On Ideas In Arithmetic

In celebration of Jean-Michel Retirement
ENS Lyon, November 7, 2025

Distinguished Professor Miloš D. Ercegovac
milos@cs.ucla.edu
http://web.cs.ucla.edu/~milos/
Computer Science Department
University of California at Los Angeles

Outline

- Renaissance of digital arithmetic: why now? Need massive computations in ML and Generative AI; hardware capable of extraordinary performance - the engine is efficient arithmetic
- AI dominates demands on processor performance
- AI hardware is optimized for arithmetic accelerators (GPUs, TPUs, DSAs) operating on massive volumes of data in parallel.
- Growing problem with power - Extraordinary increase in demand: 30-50% in data center usage by 2030.
- Cost of the interconnect is also significant and growing.

- A prevailing design approach: massively parallel arithmetic causing large increase in power and interconnect

- Arithmetic designs - fairly traditional except for use of low precision.

- Some attempts: logarithmic arithmetic

- We look here at something old but different: doing arithmetic from left to right, and terminating at will.

  Goal to reduce power and interconnect cost in Domain Specific Accelerators

- Celebrate 38 years with Jean-Michel Muller

Trend-setting Changes over Time

- A. Changes in semiconductor technology and system architecture enabled performance increases over years
  - The transistor, the ubiquitous basis of computing reaching the limit in downsizing and power efficiency but per chip count keeps growing - thanks to progress in extreme lithography and new types of transistors.
    - A1. Moore's Law of doubling the transistor count every 2 years slowed down around 2010 but it hasn't stop.

The highest transistor counts 2024-2025

- NVIDIA Blackwell GPU: 208 billion transistors, 4nm technology, 120 million transistors mm^2, OEM cost $40K for B200 SXM.

- The highest wafer-scale transistor count
  Cerebras WSE-3: 4 trillion transistors, OEM cost $2-3 million per wafer

- Transistor advances: FinFETs non-planar vertical structure; Gate-All-Around FETs (GAAFETs) gate wrapped around the channel - better electric characteristics, reduced leakage.

- The cost of technologies 7nm and smaller is becoming high but money in AI and ML systems is also becoming large - so growth continues.

* A2. For years we kept power density the same while increasing transistor count and decreasing their size: Dennard's Scaling. It ended in 2004. Let linear transistor length shrink by factor 2. Then this leads to 4 times as many transistors. If both the current and voltage are reduced by factor 2, the power drops by factor 4. Hence, the power density remains the same. This stopped working because current and voltage could not keep dropping due to dependability issues.

* A3. Slowing down the performance gains: limited instruction-level parallelism. Go parallel: switch to multicore architectures instead of improving single CPU.

* A4. On the other hand, Amdahl's Law (1967) is there: theoretical speedup from parallelism is limited by the sequential part of the task, no matter how many processors /cores are used. That is,

benefit of more transistors leading to more processors/cores has limits

- B. Changes in applications, computing demands and storage:
  - Widespread AI/Machine Learning/Generative AI became the main driving forces
  - Large and growing data sets dominate computing - huge data centers.

Implications of A and B: needed improvements in performance and efficiency cannot come from traditional approaches; the emerging trends are

DOMAIN-SPECIFIC ACCELERATORS (DSAs) AND ARRAYS OF GPUs.[6,7]

The rising role of arithmetic

- Arithmetic is essential in making GPUs successful for compute-intensive workloads.
- DSAs use extensively arithmetic and customized design.
- Arithmetic is dominated by matrix multiplication, multiply-accumulate, inner products, optimization of precision in floating-point, and efficient nonlinear functions for activations.
- Arithmetic algorithms for these operations are traditional; main difference is high expected performance and massive size.

The issues

- Interconnections are a critical factor for scaling AI and Machine Learning workloads
  - Between processors, memory modules and networking infrastructures
  - In DSAs between operators
- Voracious increase in power demands: AI share 35-50% in data centers by 2030.
  - GPT-4 training: about 50 GWh
  - The energy demand to train new models doubles every 12-18 months

A Few Observations

- Over the years research in digital arithmetic successfully enabled high performance in general-purpose (GP) processors. Its main approach has been fast, reliable, and energy-efficient implementation of *standardized* operations, compatible with established instruction-set architectures.

- Every new generation of processors, such as GPUs, super-scalar and multi-core processors, digital signal processors, and processors for AI and machine learning, have relied on progress in computer arithmetic, besides memory and on-chip communications, to increase performance and reduce energy and cost.

- Arithmetic has become crucial in domain-specific accelerators. With the rise of domain-specific applications, *standardizing arithmetic operations is no longer desirable* - flexibility is needed.

- *Fusing* operations reduces time and area costs of standardized, atomic interfaces between operators.

- This opens opportunities and presents challenges in developing new arithmetic solutions and suitable accelerators - an active area of research and development.

Started 50 Years Ago - and Still Thinking About It

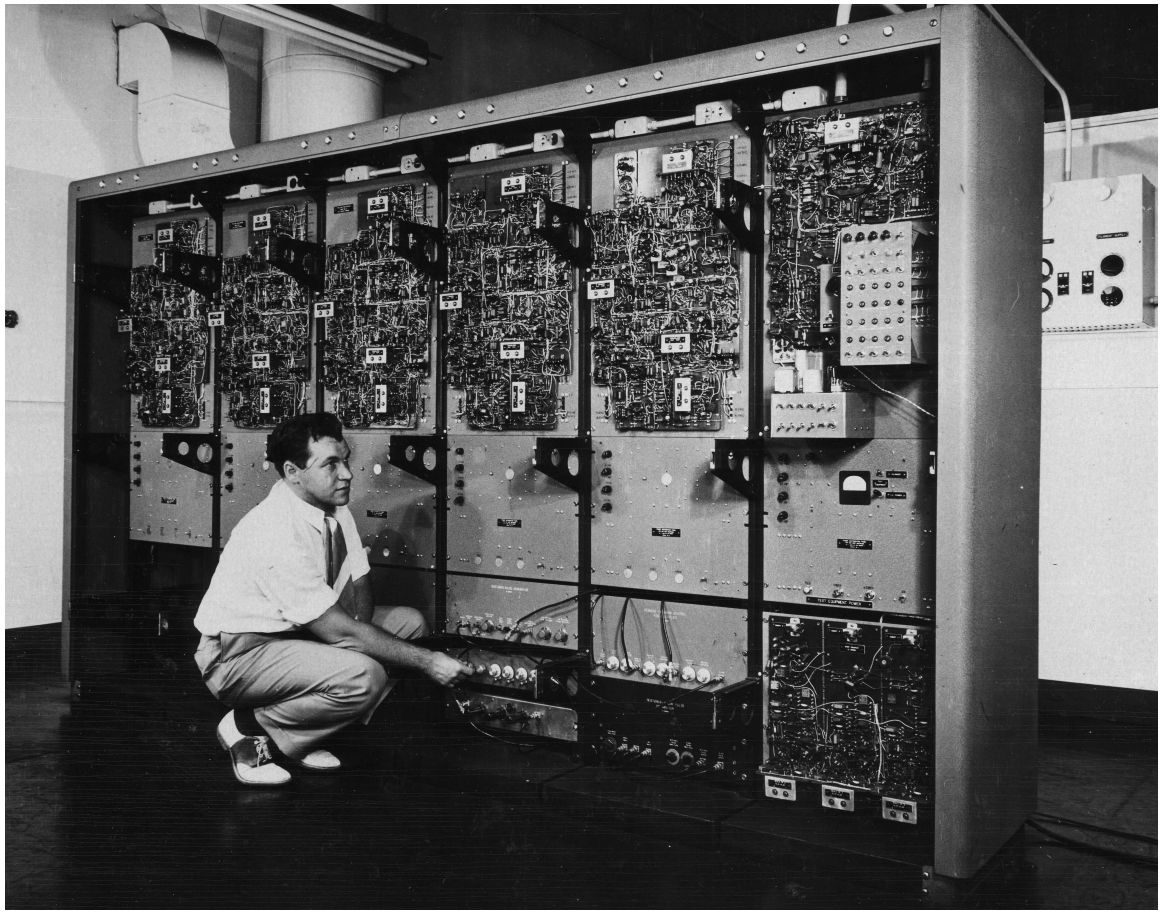
- This talk tries to illuminate some ideas and techniques of non-conventional arithmetic solutions

- Main question: Why compute from right-to-left when more significant information is on the left? We already do it in division (recurrence type)

- We argue that computing from the most significant positions has several desirable properties. This model of arithmetic we call *left-to-right (LR) arithmetic* while the conventional model *right-to-left (RL* arithmetic. Alternatively, these models are also known as *Most Significant Digit First (MSDF)* and *Least Significant Digit First (LSDF)* models.

