Approximate Computing with Approximate Circuits: Methodologies and Applications

ESWEEK 2017 Tutorial

Lukáš Sekanina
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
sekanina@fit.vutbr.cz

Jie Han
Department of Electrical and Computer Engineering, University of Alberta
Edmonton, AB, Canada
jhan8@ualberta.ca

UNIVERSITY OF ALBERTA

Part I: Approximate Arithmetic Circuits and Applications

Jie Han

Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, Canada

Cite: Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, and Jie Han. "A review, classification, and comparative evaluation of approximate arithmetic circuits." ACM Journal on Emerging Technologies in Computing Systems (JETC) 13, no. 4 (2017): 60.

Tutorial Outline – Part II. Design automation methods

- Introduction
- Design automation methods for approximate circuits
  - Classification and overview
  - Circuit parameter estimation
  - Error computation
  - Relaxed equivalence checking
  - Evaluation methodology
- Examples of design automation methods for approximate circuits
  - Minterm complements, SASIMI, AIG rewriting, ABACUS, GRATER
- Evolutionary algorithms, CGP and circuit optimization
- Applications of CGP-based approximation methods
  - Open-source library of approximate adders and multipliers
  - Approximate TMR
  - Approximate multipliers in neural networks
  - Symbolic error analysis using BDDs/SAT solving in CGP-based tools
  - Approximate image filters
- Conclusions

Part I: Outline

- Background and Motivation
- Scope of Approximate Computing
- Classification, Review and Comparison of Approximate Adders
- Classification and Comparison of Approximate Multipliers
- Approximate Dividers
- Image Processing Applications
- Cerebellar Models using Approximate Circuits
- Conclusion and Prospects

Background: Fault Tolerance

- The continuous miniaturization of electronic devices requires fault-tolerant and variation-resilient designs:
  - to ensure operational reliability during their lifetime (due to soft errors, aging, etc.);
  - to accommodate the inevitable variations in nanoscale manufacturing processes.
- However, conventional fault-tolerant techniques result in significant overhead in energy consumption.
  - Including techniques using hardware, time and/or information redundancies.
- The conflict between reliability and energy efficiency presents significant design challenges.

Error: Essential Part of the Design Process

John von Neumann's View on Error (1952):

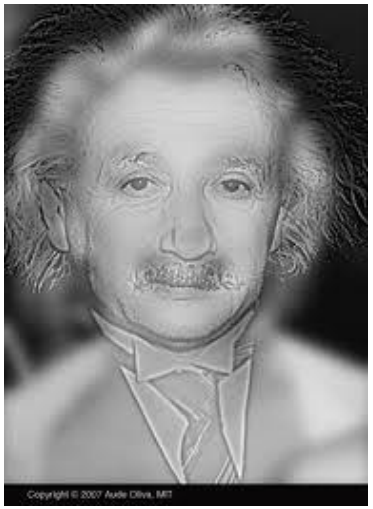
"Our present treatment of error is unsatisfactory and ad hoc. … Error is viewed (in this work), therefore, not as an extraneous and misdirected or misdirecting accident, but as an essential part of the process under consideration …" [1]

This view, that noise or error is an integral part of a system, is as valid today as it was in the early days of computers.

[1] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," Automata Studies, Shannon C.E. & McCarthy J., eds., Princeton University Press, pp. 43-98, 1956.

Motivation for Approximate Computing

- At the nanoscale era, improving performance of digital circuits and systems becomes increasingly difficult.
  - Energy efficiency is of paramount concern in digital system design.
- Computing becomes increasingly heavy with multimedia processing (audio, video, graphics, and image), recognition, search, machine learning and data mining.
- A common characteristic: a perfect result is not necessary and an approximate or less-than-optimal result is sufficient
  - Human perception is not sensitive to high frequency changes.
  - Natural noise floor due to quantization noise.



Motivation for Approximate Computing

- At the nanoscale era, improving performance of digital circuits and systems becomes increasingly difficult.
  - Energy efficiency is of paramount concern in digital system design.
- Computing becomes increasingly heavy with multimedia processing (audio, video, graphics, and image), recognition, search, machine learning and data mining.
- A common characteristic: a perfect result is not necessary and an approximate or less-than-optimal result is sufficient
  - Human perception is not sensitive to high frequency changes.
  - Natural noise floor due to quantization noise.

Error-Resilient Paradigms

How can we exploit a system's ability of imprecision tolerance and error resilience for energy efficiency?

- **Approximate Computing**
  - Does not involve assumptions on the stochastic nature of any underlying processes implementing the system. Utilizes statistical properties of data and algorithms to trade quality for energy reduction.
- **Stochastic Computing**
  - Real numbers are represented by random binary bit streams that are implemented in series (or parallel) and in time (or space). Information is carried on the statistics of the binary streams.
- **Probabilistic Computing**
  - Exploits intrinsic probabilistic behavior of the underlying circuit fabric, most explicitly, of the stochastic behavior of a binary switch under the influence of thermal noise.

[2] J. Han and M. Orshansky, "Approximate Computing: An Emerging Paradigm For Energy-Efficient Design," In ETS, pages 1-6, Avignon, France, 2013.

Spectrum of Approximate Computing

- In contrast to the *passive* use of redundancies, approximate computing (AC) employs *active* design methodologies that exploit the feature that many systems and applications can tolerate some loss of accuracy in the computation result.
- Effort in approximate computing covers a broad spectrum of research, ranging from those addressing issues at circuit and system levels, up to those at software and application levels.

Approximate Hardware Design

We focus on how hardware is re-designed.

- Arithmetic circuit design at the transistor and logic levels [3]
  - Adders, multipliers and dividers
- Approximate memory and storage [4]
  - SRAM, DRAM and non-volatile memories
- Approximate processor architectures [5]
  - Neural networks, general-purpose and reconfigurable processors such as instruction set architectures (ISAs), graphic processing units (GPUs) and FPGAs

[3] S. Venkataramani, S.T. Chakradhar, K. Roy, and A. Raghunathan, "Computing approximately, and efficiently," In Design, Automation & Test in Europe Conference & Exhibition, pp. 748-751, 2015.

[4] A. Sampson, J. Nelson, K Strauss, and L. Ceze, "Approximate storage in solid-state memories," ACM Transactions on Computer Systems (TOCS), vol. 32, no. 3, 9, 2014.

[5] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Architecture support for disciplined approximate programming," ACM SIGPLAN Notices, vol. 47, no. 4, pp. 301-312, 2012.

Approximate Adders: A Classification

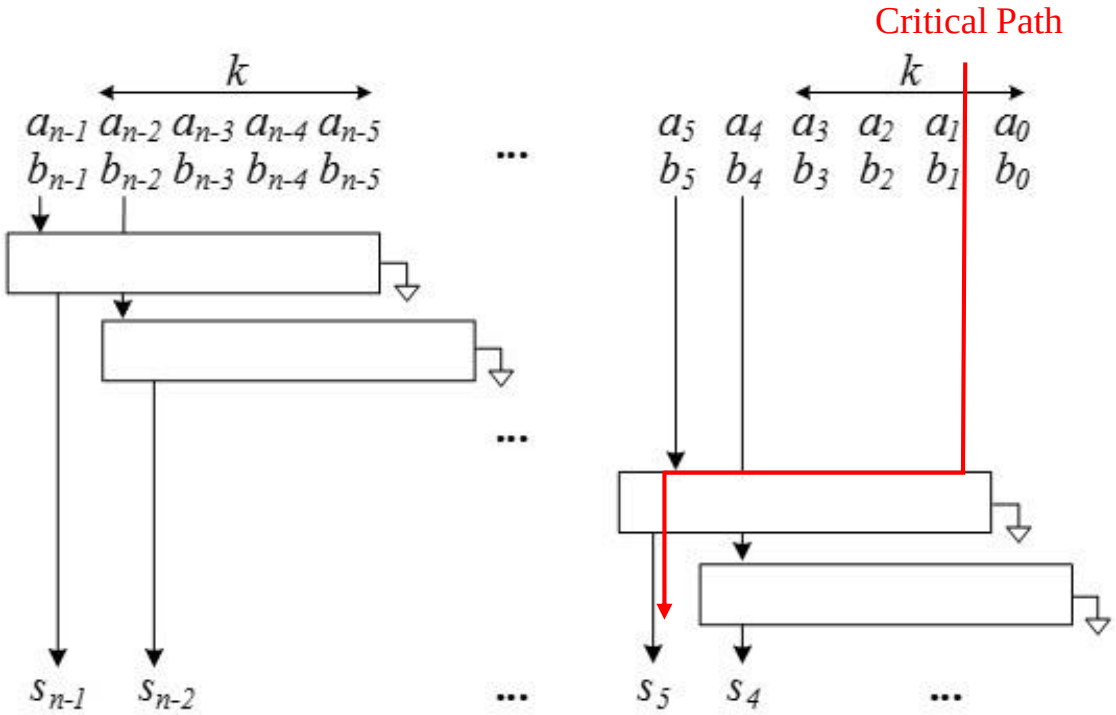
We classify approximate adders into four categories:

- **Speculative Adders**
  - For a 128-bit adder, the probability that the carry propagation chain is longer than 12 and 18 are 1% and 0.01%, respectively [6].
  - Therefore, k bits are used to speculate the carry for each bit of sum ($k < n$).
- **Segmented Adders**
  - An n-bit adder is divided into a number of smaller k-bit sub-adders.
  - The carry is generated by using different methods.
- **Carry-Select Adders**
  - Multiple sub-circuits are used to compute the sum for different carry values, and the result is selected by the carry of a sub-circuit.
- **Approximate Full Adders**
  - Including truncated adders with a lower resolution.

[6] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," In DATE, pages 1250 - 1255, 2008.

Speculative Adders

The almost correct adder (ACA):



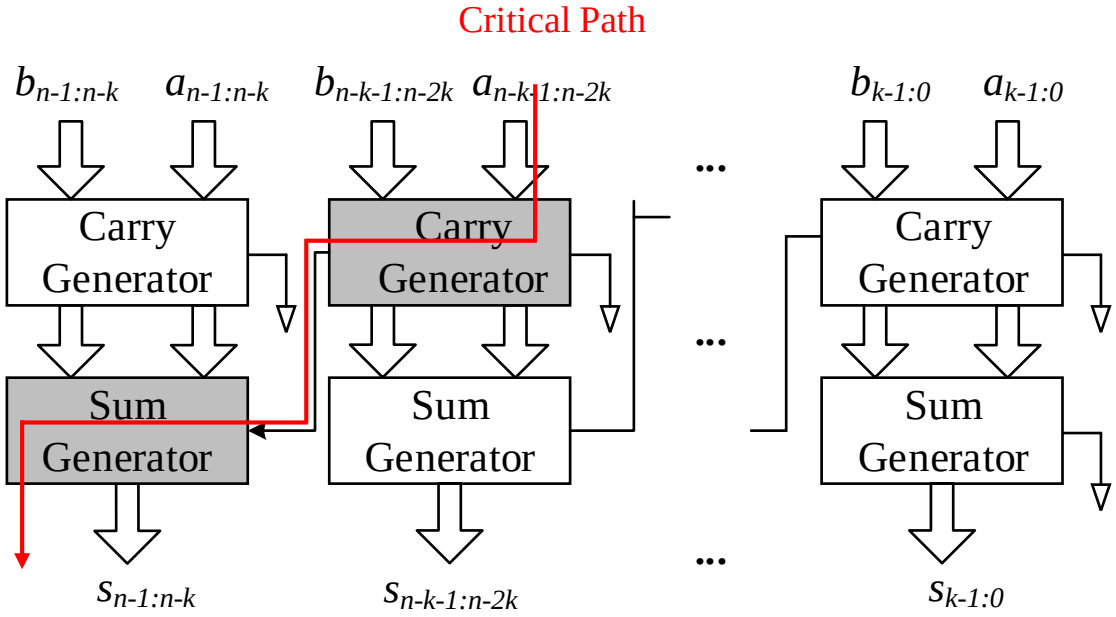
The *n*-bit almost correct adder (ACA).

$Critical\ path\colon\ O(\log(k))$ $\quad$ $Circuit\ area\colon\ O((n-k)k\log(k))$

[6] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," In DATE, pages 1250 - 1255, 2008.

