56TH EDITION IEEE ISCAS 2023

IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS

May 21-25, 2023 || Monterey, California

PDPU: An Open-Source Posit Dot-Product Unit for Deep Learning Applications

Qiong Li, Chao Fang, Zhongfeng Wang(✉)

ICAIS Lab, School of Electronic Science and Engineering, Nanjing University, China

Outline

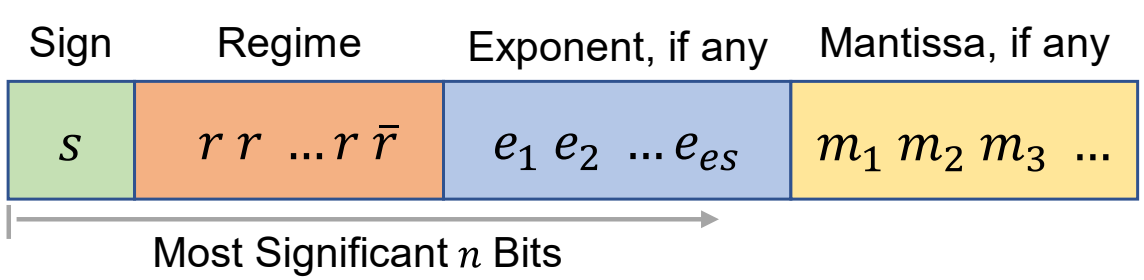
- Backgrounds & Motivations
- Overall Architecture
- Configurable PDPU Generator
- Experimental Results
- Conclusion

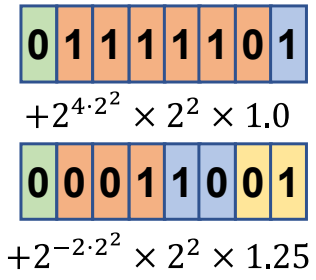
Outline

- **Backgrounds & Motivations**
- Overall Architecture
- Configurable PDPU Generator
- Experimental Results
- Conclusion

Background: Posit Format

- **Many Fields have Benefited from Posit Format**
  - **Irregular graphs processing** [N. Shah et al., JSSC'22]
  - **Posit enabled RISC-V core** [S. Tiwari et al., TACO'21]
  - **Weather forecasting** [M. Klower et al., JAMES'20]
  - **Deep Learning**
    - Posit **arithmetic units** [H. Zhang et al., ISCAS'19]
    - Posit-based **DNN training or inference evaluation** [J. Lu et al., TC'20]
    - Posit-based **DNN processor or accelerator** [Y. Wang et al., TCAS-I'22]
- Draw Attention from Institutions such as **NUS**, **UCM**, **IIT**, **THU**, **ULisboa,** etc.

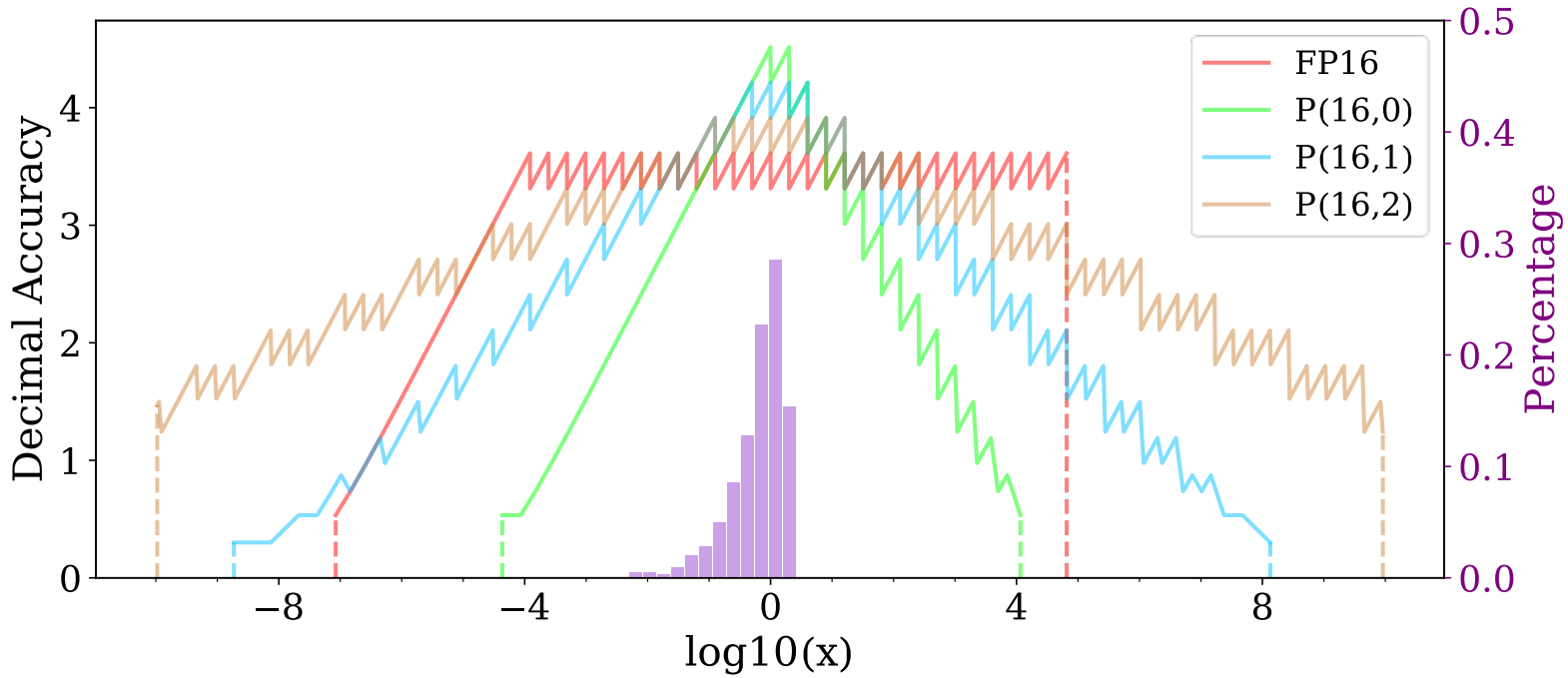




- Posit is defined by word size (n) and exponent size (es), i.e., P(n, es)
- Four fields: sign, **regime**, exponent, mantissa

Background: Posit Format (Cont'd)

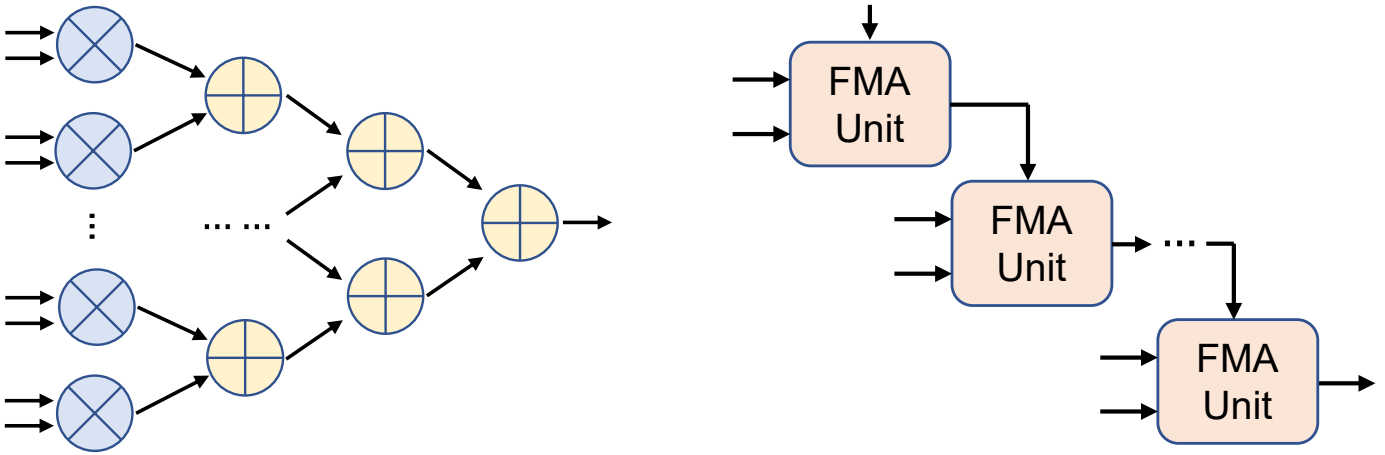
- **Posit: Promising Alternative to IEEE-754** for Deep Learning Applications
  - Better trade-off between dynamic range & accuracy
  - Simplified exceptions handling
  - Symmetrical tapered accuracy