- A generalization of the MSDF model is *online arithmetic* [1], in which all operations are done MSDF and processing of digits is done in parallel with input/output. That is, computation and communication are overlapped.

What in the world is this?



It's a 5-bit by 5-bit multiplier! Project Whirlwind MIT 1946.

It is a carefully designed prototype

- Built as a proof of concept, to learn about technology (vacuum tubes), to check circuitry, and to understand errors.

- 2MHz clock; 5μsec multiplication time

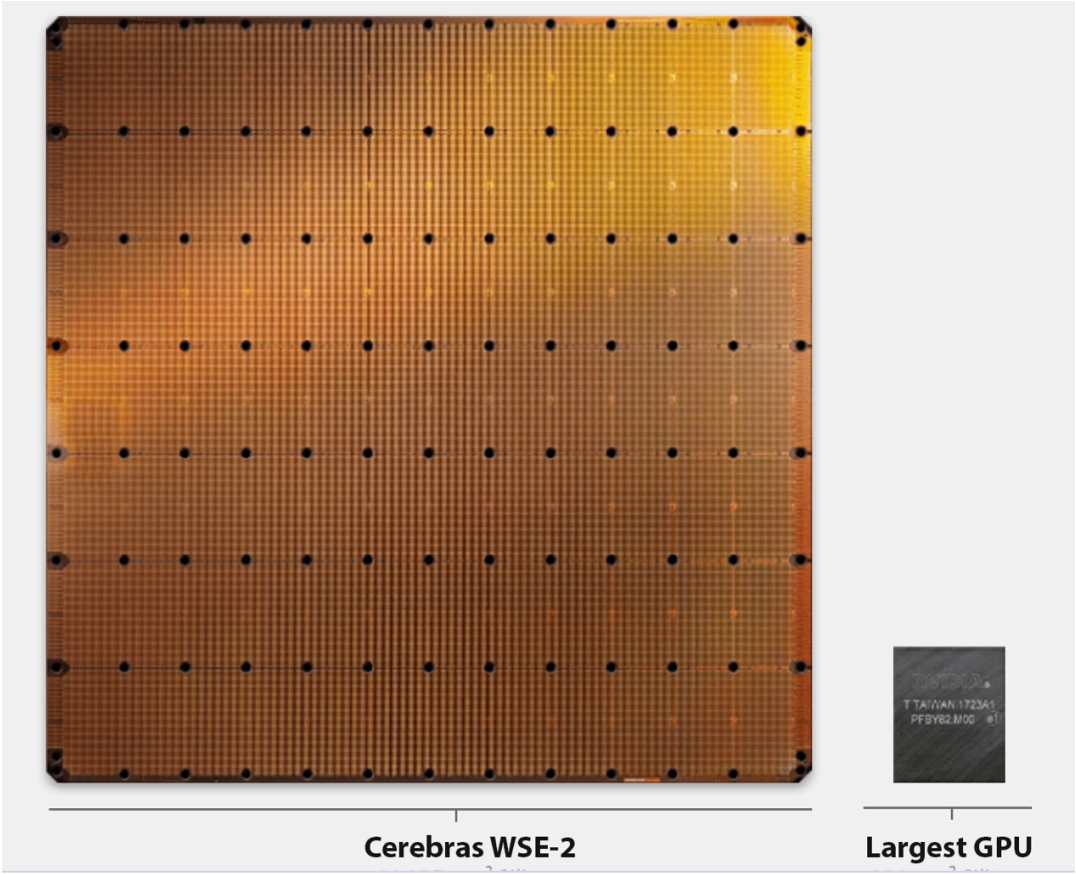
- 10 cycles: in odd cycles add, in even cycles perform "shift and carry" operation (carry-save)

- probably the first electronic implementation of the carry-save concept

- 400 vacuum tubes; a few weeks between error

75 Years later: Cerebras Wafer-Scale Engine 3, AI chip

- The largest chip ever built - Wafer Scale Cluster
- 46,225 mm^2 silicon; 4 Trillion transistors
- 900,000 AI optimized cores; Matrix Mult 100K x 100K
- 125 petaflops peak AI compute, 23kW
- 44 Gigabytes on-chip memory
- 20 Petabytes memory bandwidth
- $2 - 3 million per wafer; 7nm Process technology at TSMC


Cerebras WSE-2
Largest GPU

So Back to the Questions

- Why compute less significant digits before more significant digits?

Left to right instead of right to left?

$$|---\longrightarrow \quad not \;\; that \quad \longleftarrow---|$$

- Why not compute while communicating?
- Why not start operation before previous operation is finished?
- Why not stop when you have enough? Variable precision by default?

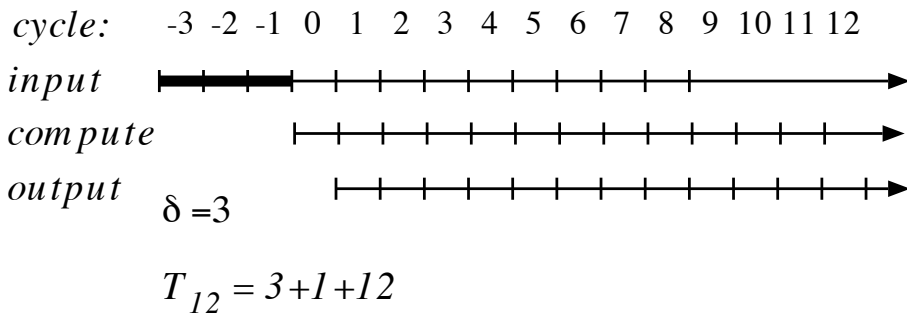
If we can do that, then ...

- We can overlap successive operations
- We can expose digit-level parallelism between operations
- We can minimize inter-operator communication bandwidth and memory access: reduce energy and cost
- We can ”fuse” operations and obtain composite operators

- *Idea that does all of that and more*

Online/LR Arithmetic[1,2]

SIMPLE PRINCIPLE



- Operands and results are streamed serially left-to-right to reduce interconnect lines, simplify interface, and reduce area and energy dissipation.
- Computation is done in parallel with input/output operation.
- This allows overlap among successive operations, compensating for serial operation.

- Due to redundancy in representation, operations are carry-free, i.e., the cycle time does not depend on the precision.

- Online arithmetic allows variable precision computation with graceful termination. Rounding by truncation is unbiased.

- Many interesting questions in on-line arithmetic have been considered in the past. We highlight here two topics
  - Dynamics of signal activities in space and time.
  - Higher level linear operator for computing $f(x)$.

