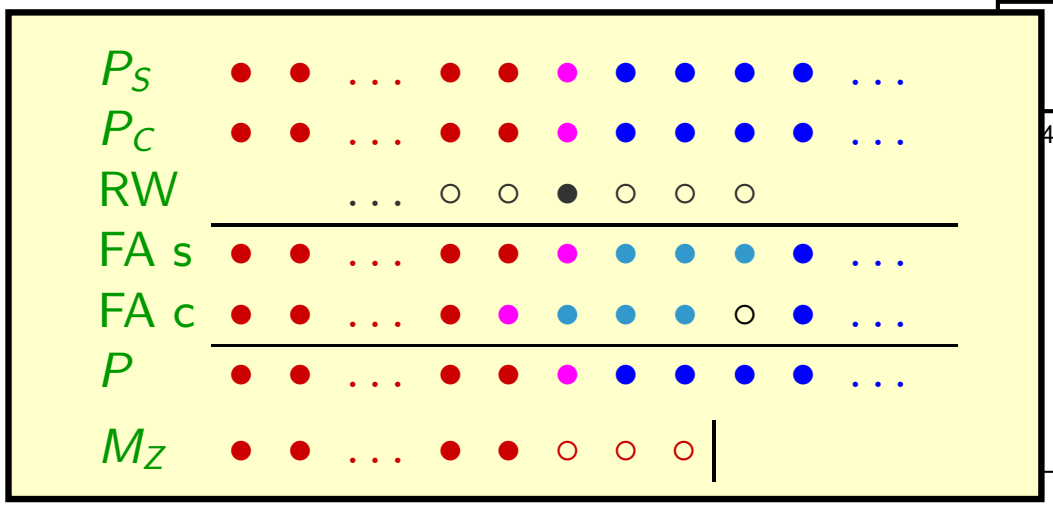
- Decoder
  needed to generate RW from *m*
- HA changed in FA (3:2 CSA)

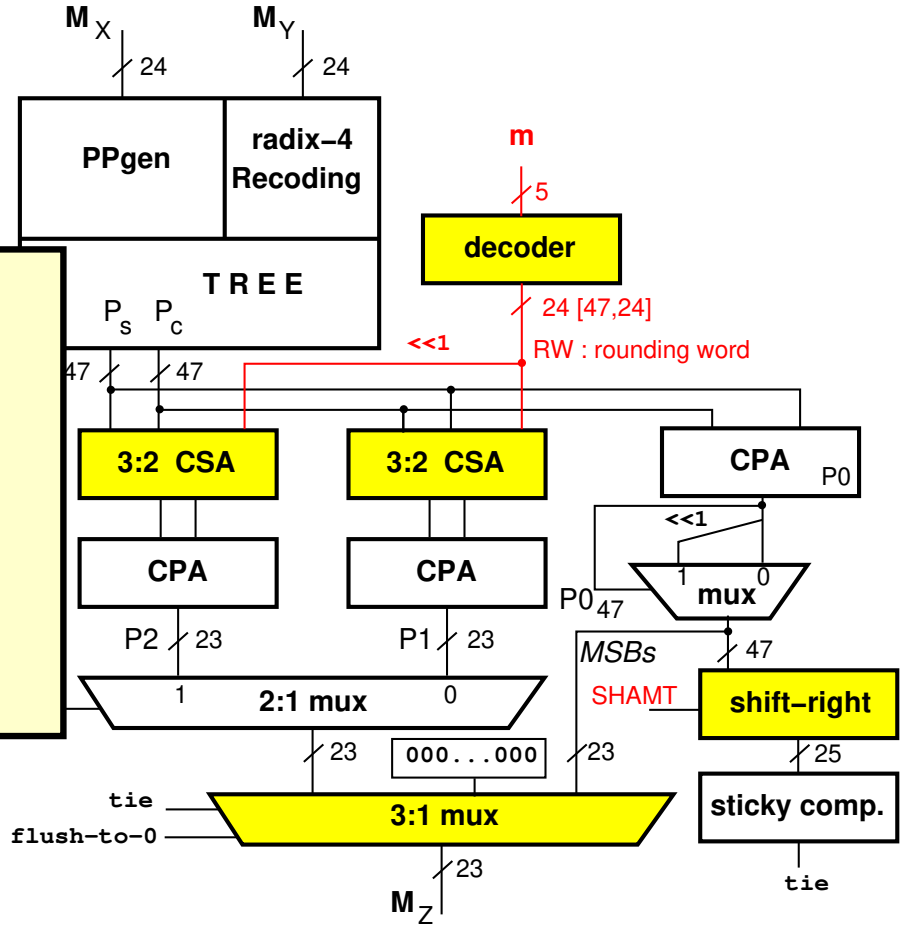


TFP Multiplier (Significand)

Hardware to support TFP

- Decoder
  needed to generate RW from m
- HA changed in FA (3:2 CSA)

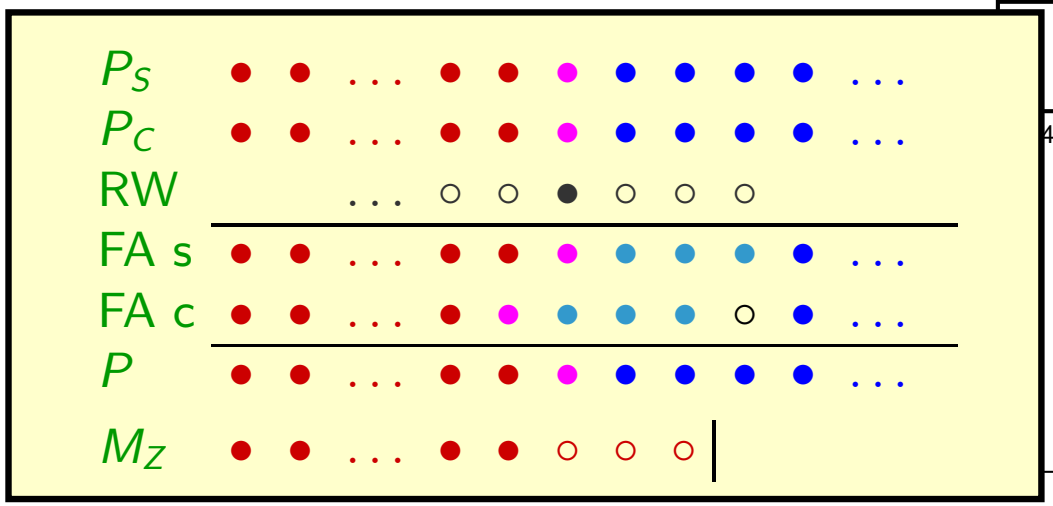


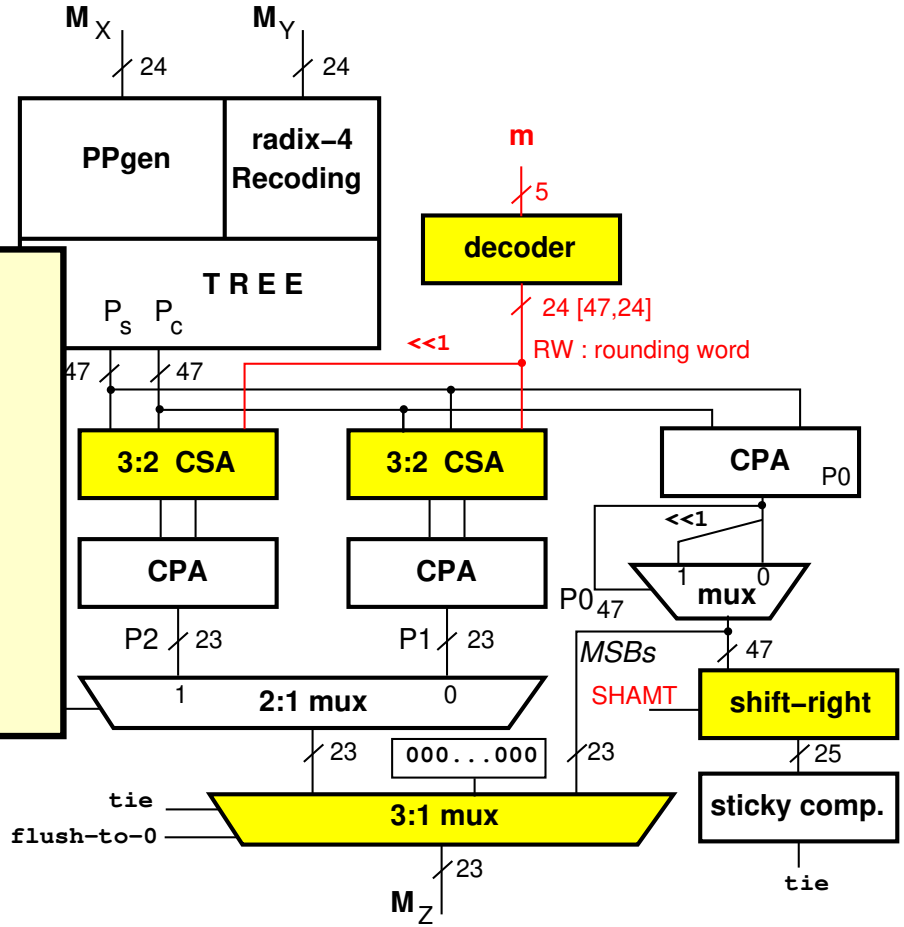


TFP Multiplier (Significand)

Hardware to support TFP

- Decoder
  needed to generate RW from *m*
- HA changed in FA (3:2 CSA)

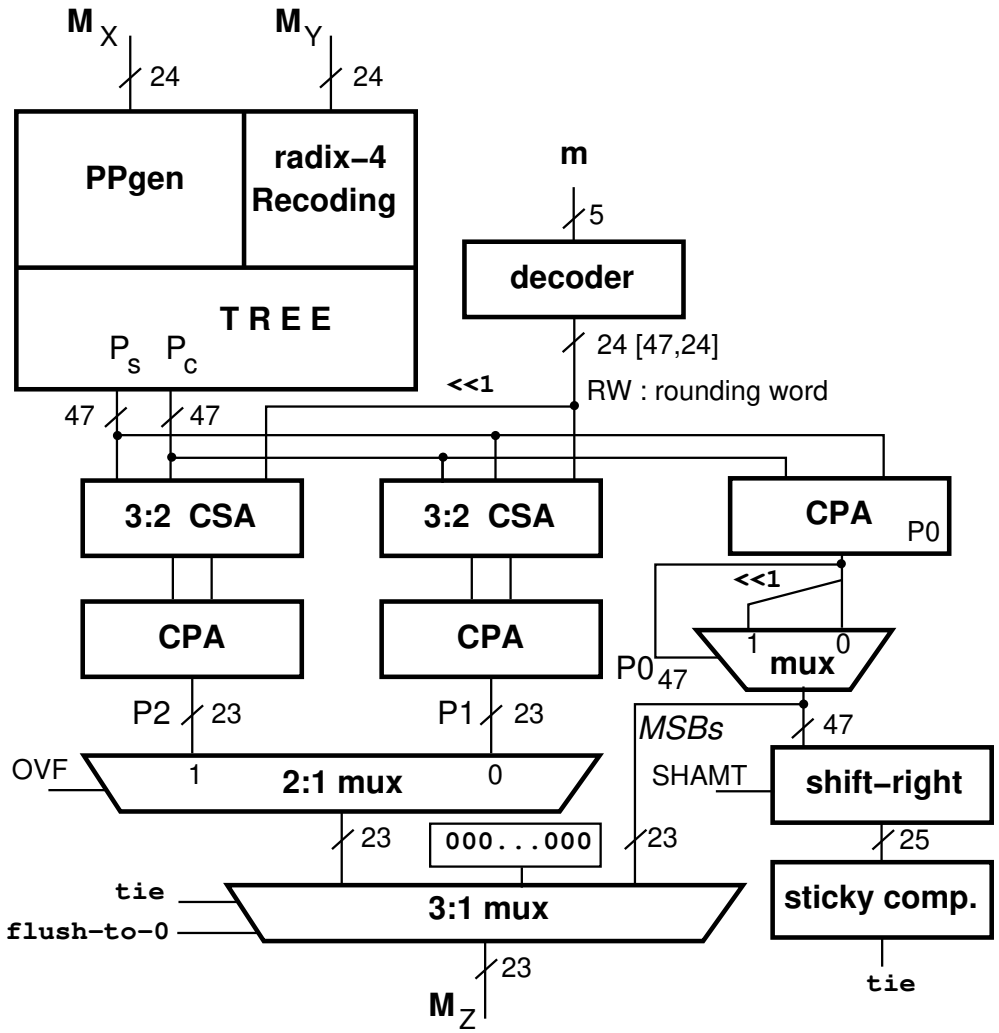




- Barrel shifter (to right)
  to select *"blue"* bits for sticky bit computation
- Modified selection multiplexer to *"zero"* LSBs of result