Segmented Adders

The error-tolerant adder type II (ETAII):



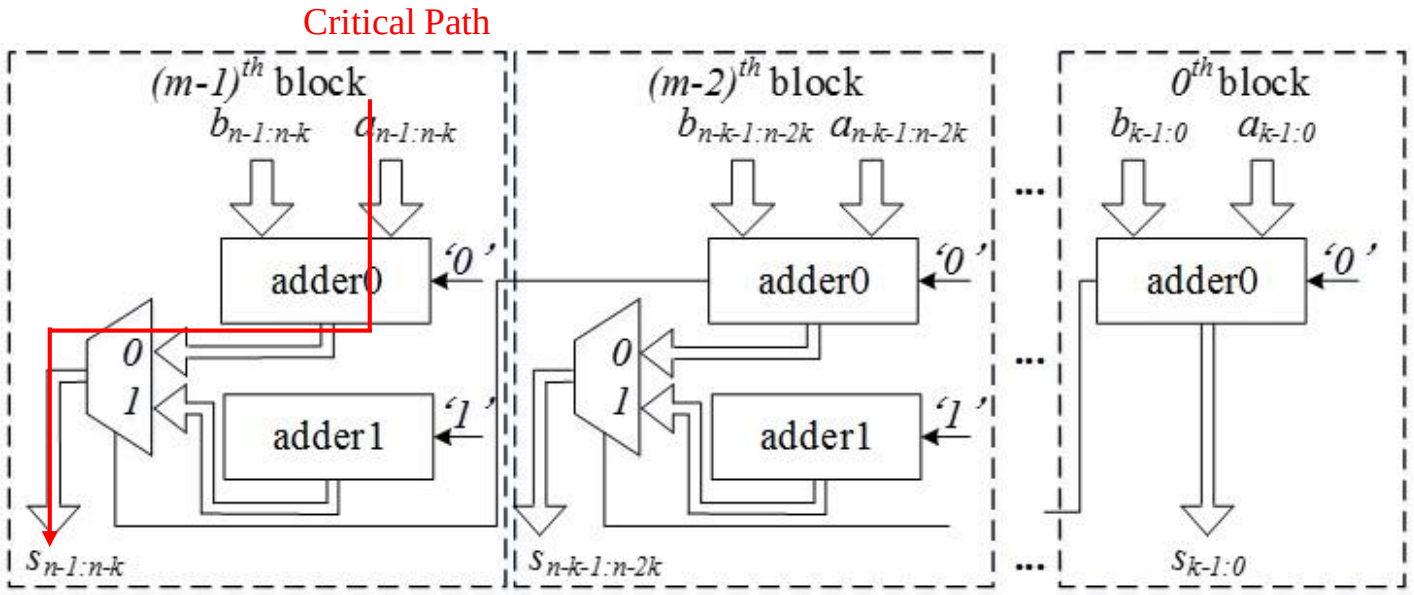
The *n*-bit error-tolerant adder type II (ETAII).

$Critical\ path\colon O(\log(k))$ $Circuit\ area\colon O(n\log(k))$

[7] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," In ISIC 2009, pages 69-72, 2009.

Carry Select Adders

The speculative carry selection adder (SCSA):



The *n*-bit speculative carry selection adder (SCSA).

$Critical\ path$: $t_{adder} + t_{mux}$

t_{adder}: $O(\log(k))$

t_{mux}: $delay\ of\ the\ multiplexer$

$Circuit\ area$: $A_{adder} + A_{mux}$

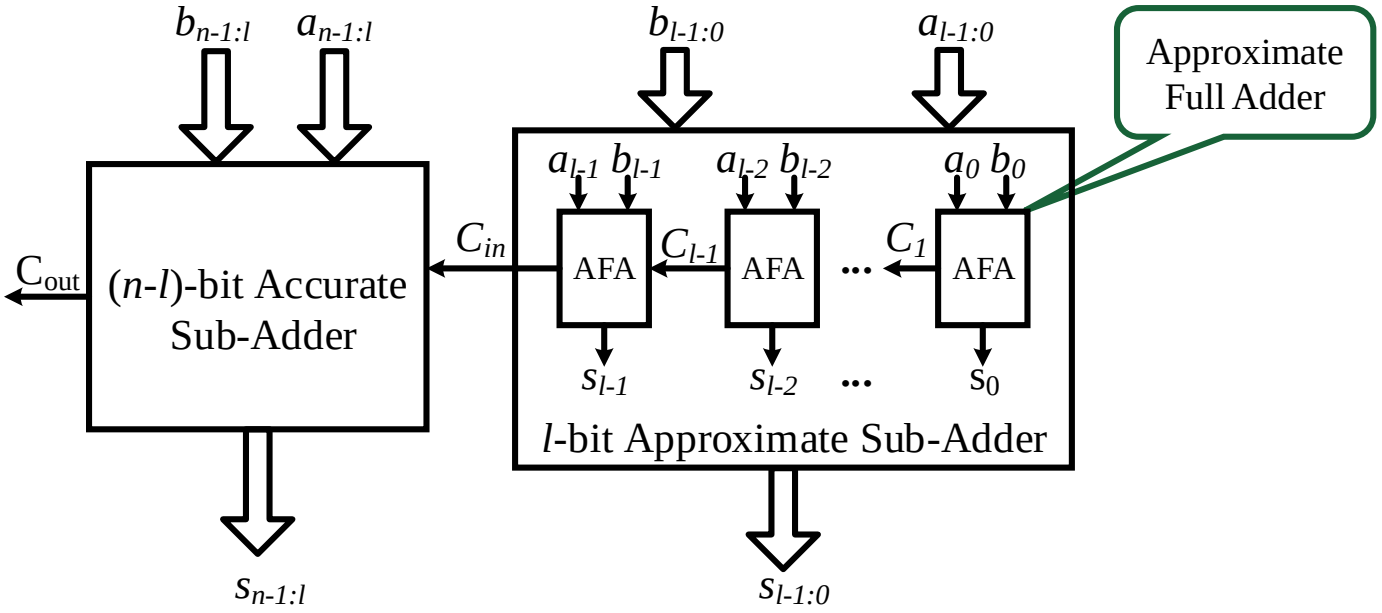
A_{adder}: $O(n\log(k))$

A_{mux}: $circuit\ area\ of\ the\ multiplexer$

[8] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," In DATE, pages 1257-1262, 2012.

Approximate Full Adders

A general schematic:



The n-bit approximate adder using approximate full adders

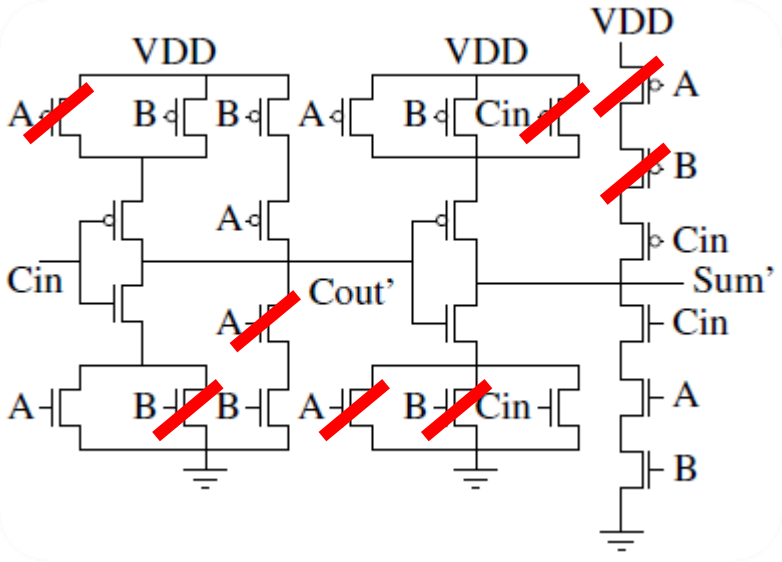
Critical path:

$t_{approximate\_adder} + t_{accurate\_adder}$

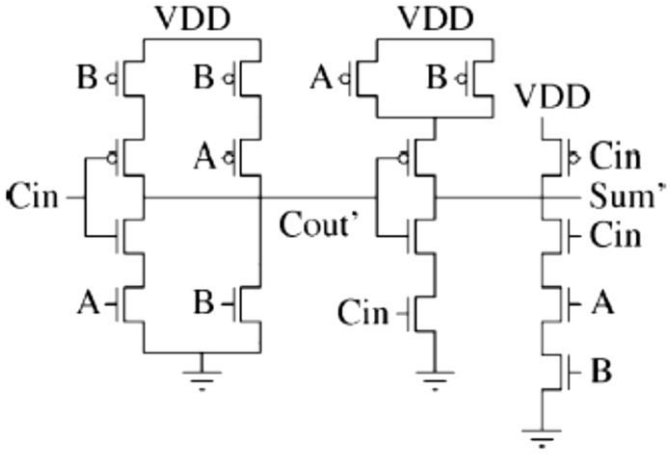
Circuit area:

$A_{approximate\_adder} + A_{accurate\_adder}$

Approximate Mirror Adders (AMAs)



The conventional mirror adder (MA).



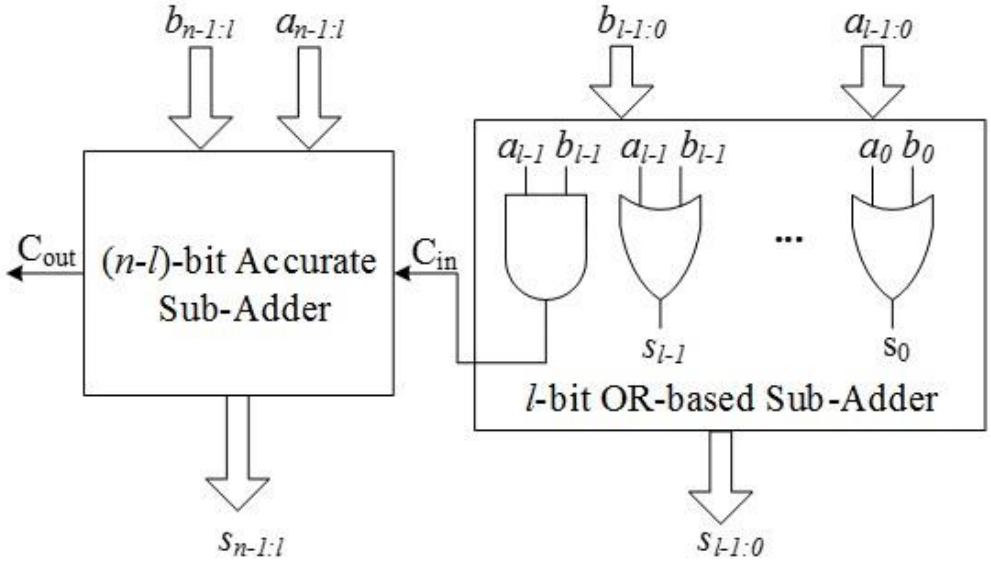
The mirror adder approximation 1 (AMA1).

The truth table for AMA1.

A	B	C_{in}	Sum'	C_{out}'
0	0	0	0	0
0	0	1	1	0
0	1	0	**0**	**1**
0	1	1	0	1
1	0	0	**0**	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

[9] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," IEEE Trans. CAD, 32(1):124-137, 2013.

Lower-part OR Adders (LOAs)



The n-bit lower-part-OR adder (LOA).

$Critical\ path\colon\ O(\log(n-l))$ $\qquad$ $Circuit\ area\colon\ A_{adder} + (l \times A_{OR})$

$A_{adder}\colon\ O((n-l)log(n-l))$ $\qquad$ $A_{OR}\ :\ circuit\ area\ of\ the\ OR\ gate.$

[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," IEEE Trans. Circuits Syst., 57(4):850-862, 2010.

A Brief Summary

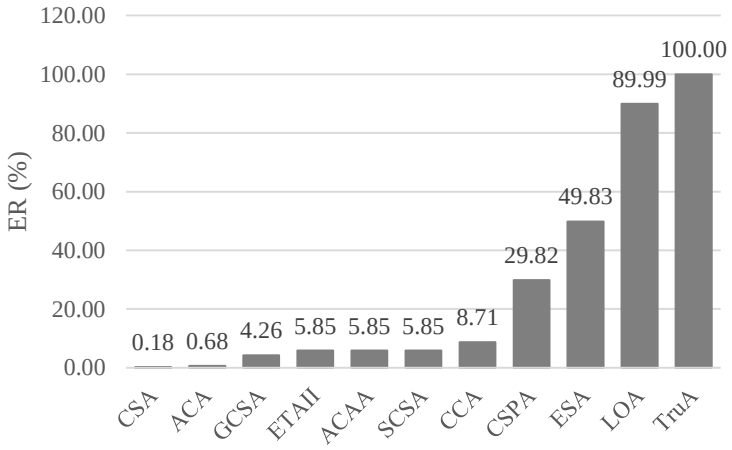
Analysis of delay and circuit complexity of approximate adders.

Adder Type		Adder Name	Delay	Circuit Area
Conventional Adders		RCA	$O(n)$	$O(n)$
		CLA	$O(log(n))$	$O(nlog(n))$
Approxim-ate Adders	Speculative Adders	ACA [6]	$O(\log(k))$	$O((n-k)k\log(k))$
	Segmented Adders	ESA [12]	$O(\log(k))$	$O(n\log(k))$
		ETAII [7]	$\boldsymbol{O(\log(k))}$	$\boldsymbol{O(n\log(k))}$
		ACAA [13]	$O(\log(k))$	$O((n-k)\log(k))$
	Carry Select Adders	SCSA [8]	$t_{adder}+t_{mux}$	$A_{adder}+A_{mux}$
		CSA [14]	$O(\log(k))$	$A_{adder}+A_{carry}$
		CSPA [15]	$t_{adder}+t_{mux}$	$A_{adder}+A_{mux}+A_{carry}$
		CCA [16]	$t_{adder}+t_{mux}$	$A_{adder}+A_{mux}$
		GCSA [11]	$O(\log(k))$	$O(n\log(k))$
	Approximate Full Adders	LOA [10]	$O(\log(n-l))$	$A_{loa}+(l\times A_{OR})$
	Truncated Adders	TruA	$O(\log(n-l))$	$O((n-l)\log(n-l))$

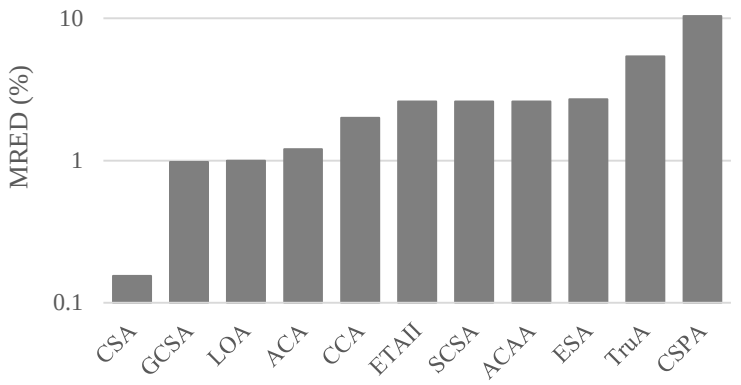
t_{adder}: $O(\log(k))$, A_{adder}: $O(n\log(k))$, A_{loa}: $O((n-l)\log(n-l))$, A_{carry}: circuit area of the carry prediction circuit

Accuracy Comparison

- 16-bit approximate adders with an equivalent 8-bit carry propagation



The error rate (ER) of approximate adders.