Dynamics of Signal Activities [3]

- Online/LR arithmetic can have a significant effect on module/signal activities, i.e., for power/energy reduction.
- Online and left-to-right arithmetic provides algorithmic reduction of the number of active modules
- Reduction of active module has two sources:
  1. Gradual use of input digits
  2. Truncation of working precision to $p < n$.
- In general, the module activities are reduced by about 50%

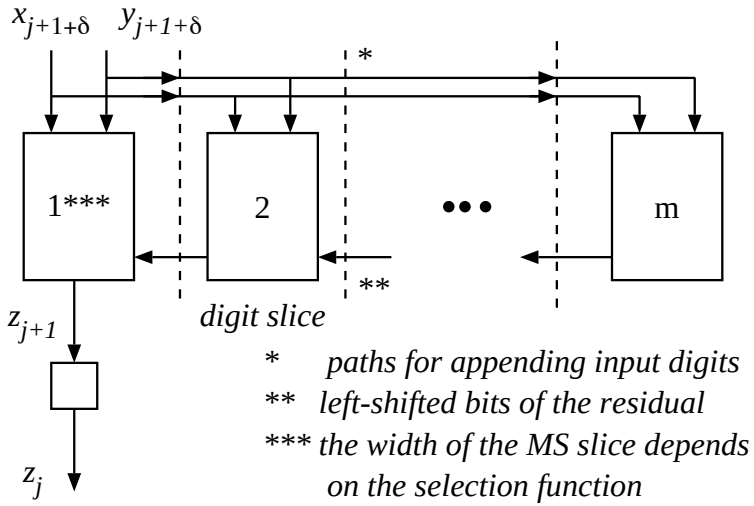
- To reduce static power
  - In 1D implementation power-gating is used on inactive modules
  - In 2D implementation inactive modules are not implemented
- The reductions given in module activities are estimates
- Implementation is needed to obtain actual energy savings
- This analysis may provide interest in further study of module activities

ORGANIZATION OF ONLINE UNITS [1]

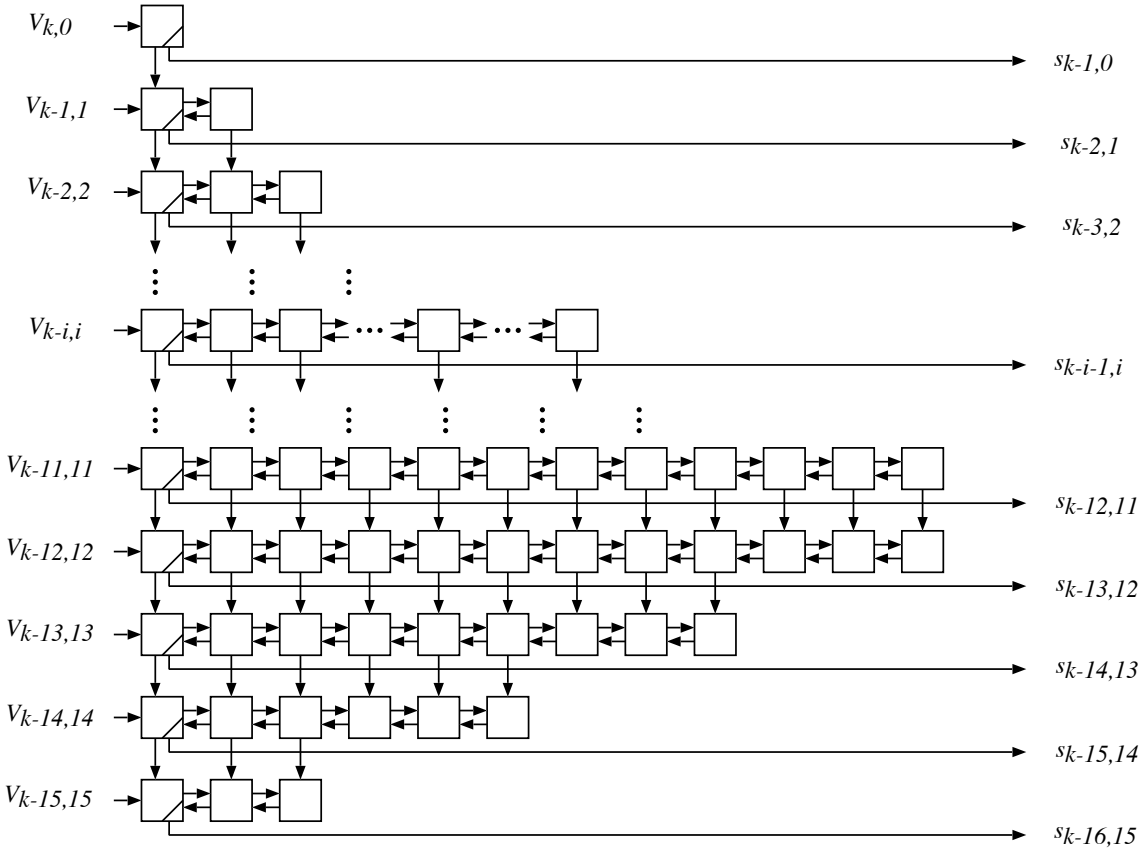
- All online algorithms have similar organization.
- Main alternatives:
  - Linear (1D) design (sequential)
  - Array (2D) design (combinational) : non-pipelined or pipelined with choice of the number of stages
  - Combination of k-bit linear and array

MAIN ORGANIZATIONS

- Basic module: Radix-r digit slice
- The design is repetitive: the design effort independent of precision
- Linear (1D) organization: [Ref. 1]



- Array (2D) organization: [2] If pipelined, Max throughput $1/t_{slice}$



IMPLICATIONS OF ONLINE/LR MODE ON WORKING PRECISION AND MODULE ACTIVITIES

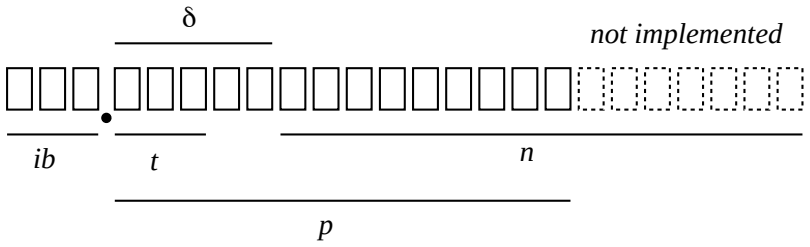
1. Gradual use of input digits: monotonically increasing number of active modules. No circuit modification needed to exploit this property - it is a no-cost power reduction.

2. The precision of online units can be truncated to $p < n$ and still get n output digits. Modules need to be gradually power-gated in linear design. In 2D design these are not implemented - another no-cost power reduction.

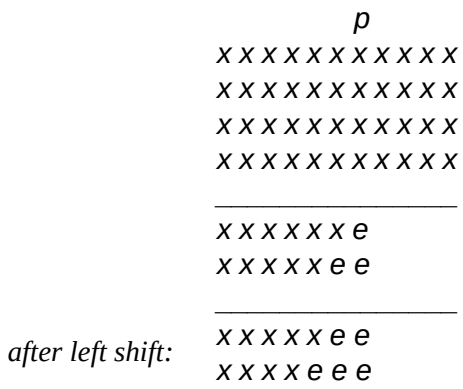
3. These two properties are combined to minimize activities of modules.