TFP Multiplier Rounding Modes

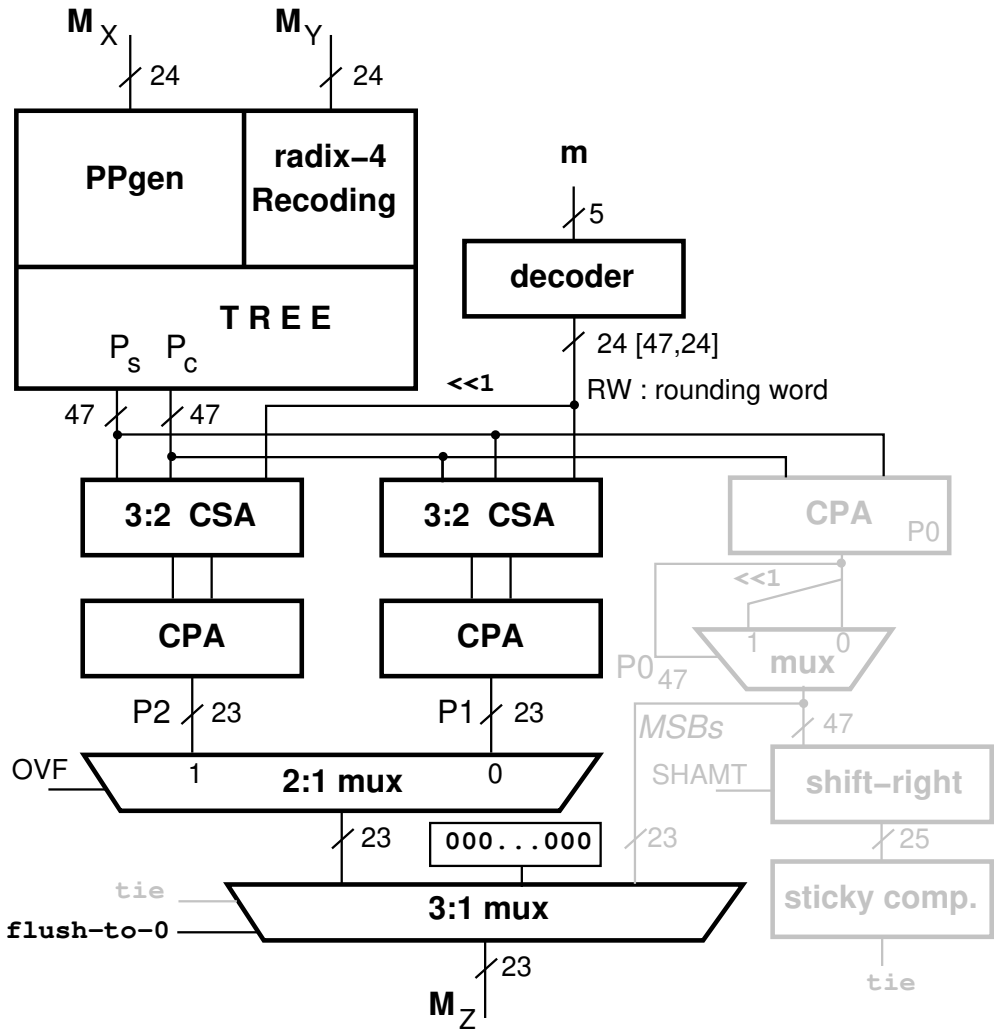
- RTNE
  IEEE *roundTiesToEven*
  Round to the nearest
  (tie to even)



TFP Multiplier Rounding Modes

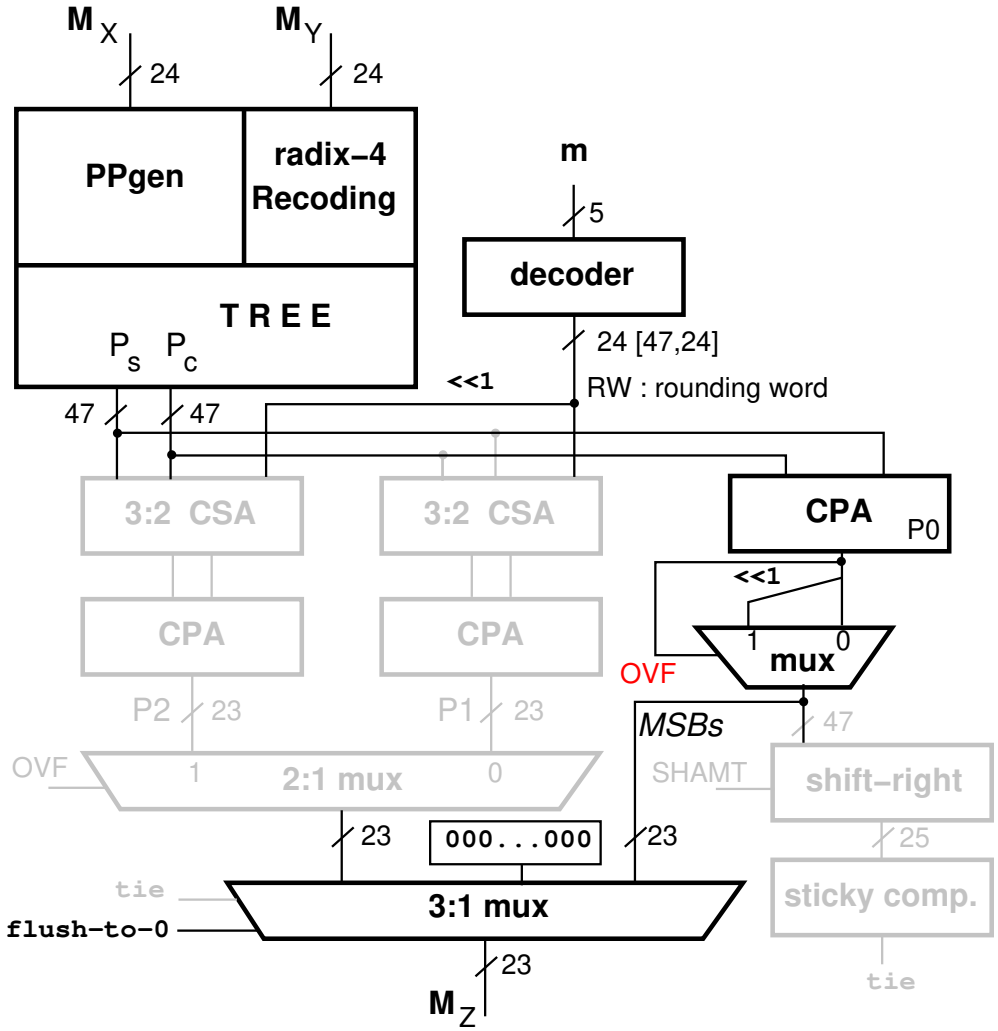
- RTN

  Round to the nearest



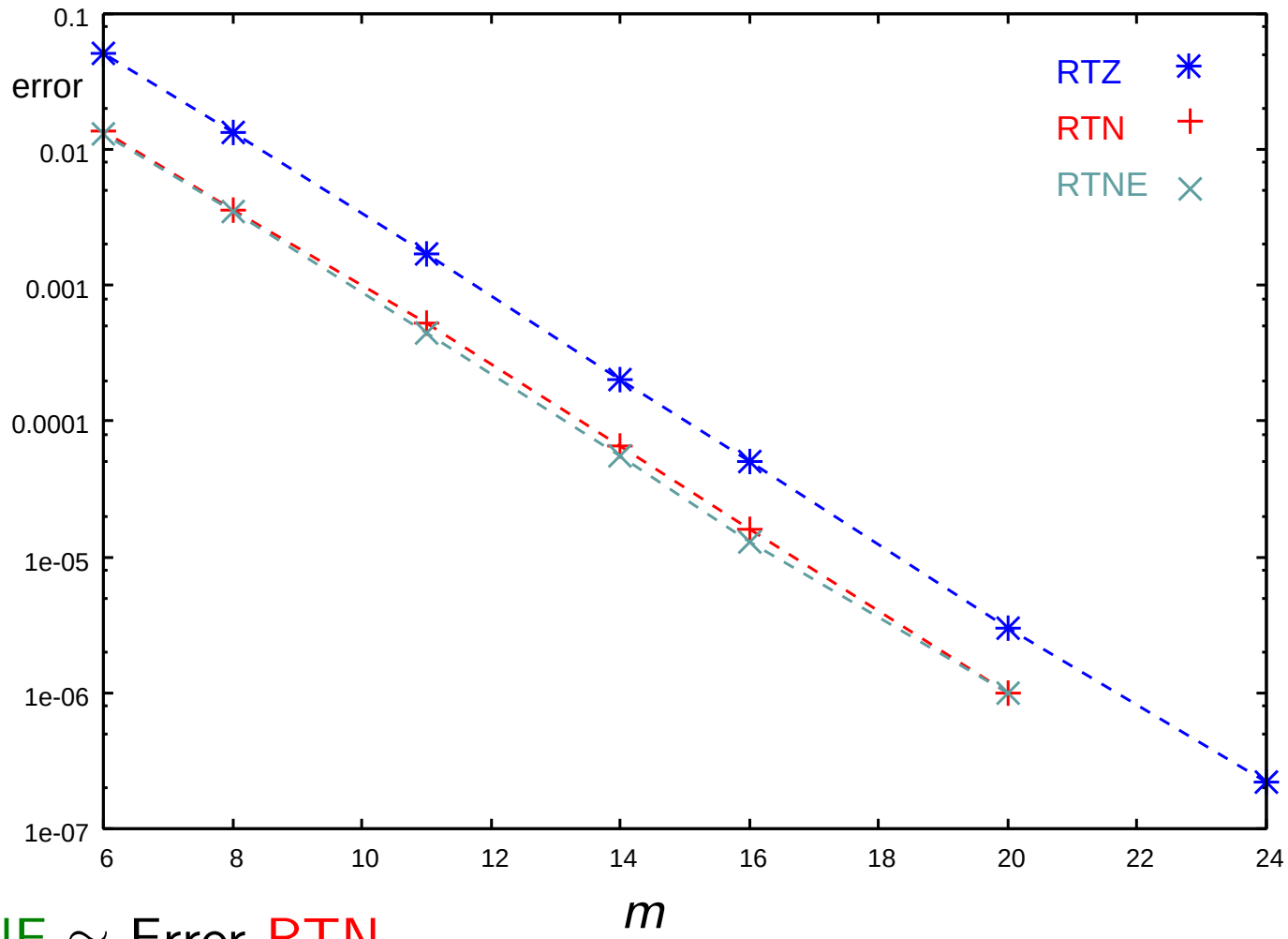
TFP Multiplier Rounding Modes

- RTZ
  Round toward zero
  (truncation)



TFP Error vs. Precision m

Matrix multiplication 8×8 for $m = \{6, 8, 11, 14, 16, 20, 24\}$



Error RTNE $\simeq$ Error RTN

Hardware Implementation

STM 45 nm low-power CMOS standard cells
Clock period 1.0 ns $\simeq$ 15 FO4 $\Rightarrow$ 2-stage pipeline

Post synthesis comparison

	Area$^{(1)}$	Power$^{(2)}$	$\overline{\text{error}}$
RTNE	15,080	21.57	$2.44 \cdot 10^{-3}$
RTN	12,860	18.98	$2.56 \cdot 10^{-3}$
RTZ	10,400	16.60	$9.42 \cdot 10^{-3}$