- **Tapered accuracy of posit fits the DNN data distribution**

Background: Dot-Product

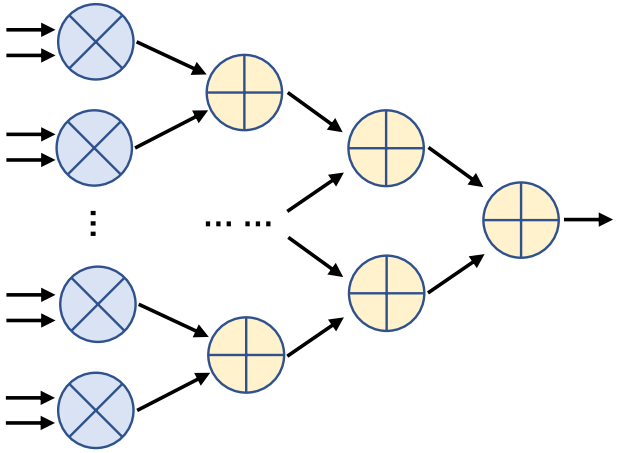
- Dot-product Operations, **Dominant in Deep Neural Network** (DNN)
- Existing Posit-Based Dot-Product Implementation:
  - **Combination of multipliers and an adder tree**
    - [ICCD'18], [IEEE Access'19], [ISCAS'20], [TCAS-II'20], [ISCAS'21], …
  - **Cascaded fused multiply-add units**
    - [ISCAS'19], [SOCC'19], [ICCD'21], [TCAS-II'22], …

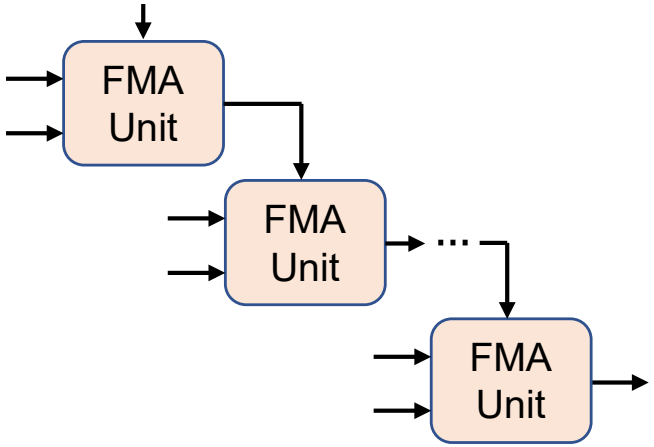


➢ **Existing discrete dot-product architecture** implemented by (a) multipliers and adders or (b) FMA units

Background & Motivations

- **Challenges** of Existing Discrete Implementation
  - **Extensive redundant operations** → high latency and hardware overhead
  - **Frequent hardware rounding** → precision loss
  - **Inflexible design** → limited application scenarios
  - **Lack support for mixed-precision arithmetic in DNNs** → limited computational efficiency





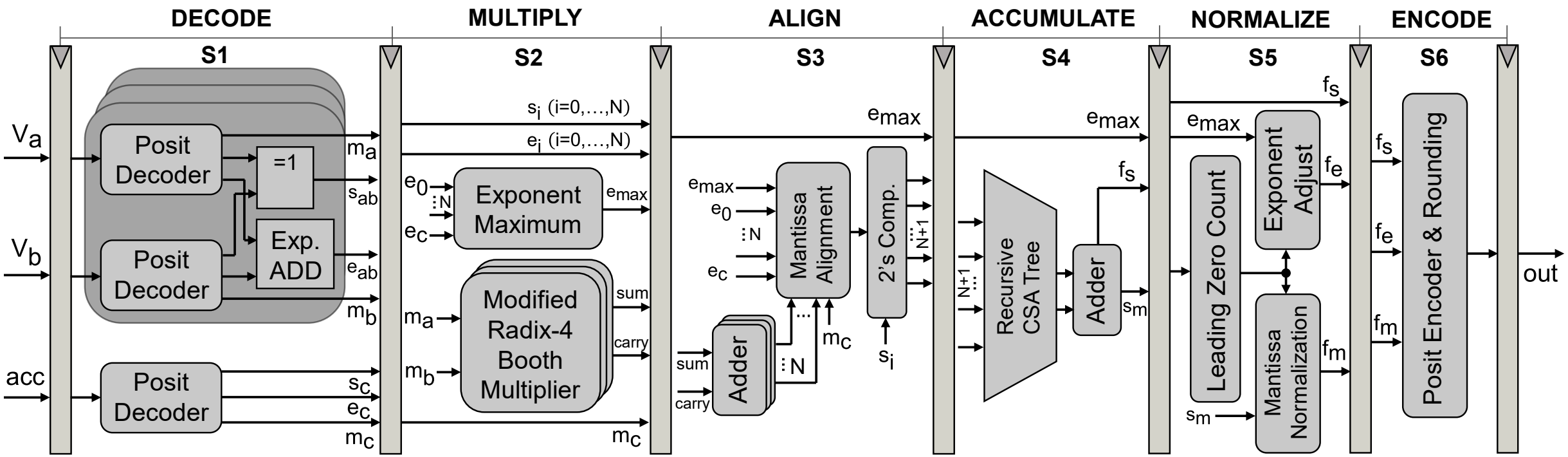
Fused, mixed-precision, and highly configurable posit-based dot-product unit

Outline

- Backgrounds & Motivations
- **Overall Architecture**
- Configurable PDPU Generator
- Experimental Results
- Conclusion

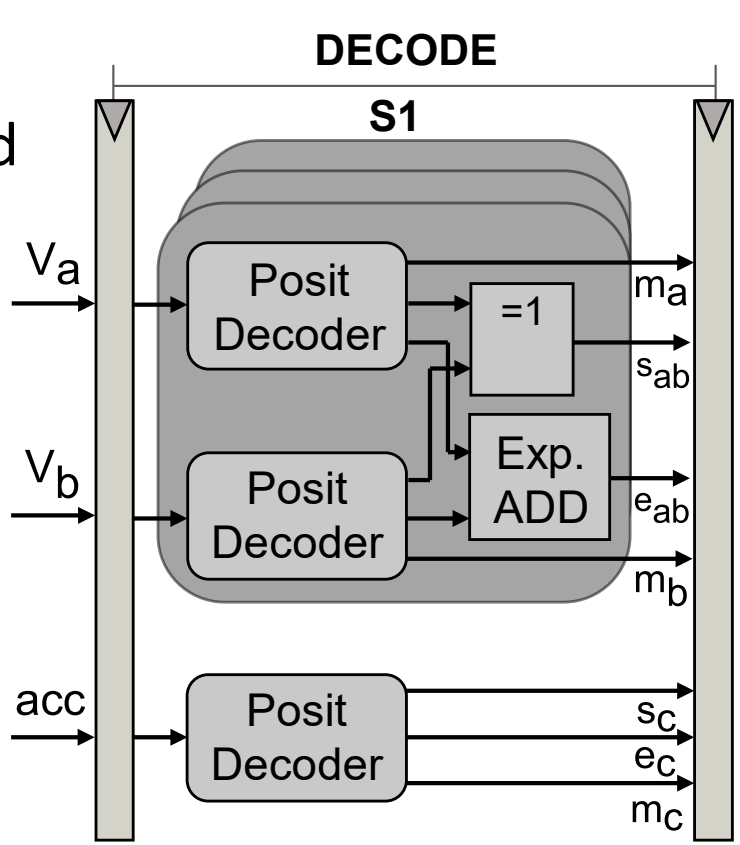
Overall Architecture

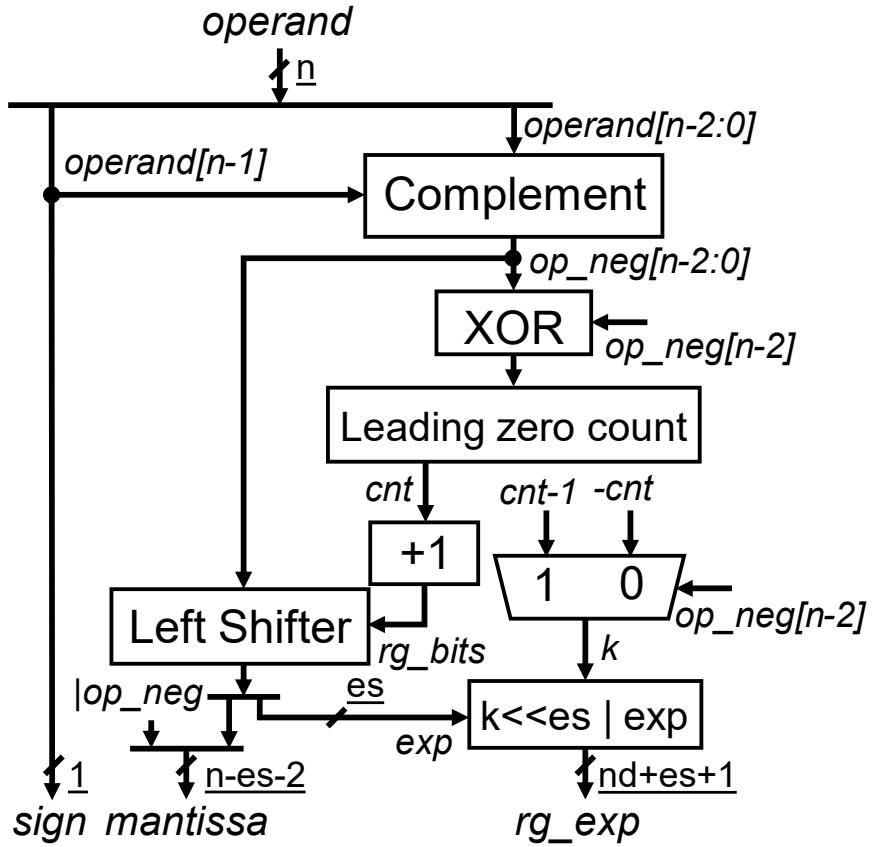
- $out = acc + V_a \times V_b = acc + a_0 \cdot b_0 + a_1 \cdot b_1 + \cdots + a_{N-1} \cdot b_{N-1}$
- Fine-Grained **6-stage Pipeline**