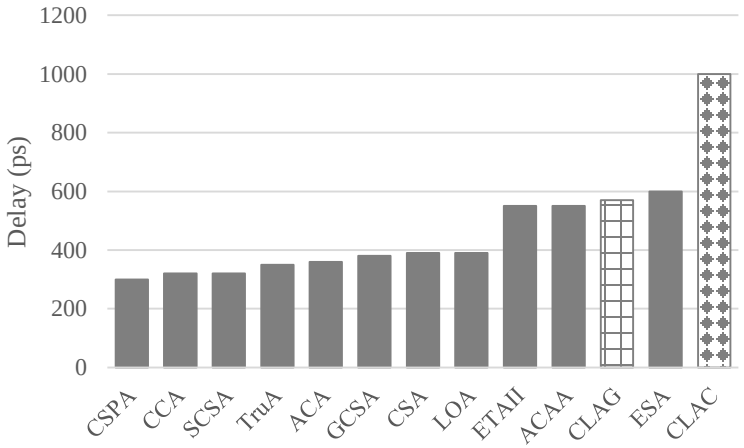
The mean relative error distance (MRED) of approximate adders.

- ETAII, ACAA and SCSA have the same error characteristics.
- The carry select adders (CSA, CSPA, CCA, GCSA) and the speculative adder (ACA) are very accurate with small values of ER and MRED (except for CSPA).
- The approximate full adder (LOA) has a moderate MRED but very large ER.
- The segmented adders (ESA, ETAII, ACAA) are not very accurate.
- The truncated adder (TruA) is the least accurate in terms of ER among the equivalent designs.

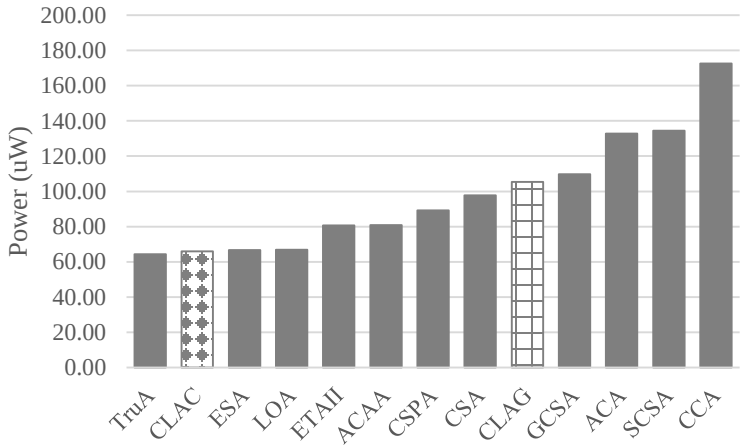
UNIVERSITY OF ALBERTA

Hardware Comparison

- Delay and power of 16-bit equivalent approximate adders



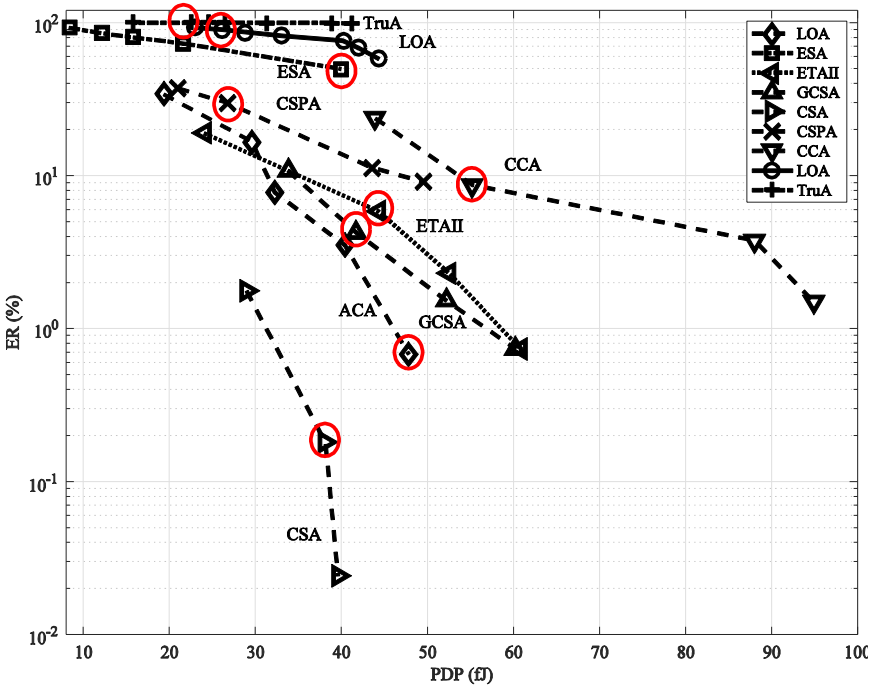
The delay of approximate adders.



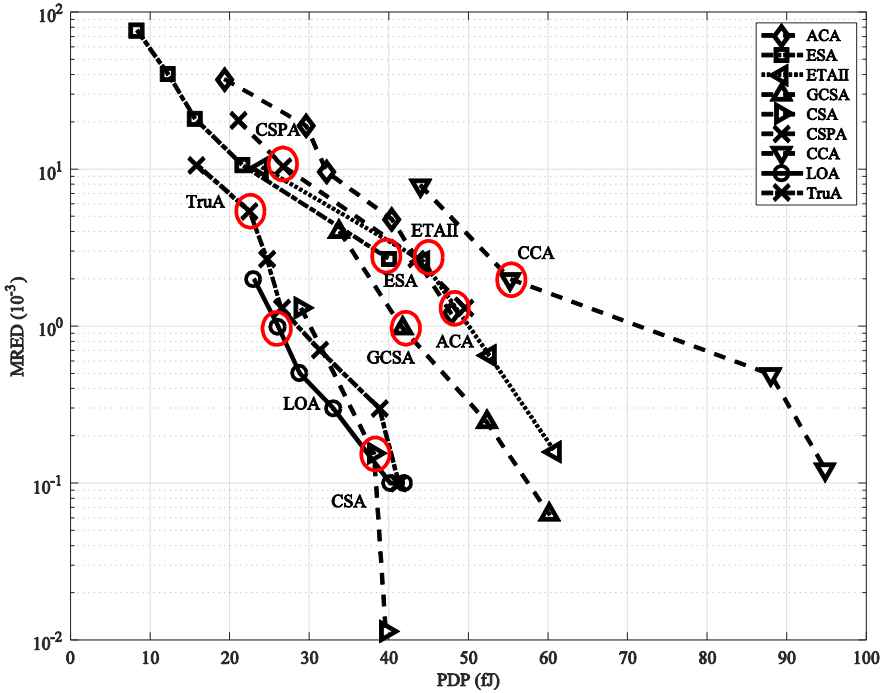
The power of approximate adders.

- The carry select adders (CSA, CSPA, CCA, GCSA) tend to consume large power at a relatively high performance.
- The speculative adder (ACA) is very fast but very power consuming.
- The approximate full adder (LOA) is slow, but it consumes a low power and area.
- The segmented adders (ESA, ETAII, ACAA) are power and area efficient.
- The truncated adder (TruA) is very power and area efficient, but with a relatively long delay.

Accuracy and Hardware Tradeoffs



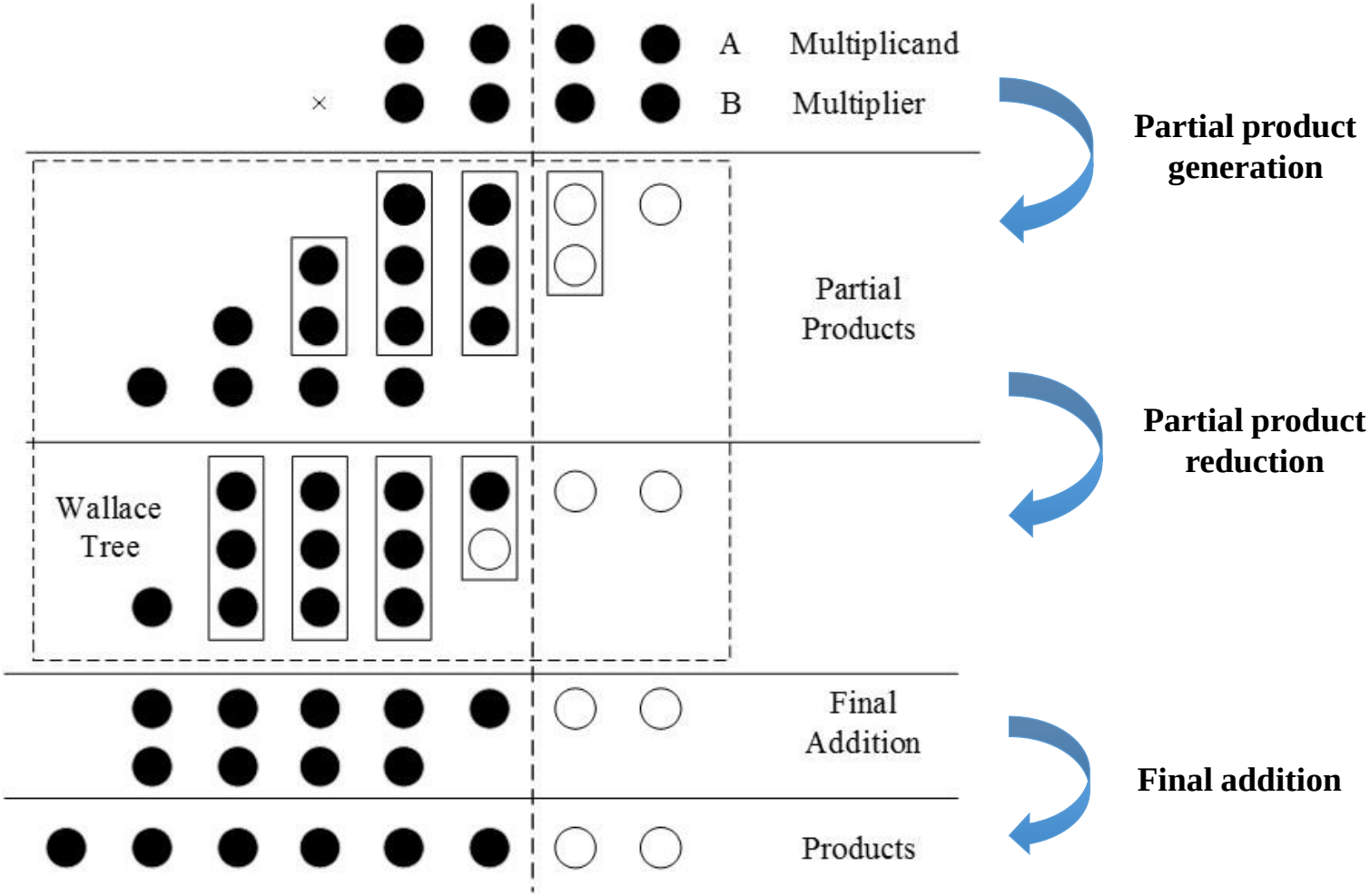
The ER and PDP of approximate adders.



The MRED and PDP of approximate adders.

- Error rate (ER) is the probability of producing an incorrect result.
- MRED (mean relative error distance (RED) is used to evaluate the mean relative difference between an approximate result and the accurate result.