$^{(1)}$[NAND2 equiv.], $^{(2)}$[mW] at 1 GHz

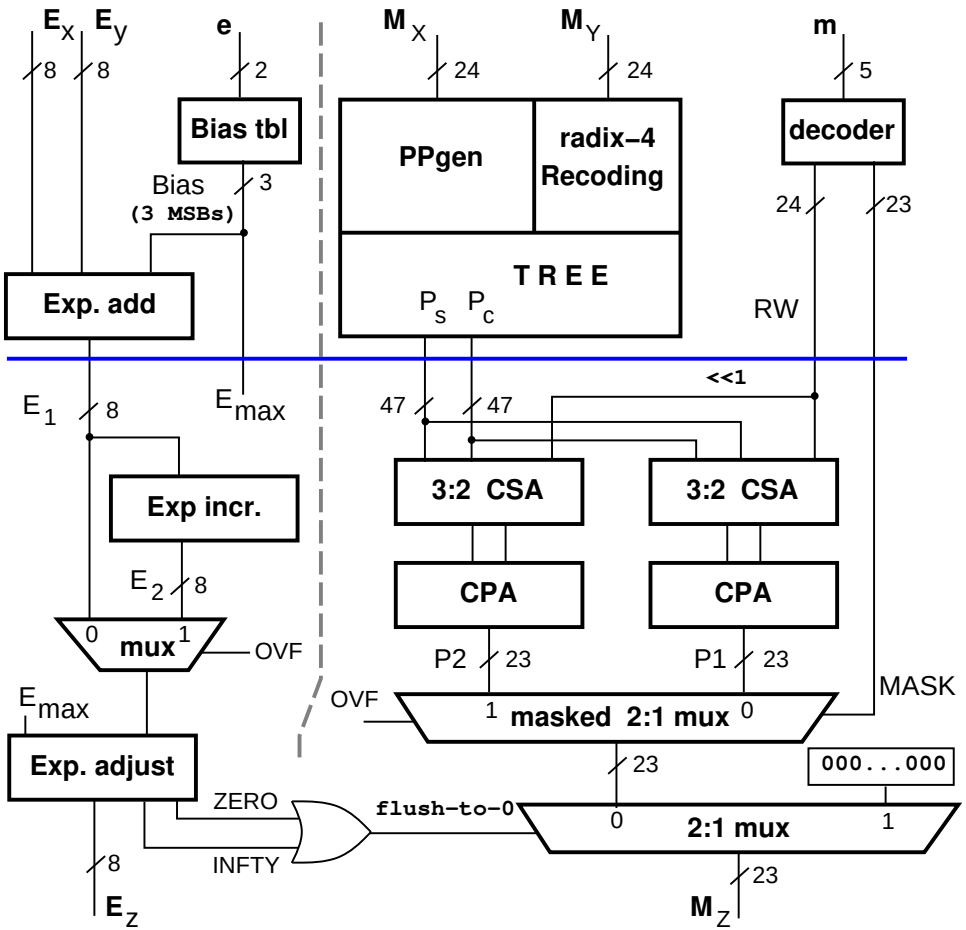
Trade-offs

	RTNE	RTN	RTZ
	Power (ratio)		
RTNE	1.00	0.88	0.77
RTN	0.95	1.00	0.89
RTZ	0.26	0.27	1.00
	Accuracy (ratio)		

Best tradeoffs for TFP multiplier RTN mode

TFP multiplier RTN

Physical implementation (layout)



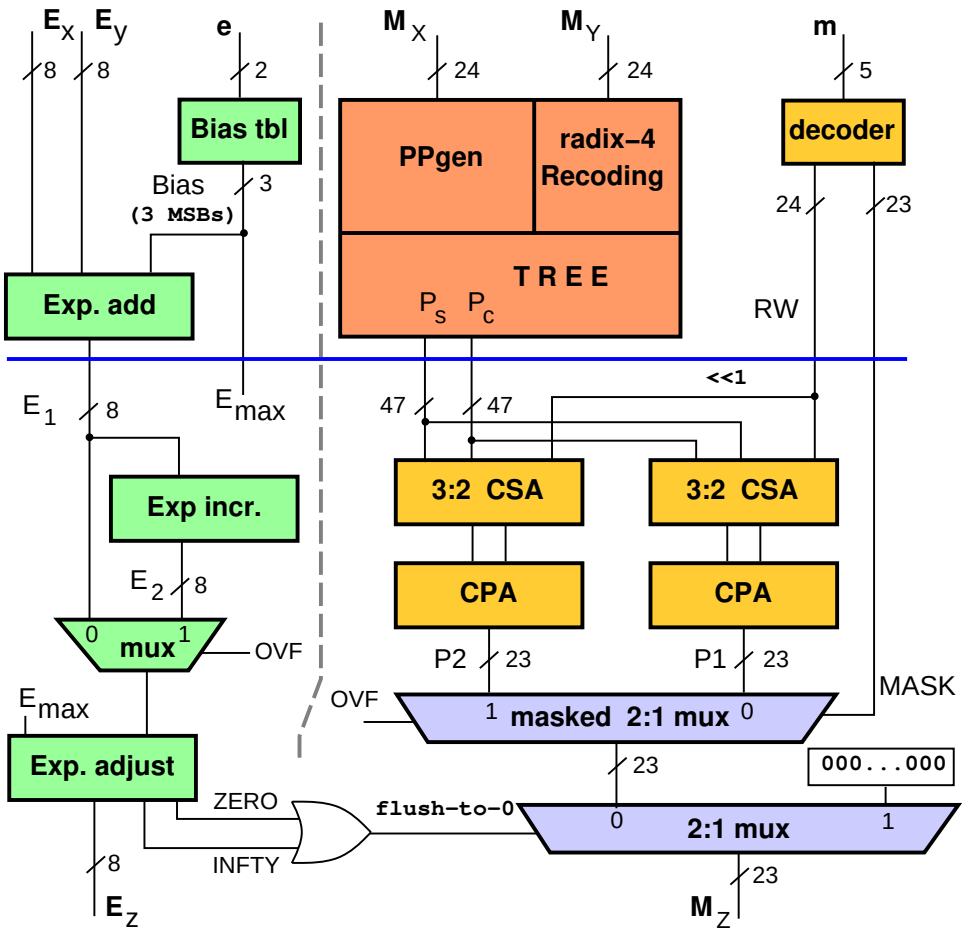
Clock period 1.0 ns (2-stage)

Area 10,930 [NAND2 equiv.]

Power dissipation 15.5 mW (1 GHz)
(*binary32* operands)

TFP multiplier RTN

Physical implementation (layout)



Clock period 1.0 ns (2-stage)

Area 10,930 [NAND2 equiv.]

Power dissipation 15.5 mW (1 GHz)

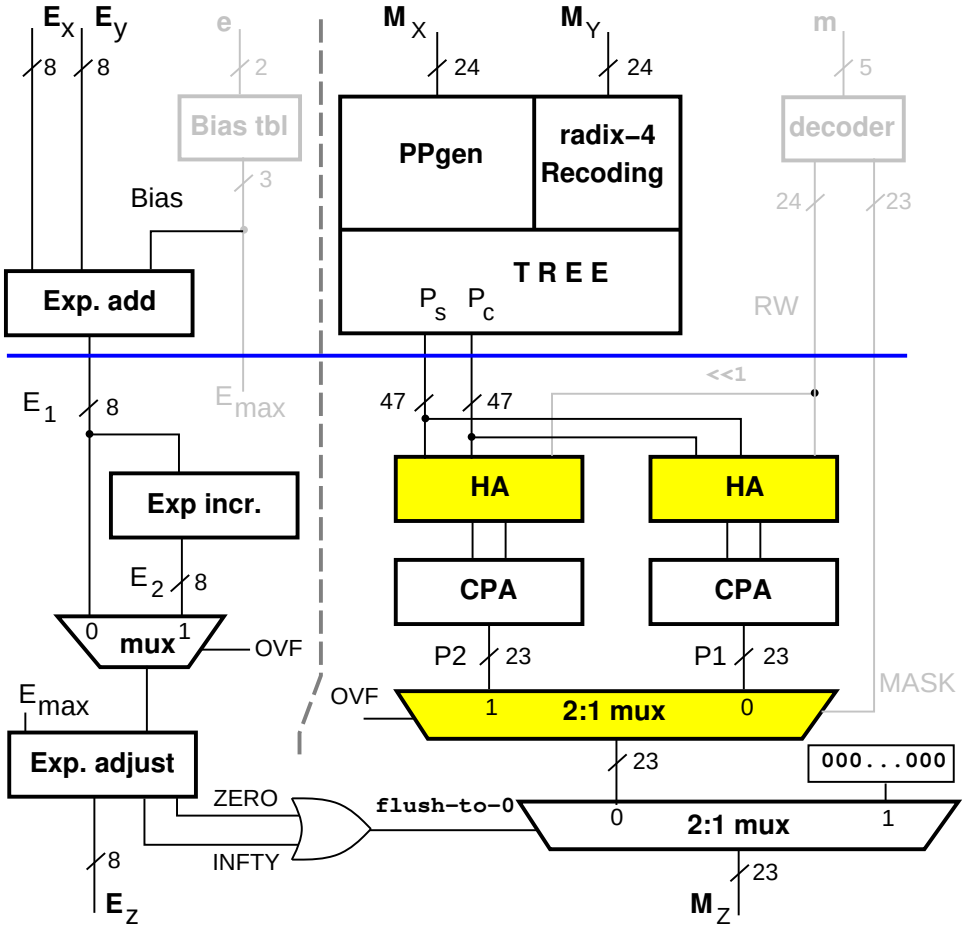
(*binary32* operands)

Power dissipation breakdown

	percent
Mult. array	75
Spec. rounding	13
Norm. & selection	2
Exponent	3
Regs. etc.	6

binary32 multiplier RTN

Physical implementation (layout)



Comparison multiplier TFP vs. *binary32* (RTN mode)

	Area (post-layout)	
b32	9,790	
TFP	10,930	(+11%)

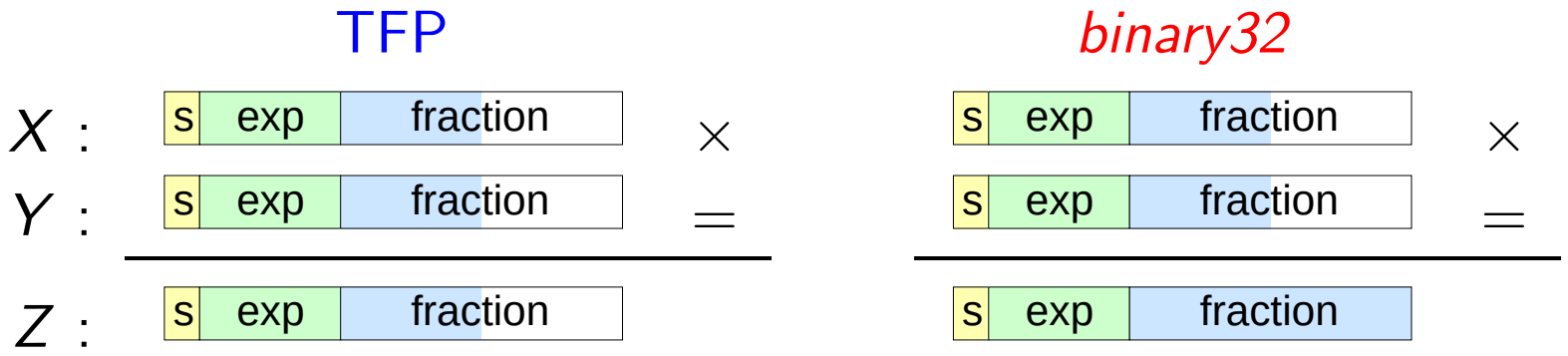
[NAND2 equiv.]