Architecture: Pipeline S1 – Decode

- **S1: Decode**
  - **Decoding**: extract valid components of inputs
    - sign (s)
    - rg_exp (e)
    - mantissa (m)
  - **Sign handling**
    - XOR → s_{ab}
  - **Exponent handling**
    - Addition → e_{ab}

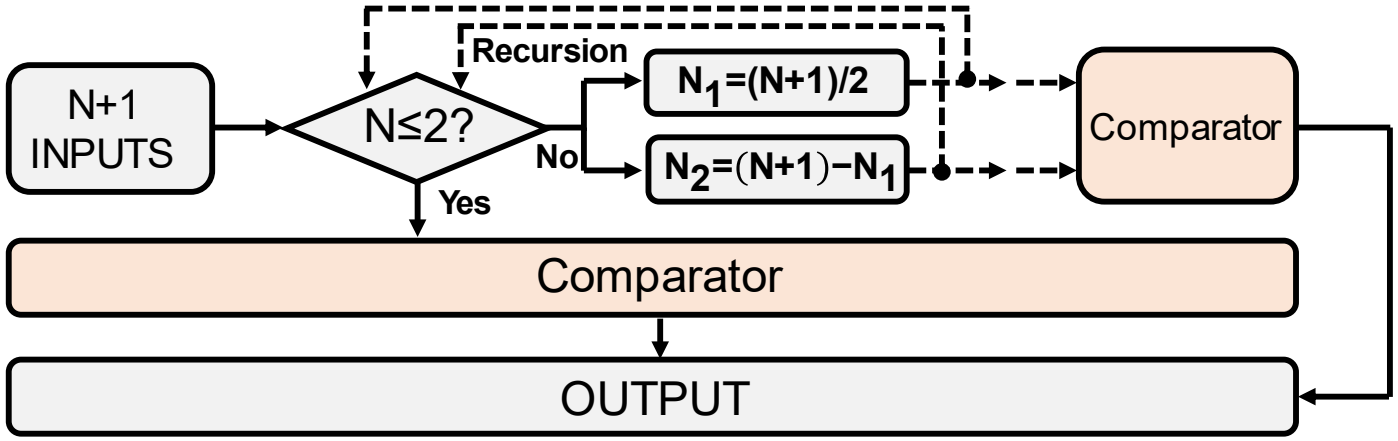


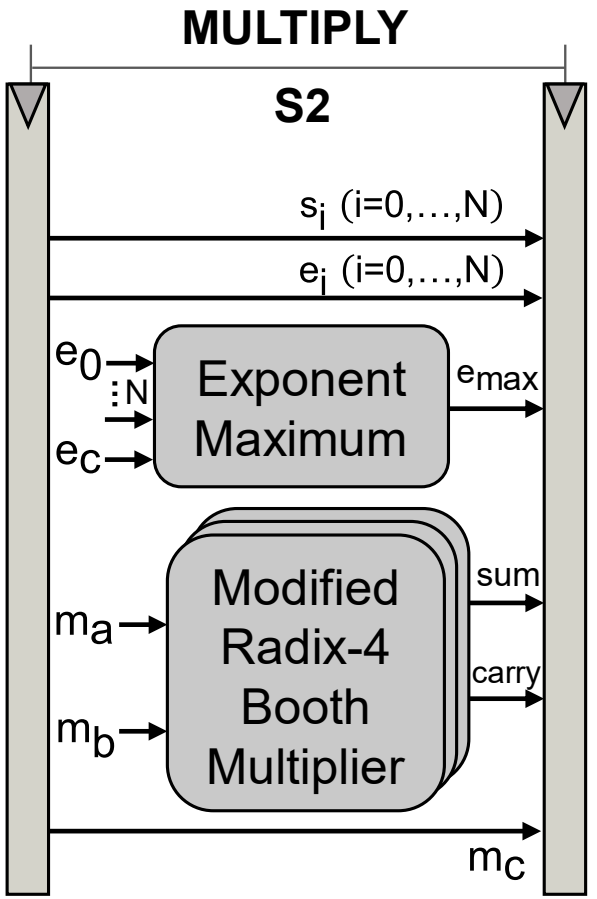


➢ Architecture of posit decoder

Architecture: Pipeline S2 – Multiply

- S1: Decode
- **S2: Multiply**
  - **Exponent comparison**
    - Comparator tree → e_{max}
  - Mantissa multiplication

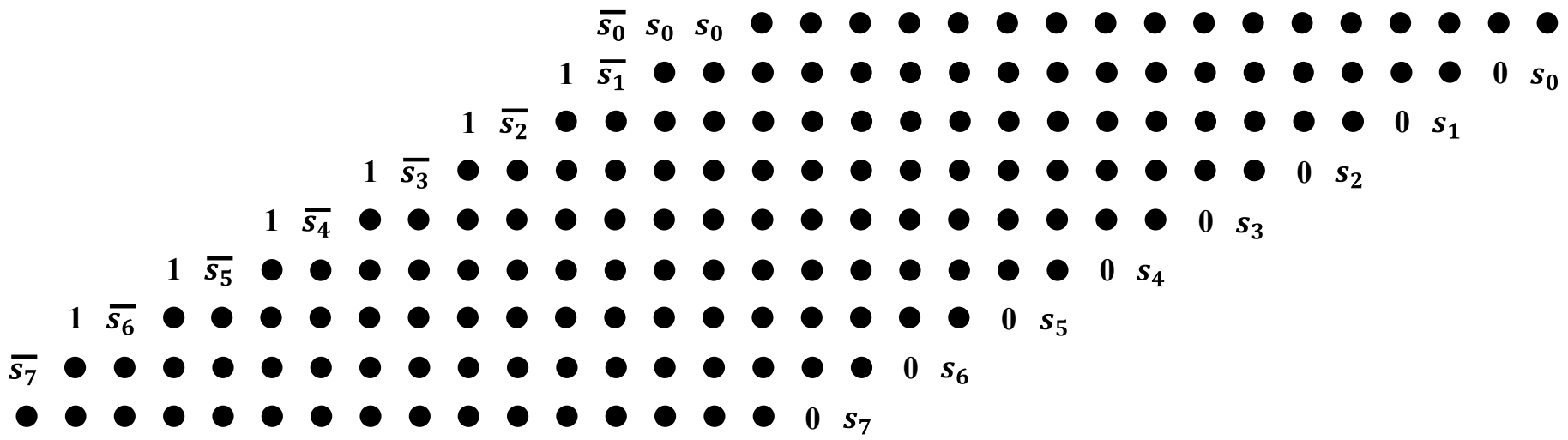


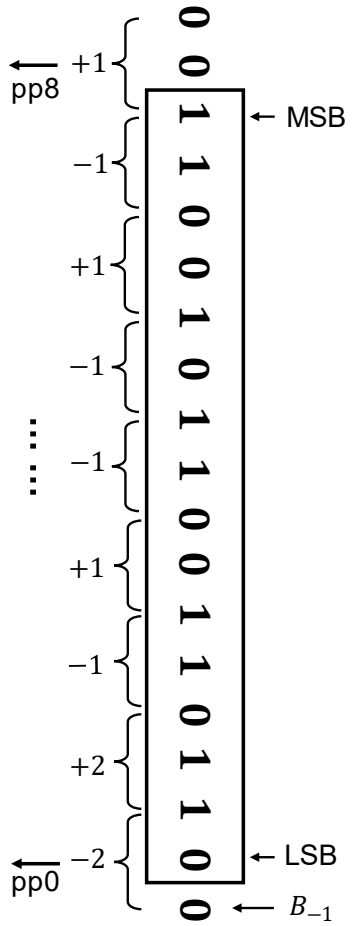


➢ Recursive design method of comparator tree

Architecture: Pipeline S2 – Multiply (Cont'd)