Unsigned Multiplier with Wallace tree



$Critical\ path\colon\ O(\log(n))$

Approximate Multipliers: A Classification

We classify the approximate multipliers into four categories:

- **Approximation in Generating Partial Products**
  - Using simpler structure to generate partial products.
- **Approximation in the Partial Product Tree**
  - Omitting some partial products.
  - Dividing partial products into several sections and applying approximation in the less significant sections.
  - Truncated multipliers with a lower precision in input operands.
- **Using Approximate Counters or Compressors in the Partial Product Tree**
  - Approximating adders, counters or compressors
- **Approximate Booth Multipliers**

Approximate Multipliers

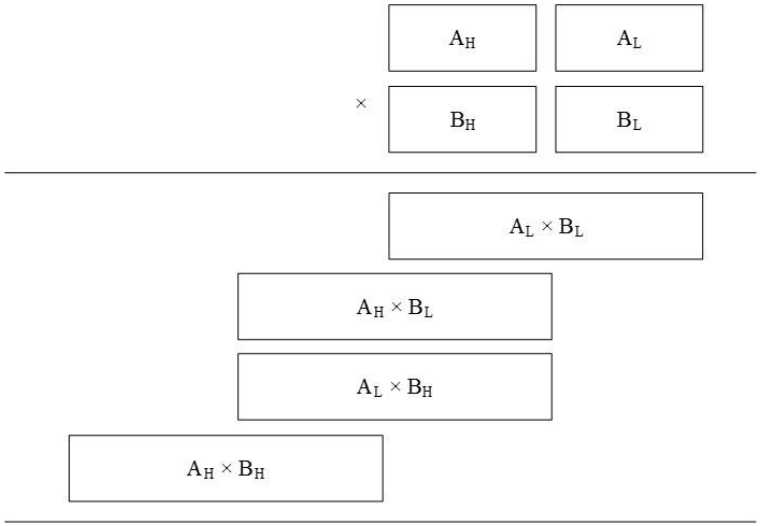
Classification	Multiplier
Approximation in Generating partial products	Under-Designed Multiplier (UDM) [17]
Approximation in the partial products	Broken Array Multiplier (BAM) [10] Error Tolerant Multiplier (ETM) [19] Approximate Wallace Tree Multiplier (AWTM) [20] Truncated Wallace Multiplier (TruMW) Truncated Array Multiplier (TruMA)
Using approximate counters or compressors	Inaccurate Compressor based Multiplier (ICM) [21] Approximate Compressor based Multiplier (ACM) [22] Approximate Multiplier 1/2 (AM1/AM2) [18] Truncated AM1/AM2 (TAM1/TAM2) [23]
Approximate Booth multipliers	Fixed-width Booth multipliers

Approximation in Generating Partial Products

The Underdesigned Multiplier (UDM):

K-Map for the 2 x 2 bit underdesigned multiplier block

A_1A_0 \ B_1B_0	00	01	11	10
00	000	000	000	000
01	000	001	011	010
11	000	011	**1001**	110
10	000	010	110	100

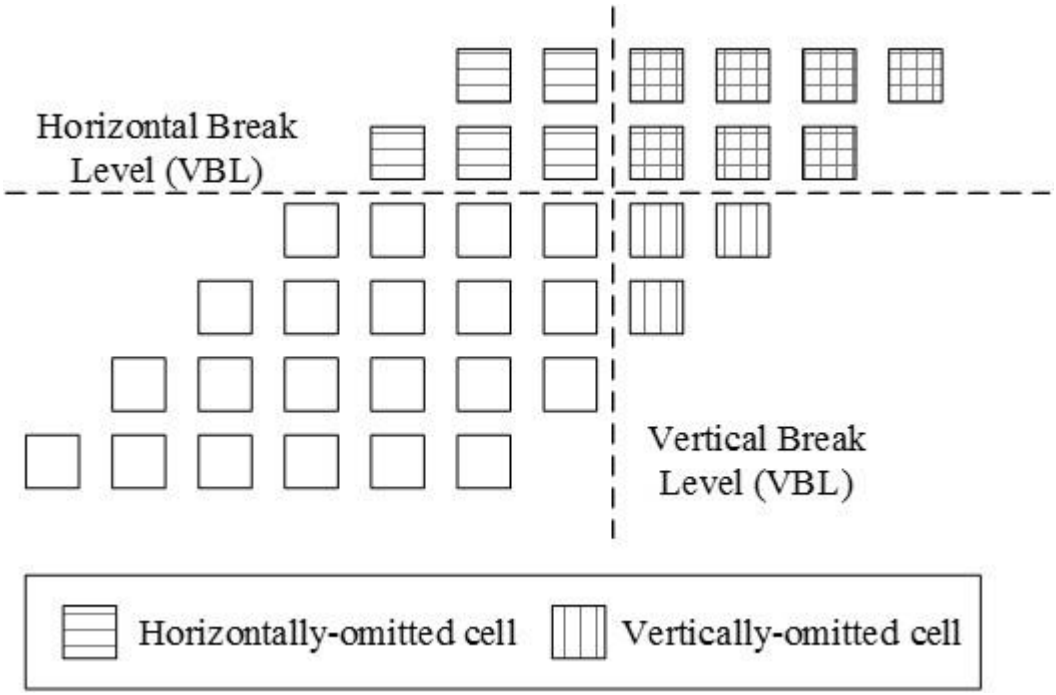


A 4 x 4 bit multiplier built on 2 x 2 bit block.

[17] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in Proceedings of the 24th IEEE International Conference on VLSI Design, 2011, pp. 346–351.

Approximation in the Partial Product Tree

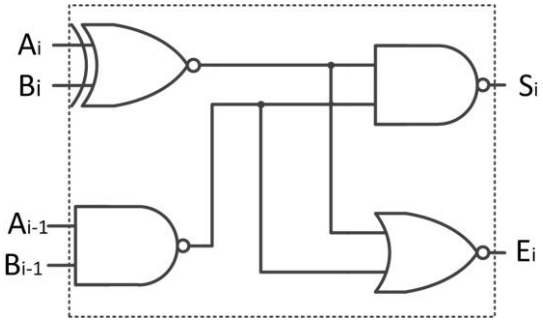
The Broken-Array Multiplier (BAM):



[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," IEEE Transactions on Circuits and Systems, vol. 57, no. 4, pp. 850–862, Apr. 2010.

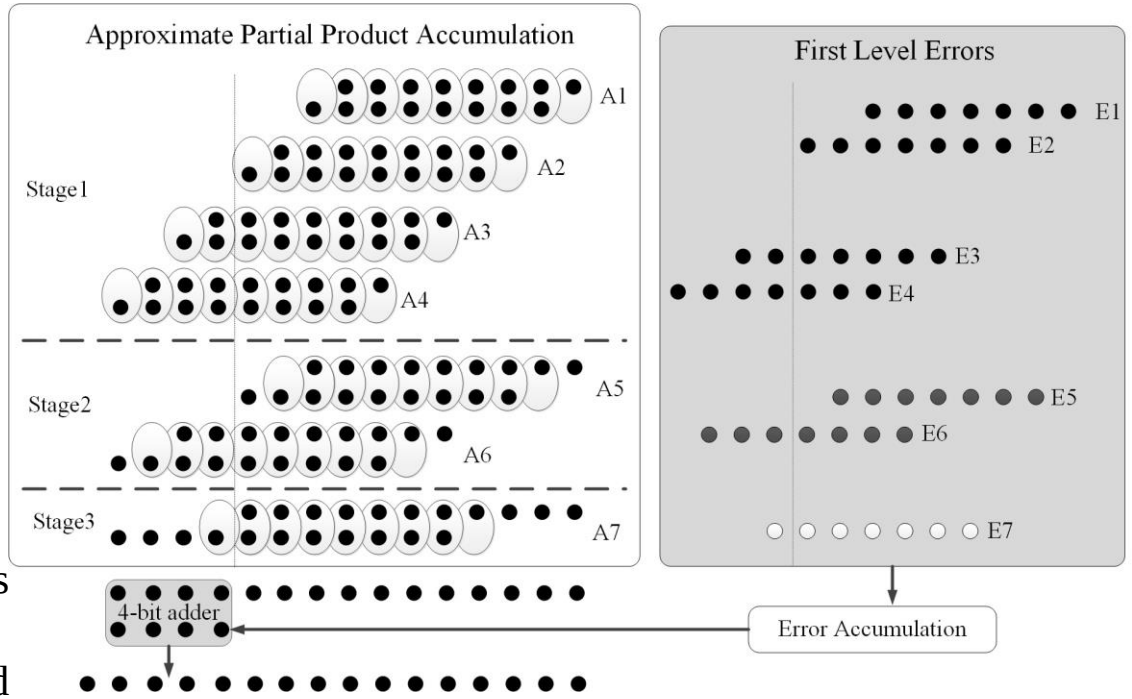
Approximate Multiplier using Approximate Counters or Compressors

Approximate Multiplier (AM) with Configurable Partial Error Recovery and Truncated AM (TAM):



The approximate adder cell.

- Two error accumulation trees for AM1 and AM2.
- TAM1 and TAM2 are obtained by truncating half LSBs of the partial products.

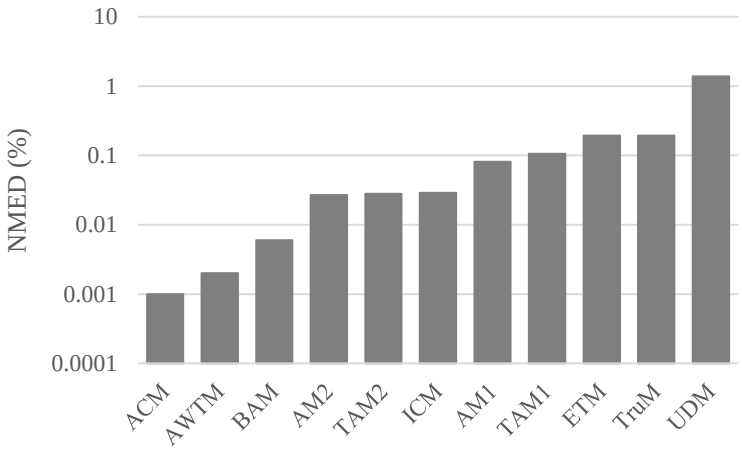


The approximate multiplier with 4-bit error recovery.

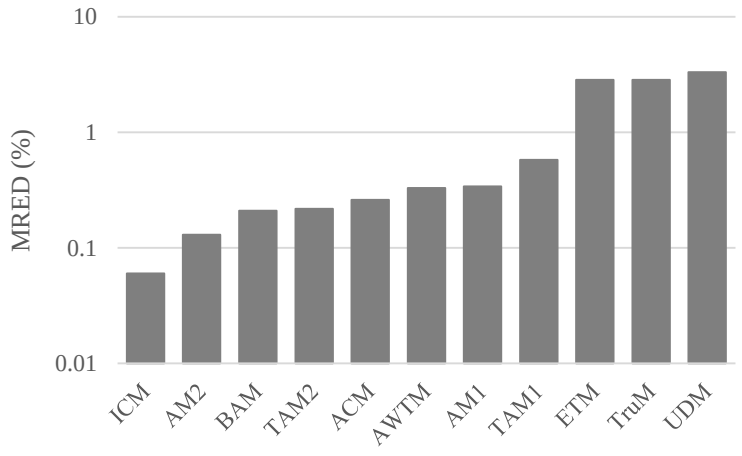
[19] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," DATE, 2014.

Accuracy Comparison

- Equivalent 16x16 approximate multipliers with16-bit accurate MSBs



The normalized mean error distance (NMED)

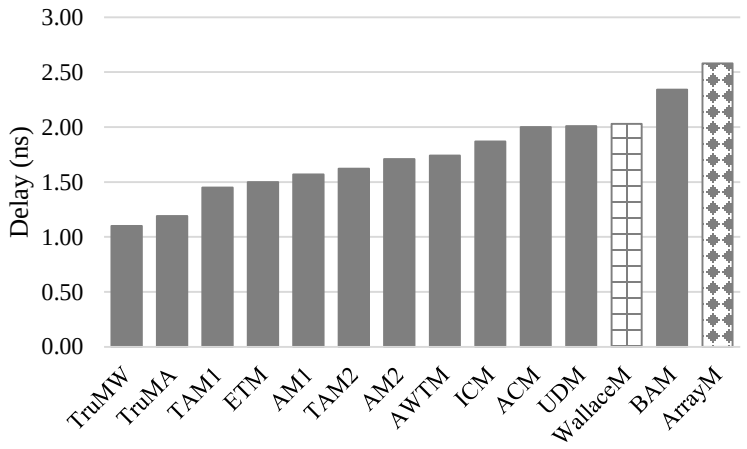


The mean relative error distance (MRED)

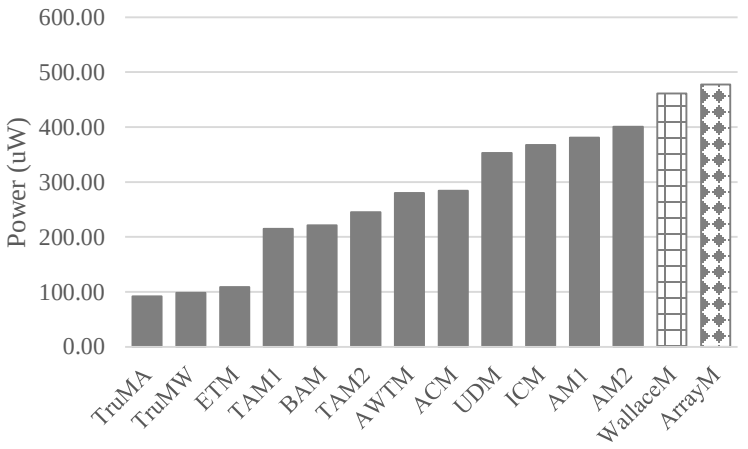
- The multiplier approximated in generating the partial product (UDM) has very large values of NMED and MRED.
- The multipliers approximated in the partial product tree (AWTM) mostly have relatively small NMEDs and moderate MREDs.
- The multipliers approximated using approximate counters or compressors (ICM, ACM, AM1/2) have smaller NMED and MRED.
- The truncated multiplier (TruM) has large values of both NMED and MRED.