TRUNCATION OF DIGIT-SLICES TO PRECISION $p < n$

– A general analysis [Ref. 1]



(a)



(b)

- Reduction of precision to p causes an error to propagate to the left after p steps (stages)
- Selection remains valid if the error does not affect t MS fractional bits
- $p + h = n + \delta$, h slices not implemented, so we get

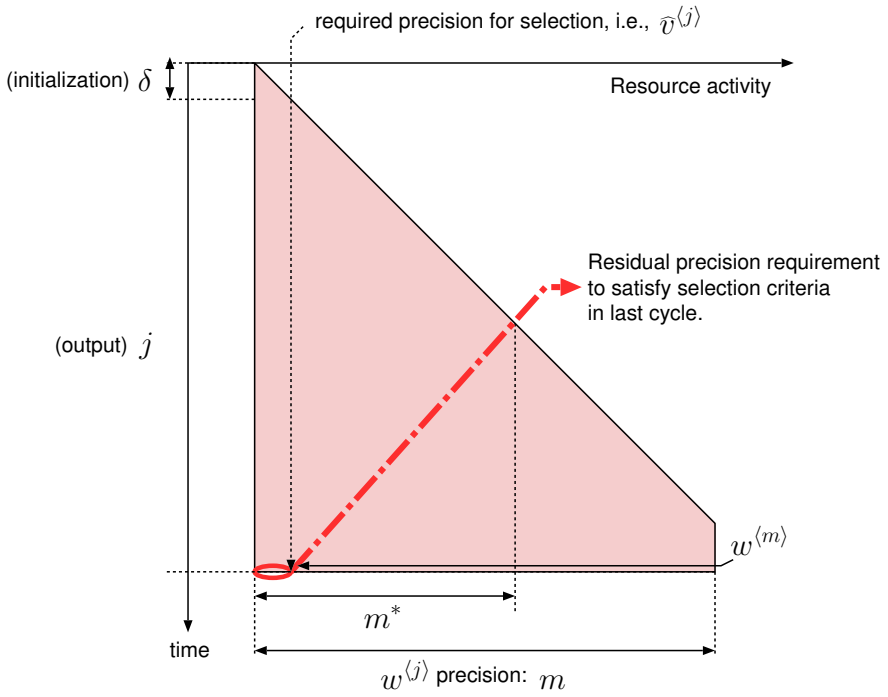
$$p = \left\lceil \frac{2n + \delta + t}{3} \right\rceil$$

- Total number of bit-slices: $ib + p$, ib - no. integer bits

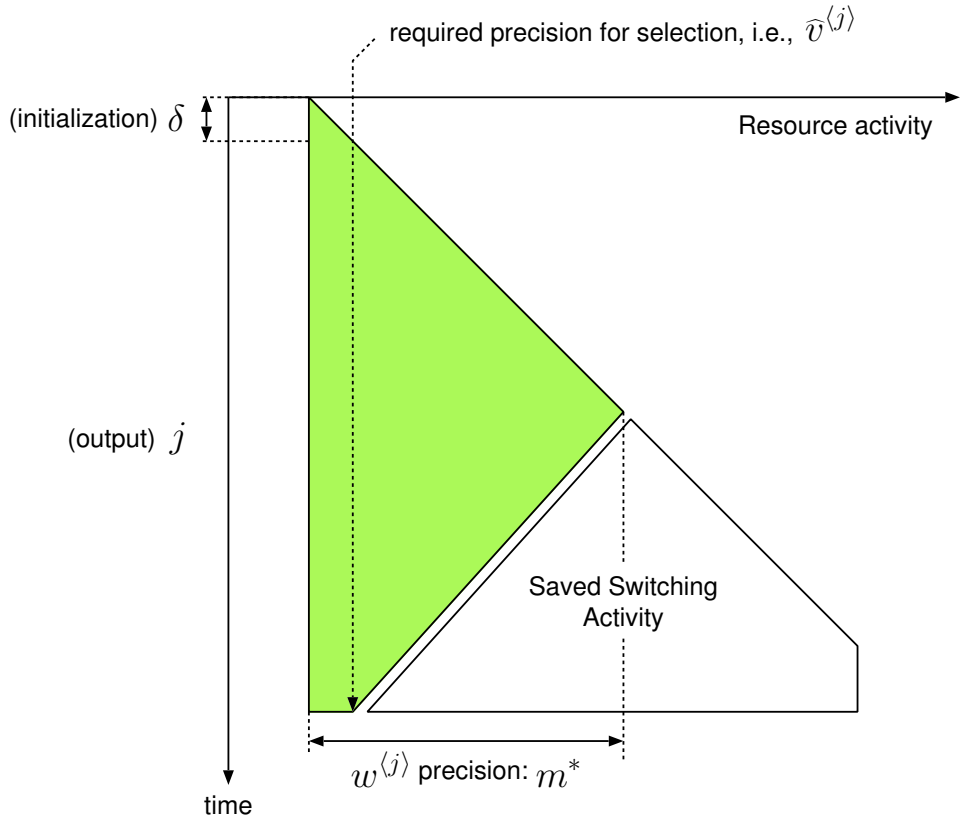
- For example, the number of bit-slices for a 32-bit radix-2 online multiplication without truncation is 34. With truncation is 25, a 26% reduction

TWO SOURCES OF ACTIVITY REDUCTION

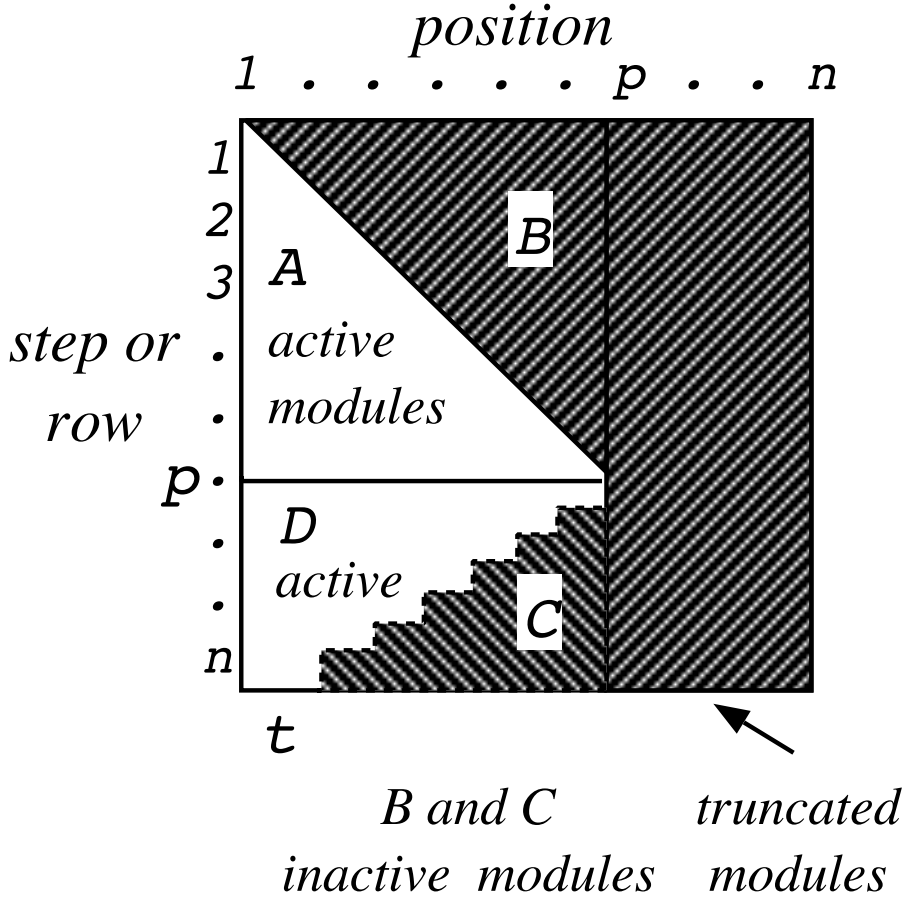
- Due to online inputs: no module activity without inputs



- Due to precision truncation to p: some modules not implemented



COMBINED ACTIVITY REDUCTION



– The regions of interest:

Region A: active following the arrival of input digits - modules gradually becoming active up to position p.
Region B: inactive - in a 1D implementation, p modules are implemented but gradually activated with the arrival of inputs;

Two options:

(a) all p modules are powered so consume static power when inactive,
(b) modules are power-gated synchronized with the input arrival so dynamic power but no static power;

in a 2D implementation, modules in the region B are not implemented: no static/dynamic power consumed.

Region C: inactive - in 1D implementation, modules need to be gradually power-gated;
- in 2D implementation, modules in the region C are not implemented: no static/dynamic power consumed.

Region D: active in the last $n - p$ cycles.