Power dissipation

1. $Z = X \times Y$
2. $Z = (X \times Y) \times W$

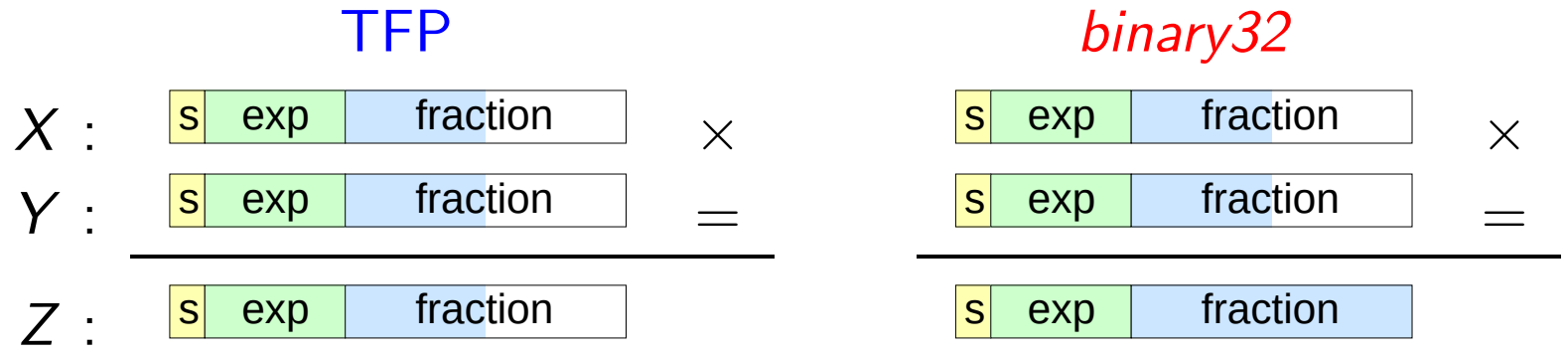
Switching Activity in TFP and binary32 Multiplier

