Left-to-Right Arithmetic Paradigm: Computing while Communicating - Terminate Gracefully at Any Moment

DDECS2020 Keynote Presentation

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

Introductory Comment

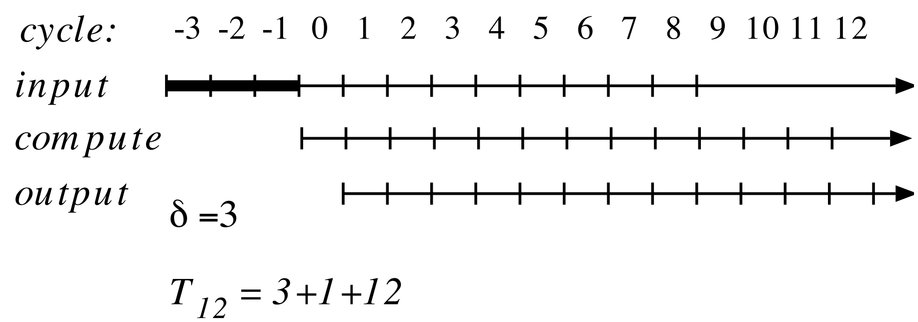
Computer arithmetic has always played an important and critical role in the design of general-purpose and special-purpose processors. It has been an active research area since the early days of computers (1950s), albeit relatively small compared to other areas in computer architecture and computer science. Computer arithmetic, in brief, investigates theoretical/hardware/software aspects in number representation, arithmetic algorithms for basic operations, function evaluation, in fixed-point and floating-point arithmetic. Computer arithmetic is a synergy of applied mathematics, algorithms, digital design, VLSI implementation, software and compilers, and applications. Every new generation of processors, such as GPUs, super-scalar and multi-core processors, digital signal processors, and, most recently, processors for AI and machine learning, rely on progress in computer arithmetic to increase performance and reduce energy, and cost.

Short Bio

Miloš D. Ercegovac
Distinguished Professor
Computer Science Department
UCLA Samueli School of Engineering
University of California, Los Angeles
468B Engineering VI
310-825-5414
310-825-2273 (Fax)
milos@cs.ucla.edu

Professor Ercegovac earned his PhD ('75) and MS ('72) in computer science from the University of Illinois, Urbana-Champaign, and BS in electrical engineering ('65) from the University of Belgrade, Serbia. He specializes in research and teaching in digital arithmetic, digital design, and computer system architecture. His recent research is in the areas of approximate arithmetic, composite algorithms, complex arithmetic, design for low power and arithmetic in application-specific architectures. His research contributions have been extensively published in journals and conference proceedings. He is a coauthor of two textbooks on digital design and of a monograph and a book in the area of digital arithmetic. Dr. Ercegovac has been involved in organizing the IEEE Symposia on Computer Arithmetic since 1978. He served as an associate editor of the IEEE Transactions on Computers and as a subject area editor for the Journal of Parallel and Distributed Computing. He received a Medal of Ecole Normale Superieure de Lyon, France, in 2015, a Distinguished Alumni Educator Award in 2013 from the Department of Computer Science, University of Illinois Urbana-Champaign, and Lockheed-Martin Corporation Excellence in Teaching Award in 2009. Dr. Ercegovac is a foreign member of the Serbian Academy of Sciences and Arts, Fellow and Life Member of the IEEE, and a member of the ACM.

MAIN IDEA of LR and ONLINE ARITHMETIC



- Why compute less significant before more significant digits?

- Compute while communicating: why waste time waiting for operands?

- Latency parameter: online delay δ - a delay before the most significant digit of a result appears. A small integer.