MODULE ACTIVITIES IN 3D VECTOR NORMALIZER

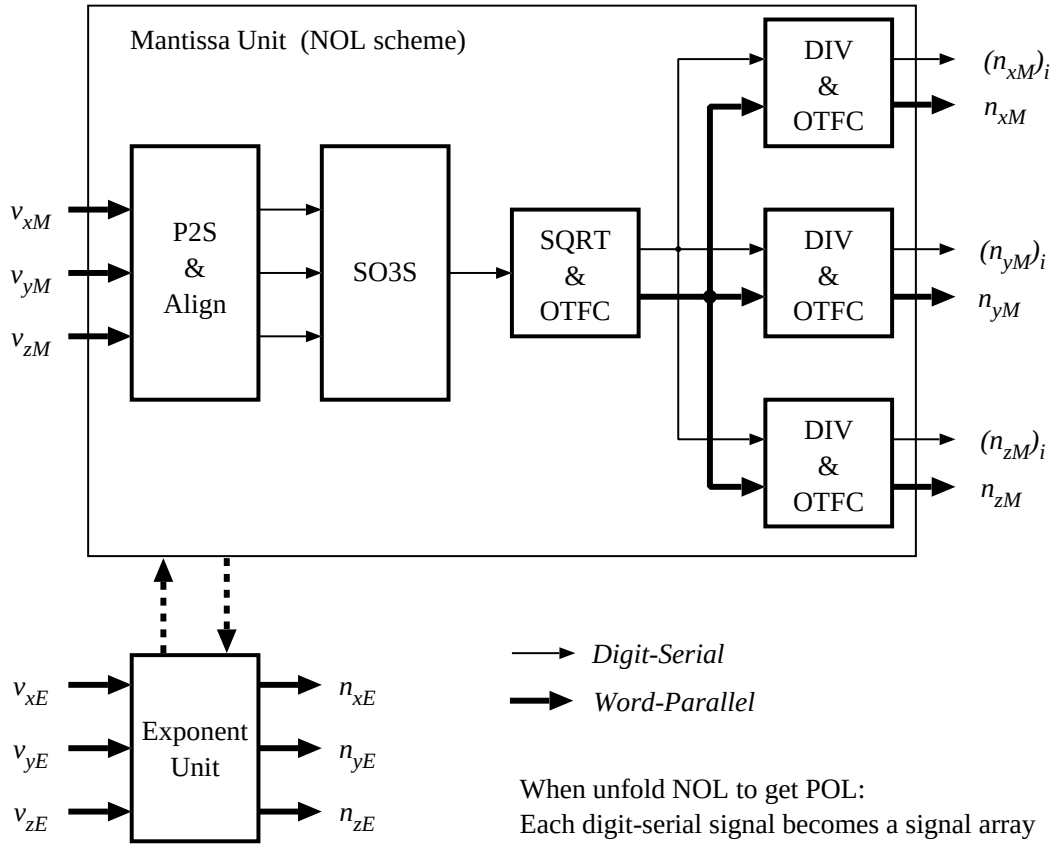
- Analyze and estimate signal/module activities in a composite arithmetic scheme with fused online units [2]
- Input vector $\vec{V} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$ and
- The normalized vector

$$\vec{N} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} v_x/d \\ v_y/d \\ v_z/d \end{bmatrix} \quad (1)$$

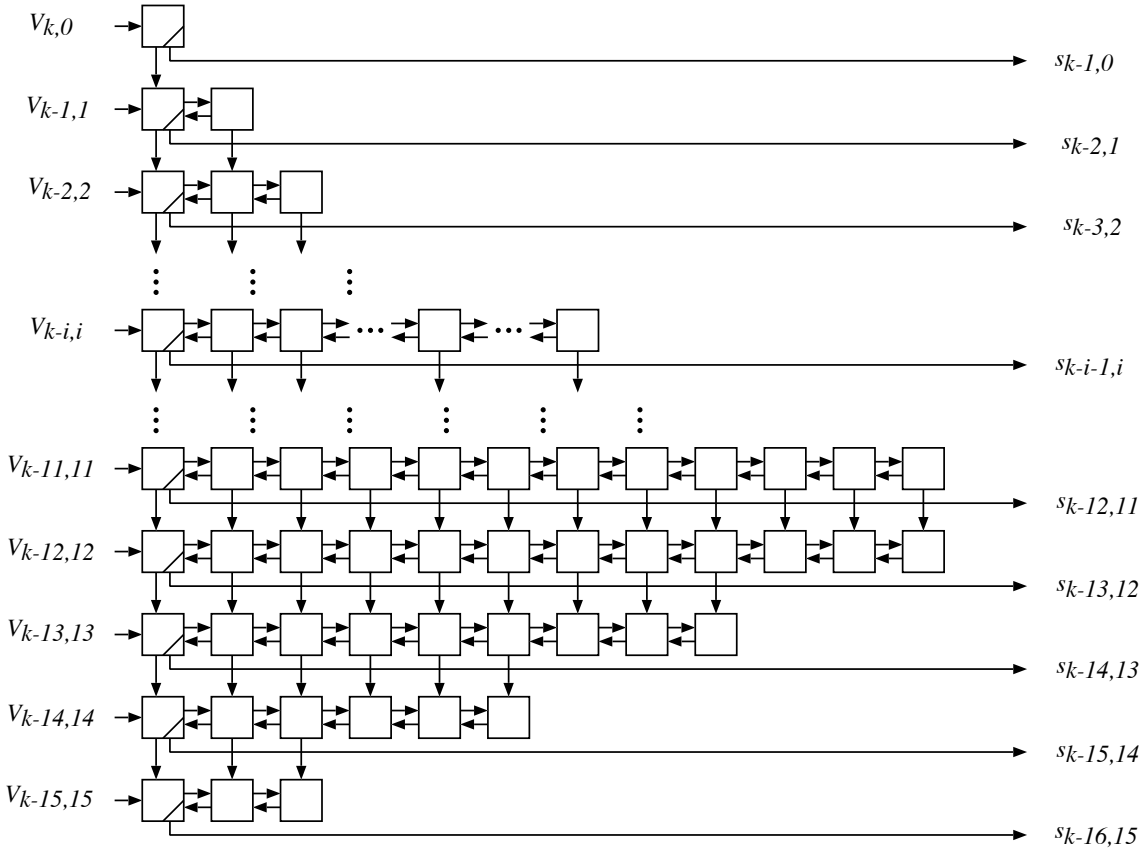
where

$$d = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2} \quad (2)$$

The accelerator scheme - all units online

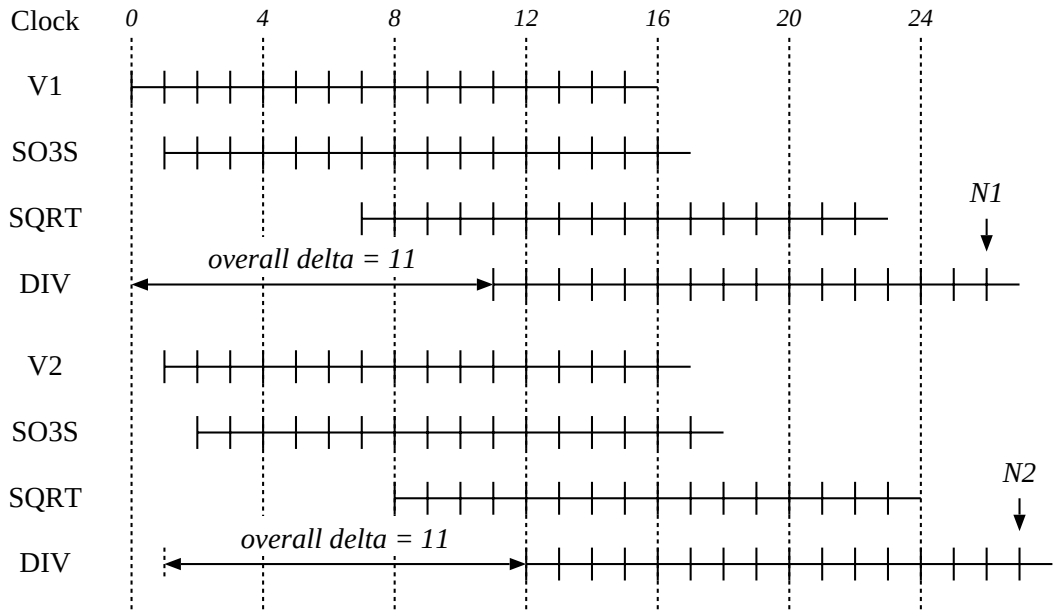


2D PIPELINED ORGANIZATION



– Pipelined array of digit modules for Max throughput $1/t_{digit-module}$

Timing: Pipelined Online



Sum of Three Squares (S03S)