Hardware Comparison

- Delay and power of equivalent approximate multipliers:



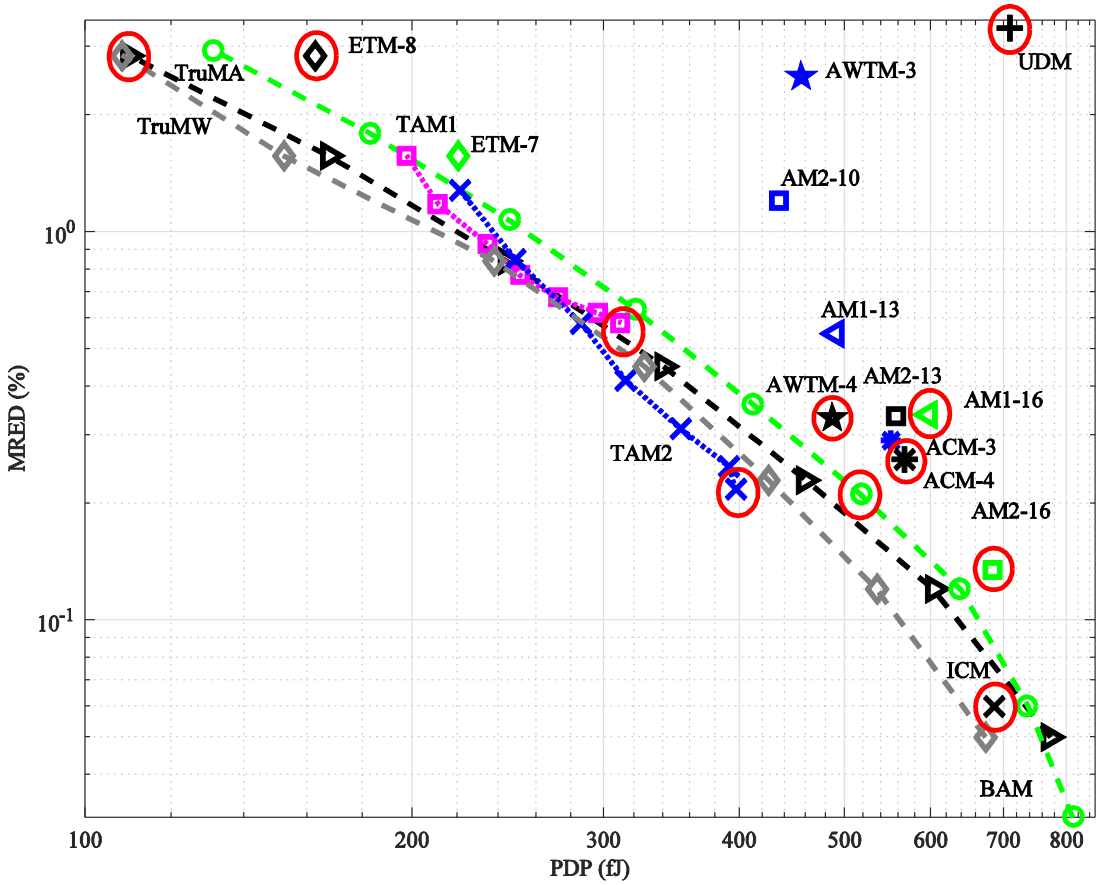
Delay



Power

- TruMA, TruMW, ETM, TAM1/TAM2 and BAM have smaller delay and power dissipation due to truncation.
- The multiplier approximated in generating the partial product (UDM) tends to have a large delay and power.
- The multipliers approximated in the partial product tree (AWTM) have moderate delay and power.
- The multipliers approximated using approximate counters or compressors (ICM, ACM, AM1/2) require higher power dissipation.

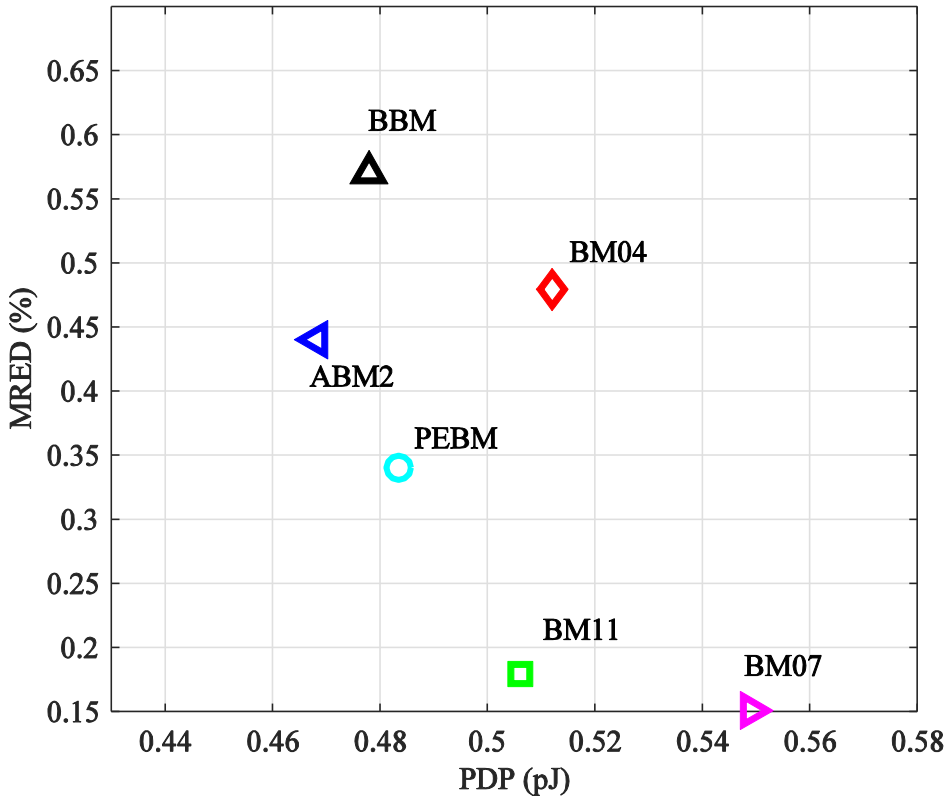
Accuracy and Hardware Tradeoffs (Unsigned)



[24] H. Jiang, J. Han, and F. Lombardi, "A comparative evaluation of approximate multipliers," IEEE Nanoarch Symposium, Beijing, China, 2016.

- Truncation is effective to save hardware. However, it incurs a large ER and moderate MRED.
- ICM and UDM have low ERs, but their MREDs are usually large because of the large errors that may occur in the more significant part of the multiplier. Moreover, they usually have rather high PDPs.

Accuracy and Hardware Tradeoffs (Booth)



- ABM2 shows the lowest PDP and a moderate accuracy.
- BM11 and BM07 area very accurate in terms of MRED but with relatively poor PDPs.
- PEBM shows both moderate PDP and MRED.

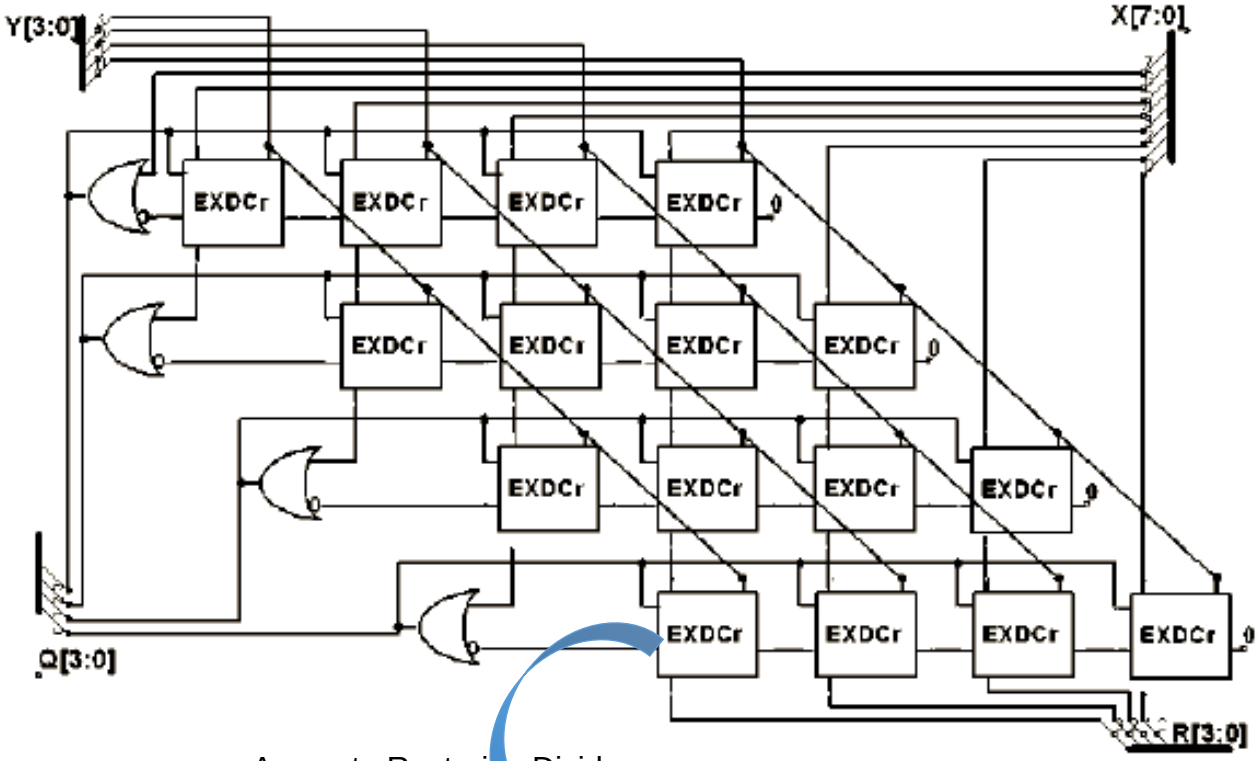
[25] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate Radix-8 Booth Multipliers for Low-Power and High-performance Operation," IEEE Transactions on Computers, 65, 8: 2638–2644, 2016.

Approximate Dividers: A Classification

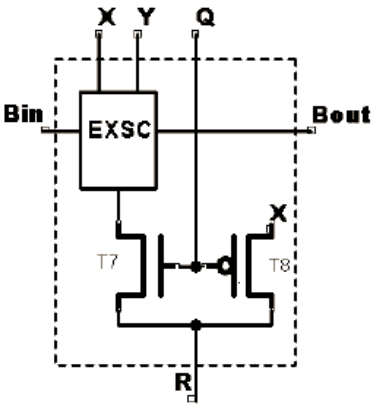
We classify approximate multipliers into three categories:

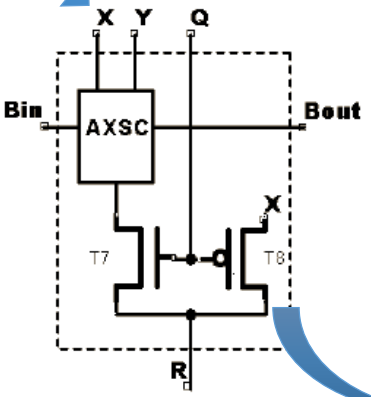
- **Approximate Array Dividers**
  - Using approximate subtractor cells or a smaller array divider.
- **Curve Fitting based Approximate Dividers**
  - Using curve fitting to approximate the binary logarithmic and antilogarithmic values.
  - Adders and subtractors are sufficient for a division.
- **Rounding based approximate dividers**
  - Transforming division to multiplication by rounding the divisor.

Approximate Array Dividers

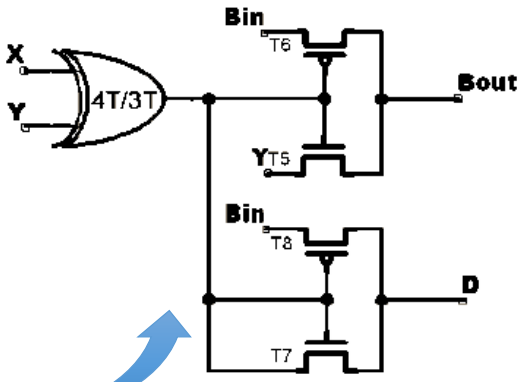


Accurate Restoring Divider





Accurate and Approximate Divider Cells



Approximate Subtractor Cell

[26] L. Chen, J. Han, W. Liu and F. Lombardi, "On the Design of Approximate Restoring Dividers for Error-Tolerant Applications," IEEE Transactions on Computers, vol. 65, no. 8, pp. 2522 - 2533, 2016.

$$D = X \oplus Y + B_{in}$$
$$B_{out} = \overline{X \oplus Y} \cdot B_{in} + \overline{X}Y$$

Approximate Divider Designs

Classification	Divider
Approximation array dividers	Approximate non-restoring divider (AXDnr) [26] Approximate restoring divider (AXDr) [27] Dynamic approximate divider (DAXD) [28]
Curve fitting based approximate dividers	High-speed divider (HSD) [29] Floating-point divider (FPD) [30]
Rounding based approximate dividers	High-speed, energy-efficient, rounding-based approximate divider (SEERAD) [31]

Image Processing Applications

Image sharpening using approximate adders and multipliers

Images sharpened using different adder and multiplier pairs.

Approximate Design	LOA-16	CSA-8	ETAII-8	CSPA-8
TAM1-16				
AM1-13				

Image Sharpening (cont'd)

Images sharpened using different adder and multiplier pairs. (cont'd)

Approximate Design	LOA-16	CSA-8	ETAII-8	CSPA-8
TAM2-13				
TAM1-13				

Image Sharpening (cont'd)

Images sharpened using different adder and multiplier pairs. (cont'd)

Approximate Design	LOA-16	CSA-8	ETAII-8	CSPA-8
BAM-17				
BAM-18				

Image Sharpening: Accuracy

Peak signal-to-noise ratios (PSNRs) of the sharpened images (*dB*).

Approximate Design	LOA-16	CSA-8	ETAII-8	CSPA-8
TAM1-16	46.97	46.97	46.97	25.01
AM1-13	45.21	45.06	36.86	24.20
TAM2-13	41.87	41.87	41.87	24.32
TAM1-13	41.42	41.42	41.42	24.35
BAM-17	40.09	40.09	40.09	25.19
BAM-18	33.99	33.99	33.99	24.21

- The images sharpened by CSPA have unacceptable defects, and some defects can be seen in the image sharpened by AM1-13 and ETAII-8.
- Other images show similar quality with the accurate result.
- The PSNRs of the images sharpened by a truncation based multiplier are fixed as the adder is changed among LOA-16, CSA-8 and ETAII-8.

Image Sharpening: Hardware

Delay, power and area of image sharpening using approximate multipliers and adders.