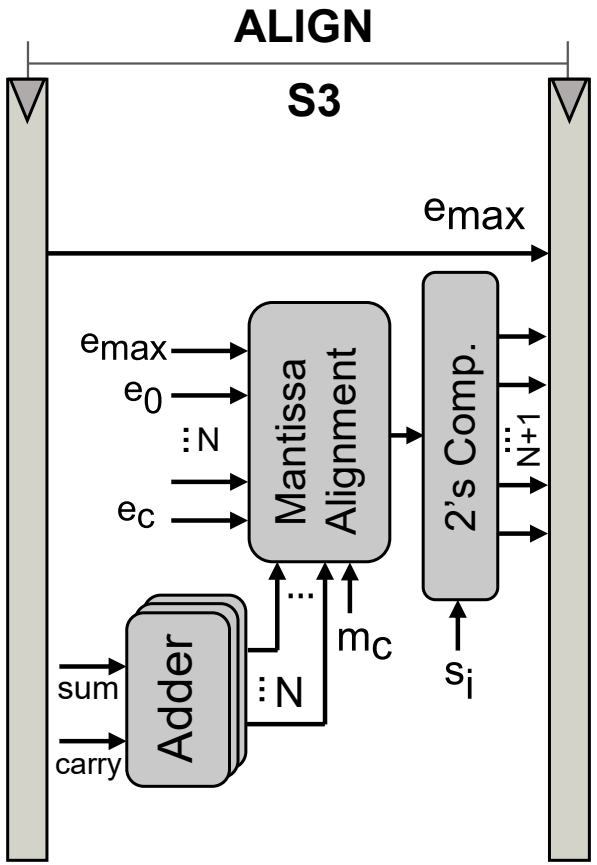
- S1: Decode
- **S2: Multiply**
  - Exponent comparison
  - **Mantissa multiplication**
    - Modified radix-4 booth multiplier → sum & carry





Architecture: Pipeline S3 – Align

- S1: Decode
- S2: Multiply
- **S3: Align**
  - **Final addition**
    - sum + carry
  - **Mantissa alignment**
    - Barrel shifters
  - **Two's complement**



$$2^5 \times 1.1110110$$
$$+2^3 \times 1.0110101$$
$$+2^0 \times 1.1011001$$

Shift Alignment

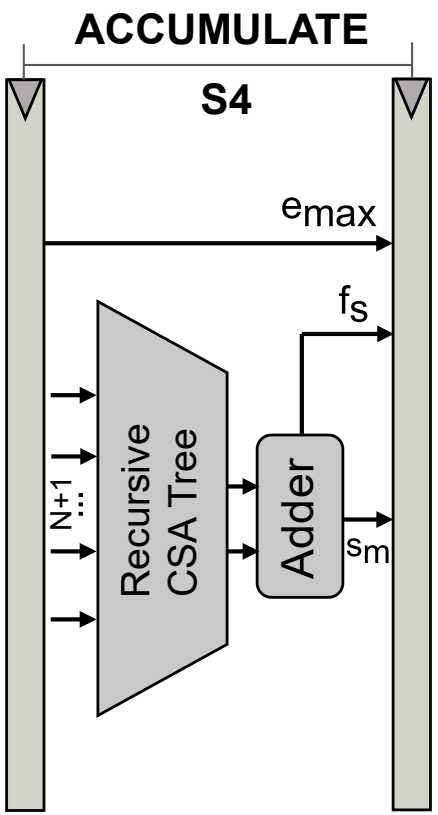
$$2^5 \times \begin{matrix} 1.1110110 \\ +\,0.010110101 \\ +\,0.000011011001 \end{matrix}$$

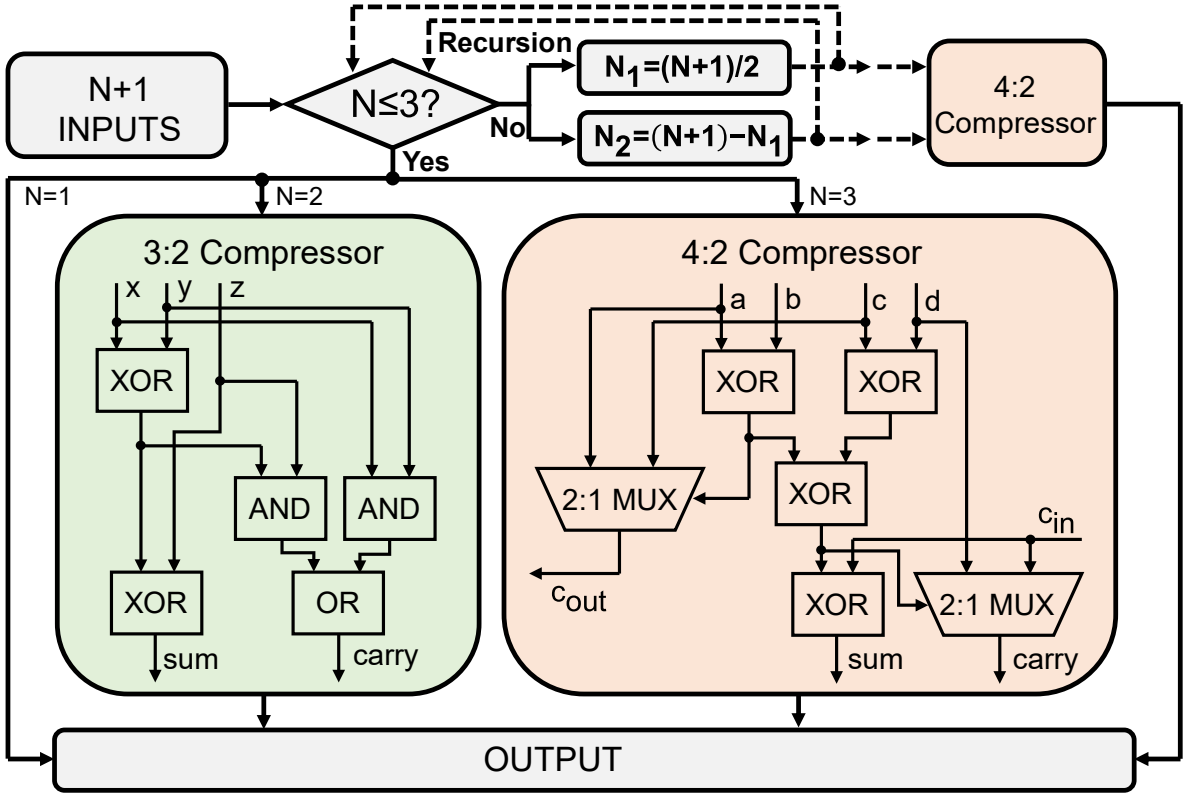
Accumulation

$$2^5 \times 10.010101000001$$

Architecture: Pipeline S4 – Accumulate

- S1: Decode
- S2: Multiply
- S3: Align
- **S4: Accumulate**
  - **Compression**
    - Recursive CSA tree → sum & carry
  - **Final addition**
    - sum + carry