1. $Z = X \times Y$

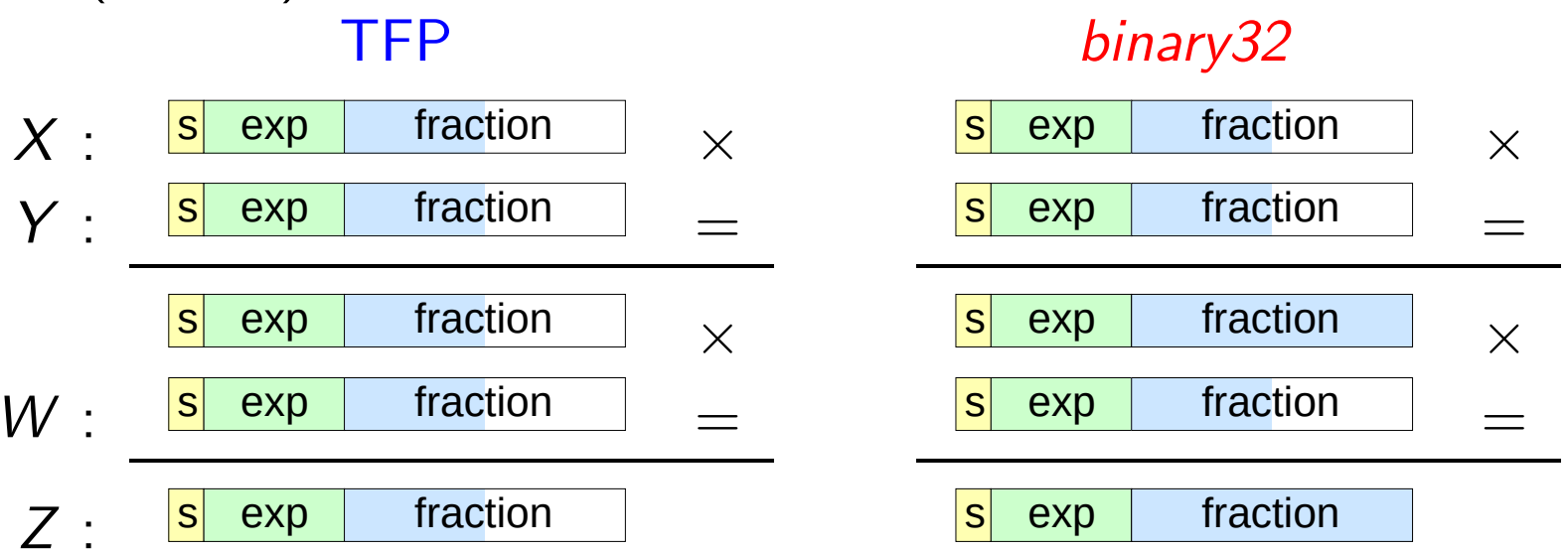


Switching Activity in TFP and binary32 Multiplier

1. $Z = X \times Y$



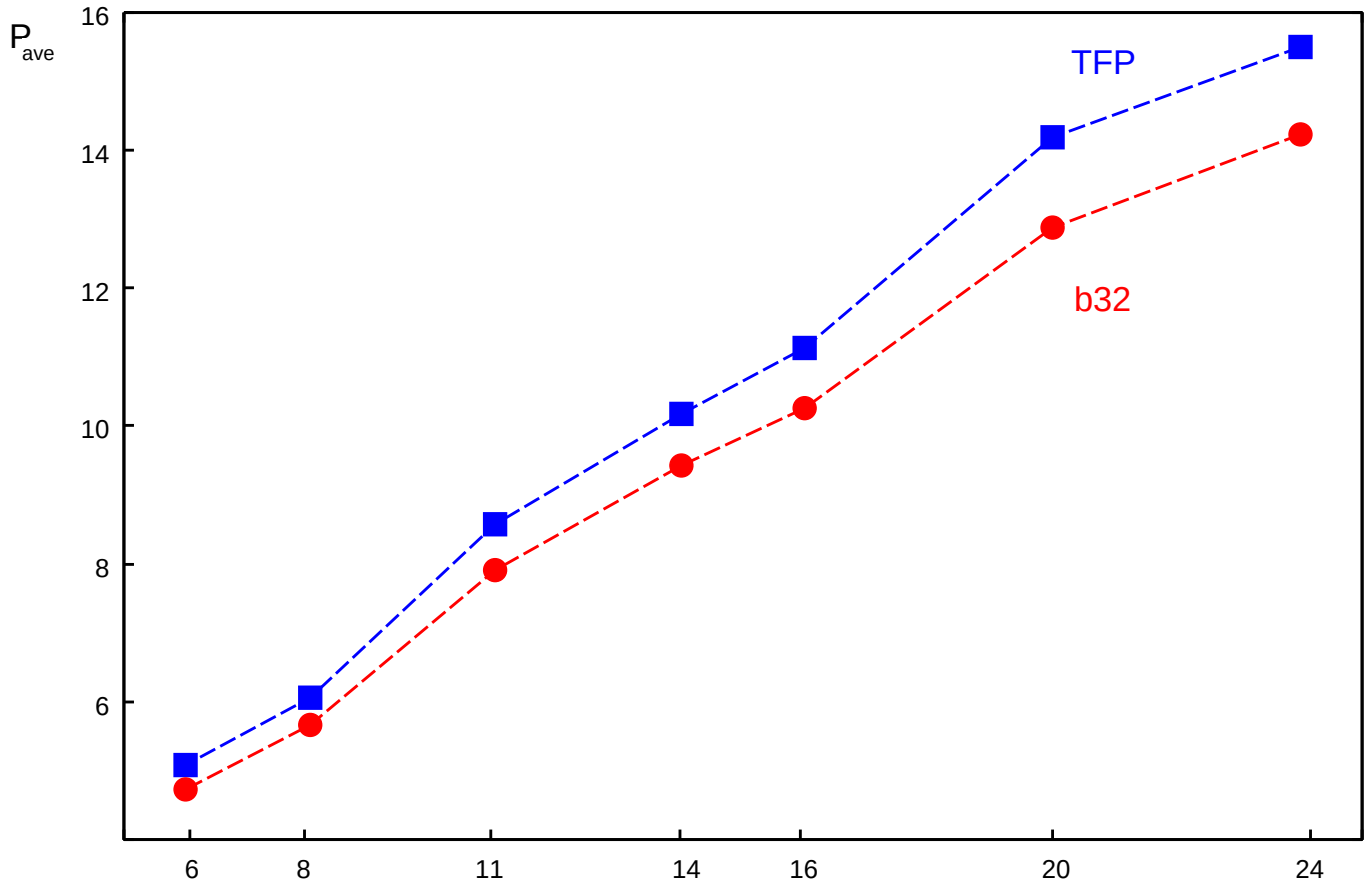
2. $Z = (X \times Y) \times W$



Power Dissipation TFP vs. binary32 Multiplier

P_{ave} under changing significand bitwidth m, $e = 8$

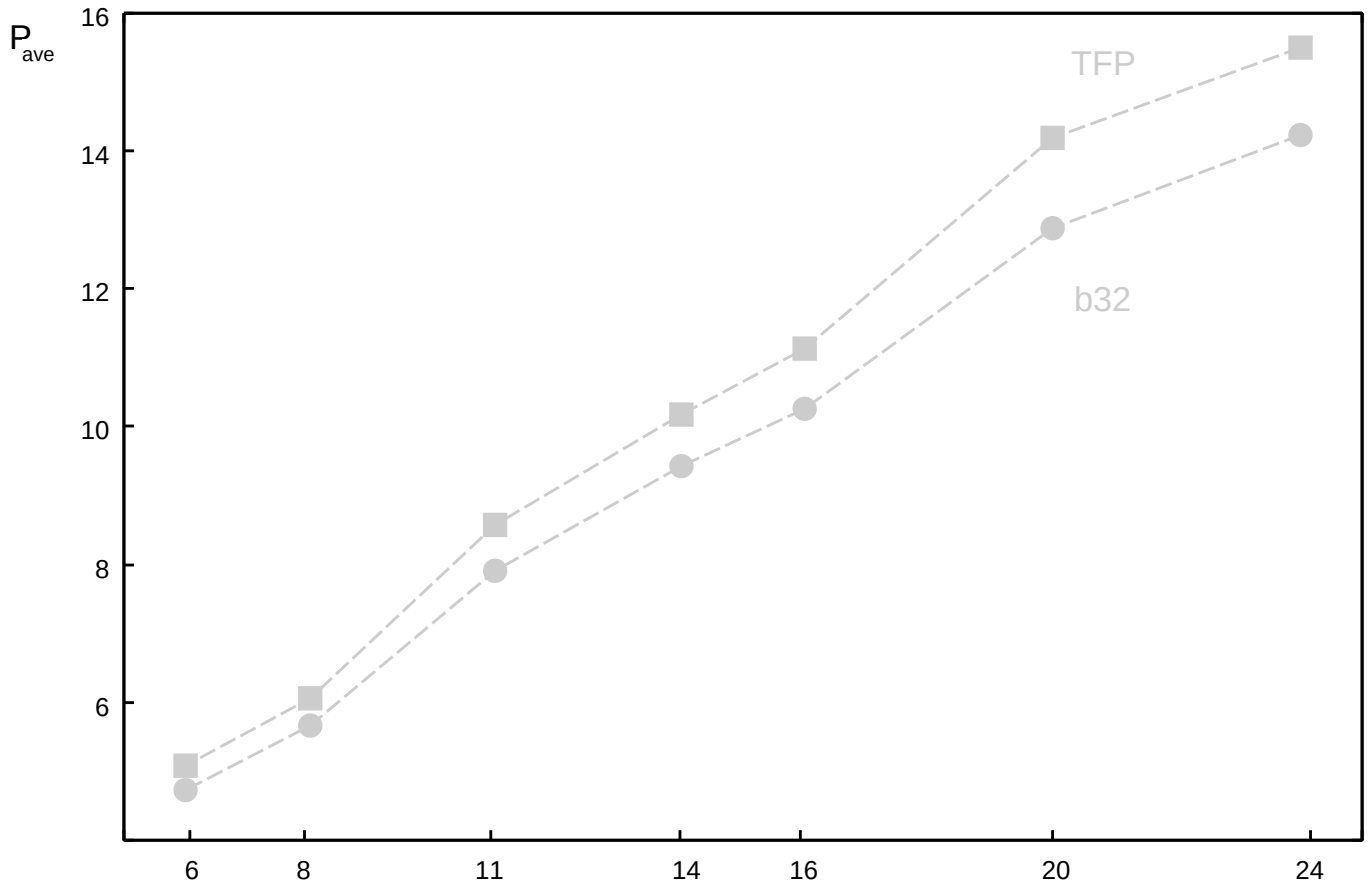
1. $Z = X \times Y$



Power Dissipation TFP vs. binary32 Multiplier

P_{ave} under changing significand bitwidth m, $e = 8$

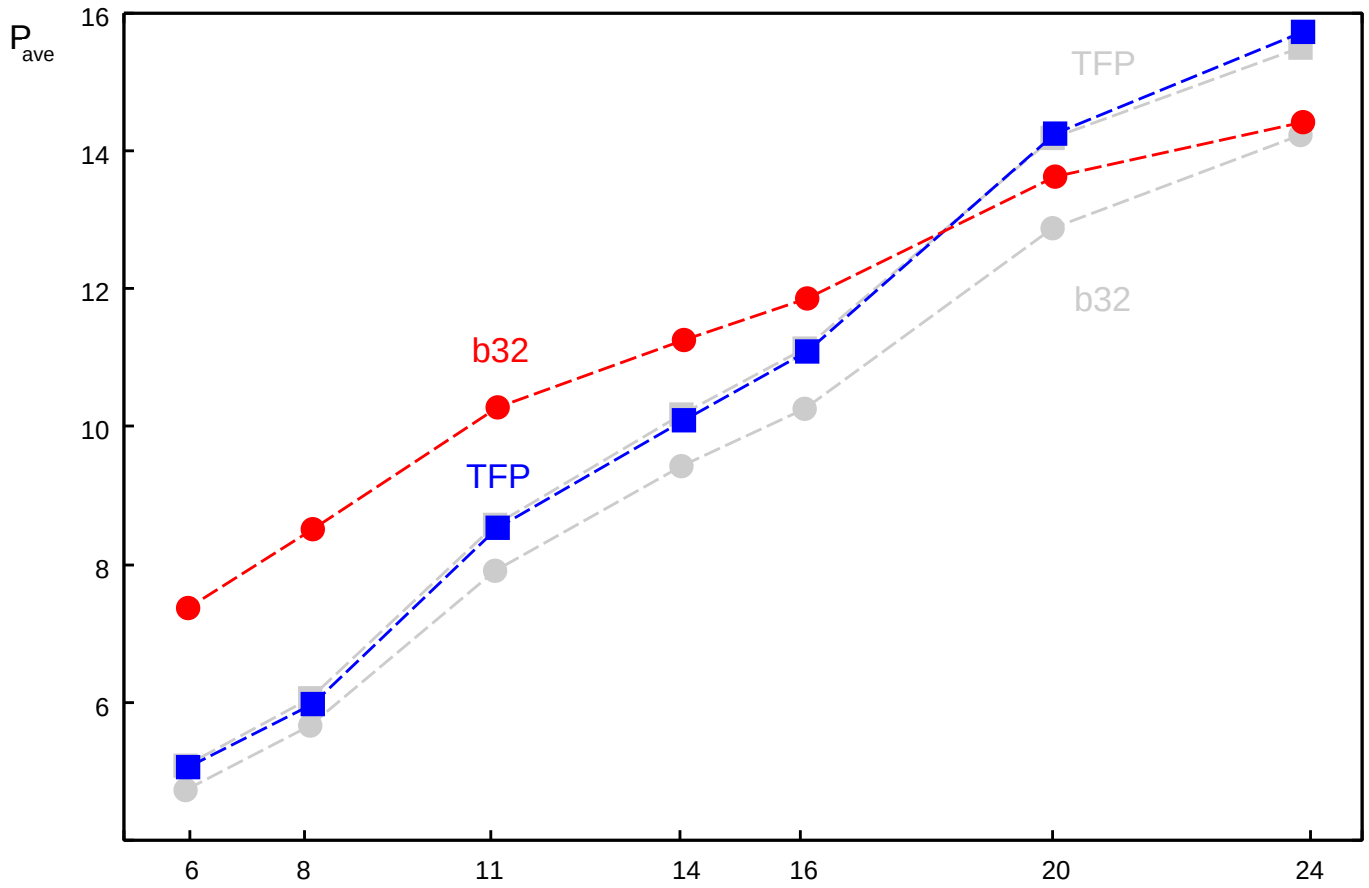
2. $Z = (X \times Y) \times W$



Power Dissipation TFP vs. binary32 Multiplier

P_{ave} under changing significand bitwidth m, $e = 8$

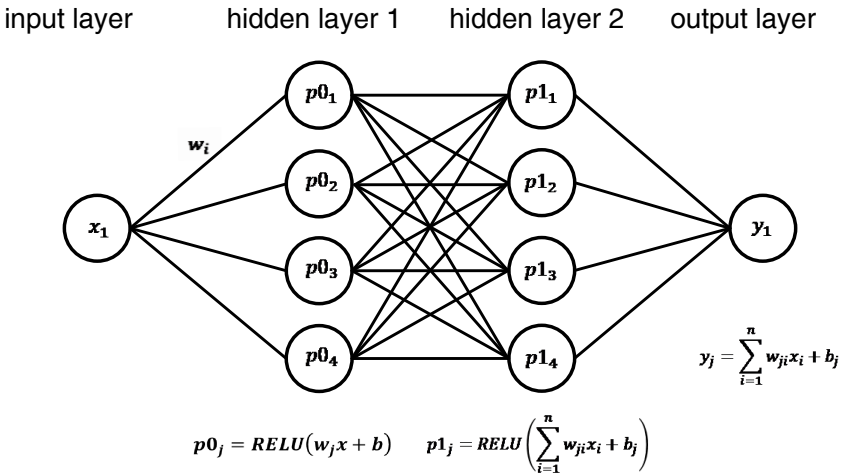
2. $Z = (X \times Y) \times W$



TFP more power efficient than b32 fom $m < 20$

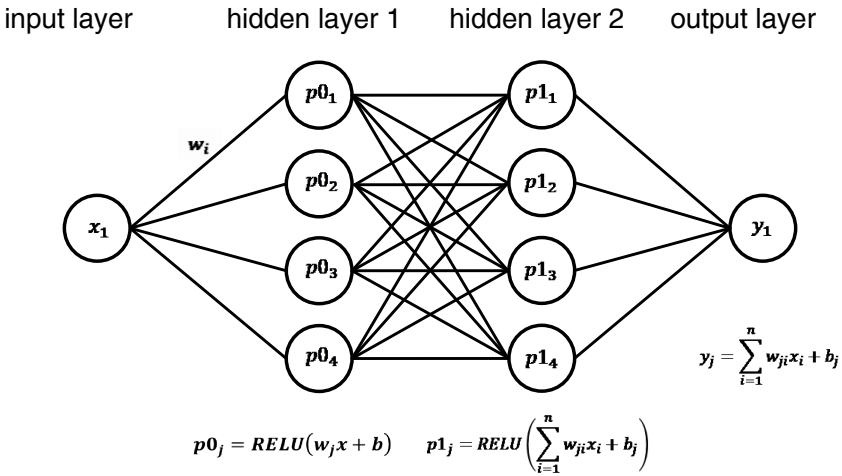
Example: Neural Network

Neural network with two hidden layers (depth=2)

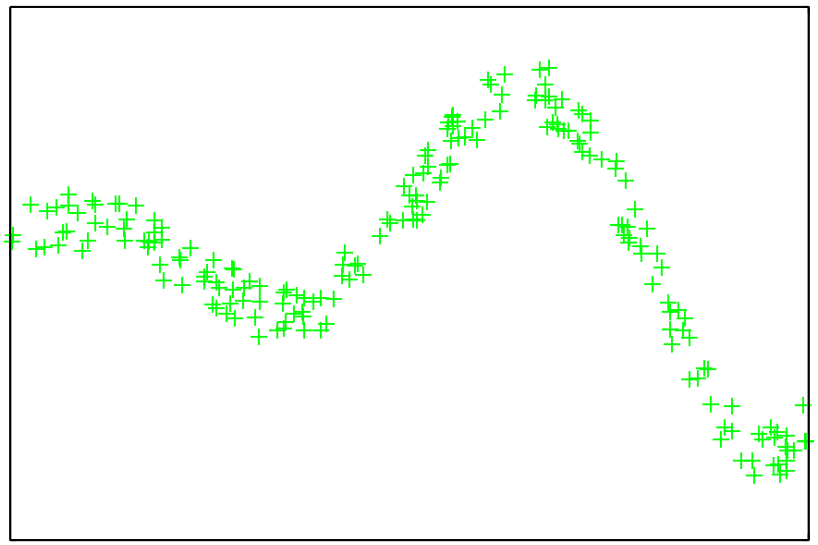


Example: Neural Network

Neural network with two hidden layers (depth=2)

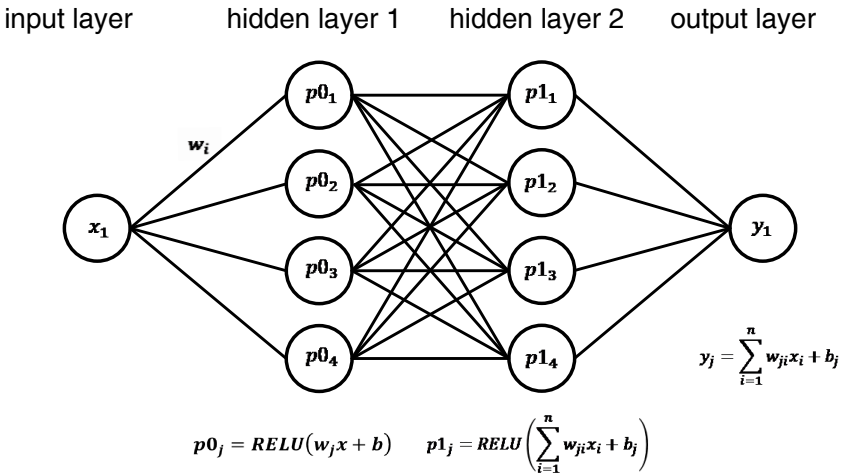


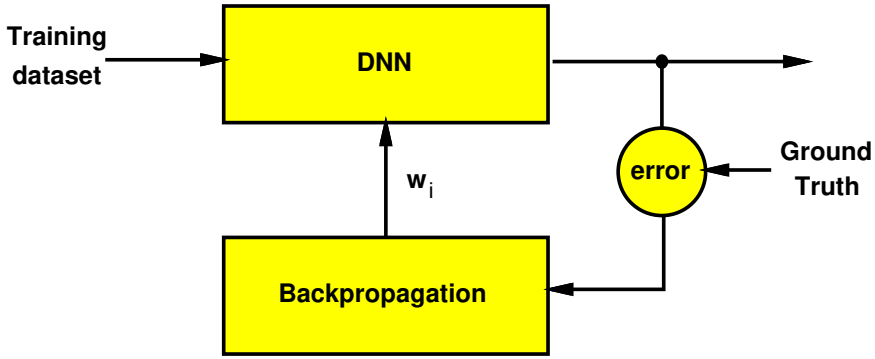
Training/Learning



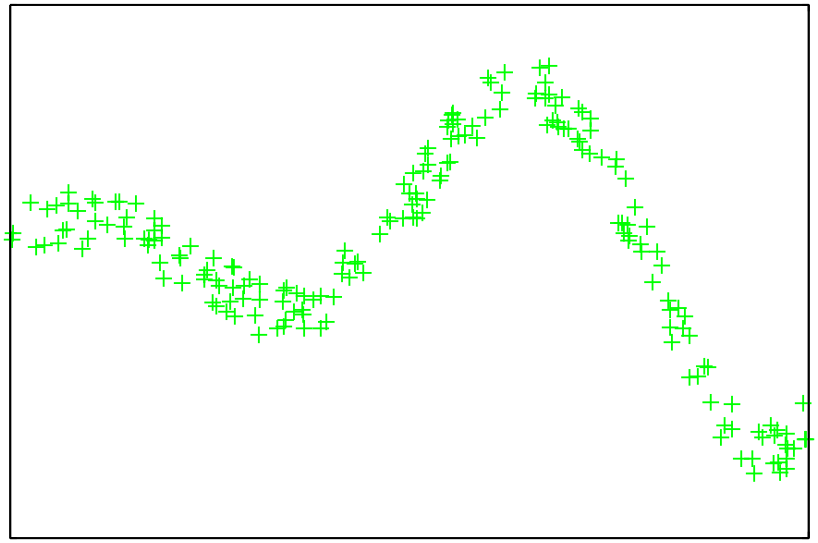
Example: Neural Network

Neural network with two hidden layers (depth=2)



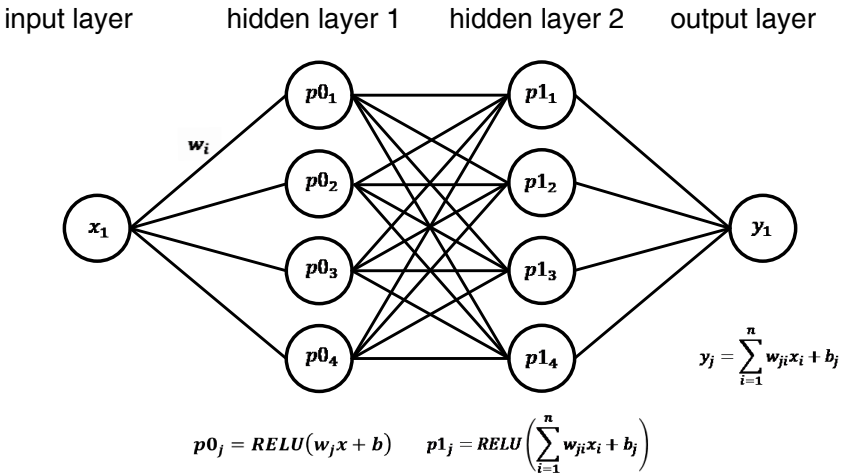


Training/Learning

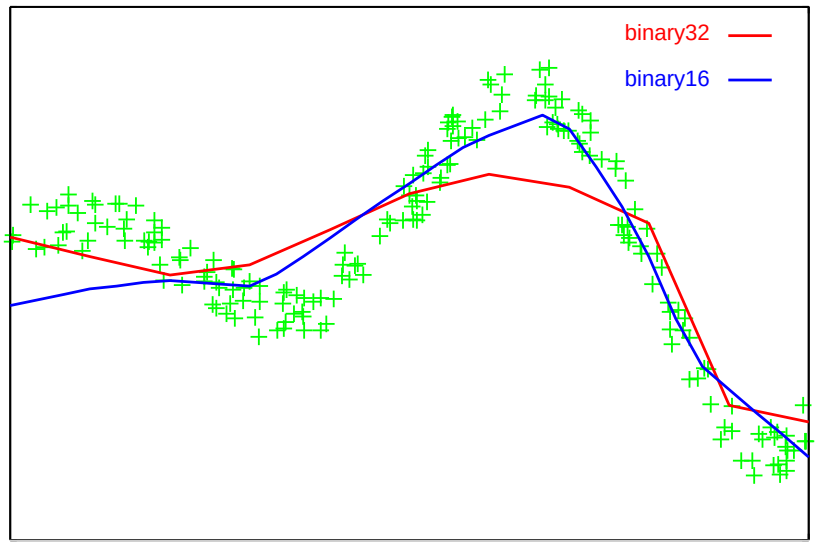


Example: Neural Network

Neural network with two hidden layers (depth=2)