Multiplier	Adder	Delay (*ns*)	Power (*mW*)	PDP (*pJ*)	Area (um^2)	ADP ($um^2.ns$)
ArrayM	**CLAG**	**6.74**	**1.995**	**13.45**	**31,183.9**	**210,179.5**
TAM1-16	LOA-16	5.36	0.9723	5.215	18,139.0	97,225.0
TAM1-16	CSA-8	7.45	1.032	7.69	23,652.1	176,208.1
TAM1-16	ETAII-8	5.34	0.9643	5.15	18,056.8	96,423.3
AM1-13	LOA-16	5.41	1.193	6.45	26,644.0	144,144.0
AM1-13	CSA-8	7.41	1.377	10.20	30,586.5	226,646.0
AM1-13	ETAII-8	6.40	1.369	8.76	28,214.7	180,574.1
TAM2-13	LOA-16	5.25	1.055	5.54	17,057.8	89,553.5
TAM2-13	CSA-8	6043	1.053	6.77	20,526.6	131,986.0
TAM2-13	ETAII-8	5.22	1.041	5.43	16,975.6	88,612.6

- For the same multiplier, LOA-16 and ETAII-8 result in similar delay, power and area (except for AM1-13), while the implementations using CSA-8 result in relatively larger values of these metrics.

Image Sharpening: Hardware (cont'd)

Delay, power and area of image sharpening using approximate multipliers and adders. (cont'd)

Multiplier	Adder	Delay (*ns*)	Power (*mW*)	PDP (*pJ*)	Area (um^2)	ADP ($um^2.ns$)
ArrayM	**CLAG**	**6.74**	**1.995**	**13.45**	**31,183.9**	**210,179.5**
TAM1-13	LOA-16	5.25	0.9467	4.97	17,221.0	90,410.3
TAM1-13	CSA-8	7.45	0.9942	7.41	22,734.1	169,369.0
TAM1-13	ETAII-8	5.34	0.9350	4.88	17,138.8	89,464.5
BAM-17	LOA-16	6.14	1.226	7.53	14,993.8	92,061.9
BAM-17	CSA-8	7.36	1.247	9.17	16,533.0	121,682.9
BAM-17	ETAII-8	6.13	1.211	7.42	14,868.5	91,143.9
BAM-18	LOA-16	5.97	1.097	6.55	13,285.3	79,313.2
BAM-18	CSA-8	6.89	1.117	7.70	16,901.8	116,453.4
BAM-18	ETAII-8	5.96	1.076	6.41	13,156.0	78,409.8

- Using the same adder, the image sharpening circuits show similar measurements except that AM1-13, BAM-17 and BAM-18 based schemes show slightly larger values.

Image Sharpening: Hardware Comparison

Compared to the accurate design

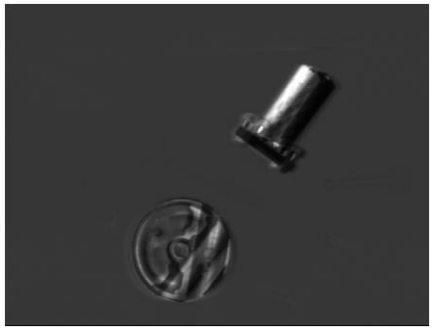
- The approximate designs using CSA-8 or AM1-13 achieve small improvement in terms of delay and area.
- By using LOA-16, ETAII-8, TAM2-13, BAM-17 or BAM-18, the image sharpening circuit can be 23% faster and saves as much as 53% in power and 58% in area.
- The PDP and ADP are improved by 64% and 62%, respectively, by using LOA-16, ETAII-8, TAM2-13, BAM-17 or BAM-18 for the image sharpening circuit.

Image Processing Applications for Dividers

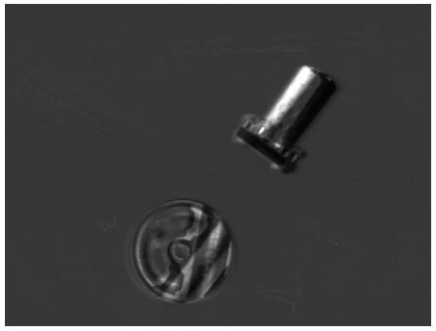
Change detection using approximate dividers

Input image 1

Input image 2

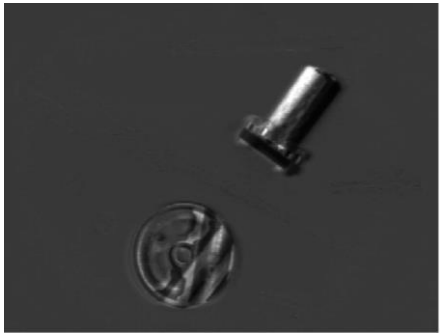


Accurate output



AXDr1 (42.75 *dB*)

AXDr2 (33.22 *dB*)



AXDr3 (43.38 *dB*)

- AXDr1 and AXDr3 with the triangle replacement of depth 8 perform well, while AXDr2 has a relatively lower performance.

Change Detection (cont'd)

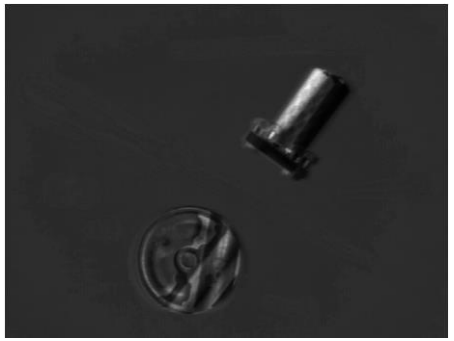
DAXD8 (21.26 *dB*)

DAXD10 (25.03 *dB*)

SEERAD1 (21.45 *dB*)

SEERAD2 (24.69 *dB*)

SEERAD3 (24.67 *dB*)



SEERAD4 (26.84 *dB*)

- The results by SEERAD4 is very good, while the results by DAXD8 and SEERAD1 are of low quality.

Change Detection: Hardware

Delay, power and area of change detection using approximate dividers.

Multiplier	PSNR (*dB*)	Delay (*ns*)	Power (*uW*)	PDP (*pJ*)	Area (um^2)	ADP ($um^2.ns$)
ArrayD	**--**	**4.08**	**54.29**	**221.50**	**425.8**	**1,737.2**
AXDr1	42.75	3.85	50.71	195.23	415.5	1,599.7
AXDr2	33.22	4.36	54.08	235.79	408.2	1,779.6
AXDr3	43.38	4.58	40.55	185.72	376.2	1,722.9
DAXD10	25.03	2.43	40.25	97.84	375.7	912.9
SEERAD3	24.67	1.83	60.27	110.29	615.8	1,126.8
SEERAD4	26.84	2.43	70.62	181.33	765.4	1,859.9

- The array-based dividers (ArrayD, AXDrs and DAXDs) are power and area efficient with a very low speed.
- The rounding based approximate dividers (SEERADs) are very fast, but they consumes more power and area due to the use of loop-up tables.

Summary

- Approximate computing is emerging as a paradigm for energy-efficient and/or high-performance design.
- It covers a broad spectrum of research from circuits and systems, to software and application levels.
- A classification and comparison of approximate adders and multipliers show that
  - Truncation is effective and introduces a low error distance but a high error rate.
  - It does not significantly improve performance.
  - The performance of other types of design varies (discussed as follows).

Summary on Approximate Adders

- In general, approximate speculative adders show high accuracy and relatively small PDPs.
- The approximate adders using approximate full adders in the LSBs are slow, but they are power efficient with high ERs (due to the approximate LSBs) and moderate NMED and MRED values (due to the accurate MSBs).
- The error and circuit characteristics of the segmented and carry select adders vary with the predictions of carry signals.
- With the highest ER, a truncated adder has a smaller MRED (an indicator of a smaller error magnitude) than most approximate designs at a similar PDP (except for LOA and CSA).
  - However, it has a lower performance than other approximate designs.
  - Due to the low power dissipation, it is useful in applications in which hardware efficiency is of the utmost importance.

Summary on Approximate Multipliers

- Truncation is an effective scheme to reduce hardware. For a similar PDP, it results in a moderate MRED (an indicator of the error magnitude) that is smaller than most other approximate designs, except for TAM1, TAM2 and ICM.
- Albeit with a relatively low ER, UDM shows a low accuracy in terms of the error distance and a relatively high circuit overhead, whereas ICM has the lowest ER among all designs.
- When truncation is not used, multipliers approximated in the partial product tree tend to have a poor accuracy (except AWTM-3 and AWTM-4) and moderate hardware consumption.
- Multipliers using approximate counters or compressors are usually very accurate with relatively high power dissipation and hardware consumption.
- The approximate Booth multipliers show different characteristics in hardware efficiency and accuracy.

Summary on Approximate Dividers

- The approximate array dividers are slow, but they are hardware efficient with variable accuracy depending on the approximation parameters.
- The dividers based on curve fitting are very accurate and fast but they require a large area and high power dissipation due to the utilization of look-up tables.
- The rounding based approximate dividers have a very high speed, large area and power dissipation, with a relatively low accuracy.

Summary on Image Processing Application

- The image sharpening circuits using approximate adders and multipliers achieves significant savings in hardware, while producing similar results as the accurate design.
- The change detection circuits using the approximate array dividers (AXDr1 and AXDr3) are power and area efficient but very slow.
- The circuit using the rounding based approximate divider (SEERAD4) consumes more power and area with a high performance for an excellent detection accuracy.

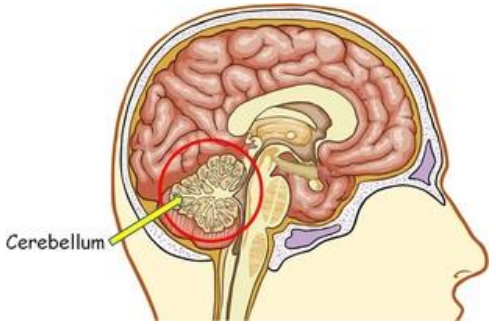
Efficient Implementation of Cerebellar Models using Approximate Circuits

- **Background and Motivation**
- **Cerebellar Model Design**
  - Adaptive Filter based Cerebellar Model
  - Proposed Hardware Implementation
- **Evaluation**
  - Accuracy
  - Circuit measurements
- **Conclusion**

[37] H. Jiang, L. Liu and J. Han, “Special Session Paper: An Efficient Hardware Design for Cerebellar Models using Approximate Circuits,” CODES/ISSS’17, October 15-20, 2017, Seoul, Korea.

Background: Cerebellum

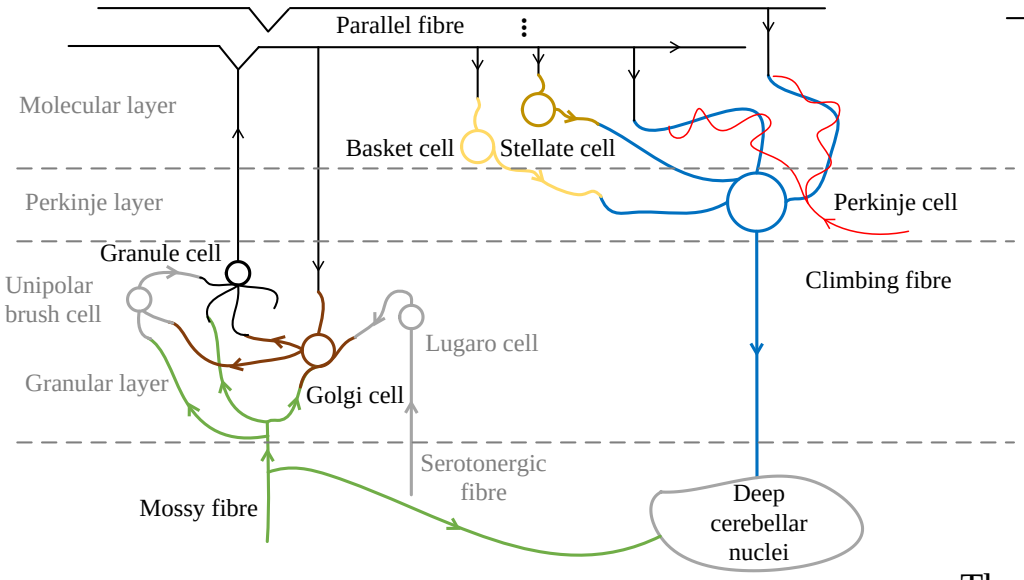
- The cerebellum is a very important part of the brain.
  - Keeping balance
  - Smoothing movements
  - Coordinating muscles
- Cerebellar models
  - Perceptron based model
  - Continuous spatio-temporal model
  - Higher-order lead-lag compensator model
  - Adaptive filter based model



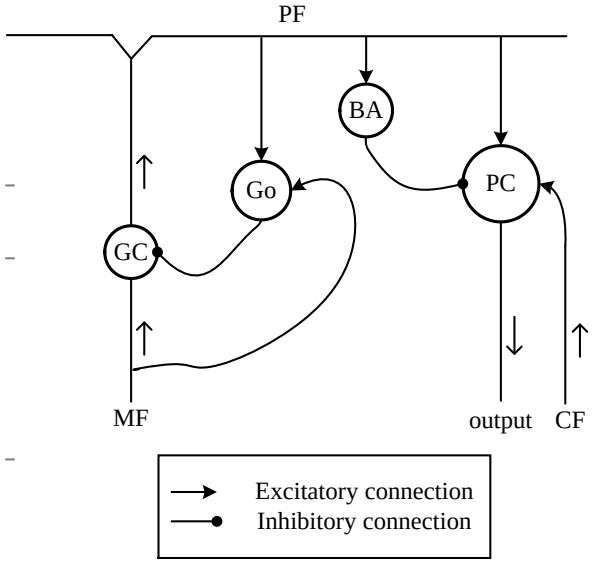
- The adaptive filter based cerebellar model is the most widely used due to its low complexity and high structure-resemblance to the cerebellum.
- The cerebellar models are inherently error-tolerant.