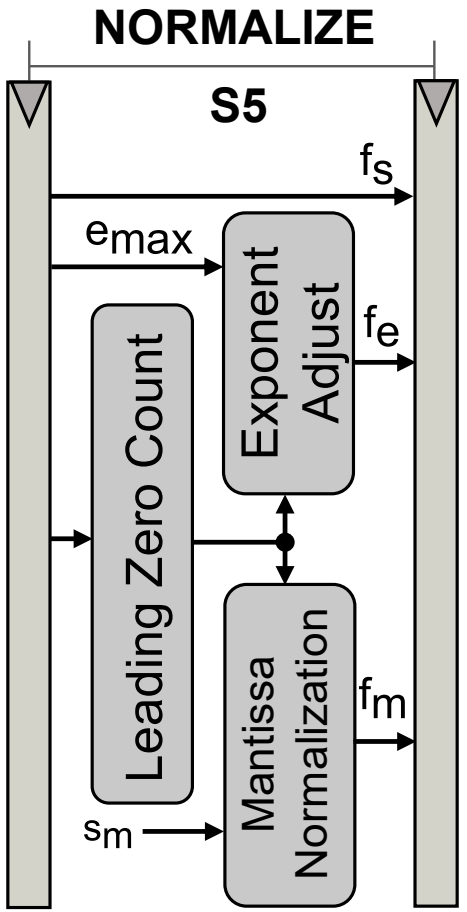


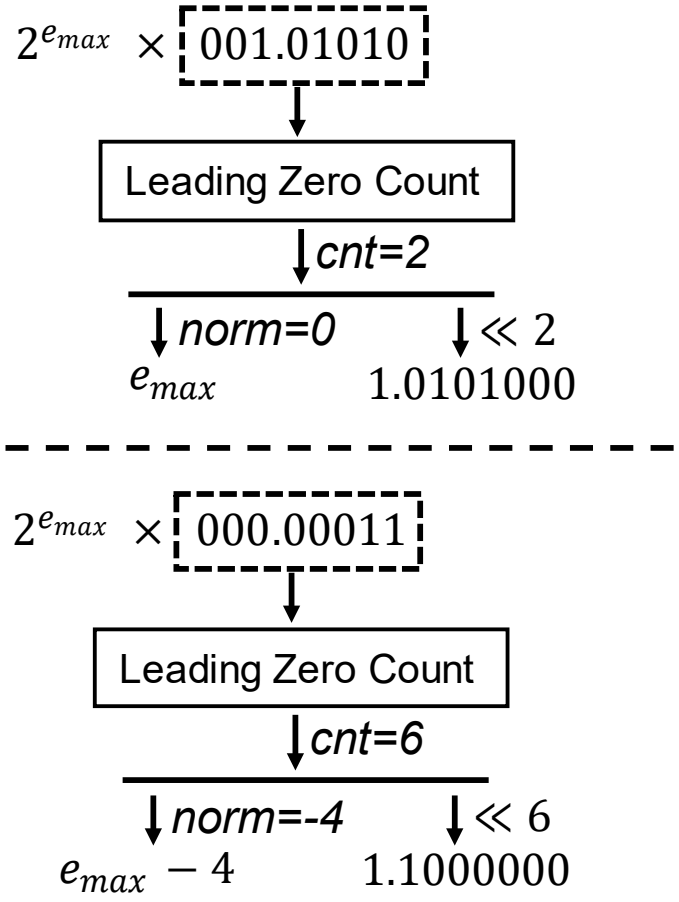


➢ Recursive design method of Carry-Save-Adder (CSA) tree

Architecture: Pipeline S5 – Normalize

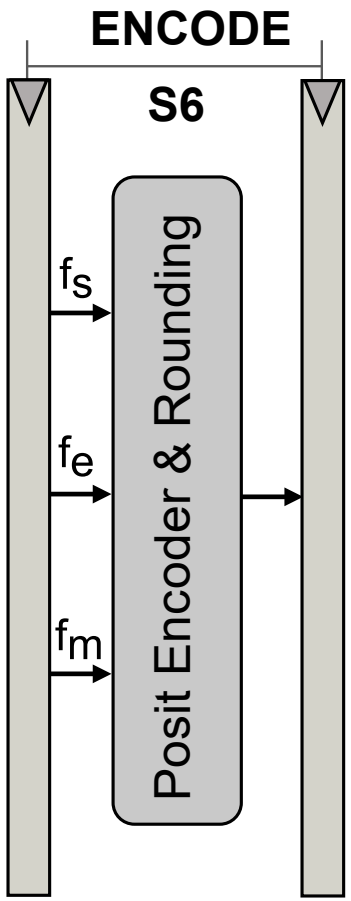
- S1: Decode
- S2: Multiply
- S3: Align
- S4: Accumulate
- **S5: Normalize**
  - **Leading zero count**
  - **Mantissa normalization**
  - **Exponent adjustment**

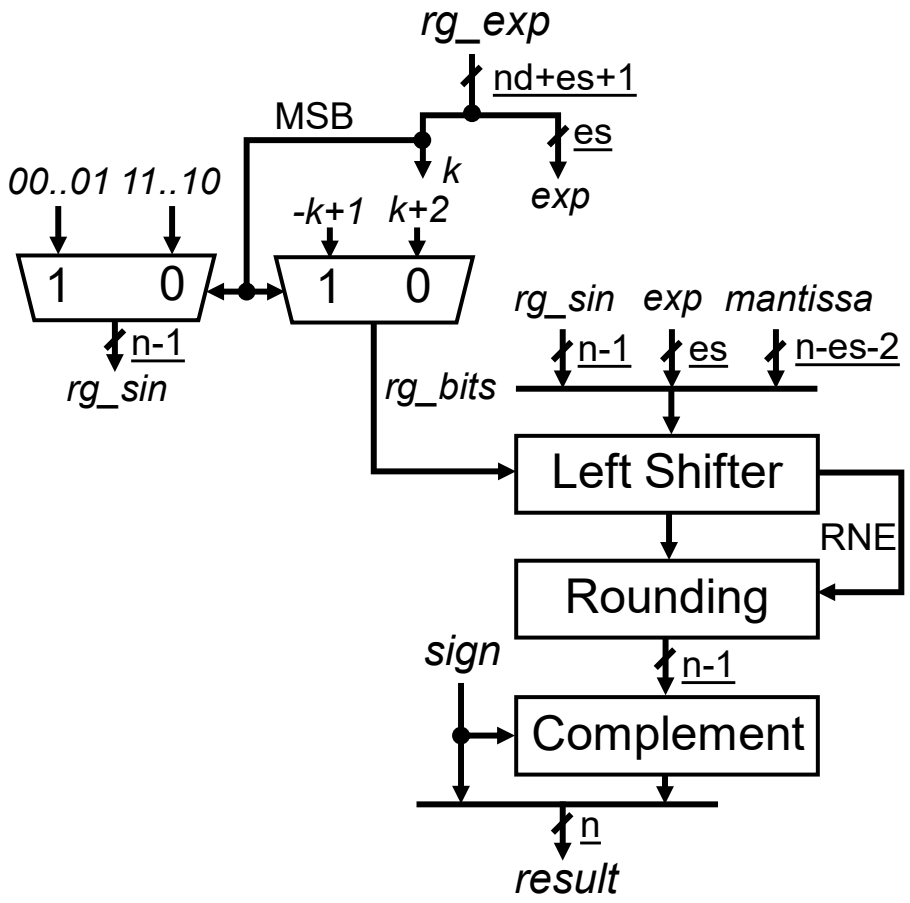




Architecture: Pipeline S6 – Encode

- S1: Decode
- S2: Multiply
- S3: Align
- S4: Accumulate
- S5: Normalize
- **S6: Encode**
  - **Encoding:** pack result components into posit output
  - **Rounding**: RNE method

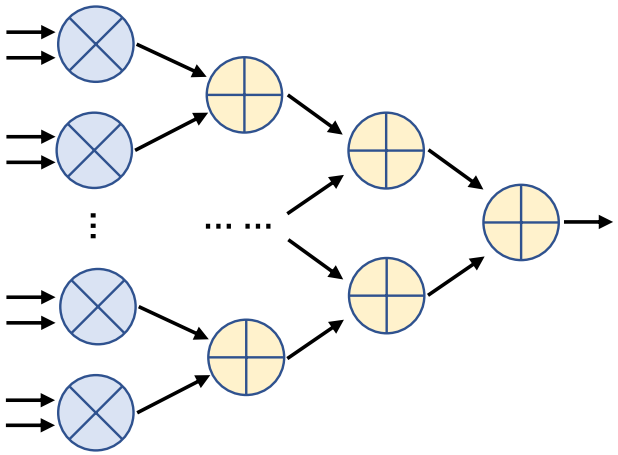




➢ Architecture of posit encoder

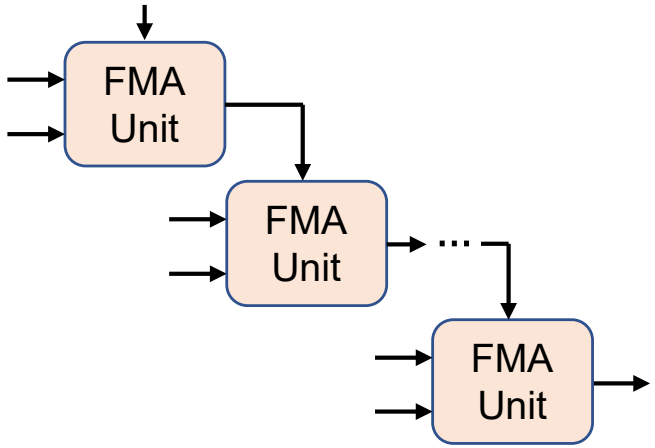
Architecture: Fused implementation

- **Fused Strategy**: perform N-terms dot-product operation as a whole
  - **High computational efficiency and area efficiency**, by removing redundant logic (e.g., repeated encoding and decoding process)
  - **High computational accuracy**, by decreasing rounding operations



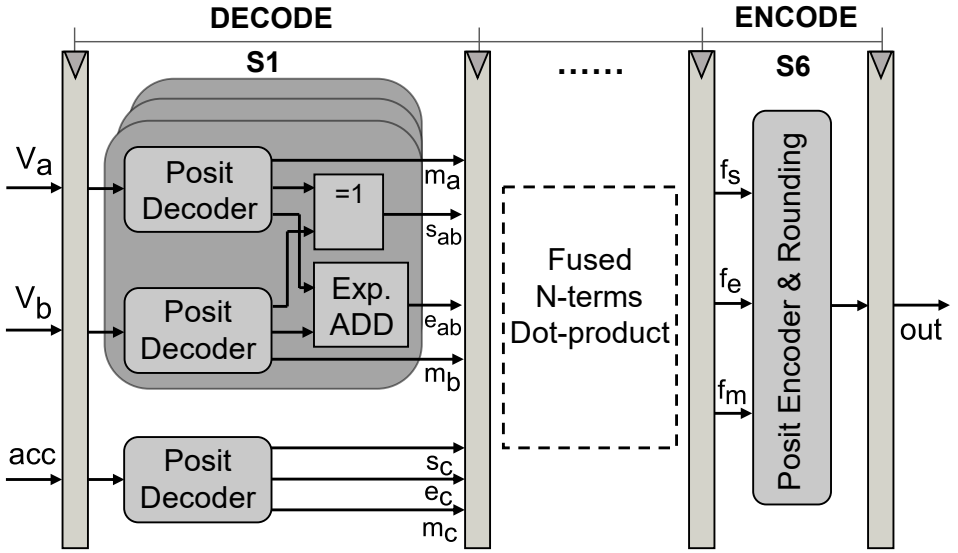
Decoders: $\geq 2N + 2^{\lfloor \log_2(N+1) \rfloor}$