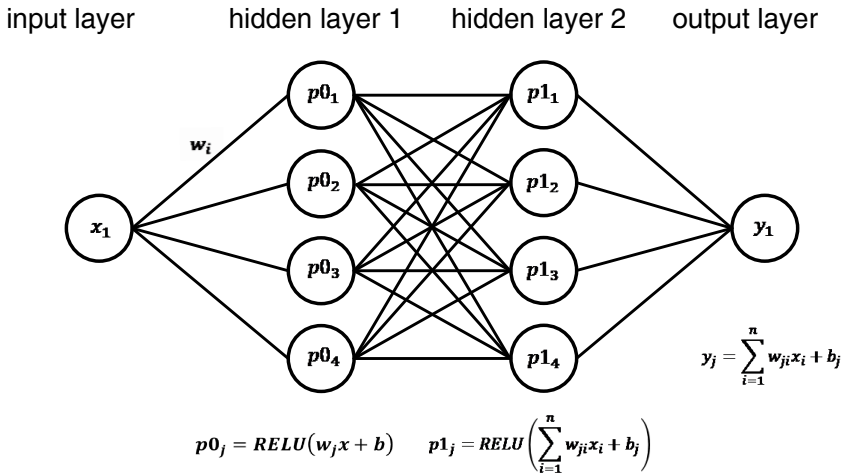
Training/Learning



m	e	ϵ_{ave}	epoch	n_{op}	P_{ave}	E_{Learn}
24	8	0.13	212	6,127	9.49	58,151
20	8	0.13	229	6,618	8.63	57,120
16	8	0.13	214	6,188	7.55	46,719
14	8	0.19	5	145	7.06	1,029
11	5	0.12	5	145	6.46	939
8	5	0.27	9	258	5.61	1,452
6	5	0.27	4	115	5.05	581
				$\times 10^3$	[mW]	[J]

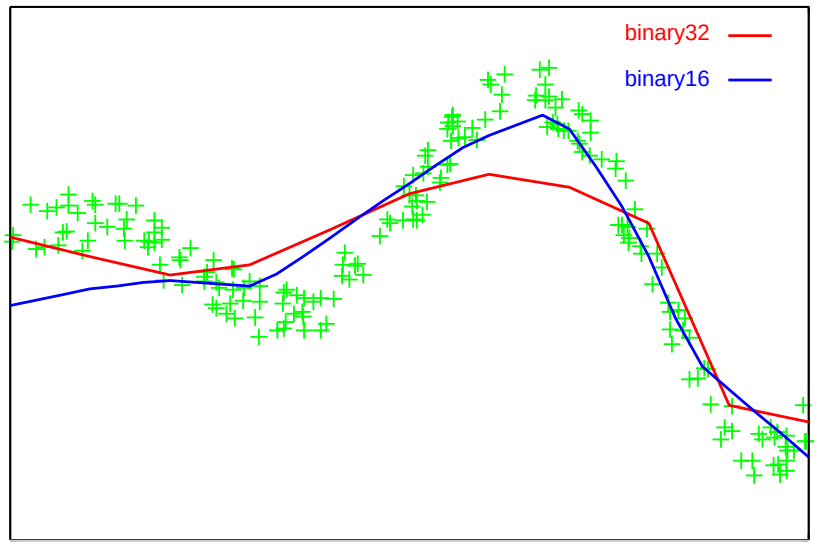
Example: Neural Network

Neural network with two hidden layers (depth=2)



- C simulator $\rightarrow \epsilon_{ave}$, epoch, n_{op}
  $\rightarrow$ test-vectors
- Synopsys VCS $\rightarrow$ switching activity
- Synopsys ICC $\rightarrow P_{ave}$
- $E_{Learn} = P_{ave} \times n_{op} \times T_{clk}$

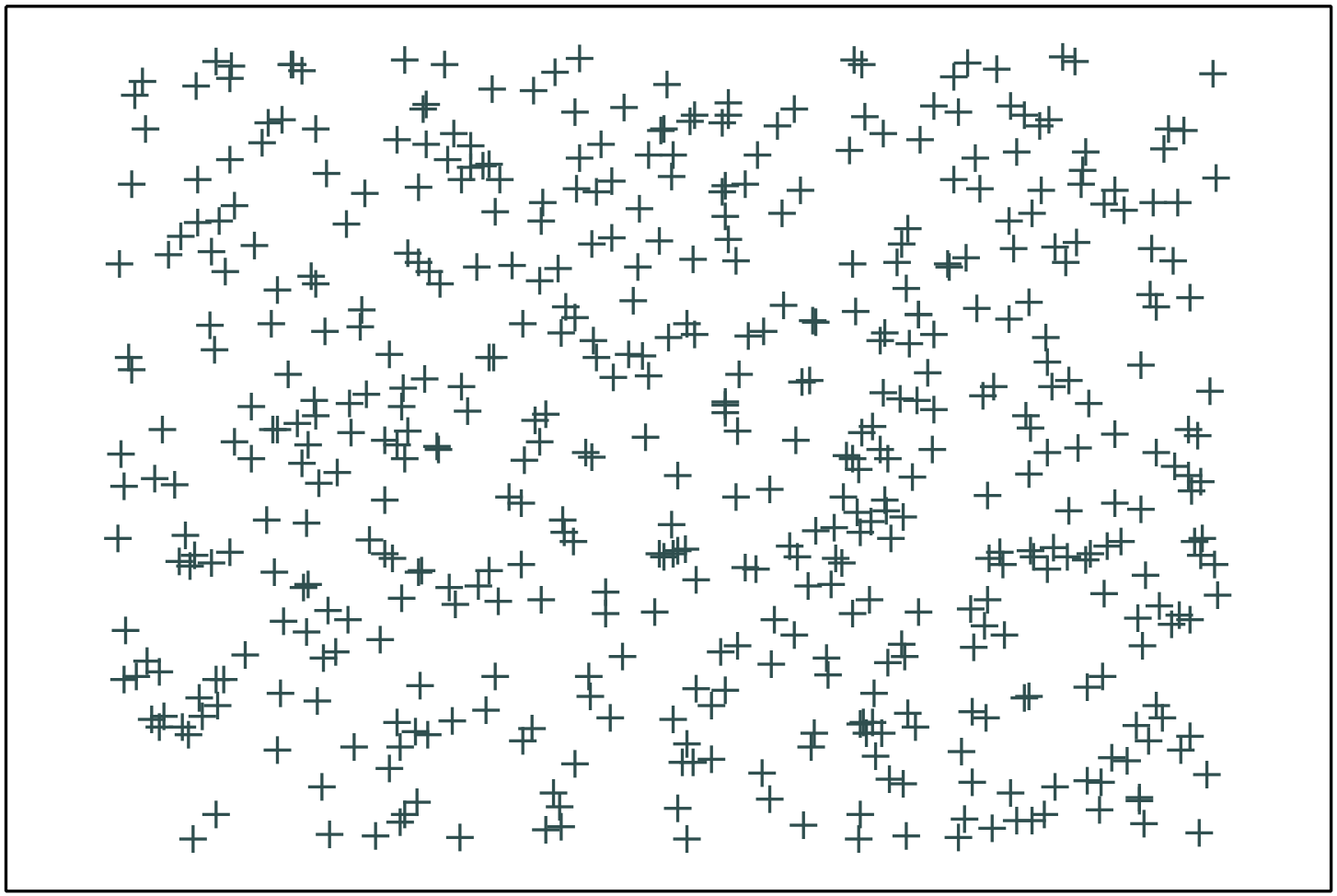
Training/Learning



m	e	ϵ_{ave}	epoch	n_{op}	P_{ave}	E_{Learn}
24	8	0.13	212	6,127	9.49	58,151
20	8	0.13	229	6,618	8.63	57,120
16	8	0.13	214	6,188	7.55	46,719
14	8	0.19	5	145	7.06	1,029
11	5	0.12	5	145	6.46	939
8	5	0.27	9	258	5.61	1,452
6	5	0.27	4	115	5.05	581
				$\times 10^3$	[mW]	[J]

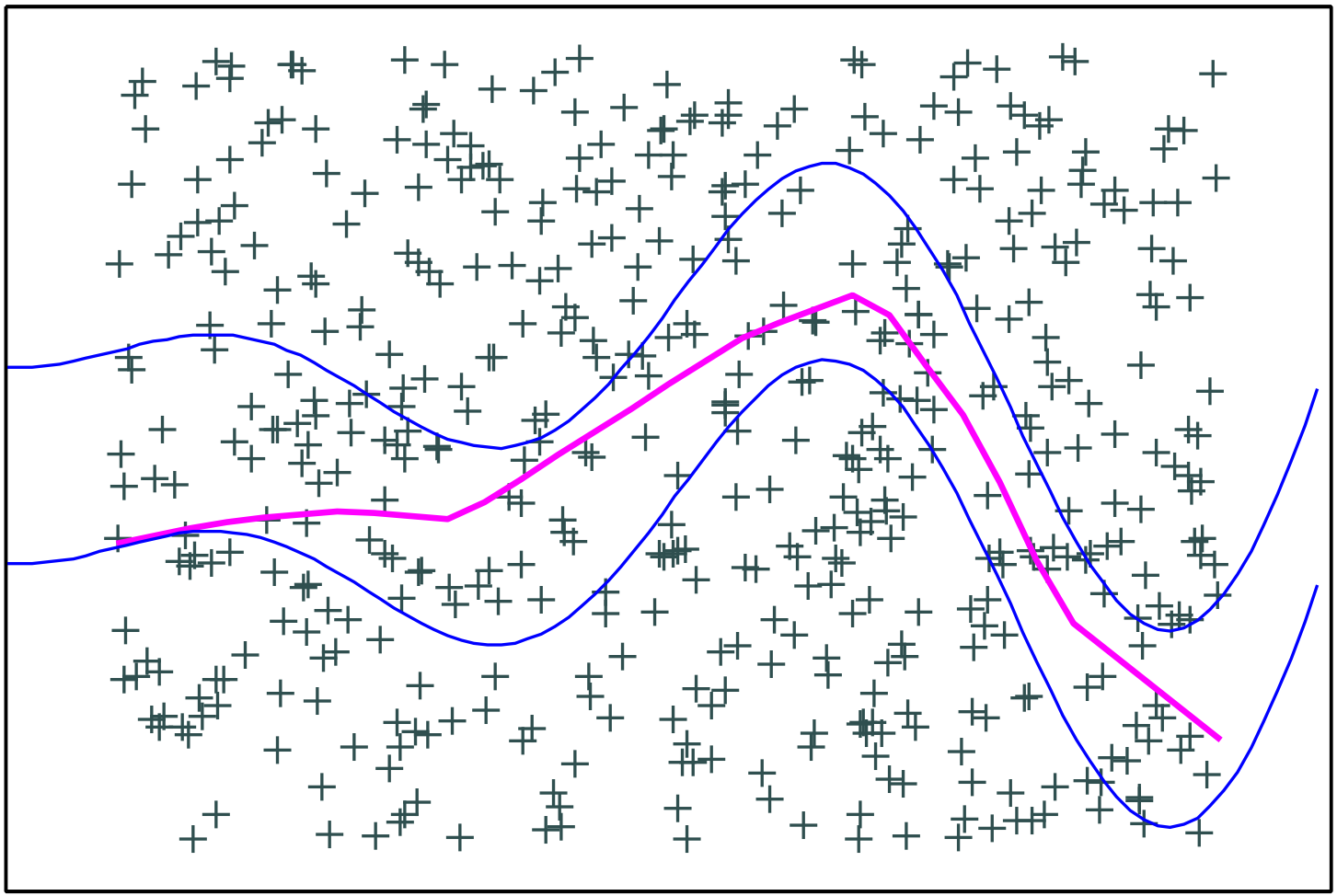
Example: Neural Network – Inference

Approximation Error (*binary32*)