Cerebellar Model Design

- The cerebellar cortex consists of three layers.
  - The granular layer: Granule cell (GC), Golgi cell (Go), mossy fibre (MF), Lugaro cell, unipolar brush cell
  - The Perkinje layer: Perkinje cell (PC) bodies
  - The molecular layer: basket and stellate cells (BA), parallel fibre (PF), climbing fibre (CF)
- The learning ability of the cerebellum is related to the plasticity of synaptic weights.



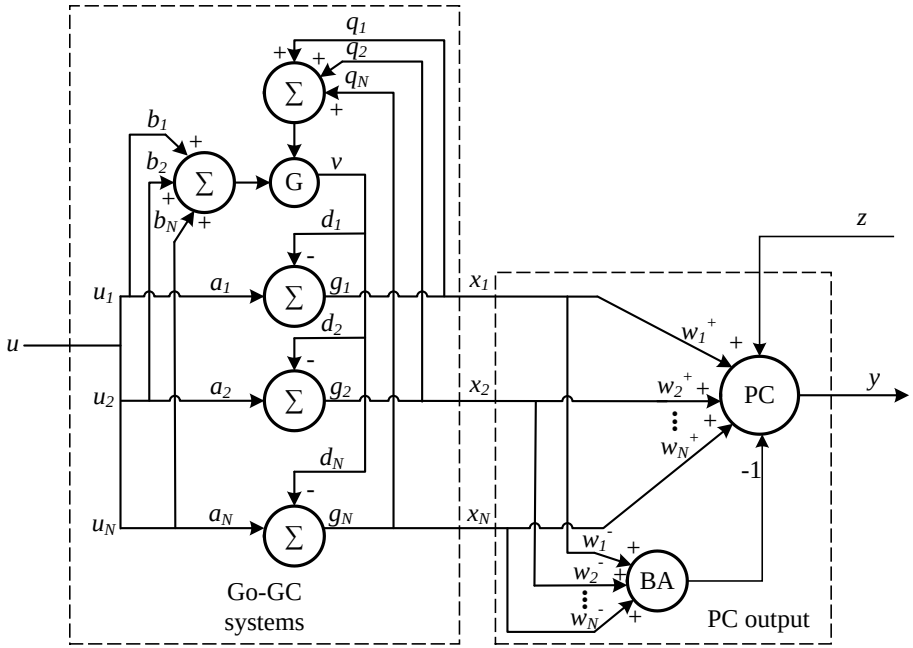
The connection networks of cerebellar cells [38].



The internal connections of cerebellar cells [39].

Adaptive Filter based Cerebellar Model

- The model acts as a lead-lag compensator with learning capability.
- The Go is assumed to be a lag element with a time constant of T.
- The Go-GC is simplified to a lead-lad compensator.
- The cerebellar cortex consists of three layers.



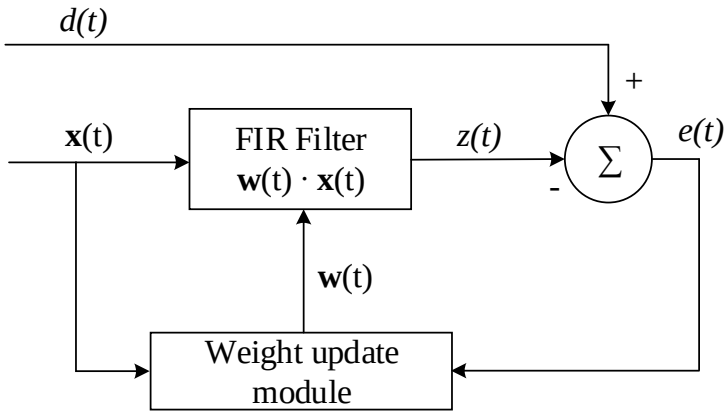
$$y = \sum_{j=1}^{N} w_j^{+} x_j - \sum_{j=1}^{N} w_j^{-} x_j = \sum_{j=1}^{N} w_j x_j$$

$$dw_j^{+}(t) = -\mu z(t) x_j(t) dt$$
$$dw_j^{-}(t)/dt = 0$$

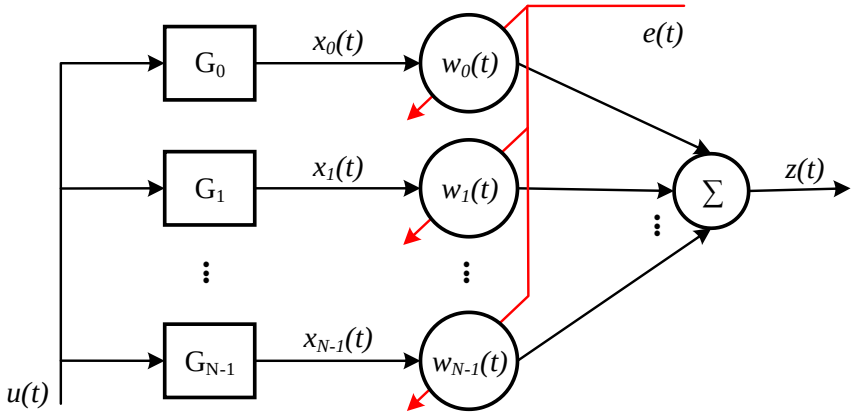
Block diagram of the cerebellar model based on the adaptive filter [40].

[40] M. Fujita, “Adaptive filter model of the cerebellum. Biological cybernetics,” 45(3):195–206, 1982.

Adaptive Filter



An adaptive filter.



Simplified adaptive filter based cerebellar model [41].

Error computation

$$z(t) = \sum_{i=0}^{N-1} w_i(t) \cdot x_i(t)$$

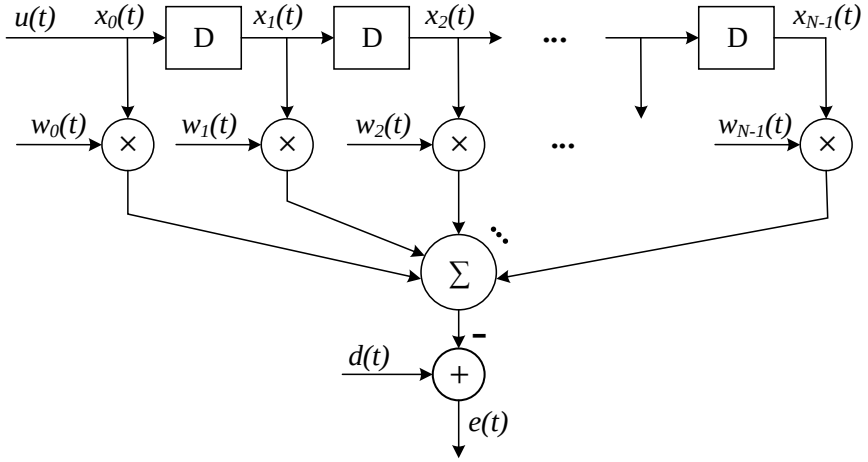
$$e(t) = d(t) - y(t)$$

Weight update using
the least mean square algorithm

$$w_i(t+T) = w_i(t) + \mu \cdot e(t) \cdot x_i(t)$$

[41] Alexander Lenz, Sean R Anderson, Anthony G Pipe, Chris Melhuish, Paul Dean, and John Porrill, "Cerebellar-inspired adaptive control of a robot eye actuated by pneumatic artificial muscles," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 39, 6: 1420–1433, 2009.

Proposed Hardware Implementation

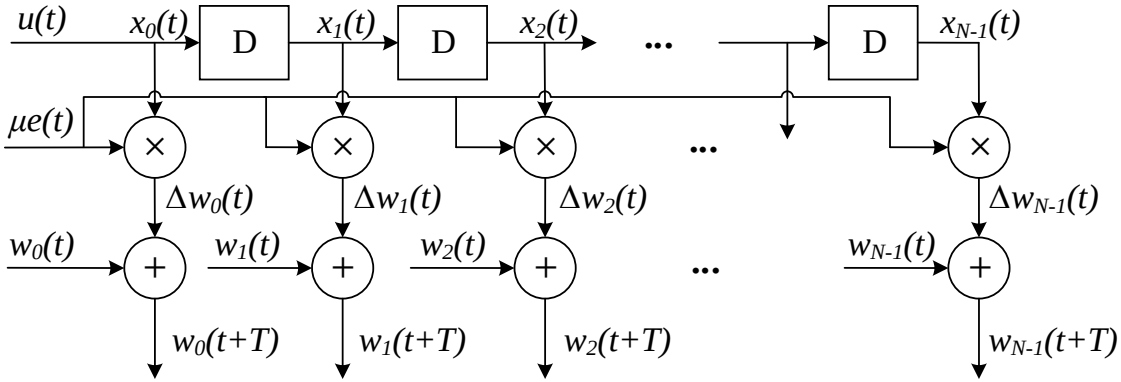


Hardware implementation of error computation module.

×: Approximate radix-8 Booth multiplier [25]

+: Lower-part-OR adder [10]

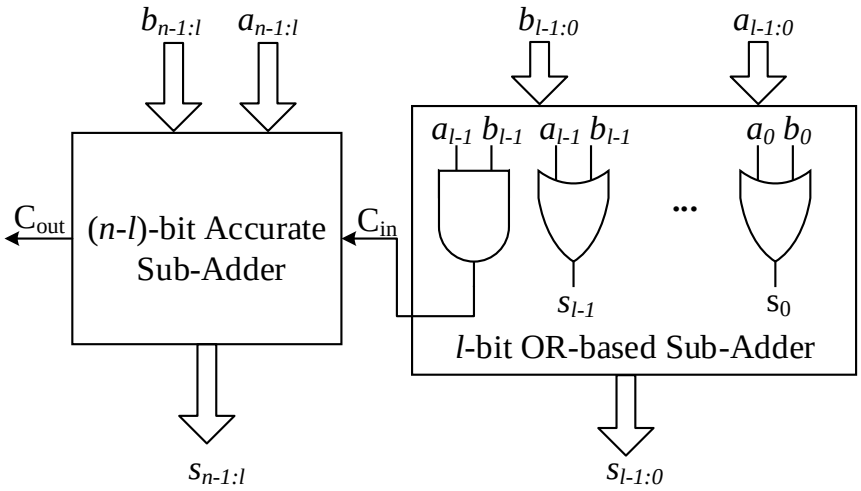
3N multipliers and 2N adders are required for an N-tap adaptive filter.



Hardware implementation of weight update module.

Approximate Arithmetic circuits

- Approximate radix-8 Booth multiplier (ABM2_C15) [25]
  - ABM2_C15 is redesigned as an $n \times n$ fixed-width multiplier.
  - Approximate recoding adder with n approximated bits.
  - (n-1) least significant bits of partial products are truncated.
- Lower-part-OR adder (LOA) [10]



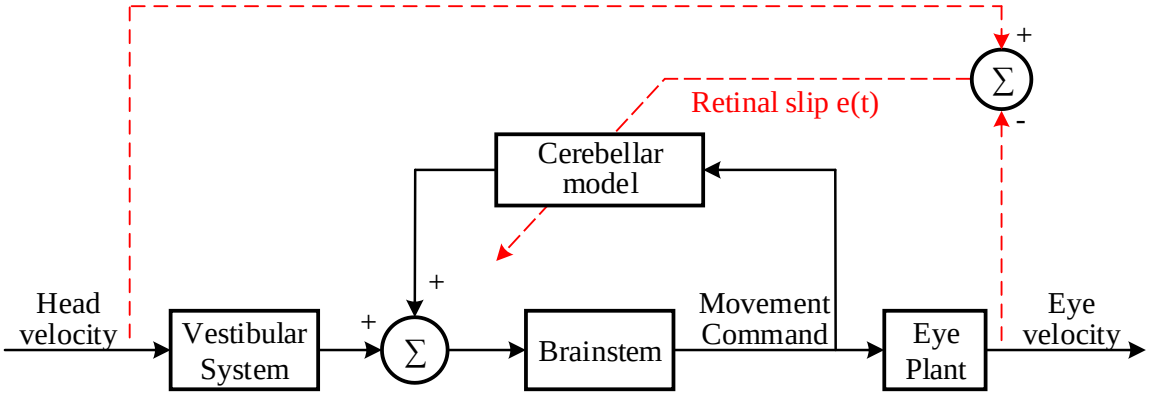
System Architecture for VOR

- Vestibulo-ocular reflex (VOR)
  - The VOR stabilizes a stimulus into the center of the retina for a clear vision when the head moves .
  - The cerebellum predicts the eye plant output to compensate the movement command.
  - The horizontal canal in the vestibular system is modeled as a high-pass filter [42].

$$V(s) = \frac{s}{s + 1/T_c}$$

  - The transfer functions of the brainstem and the eye plant are given by [43].