Encoders: $\geq N + 2^{\lfloor \log_2(N+1) \rfloor}$



Decoders: $3N$

Encoders: N



Decoders: $2N + 1$

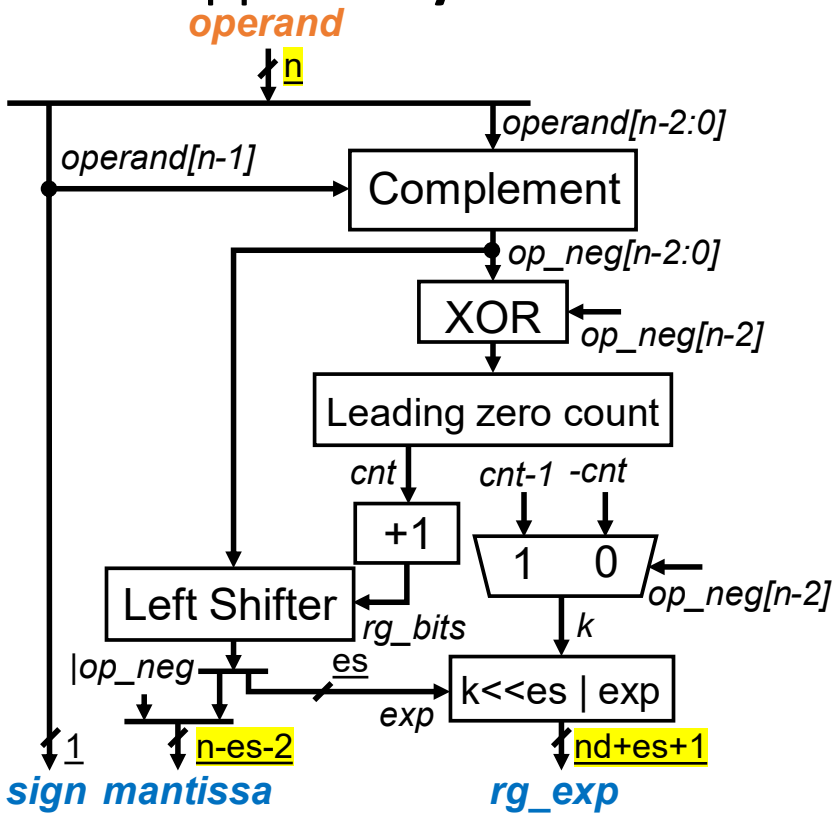
Encoders: 1

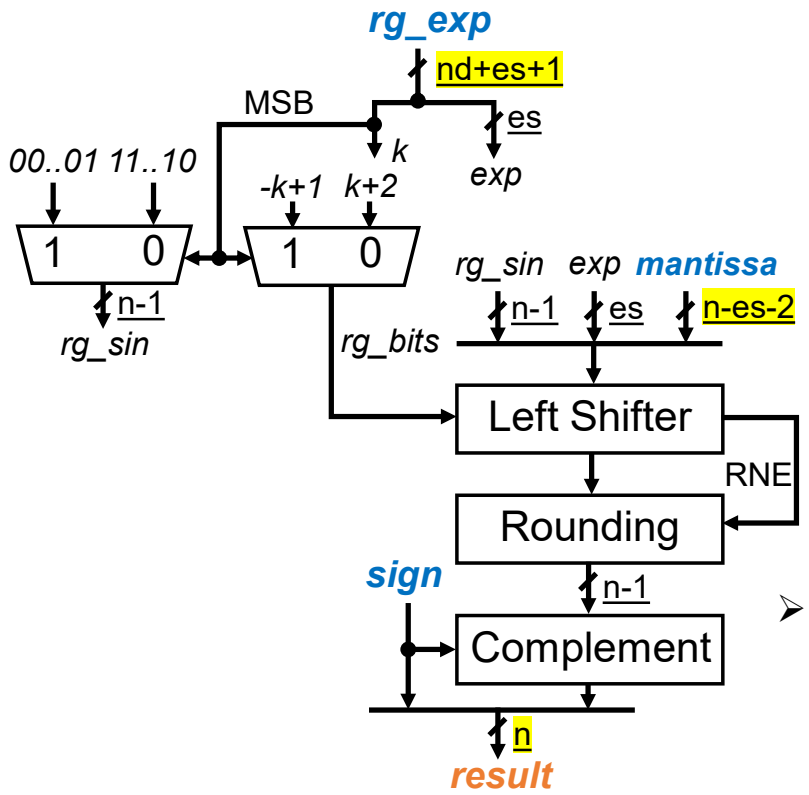
Outline

- Backgrounds & Motivations
- Overall Architecture
- **Configurable PDPU Generator**
- Experimental Results
- Conclusion

PDPU Generator – 1

- **Supporting Custom Posit Formats**
  - Support **any combination of n and es** both for inputs and outputs

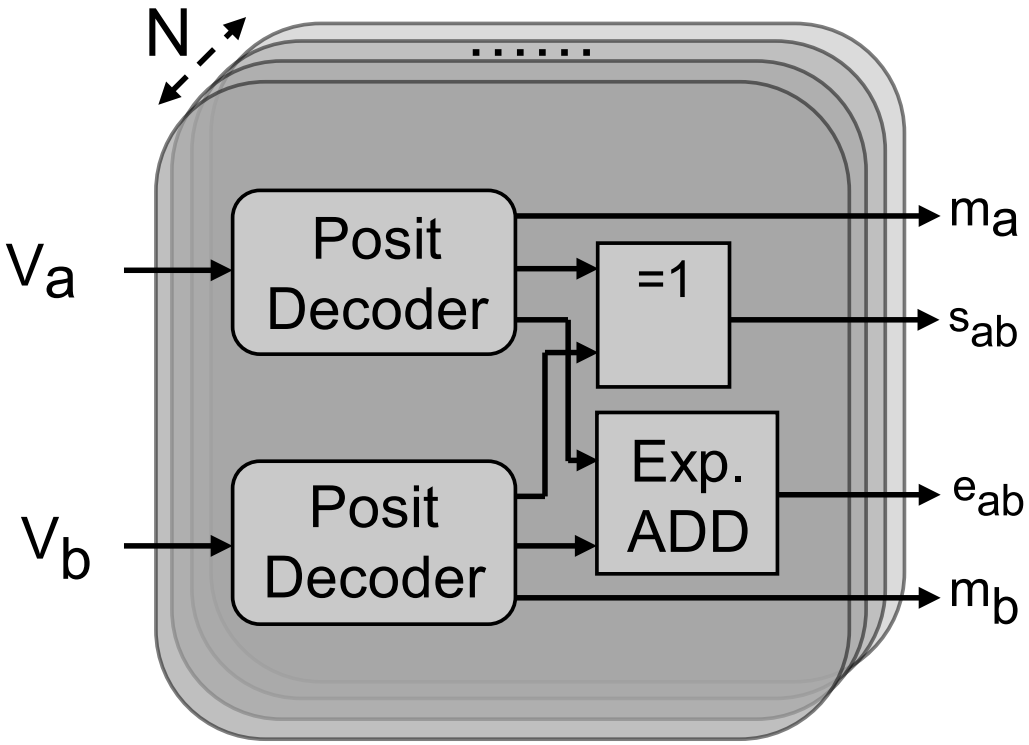


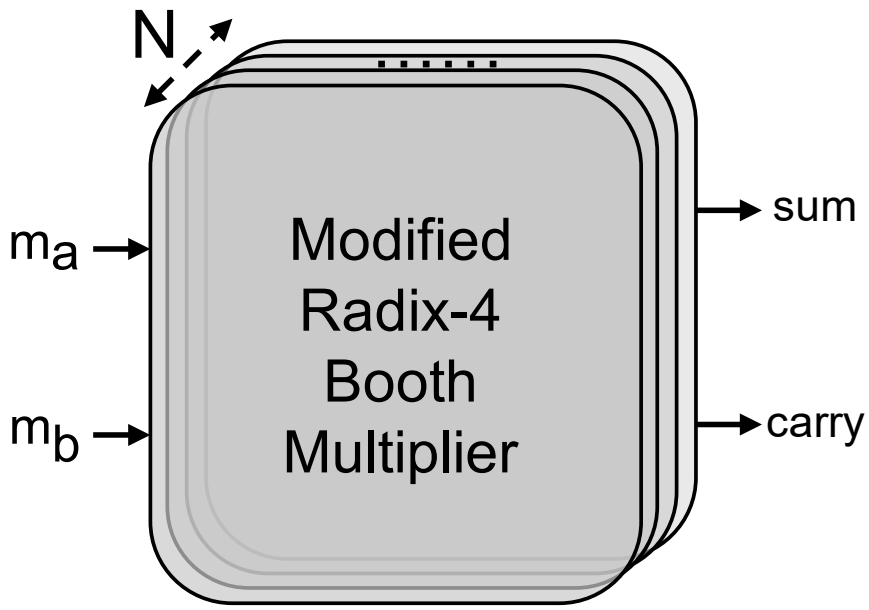


- The flexible format supports of decoder and encoder **enable mixed-precision strategy**

PDPU Generator – 2

- **Supporting Diverse Dot-Product Sizes**
  - Method 1: **Instantiate sub-modules in parallel**