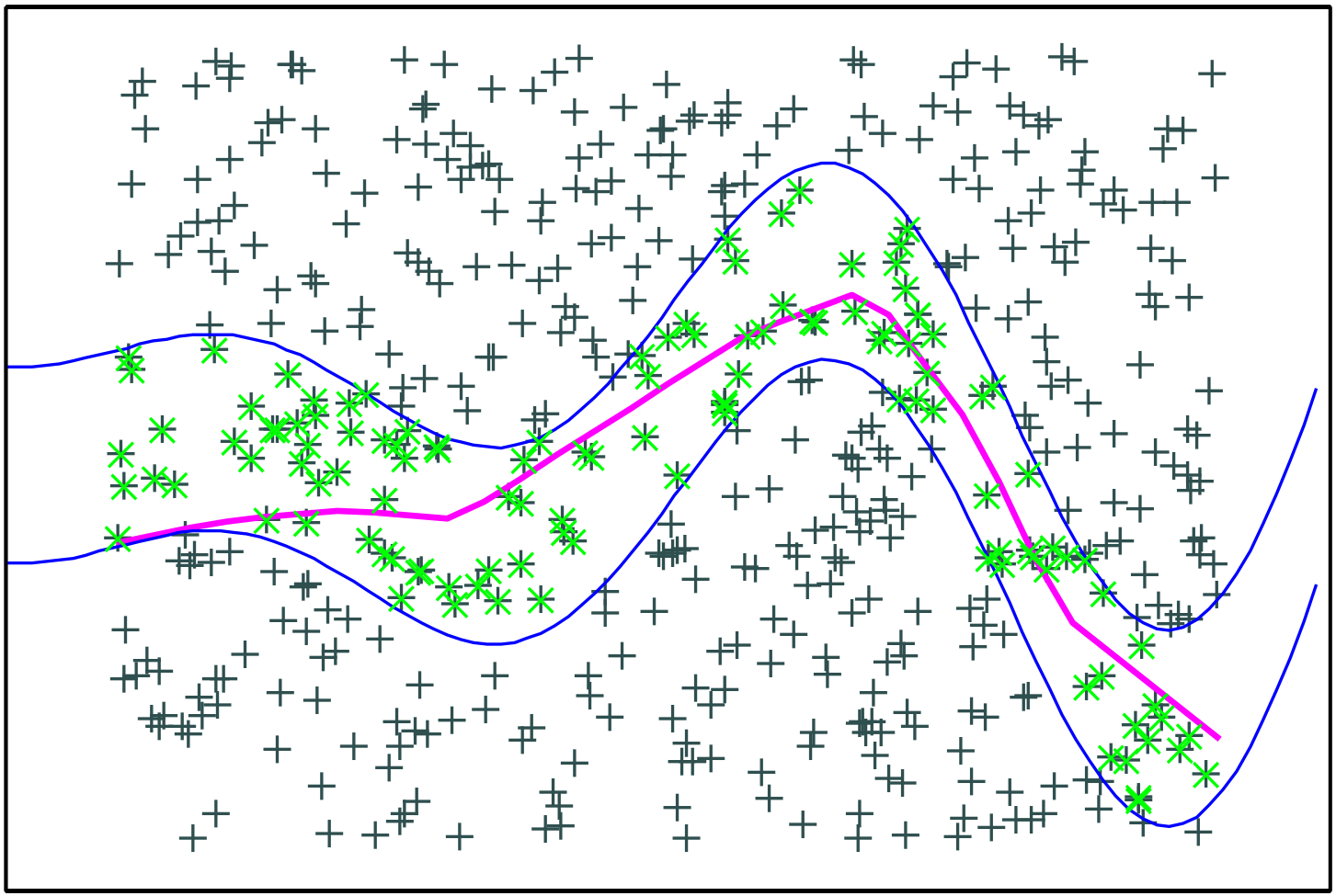
Example: Neural Network – Inference

Approximation Error (*binary32*)



Example: Neural Network – Inference

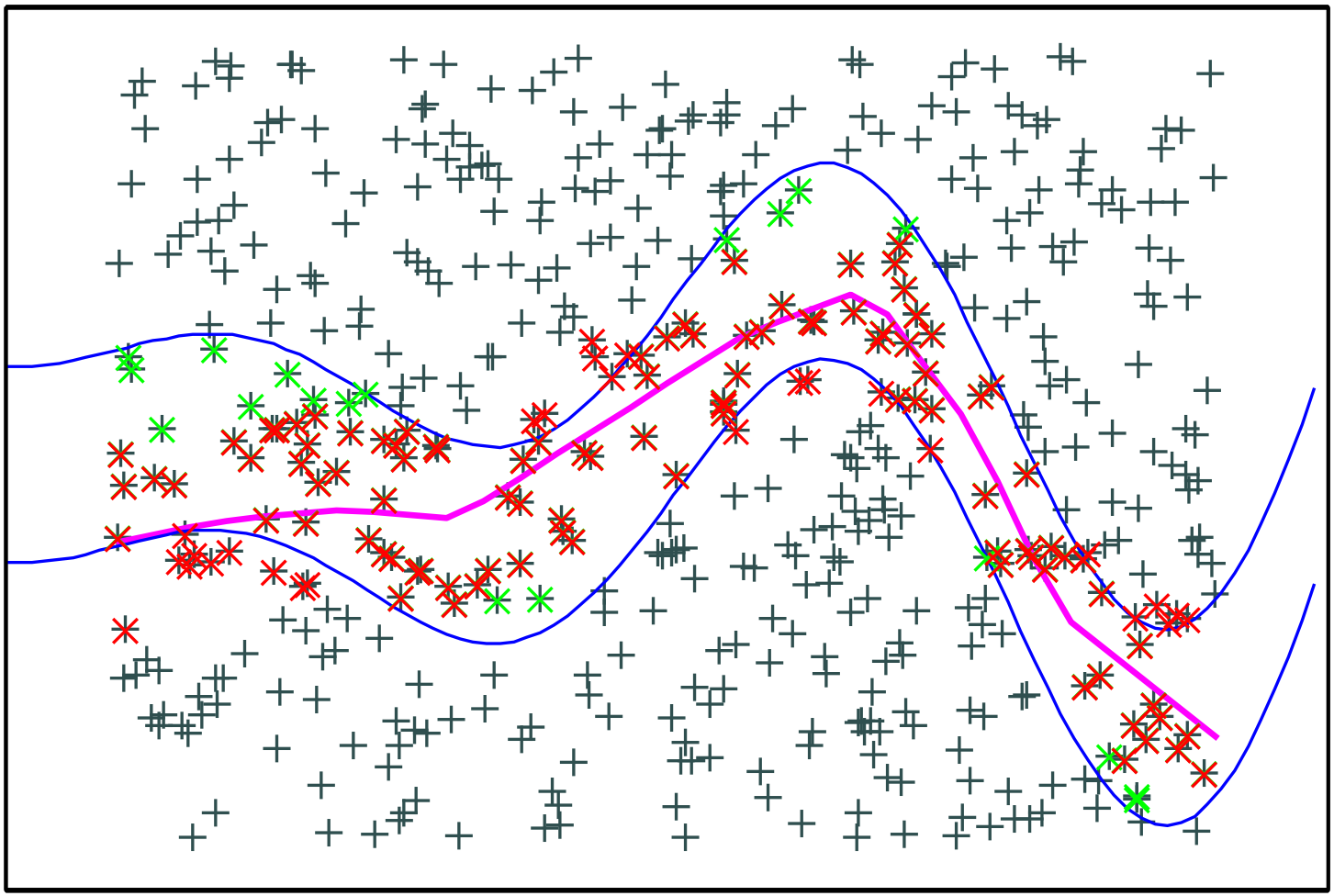
Approximation Error (*binary32*)



- Ideal
  $n = 120$

Example: Neural Network – Inference

Approximation Error (*binary32*)



- Ideal
  $n = 120$
- NN approx.
  $n = 129$

$(* \cap *) \rightarrow 97$

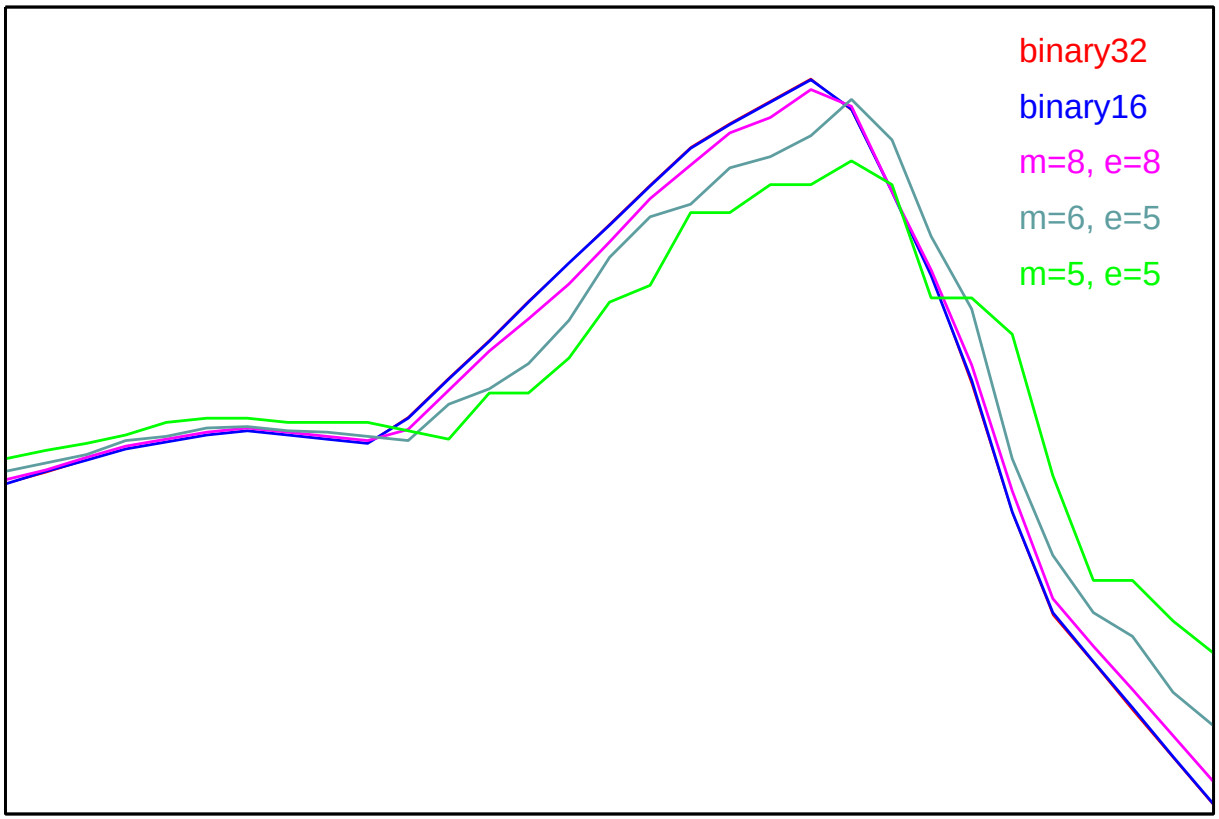
Miss-classified

$* \rightarrow 23$ (in)

$* \rightarrow 32$ (out)