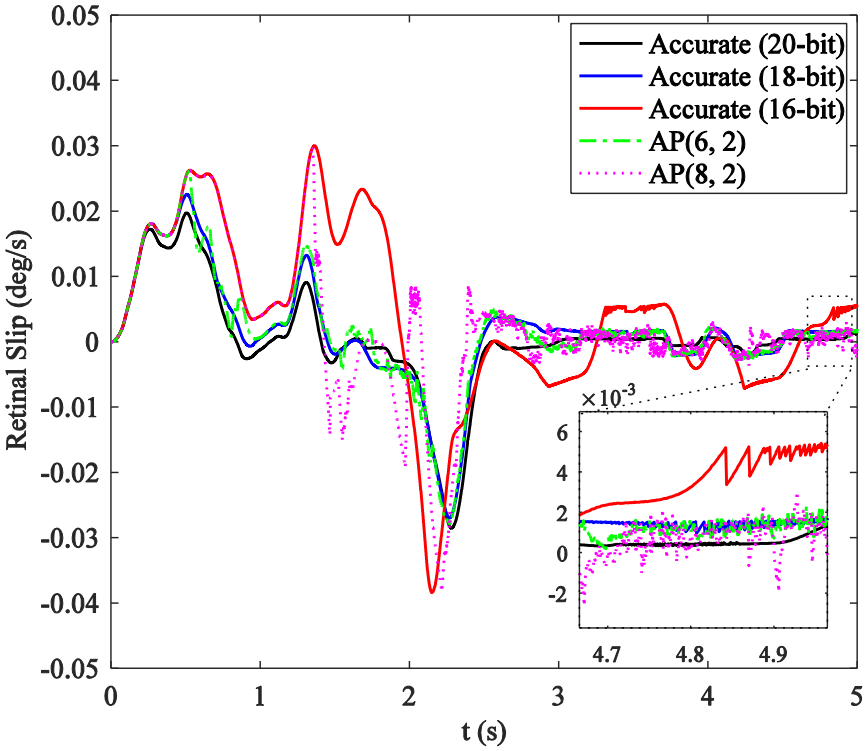
$$B(s) = G_d + \frac{G_i}{s + 1/T_i} \qquad P(s) = \frac{s(s + 1/Tz)}{(s + 1/T_1)(s + 1/T_2)}$$



The simplified model of the vestibule-ocular reflex in a saccade system.

Simulation Results: Accuracy

- Parameters
  - The constant delay T is $1ms$.
  - The length of the adaptive filter N is 128.
  - The step size μ is 2^{-8}.
- Accuracy
  - The accurate 20-bit fixed-point cerebellar model shows the lowest stable retinal slip.
  - The retinal slip of the 16-bit implementation does not converge.
  - AP (6, 2), all multipliers are implemented by 20×20 approximate Booth multipliers, the adder tree in the error computation module is implemented by LOA-6's, LOA-2's are used in the weight update module.
  - AP(6, 2) generates a similar retinal slip with Accurate (18-bit), while the retinal slip of AP(8, 2) converges to a slightly larger value.



The output retinal slip during a $5s$ VOR training.

Simulation Results: Hardware

- Synthesis tool and configuration
  - Synopsys design compiler
  - STM 28nm CMOS process
  - The clock frequency is 125MHz.
  - The supply voltage is 1V.
- Circuit measurements
  - With a similar accuracy, the AP(6, 2) is faster by 17.3%, and consumes a smaller area by 37.3% and a lower power by 29.7% than the accurate 18-bit design.
  - A saving of 41.9% (power-delay product (PDP) is obtained by using approximate multipliers and adders in the adaptive filter based cerebellar model.

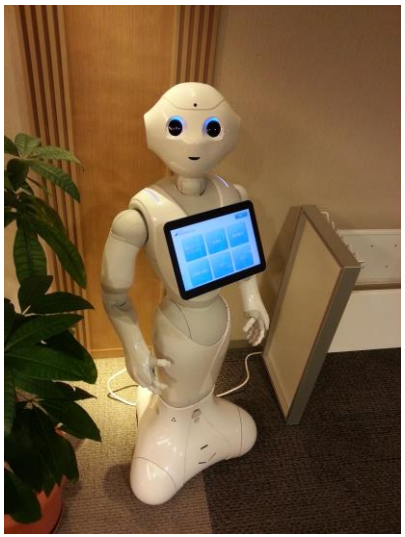
Design	Delay (ns)	Area (um^2)	Power (mW)	PDP (pJ)
Accurate (18-bit)	7.01	332,616	12.08	84.68
AP (6,2)	5.80	208,696	8.49	49.24
AP (8,2)	5.76	207,274	8.41	48.44

Conclusion

- An efficient hardware design is proposed for the adaptive filter based cerebellar model.
- Approximate multipliers and approximate adders are used in the proposed design.
- The simulation results show that the approximate cerebellar model (AP(6, 2)) achieves a similar accuracy as the exact 18-bit design.
- AP(6, 2) is more efficient in hardware than the accurate 18-bit design, with a reduction of PDP by 41.9%.

Prospects

- In parallel with the advances in approximate computing, brain-inspired computing has gained momentum. Approximate computing techniques appear promising to be integrated into the algorithms and architectures of a brain-inspired computing system.
- Approximate arithmetic circuits are applicable in many computational models for robotic control (as robot brains).
- Approximate computing is appealing to implementations of deep neural networks (DNNs); it remains to be investigated.


Acknowledgement

- **Graduate and undergraduate students:**
  - Honglan Jiang, Cong Liu and Jinghang Liang
  - Several summer intern students

- **Collaborators:**
  - Prof. Fabrizio Lombardi (Northeastern University, Boston, MA, USA)
  - Prof. Leibo Liu and Prof. Fei Qiao (Tsinghua University, Beijing, China)
  - Prof. Weiqiang Liu (Nanjing University of Aeronautics and Astronautics, Nanjing, China)
  - Prof. Lukáš Sekanina (Brno University of Technology, Brno, Czech Republic)

- **Funding:**
  - Natural Sciences and Engineering Research Council (NSERC) of Canada
  - University of Alberta China Opportunity Grants

UNIVERSITY OF ALBERTA

References

[1] J. von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” Automata Studies, Shannon C.E. & McCarthy J., eds., Princeton University Press, pp. 43-98, 1956.

[2] J. Han and M. Orshansky, “Approximate Computing: An Emerging Paradigm For Energy-Efficient Design,” In ETS, pages 1-6, Avignon, France, 2013.

[3] S. Venkataramani, S.T. Chakradhar, K. Roy, and A. Raghunathan, “Computing approximately, and efficiently,” In Design, Automation & Test in Europe Conference & Exhibition, pp. 748-751, 2015.

[4] A. Sampson, J. Nelson, K Strauss, and L. Ceze, “Approximate storage in solid-state memories,” ACM Transactions on Computer Systems (TOCS), vol. 32, no. 3, 9, 2014.

[5] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, “Architecture support for disciplined approximate programming,” ACM SIGPLAN Notices, vol. 47, no. 4, pp. 301-312, 2012.

[6] A. K. Verma, P. Brisk, and P. Ienne, “Variable latency speculative addition: A new paradigm for arithmetic circuit design,” In DATE, pages 1250 - 1255, 2008.

[7] N. Zhu, W. L. Goh, and K. S. Yeo, “An enhanced low-power high-speed adder for error-tolerant application,” In ISIC 2009, pages 69-72, 2009.

[8] K. Du, P. Varman, and K. Mohanram, “High performance reliable variable latency carry select addition,” In DATE, pages 1257-1262, 2012.

[9] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” IEEE Trans. CAD, 32(1):124-137, 2013.

[10] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, “Bio-Inspired Imprecise computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications,” IEEE Trans. Circuits Syst., 57(4):850-862, 2010.

[11] J. Hu and W. Qian, “A New Approximate Adder with Low Relative Error and Correct Sign Calculation,” In DATE. 1449–1454, 2015.

[12] D. Mohapatra, V.K. Chippa, A Raghunathan, and K. Roy, “Design of voltage-scalable meta-functions for approximate computing,” In DATE, pages 1–6, March 2011.

UNIVERSITY OF ALBERTA

References

[13] Andrew B Kahng and Seokhyeong Kang. Accuracy-configurable adder for approximate arithmetic designs. In Proceedings of the 49th ACM Annual Design Automation Conference, pages 820–825, June 2012.

[14] Yongtae Kim, Yong Zhang, and Peng Li. An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems. In ICCAD, pages 130–137, November 2013.

[15] IngChao Lin, YiMing Yang, and ChengChian Lin. High-performance low-power carry speculative addition with varible latency. IEEE Trans. VLSI Syst., 23(9):1591–1603, 2015.

[16] Li Li and Hai Zhou. On error modeling and analysis of approximate adders. In ICCAD, pages 511–518, November 2014.

[17] P. Kulkarni, P. Gupta, and M. Ercegovac, “Trading accuracy for power with an underdesigned multiplier architecture,” in Proceedings of the 24th IEEE International Conference on VLSI Design, 2011, pp. 346–351.

[18] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” DATE, 2014.

[19] Khaing Yin Kyaw, Wang Ling Goh, and Kiat Seng Yeo. Low-power high-speed multiplier for error-tolerant application. In EDSSC, pages 1–4, 2010.

[20] Kartikeya Bhardwaj, Pravin S. Mane, and Jorg Henkel. Power- and area-efficient Approximate Wallace Tree Multiplier for error-resilient systems. In ISQED, pages 263–269, March 2014.

[21] Chia-Hao Lin and Ing-Chao Lin. High accuracy approximate multiplier with error correction. In ICCD, pages 33–38. IEEE, October 2013.

[22] Amir Momeni, Jie Han, Paolo Montuschi, and Fabrizio Lombardi. Design and Analysis of Approximate Compressors for Multiplication. IEEE Transactions on Computers, 64(4):984–994, 2015.

[23] Cong Liu. Design and analysis of approximate adders and multipliers. Master’s thesis, University of Alberta, Canada, 2014.

[24] H. Jiang, J. Han, and F. Lombardi, “A comparative evaluation of approximate multipliers,” IEEE Nanoarch Symposium, Beijing, China, 2016.

UNIVERSITY OF ALBERTA

References

[25] H. Jiang, J. Han, F. Qiao, and F. Lombardi, “Approximate Radix-8 Booth Multipliers for Low-Power and High-performance Operation,” IEEE Transactions on Computers, 65, 8: 2638–2644, 2016.

[26] Linbin Chen, Jie Han, Weiqiang Liu and Fabrizio Lombardi, “Design of Approximate Unsigned Integer Non-restoring Divider for Inexact Computing,” in GLSVLSI’15, the 25th IEEE/ACM Great Lakes Symposium on VLSI, Pittsburgh, PA, USA, 2015.

[27] Linbin Chen, Jie Han, Weiqiang Liu, and Fabrizio Lombardi, “On the Design of Approximate Restoring Dividers for Error-Tolerant Applications,” IEEE Transactions on Computers, vol. 65, no. 8, pp. 2522 - 2533, 2016.

[28] Soheil Hashemi, R Bahar, and Sherief Reda. A low-power dynamic divider for approximate applications. In Proceedings of the 53rd Annual Design Automation Conference. ACM, 105, 2016.

[29] Joshua Yung Lih Low and Ching Chuen Jong. Non-iterative high speed division computation based on Mitchell logarithmic method. In IEEE International Symposium on Circuits and Systems (ISCAS). 2219–2222, 2013.

[30] Lei Wu and Ching Chuen Jong. A curve fitting approach for non-iterative divider design with accuracy and performance trade-off. In IEEE 13th International New Circuits and Systems Conference (NEWCAS), 2015.

[31] Reza Zendegani, Mehdi Kamal, Arash Fayyazi, Ali Afzali-Kusha, Saeed Safari, and Massoud Pedram. SEERAD: A high speed yet energy-efficient rounding-based approximate divider. In Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 1481–1484, 2016.

[32] Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung, and Keshab K Parhi. Design of low-error fixed-width modified booth multiplier. IEEE Transactions on VLSI Systems, 12(5):522–531, 2004.

[33] SONG Min-An, VAN Lan-Da, and KUO Sy-Yen. Adaptive low-error fixed-width booth multipliers. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 90(6):1180–1187, 2007.

[34] Jiun-Ping Wang, Shiann-Rong Kuang, and Shish-Chang Liang. High-accuracy fixed-width modified booth multipliers for lossy applications. IEEE Transactions on VLSI Systems, 19(1):52–60, 2011.

[35] Yuan-Ho Chen and Tsin-Yuan Chang. A high-accuracy adaptive conditional-probability estimator for fixed-width booth multipliers. IEEE Trans. Circuits and Systems I: Regular Papers, 59(3):594–603, 2012.

References

[36] Farzad Farshchi, Muhammad Saeed Abrishami, and Sied Mehdi Fakhraie. New approximate multiplier for low power digital signal processing. In IEEE CADS, pages 25–30, October 2013.

[37] H. Jiang, L. Liu and J. Han, “Special Session Paper: An Efficient Hardware Design for Cerebellar Models using Approximate Circuits,” CODES/ISSS’17 Companion, October 15-20, 2017, Seoul, Korea.

[38] Masao Ito. Cerebellar circuitry as a neuronal machine. Progress in neurobiology, 78(3):272–303, 2006.

[39] Thomas W Calvert and Frank Meno. Neural systems modeling applied to the cerebellum. IEEE Transactions on Systems, Man, and Cybernetics, (3):363–374, 1972.

[40] M Fujita. Adaptive filter model of the cerebellum. Biological cybernetics, 45(3):195–206, 1982.

[41] Alexander Lenz, Sean R Anderson, Anthony G Pipe, Chris Melhuish, Paul Dean, and John Porrill. Cerebellar-inspired adaptive control of a robot eye actuated by pneumatic artificial muscles. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39, 6: 1420–1433, 2009.

[42] Mina Ranjbaran and Henrietta L Galiana. Hybrid model of the context dependent vestibulo-ocular reflex: implications for vergence-version interactions. Frontiers in computational neuroscience 9, 2015.

[43] Paul Dean, John Porrill, and James V Stone. Visual awareness and the cerebellum: possible role of decorrelation control. Progress in brain research 144: 61–75, 2004.

UNIVERSITY OF ALBERTA

Thanks for your attention.
Questions?