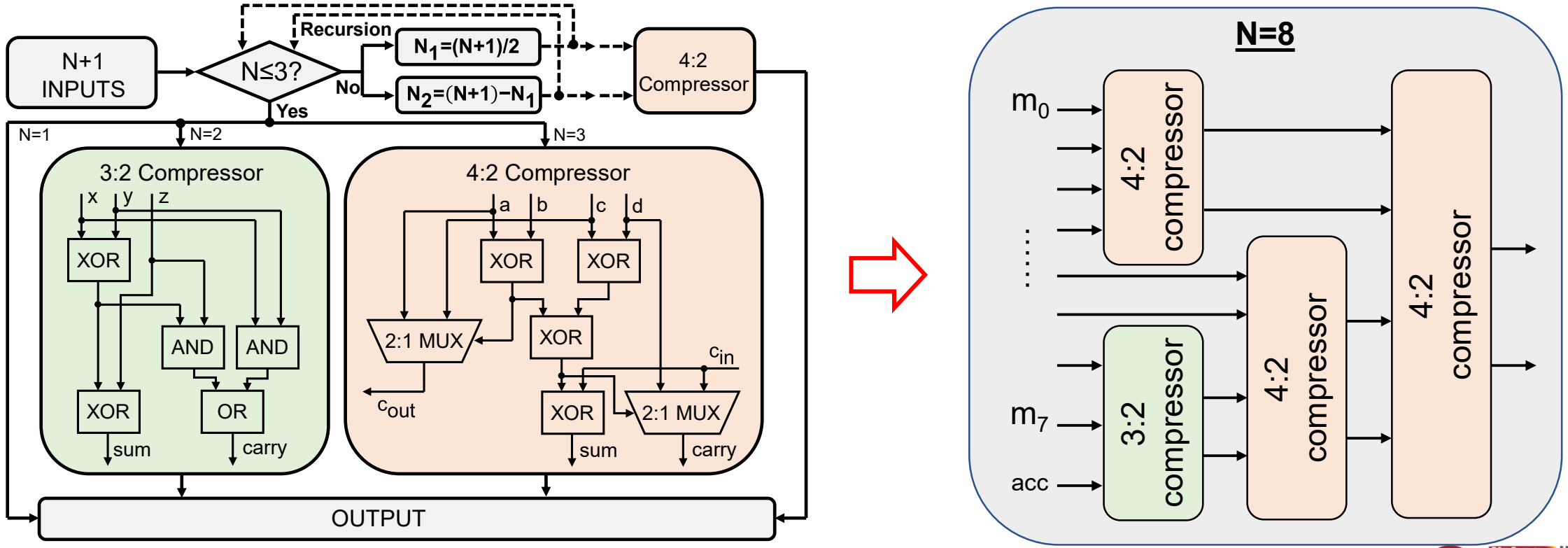




- The modules are **instantiated in parallel** to handle N groups of inputs simultaneously

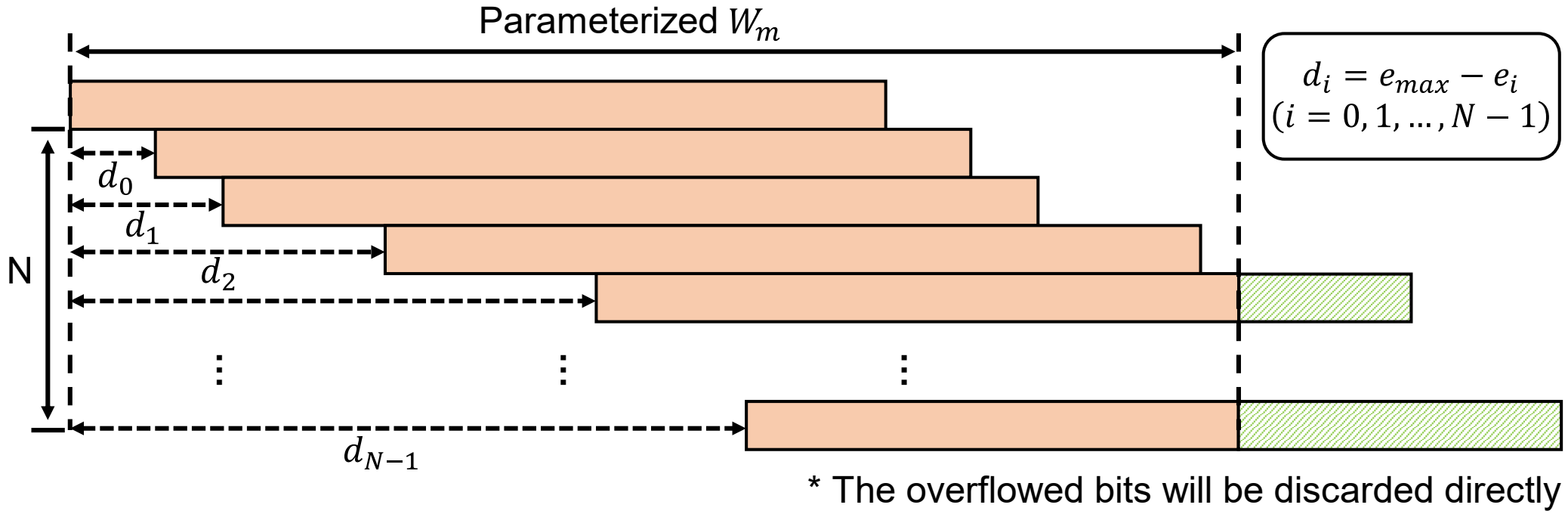
PDPU Generator – 2 (Cont'd)

- **Supporting Diverse Dot-Product Sizes**
  - Method 2: **Recursively implement submodules** → tree structure



PDPU Generator – 3

- **Supporting Suitable Alignment Width**
  - Determine the width of aligned mantissa (i.e., W_m) based on DNN accuracy requirements → **minimize the hardware cost while meeting precision**



Outline

- Backgrounds & Motivations
- Overall Architecture
- Configurable PDPU Generator
- **Experimental Results**
- Conclusion

Experimental Settings

- Design in SystemVerilog, Validated using Extended SoftPosit Library
- Synopsys DC, TSMC-28nm with Normal Case
- **Comparison with the SOTAs**
  - **Baselines**
    - PACoGen [M. K. Jaiswal et al., IEEE Access'19]
    - FPnew [S. Mach et al., TVLSI'20]
    - Posit FMA [H. Zhang et al., ISCAS'19]
  - **Combinationally implemented** → avoid impacts of different pipeline schemes
  - Accuracy: **computational accuracy** of the first convolution layer of ResNet18
  - **Mixed-Precision PDPU**: P(13/16,2) means P(13,2) for a, b & P(16,2) for acc, out
- **Evaluation of 6-stage Pipeline**

Experimental Results – I

- **Performance of PDPU under Different Configurations**
  - Inappropriate data formats or alignment width **may result in 10% higher computational loss of accuracy**
  - **Improve area and energy efficiency by 5.0x and 2.1x**, compared with quire PDPU

Architecture	Formats	N	W_m	Accuracy	Area (um^2)	Delay (ns)	Power (mW)	Perf. (GOPS)	Area Eff. ($GOPS/mm^2$)	Energy Eff. (GOPS/W)
	P(16/16,2)	4	14	99.10%	9579.15	1.62	4.49	2.47	257.76	550.37
	P(13/16,2)	4	14	98.69%	**7694.82**	**1.60**	**3.66**	**2.50**	**324.89**	**682.82**
Proposed PDPU	P(13/16,2)	8	14	98.68%	13560.37	1.69	5.80	4.73	349.09	816.16
	P(**10**/16,2)	8	14	**89.58%**	10006.42	1.70	4.24	4.71	470.29	1110.95
	P(13/16,2)	8	**10**	**88.90%**	12157.11	1.66	5.06	4.82	396.42	953.14
Quire PDPU	P(13,16,2)	4	**256**	98.79%	**29209.45**	**2.10**	**5.87**	**1.90**	**65.21**	**324.50**

10% Accuracy Drop

5.0x

2.1x

Experimental Results – I (Cont'd)

- **Comparison of PDPU with Existing DPU Implementations under N=4**
  - P(16, 2) can maintain the precision of FP32
  - **Mixed-precision PDPU reduce area, latency and power by 43%, 64%, and 70%**, compared with PACoGen DPU

Architecture	Formats	N	W_m	Accuracy	Area (um^2)	Delay (ns)	Power (mW)	Perf. (GOPS)	Area Eff. ($GOPS/mm^2$)	Energy Eff. (GOPS/W)
FPnew DPU	FP32	4	\	100%	28563.19	3.45	7.60	1.16	40.59	152.65
	FP16	4	\	91.21%	13448.99	2.75	4.29	1.45	108.15	338.85
PACoGen DPU	P(16,2)	4	\	**98.86%**	**13433.11**	**4.45**	**12.21**	**0.90**	**66.91**	**73.59**
Proposed PDPU	P(16/16,2)	4	14	**99.10%**	9579.15	1.62	4.49	2.47	257.76	550.37
	P(13/16,2)	4	14	**98.69%**	**7694.82**	**1.60**	**3.66**	**2.50**	**324.89**	**682.82**