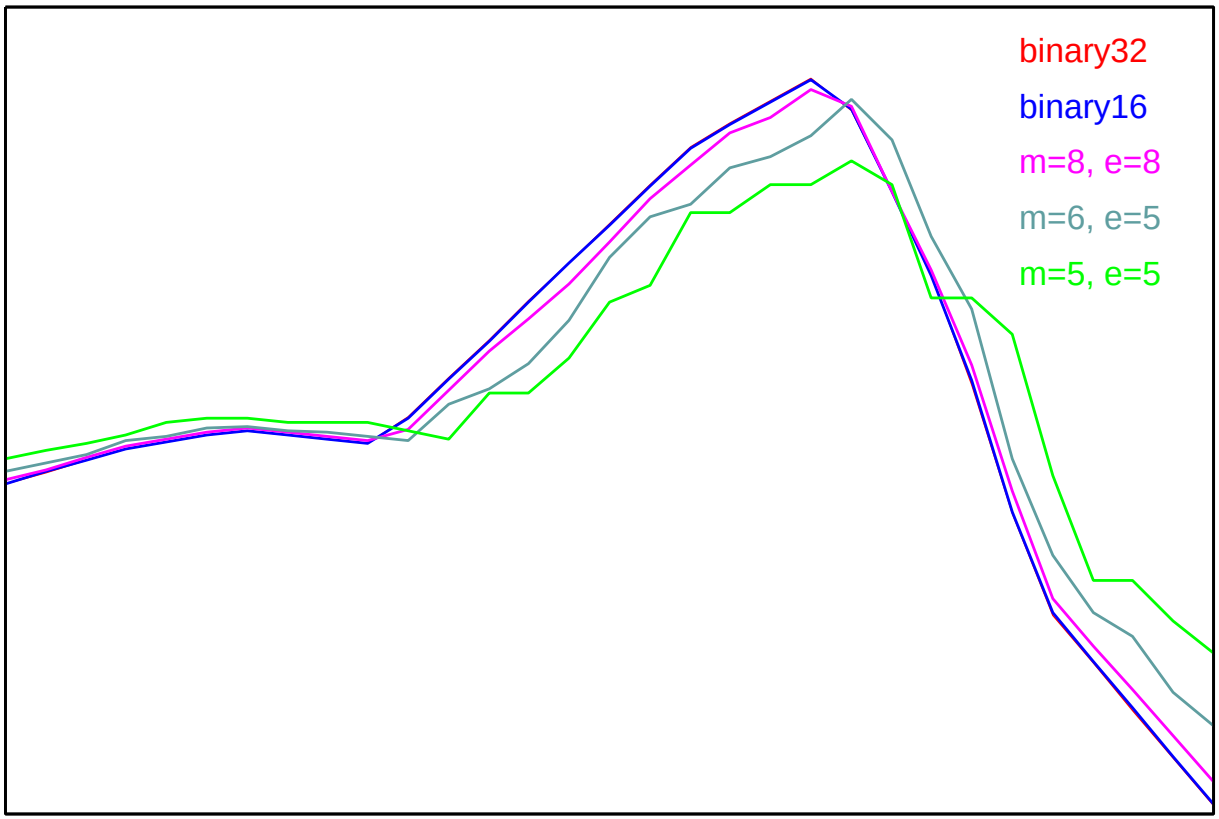
Example: Neural Network – Inference

Quantization Error / Reduced Precision



Example: Neural Network – Inference

Quantization Error / Reduced Precision



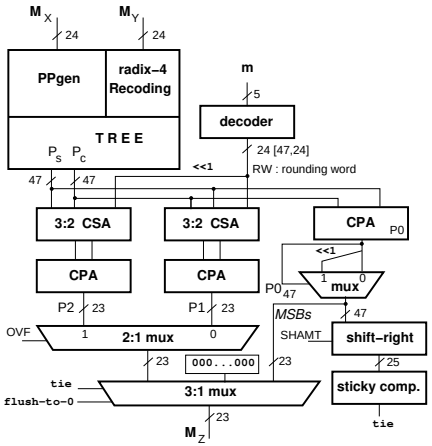
m	e	Np_{TOT}	$(*\cap*)$	$*$	$*$
24	8	129	–	–	–
20	5	129	129	0	0
11	5	128	127	2	1
9	5	129	127	2	2
8	8	131	124	5	7
8	5	131	124	5	7
7	5	132	121	8	11
6	5	129	116	13	13
5	5	129	97	32	32

$*$ points in b32 and not in (m,e)

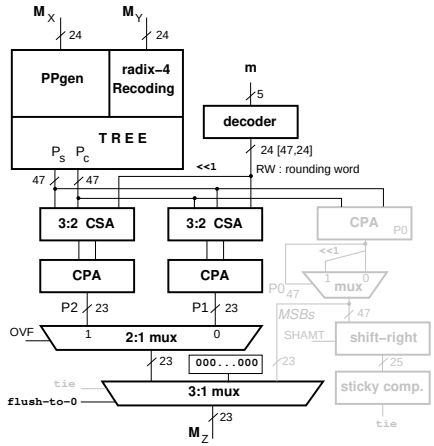
$*$ points in (m,e) and not in b32

Improvements

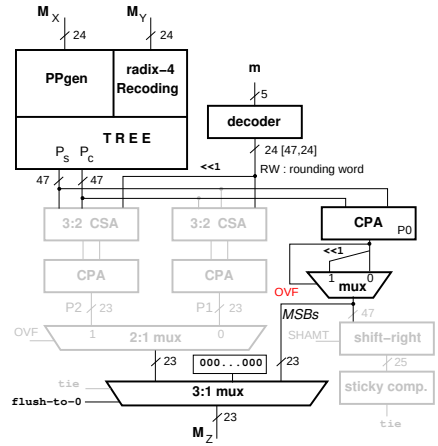
- Use compound adder in speculative rounding
- Implement programmable rounding modes



RTNE



RTN



RTZ

- RTNE for *binary32* and *binary16* only
  $\Rightarrow$ replace barrel shifter with multiplexer
- Split the unit in two when $m < 12$
  $\Rightarrow$ two *binary16* in parallel

Conclusions and Future Work

Tunable Floating-Point multiplier for accelerator use

- correct rounding for the selected precision
- reduced power dissipation

Future Work

- Currently testing a TFP-adder (2-stage)
- Integrating multiplication and addition in FMA
- Division

Additional Slides

TFP-mult vs. b32-mult: Power Dissipation

Average power dissipation for TFP-mult and b32-mult (RTN)

P_{ave}	$Z = X \times Y$			$Z = (X \times Y) \times W$		
m	TFP-mult	b32-mult	ratio	TFP-mult	b32-mult	ratio
24	15.51	14.24	1.09	15.72	14.42	1.09
20	14.19	12.89	1.10	14.26	13.63	1.05
16	11.14	10.26	1.09	11.10	11.86	**0.94**
14	10.17	9.42	1.08	10.09	11.26	**0.90**
11	8.58	7.91	1.08	8.54	10.28	**0.83**
8	6.05	5.67	1.07	5.98	8.52	**0.70**
6	5.08	4.72	1.08	5.07	7.36	**0.69**

P_{ave} [mW] measured at 1 GHz.

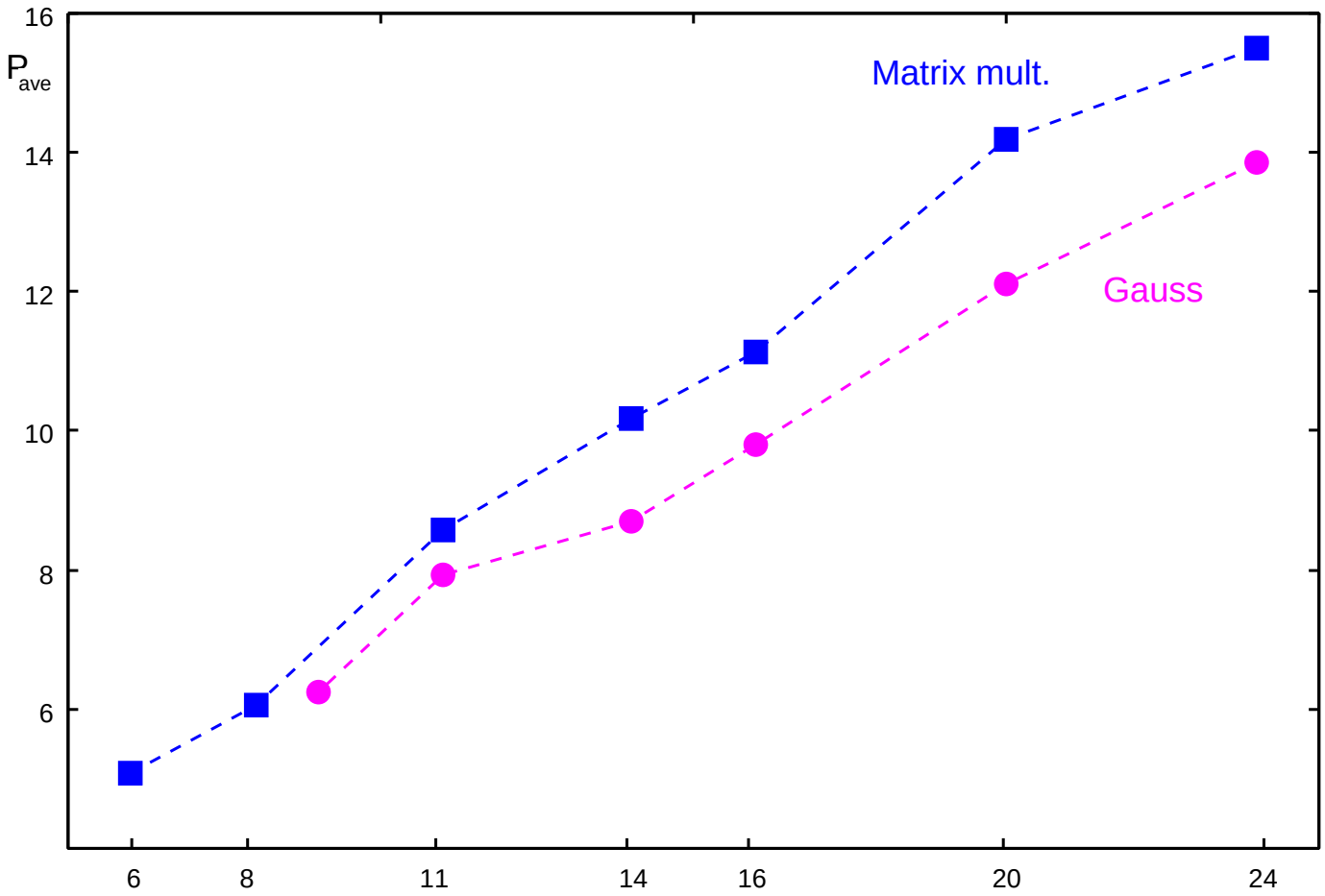
Average Power Dissipation for TFP-mult

Average power dissipation for TFP-mult (RTN) as m scales ($e = 8$)

	Matrix multiplication		Gauss elimination	
m	P_{ave} [mW]	ratio	P_{ave} [mW]	ratio
24	15.51	1.00	13.85	1.00
20	14.19	0.91	12.11	0.87
16	11.14	0.72	9.80	0.71
14	10.17	0.66	8.69	0.63
11	8.58	0.55	7.93	0.57
9			6.24	0.45
8	6.05	0.39		
6	5.08	0.33		

P_{ave} measured at 1 GHz.

TFP-mult Trends

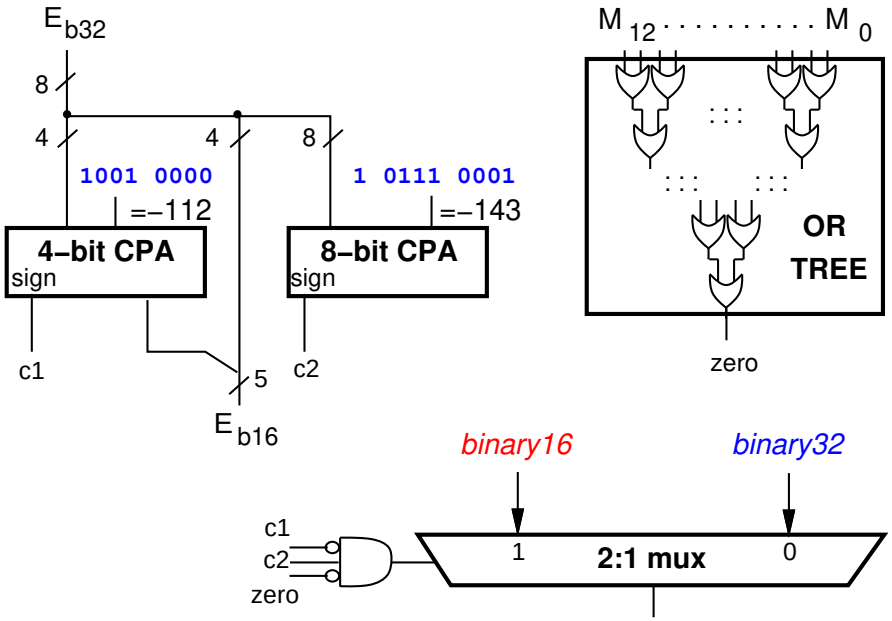


Trends of average power dissipation for TFP-mult (RTN) as m scales ($e = 8$)

Hardware for *binary32* to *binary16* Reduction

/* range checking (exponent) */
$E_{b16} = E_{b32} - B_{b32} + B_{b16} = E_{b32} - 112$
// must be positive

/* check lower bound (exponent) */
$E_{b16} - E_{max(5b)} < 0$
$\rightarrow \quad E_{b32} - 112 - 31 = E_{b32} - 143 < 0$



/* check significand for non-zero bits */
zero = 0;
for i=0 to 12 **do**
  zero = zero OR $M_{b32}(i)$;
end for

if $((E_{b16} > 0)\text{AND}(E_{b32} - 143 < 0)\text{AND}(zero = 0))$ **then**
  reduce to binary16
else
  keep binary32
end if