– Initialize–

$R[-1] = 0;$

$X[-1] = Y[-1] = Z[-1] = 0;$

– Recurrence –

for $j = 0, 1, 2, ..., n$ **do:**

$W[j] = 2R[j-1] + (2X[j-1] + x_j 2^{-j})x_j$

$+(2Y[j-1] + y_j 2^{-j})y_j + (2Z[j-1] + z_j 2^{-j})z_j;$

$s_j \leftarrow csint(W[j]);$

$R[j] \leftarrow csfrac(W[j]);$

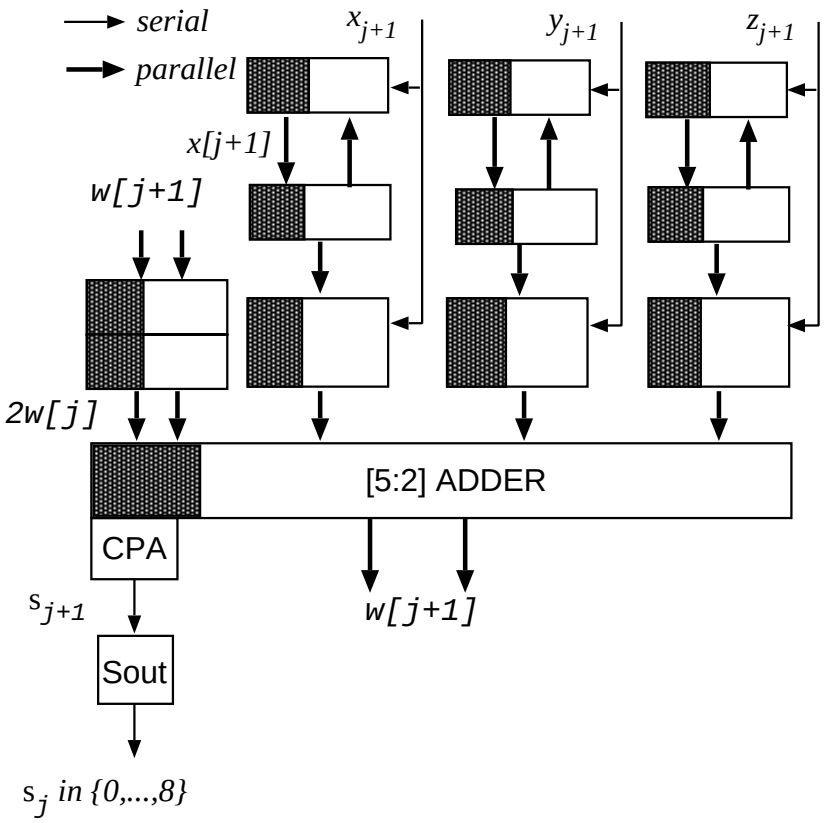
$X[j] \leftarrow append(X[j-1], x_j);$

$Y[j] \leftarrow append(Y[j-1], y_j);$

$Z[j] \leftarrow append(Z[j-1], z_j);$

end for

Active Modules at step j



- Similar activity patterns in SQRT and DIV units

Total Activity Savings

- The total reductions in module activities for the 3D Vector Normalizer are estimated around 50% compared to online design using full precision in all steps (stages) and not taking advantage of reduced activities.

- The effect on power dissipation will be smaller since online units are more complex than conventional units - to be evaluate

Power of Online Arithmetic: the E-Method

- How can we compute

$$R_{3,2}(x) = \frac{p_2x^2 + p_1x + p_0}{q_3x^3 + q_2x^2 + q_1x + 1}$$

  Of course, use standard techniques like Horner or, if you want it faster, Estrin methods for polynomials, followed by division. In full precision.

- We suggest another method:
  - Find an equivalent system of linear equations.
  - Using online operations solve it in time $T(m) = O(m)$, m precision.
  - Typically, the first component of the solution is the value of the expression.

- The method

1. Transform an arithmetic expression into a system of linear equations L:

$f(x) = E(x, \underline{p})$, $\underline{p}$ parameters

$$f(x) \Rightarrow L : \mathbf{A} \cdot \mathbf{y} = \mathbf{b} \text{ such that } y_1 = f(x).$$

2. Solve the system using digit-by-digit vector recurrences in m steps for m digit result. This is a Jacobi iterative method using digits instead of full-precision operands.

Typical recurrence: $w[j+1] = 2(w[j] - d2_j - q_1 \cdot d1_j + x \cdot d3_j)$

3. Coefficient matrix corresponds to the matrix divisor and the right-hand side vector to the vector dividend. The quotient is the solution

vector.

4. The elements of the solution vector are obtained in parallel starting with the most significant digits.

5. Redundancy makes the cycle time independent of precision and simplifies the selection of result digits.

TRANSFORMATIONS

- A rational function $R_{2,3}(x)$ is mapped to the system L:

$$\begin{bmatrix} 1 & -x & 0 & 0 \\ q_1 & 1 & -x & 0 \\ q_2 & 0 & 1 & -x \\ q_3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ 0 \end{bmatrix}$$

Solving the system produces in m steps $\mathbf{y}$ to precision m such that:

$$y_1 = R_{3,2}(x) = \frac{p_2x^2 + p_1x + p_0}{q_3x^3 + q_2x^2 + q_1x + 1}$$

Recurrences

- The recurrences are

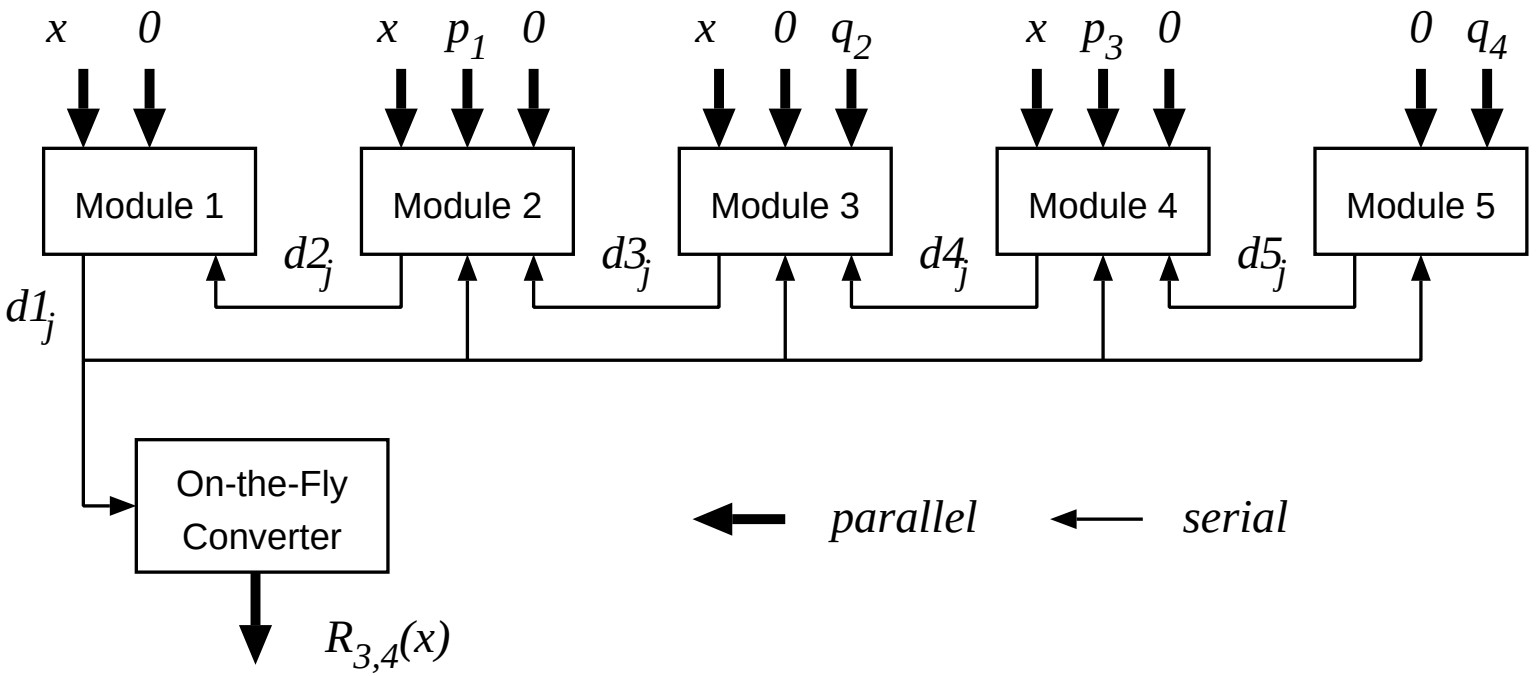
$$\begin{aligned}
w1[j+1] &= 2(w1[j] - d1_j + x \cdot d2_j) \\
w2[j+1] &= 2(w2[j] - d2_j - q_1 \cdot d1_j + x \cdot d3_j) \\
w3[j+1] &= 2(w3[j] - d3_j - q_2 \cdot d1_j + x \cdot d4_j) \\
w4[j+1] &= 2(w4[j] - d4_j - q_3 \cdot d1_j + x \cdot d5_j) \\
w5[j+1] &= 2(w5[j] - d5_j - q_4 \cdot d1_j)
\end{aligned}$$

  dk_j - a single digit

- The initial residuals are

$$(w1[0], w2[0], w3[0], w4[0], w5[0]) = (0, p_1, 0, p_3, 0)$$

The Network



– regular, serial intermodule links, easily scalable

Examples of polynomial and rational function evaluators [1,5]

A Key Observation

- Online E-method vs classical Jacobi method
- Consider an n-th order system of linear equations

$$L : \mathbf{A} \cdot \mathbf{y} = (\mathbf{I} - \mathbf{G}) \cdot \mathbf{y} = \mathbf{b}$$

- Classical Jacobi method solves L by iterating

$$\mathbf{y}[j] = \mathbf{b} + \mathbf{G} \cdot \mathbf{y}[j-1],\ j = 1, 2, \ldots$$

The step requires $(n-1)$ **full precision** multiplications and $(n-1)$ **full precision** additions per row per step.

Online method solves L by iterating a digit recurrence

$$\mathbf{d}[j] + \mathbf{z}[j] = r(\mathbf{z}[j-1] + \mathbf{G}\cdot\mathbf{d}[j-1]),\ j = 1, 2, \ldots, m$$

where $\mathbf{d}[j]$ is a vector of digits, $\mathbf{z}[j]$ a vector of m-digit fractions, and $\mathbf{G}$ is a matrix with m-digit fractions as coefficients.

Therefore, the step uses $(n-1)$ m-digit $\times$ **single digit multiplications** and $(n-1)$ **totally parallel additions**

– A significant reduction in cost, delay and energy

– How to use it in ML and NN architectures?

Rapid growth of ML/AI needs [6,7]

- Rapid evolution of hardware deployed in the cloud, on the edge, in IoTs and mobile devices
- In just 2 years: 1000x larger models, 1000x more compute
- Today, GPT-3 with 175 billion params trained on 1024 GPUs for 4 months.
- Cerebras cluster CS-2: 120-trillion parameters models
- Tomorrow, multi-trillion parameter models and beyond: need massive memory, compute and communication

What is the role of arithmetic in this growth?

- Arithmetic affects all three requirements in AI/ML systems:
  - Computing throughput, latency, size, and energy
  - Memory size, access, and bandwidth
  - Bandwidth and delay of communication paths
- All three depend heavily on arithmetic formats and algorithms

The main approaches

- Optimizing the individual arithmetic operations and precision of operands

- Massive use of arithmetic operators:
  e.g., Google Tensor Processing Unit TPU has 65 thousands 8-bit by 8 multipliers

- Domain-specific accelerators developed for:
  - Convolutional neural networks
  - Recurrent neural networks
  - Composite architectures integrating key operations in neural networks:
    * Convolution

  - Maximum pooling
  - Firing/activation decision: e.g., rectified linear unit (ReLU)
- A plethora of accelerator architectures for deep neural networks, all relying on massive, efficient arithmetic and accelerator architectures.

Google, NVIDIA, Intel, AMD ... AI/ML Accelerators [6,7]

- Google has been making ”relentless progress”:
- TPU v1, deployed 2015, 92 teraops, inference only.
- TPU v2, cloud TPU 2017, pod 2018, 180 teraflops, 64 GB HBM, training and inference, generally available. 11.5 petaflops in a pod.
- TPU v3, cloud beta 2018, 420 teraflops, 128 GB HBM, training and inference, beta. > 100 petaflops in a pod.
- TPU v4, 2021, 275 teraflops/chip, 1.1 exaflops in a pod, optical circuit switch network - rapid and efficient scaling with low latency

Effects of Online Arithmetic in CNN Accelerators [10]

- Support for adaptive precision across layers
- Reduction in number of operations: 17% to 60% wrt conventional
- Corresponding reduction in memory and access bandwidth
- Improves energy efficiency by $1.8\times$ vs. other bit-serial methods [10]

Online Arithmetic - Potential uses in ML Accelerators [8,9]

- Wide range of uses because online arithmetic is possible in all operations
- Flexible arithmetic design technique for accelerators
- In multipliers: Left-to-right carry-free multiplication (LRCF) avoids the final adder
- Applicable in the of inner products, sum of products, sum of squares, convolutions
- In composite (fused) arithmetic algorithms for matrix multiplication, norms, and sparse matrix operations

- In low-precision and variable-precision arithmetic designs
- In function approximation using polynomials and rational functions: use the E-method. m steps for m digit precision, time independent of degree, cost proportional to degree
- In recursive computations.
- In convolutional neural networks, multilayer perceptrons, and in backpropagation
- Reconfigurable architectures simplified - fused operations, minimal interconnect, and variable-precision
- Low energy arithmetic - minimal signal activity

What about AI Processor Trends

- AI dominates demands on processor performance.
- Arithmetic is essential to the performance, efficiency and design of machine learning systems and deep neural networks.
- Arithmetic innovations are focussed on precision optimization in linear algebra forms like matrix multiplication, vector addition and convolution. Also method like gradient descent and backpropagation rely on efficient use of arithmetic.
- AI hardware is optimized for arithmetic accelerators (GPUs, TPUs) on massive volumes of operations in parallel. Cost of the interconnect is significant. The latest relies on optical interconnect.

- AI acceleration is achieved through specialized hardware and heterogeneous architectures using ”chiplets,” with a focus on power efficiency and security. Memory organization and interconnection structures have evolved to support massive use of AI processors.

- Chiplets are smaller, specialized pieces of silicon which can be combined to make different types of cores and other components. These can be combined to create heterogeneous architectures. Chiplets raise both granularity and modularity - simplifying design and achieving higher performance.

- Specialized design (Domain-specific architectures) is expanding - end of ”one-size-fits-all” approach

- Well-known algorithms for multiply-accumulate and inner products dominate AI-oriented implementations

What about Quantum Arithmetic?