↓ 43% ↓ 64% ↓ 70%

Maintain Precision

Experimental Results – I (Cont'd)

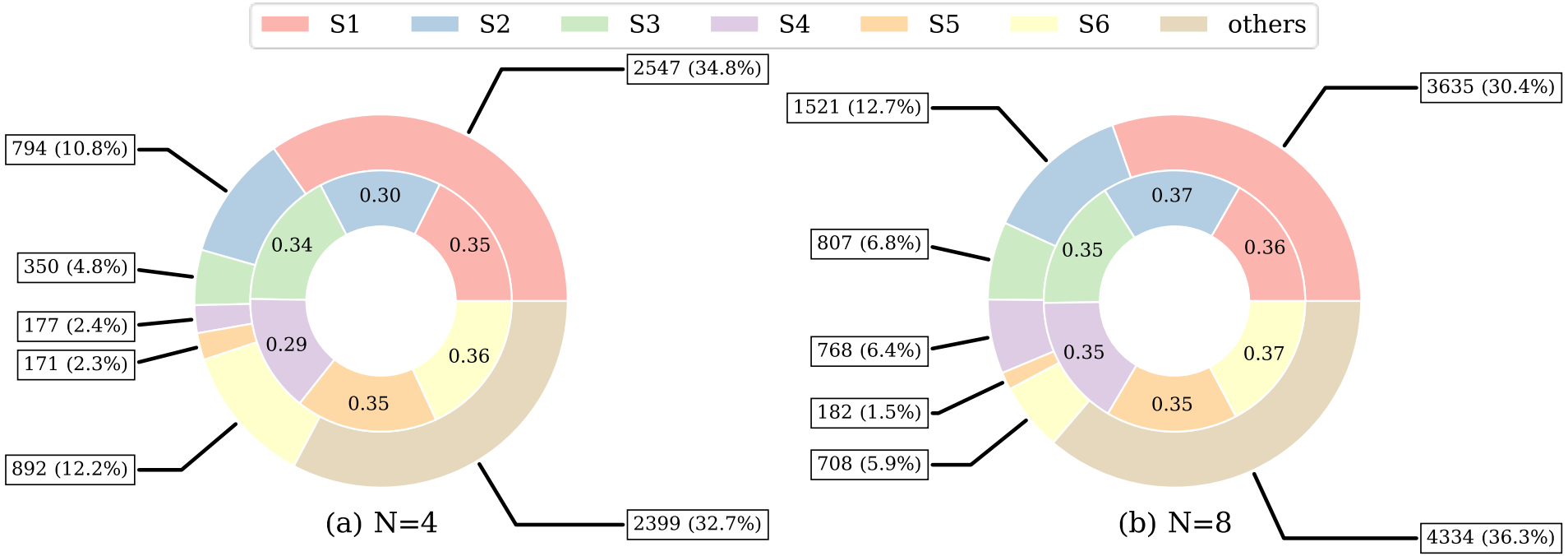
- **Comparison of PDPU with Off-the-shelf FMA Implementations**
  - PDPU provides **3.1x and 3.5x the area and energy efficiency** compared with Posit FMA unit

Architecture	Formats	N	W_m	Accuracy	Area (um^2)	Delay (ns)	Power (mW)	Perf. (GOPS)	Area Eff. ($GOPS/mm^2$)	Energy Eff. (GOPS/W)
Proposed PDPU	P(13/16,2)	4	14	98.69%	**7694.82**	**1.60**	**3.66**	**2.50**	**324.89**	**682.82**
	P(13/16,2)	8	14	98.68%	13560.37	1.69	5.80	4.73	349.09	816.16
FPnew FMA	FP32	1	\	100%	6668.17	1.20	3.97	0.83	124.97	210.00
	FP16	1	\	92.93%	3713.72	1.00	2.51	1.00	269.27	398.61
Posit FMA	P(16,2)	1	\	99.23%	**7035.34**	**1.35**	**3.79**	**0.74**	**105.29**	**195.52**

3.1x **3.5x**

Experimental Results – II

- **Evaluation of 6-stage Pipeline**
  - A balanced critical path delay of each stage, **improving throughput of PDPU by 4.4x and 4.6x**, respectively



Outline

- Backgrounds & Motivations
- Overall Architecture
- Configurable PDPU Generator
- Experimental Results
- **Conclusion**

Conclusion

- A Posit Dot-Product Unit (PDPU) for Deep Learning Applications
  - **Fused and mixed-precision** properties
  - A balanced **6-stage pipeline** scheme
  - Configurable **PDPU generator**
- PDPU achieves a significant reduction of **43%**, **64%**, and **70%** in terms of **area**, **delay**, and **power**, respectively
- Code is available at https://github.com/qleenju/PDPU for further exploration

Thank You!

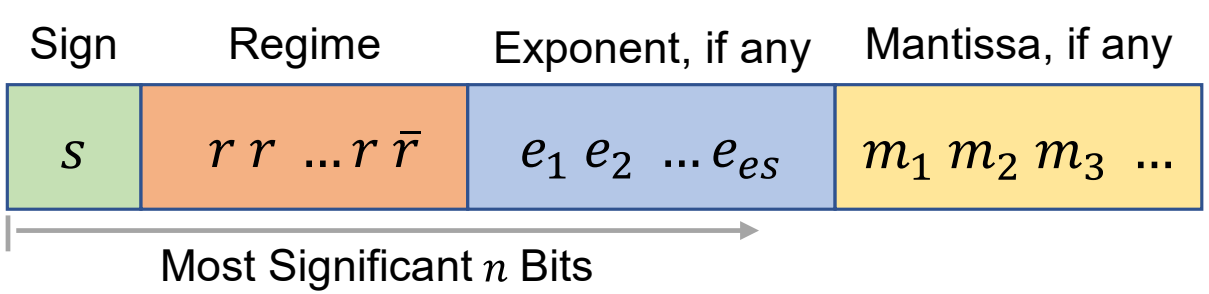
If you have any questions, please don't hesitate to drop me an email.

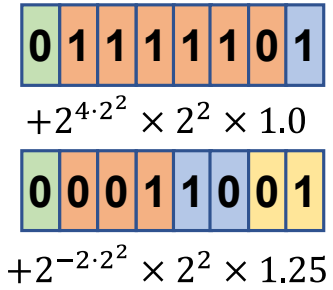
Open Source at **https://github.com/qleenju/PDPU**

Email: **qiongli@smail.nju.edu.cn**, fantasysee@smail.nju.edu.cn, zfwang@nju.edu.cn

Appendix A: Posit Format Encoding

- Defined by **word size (n)** and **exponent size (es)**, i.e., P(n, es)
- $$p = \begin{cases} \pm 0, & 000 \ldots 000, \\ \pm\infty, & 100 \ldots 000, \\ (-1)^{s_p} \times \mathbf{2^{k \cdot 2^{es}}} \times 2^{e_p} \times 1.m_p, & otherwise, \end{cases}$$
  - where s_p, e_p, m_p is the value of sign, exponent and mantissa field, respectively
  - ***k* is the factor represented by regime bits**





- The regime field is composed of ***m* consecutive identical bits *r* and an opposite bit $\bar{r}$**, indicating a scale factor of $2^{k \cdot 2^{es}}$
- $k = \begin{cases} -m, & r = 0 \\ m - 1, & r = 1 \end{cases}$

Appendix B: Complete experimental results

Architecture	Formats	N	W_m	Accuracy	Area (um^2)	Delay (ns)	Power (mW)	Perf. (GOPS)	Area Eff. ($GOPS/mm^2$)	Energy Eff. (GOPS/W)
FPnew DPU	FP32	4	\	100%	28563.19	3.45	7.60	1.16	40.59	152.65
	FP16	4	\	91.21%	13448.99	2.75	4.29	1.45	108.15	338.85
PACoGen DPU	P(16,2)	4	\	98.86%	13433.11	4.45	12.21	0.90	66.91	73.59
Proposed PDPU	P(16/16,2)	4	14	99.10%	9579.15	1.62	4.49	2.47	257.76	550.37
	P(13/16,2)	4	14	98.69%	**7694.82**	**1.60**	**3.66**	2.50	324.89	682.82
	P(13/16,2)	8	14	98.68%	13560.37	1.69	5.80	**4.73**	**349.09**	**816.16**
	P(10/16,2)	8	14	89.58%	10006.42	1.70	4.24	4.71	470.29	1110.95
	P(13/16,2)	8	10	88.90%	12157.11	1.66	5.06	4.82	396.42	953.14
Quire PDPU	P(13,16,2)	4	256	98.79%	29209.45	2.10	5.87	1.90	65.21	324.50
FPnew FMA	FP32	1	\	100%	6668.17	1.20	3.97	0.83	124.97	210.00
	FP16	1	\	92.93%	3713.72	1.00	2.51	1.00	269.27	398.61
Posit FMA	P(16,2)	1	\	99.23%	7035.34	1.35	3.79	0.74	105.29	195.52