- Quantum arithmetic is very hard and extremely device costly. Why? Quantum computers do not handle arithmetic (addition, multiplication, etc.) the way classical computers do. Classical operations erase and overwrite values all the time—but quantum operations must be reversible and cannot simply discard information.

  Digital logic is directly supportive of arithmetic primitives. We can build arbitrarily complex arithmetic networks because a composition of switching functions is a switching function. No equivalent exists in quantum in quantum logic.

- In quantum operations no information is discarded (overwritten): the operations must be reversible. This requires extra qubits called ancillas. Even for moderate problem size, the required number of qubits is in thousands or even millions.

- Constructing quantum arithmetic networks is very difficult because fanout is hard to implement - its fundamental issue is due to no-cloning theorem which does not allow duplication - so no quantum data replication.

  For example, multiplying a vector of bits with a single bit, frequent in arithmetic algorithms, is not allowed. Quantum fanout operation can only copy classical information, not superpositions. There are some solutions to fanout problem with complicated realizations.

  Most of the well-developed algorithms in classical arithmetic cannot be mapped to quantum implementations.

- Quantum circuit depth causes long circuits to accumulate errors leading to decoherence, reversal of quantum information to traditional. Quantum arithmetic also needs many gates which causes decoherence. To stabilize qubits, a quantum error correction is

needed which increases hardware demands dramatically.

- Quantum arithmetic is noise-sensitive leading to decoherence. Quantum states only last for a short time before they decay (lose information).

- Quantum gates are analog (continuous), not digital - two state.

- Representations of fixed and floating-point numbers are tricky.

- Hardware limitations: There are no native arithmetic units in quantum chips

Literature on quantum arithmetic is very limited. Years of arithmetic research is not seriously considered. Mention carry-propagate adder, restoring and nonrestoring division and simple multiplication.

Quantum computers are best when:

- The algorithm avoids measurement
- Reversible logic is already built into the math
- Problems use linear algebra-like transformations
- Examples:
  - Shor's factoring algorithm
  - Quantum phase estimation
  - Grover's search algorithm

Bibliography

1. *Digital Arithmetic.* M.D. Ercegovac and T. Lang, Morgan Kaufmann Publishers, 2004, An Imprint of Elsevier. Chapters 9 and 10 cover online arithmetic.

2. M. D. Ercegovac, "On Left-to-Right Arithmetic", *Proc.51st Asilomar Conference on Signals, Systems and Computers*, 2017.

3. M. D. Ercegovac, "On Reducing Module Activities in Online Arithmetic Operations", *Proc.54th Asilomar Conference on Signals, Systems and Computers*, 2020.

4. W. Yan, M.D. Ercegovac and H. Chen, An Energy-Efficient Multiplier With Fully Overlapped Partial Products Reduction and Final

Addition, *IEEE Transactions on Circuits and Systems*, 63(11):1954-1963, 2016.

5. N. Brisebarre, G. Constantinides, M. D. Ercegovac, S.-I. Filip, M. Istoan and J-M. Muller, "A High Throughput Polynomial and Rational Function Approximations Evaluator", *Proc. of the IEEE Symposium on Computer Arithmetic*, pp. 95-102, June 2018.

6. N. Jouppi et all, "A Domain-Specific Architecture for Deep Neural Networks", *CACM*, Sept 2018.

7. N. Jouppi et al., "In-Data Center Performance Analysis of a Tensor Processing Unit", *ISCA* 2017.

8. M.Usman, J-A Lee and M. D.Ercegovac, "Multiplier with Reduced Activities and Minimized Interconnect for Inner Product Arrays",

Proc.55th Asilomar Conference on Signals, Systems and Computers, 2021.

9. T. Arifeen, A. S.Hassan, J-A Lee, and M. D. Ercegovac, "Adder with Reduced Latency and Minimized Interconnect for Streaming Inner Products", *Proc.55th Asilomar Conference on Signals, Systems and Computers*, 2021.

10. A. S. Hassan, T. Arifeen and J-A. Lee, "Data Footprint Reduction in DNN Inference by Sensitivity-Controlled Approximations with Online Arithmetic", *2020 23rd Euromicro Conferenceon Digital System Design.*

Jean-Michel Muller - Illustrious and Productive Career

- Jean-Michel consistently made significant research contributions over many years in several areas of arithmetic and established himself as a leading researcher.

- His work is characterized by original ideas, an excellent mathematical depth, and connection to practical, implementation aspects of arithmetic.

- Jean-Michel has had a leading role in floating-point arithmetic. The *Handbook of Floating-Point Arithmetic* is an outstanding reference book, not surpassed in depth and scope.

- The book remains an invaluable source about floating-point computation. It covers in depth software, hardware , and theoretical aspects of floating-point arithmetic.

- He has several books on arithmetic in French and a widely recognized book *Elementary Functions, Algorithms, and Implementations* .

Let's mention some of his works

- In floating-point: correct rounding - many problems, FMA, accurate arithmetic, work on Table Maker's Dilemma, correctly rounded sums, analysis of rounding errors and bounds in various problems, IEEE Floating-Point Standard.

- In on-line arithmetic: characterization of on-line computable functions, very high radix on-line arithmetic for accurate computations, table-lookup methods, function approximations, and elementary real and complex operations.

- An elegant solution to CORDIC scaling problem: Branching CORDIC algorithm with constant-time iteration and a constant scale factor
- Evaluation of complex functions: BKM - a hardware algorithm for complex elementary functions
- Variable radix real and complex division
- Analysis of table methods: bipartite and multipartite approaches
- Elementary operations like division and square root with small multipliers
- Complex arithmetic: multiplication, division, square root
- Machine-efficient polynomial approximations and evaluation of polynomials and rational functions

- An early original mathematical formulation of optimization of carry-skip chains. This paper led to many papers on improving carry-skip adders.

- There are many other contributions Jean-Michel made by himself and with his students and collaborators.

- The CNRS group at ENS Lyon he directed produced a leading arithmetic research. He mentored more than 20 students, some became leaders in arithmetic research. Jean-Michel and his students have been among most frequent contributors to the ARITH symposia and IEEE special issues on arithmetic.

- Jean-Michel has played a significant role in ARITH IEEE Symposia over many years: as a general chair and a program chair. He has been a member of the ARITH Steering Committee and a frequent

author/coauthor. He always had a visible role in the program committee meetings: his comments always appreciated.

- My relation with Jean-Michel began in 1987, at ARITH in Como, Villa Olmo, Italy. He presented a paper on FELIN VLSI processor. During a cruise on the lake we had a long discussion about arithmetic, especially on-line, which continued over many years and many espressos.

- We exchanged visits to ENS Lyon, Marseilles and UCLA. Besides ARITH, we frequently participated in the Asilomar Conference on Systems, Signals and Computers, in Pacific Grove, and in special sessions at the SPIE conference in San Diego. Several students from ENS/UCLA exchanged visits.

- I collaborated frequently with Jean-Michel and his colleagues Nicolas Brisebarre, Jean-Claude Bajard, Florent de Dinechin, and Nathalie Revol, and students including Arnaud Tisserand, Vincent Lefévre, Christoph Mazenc, Sylvain Chevillard, Serge Torres.

- We worked on many different topics, most notably algorithms and implementations for complex division, square root and polynomials. He proposed transformation, which allowed the E-method to result in real recurrences to compute complex polynomials. With Nicolas, a lattice approximation method for rational functions was developed producing coefficients compatible with the E-Method. Other problems included variable radix algorithms (real and complex), and algorithms for all basic operations using small multipliers. Arnaud Tisserand participated in many papers.

On Santa Monica Pier

In the Sierras

On the trail to Rainbow Falls and Devil's Postpile with Muller's family

Going to Mammoth Crest on Muir Trail, 11,000 feet.

THANK YOU

It has been a pleasure knowing Jean-Michel and working with him.

I thank him for many contributions he made to the arithmetic field and for keeping it and ARITH alive.

I wish him happy and free time to pursue what he likes

- No more deadlines, just a few roundings here and there.