LEFT-TO-RIGHT AND ONLINE ARITHMETIC

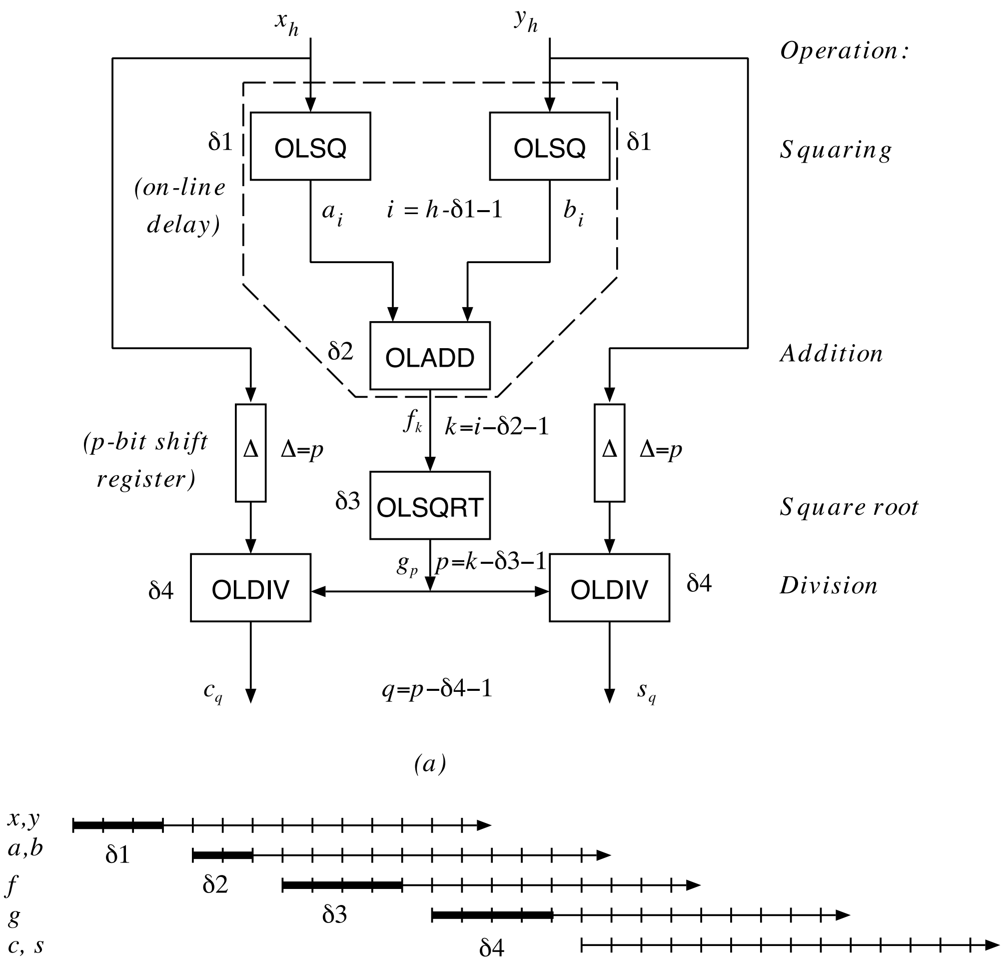
- The inputs applied and output delivered one digit at a time - serially. All digits of the same operands/results share the the same digit lines. The systems are usually clocked so that one digit is applied per clock cycle. It is possible to have some operands in parallel form.

- There are two serial modes: Least-significant digit first (LSDF) and most-significant digit first (MSDF)

- In left-to-right arithmetic the result is generated most significant digit first (MSDF mode) and one or more operands are in digit-parallel form and one or more operands are in digit-serial form.

- In on-line arithmetic the results and the operands are processed one-digit at a time, most significant digit first (MSDF mode)

DIGIT-LEVEL OVERLAP BASIS FOR PERFORMANCE



ONLINE ALGORITHM MODEL

- Operands x and y, result z: n radix-r digits, redundant
- In cycle j the result digit z_{j+1} is computed
- Cycles labeled $-\delta, \ldots, 0, 1, \ldots, n$
- In cycle j receive the operand digits $x_{j+1+\delta}$ and $y_{j+1+\delta}$, and output z_j
- $x[j]$, $y[j]$ and $z[j]$ are the numerical values of the corresponding signals when their representation consists of the first $j+\delta$ digits for the operands, and j digits for the result.

In cycle j

online operands

$$\begin{aligned} x[j+1] &= (x[j], x_{j+1+\delta}) \\ y[j+1] &= (y[j], y_{j+1+\delta}) \end{aligned}$$

result digit

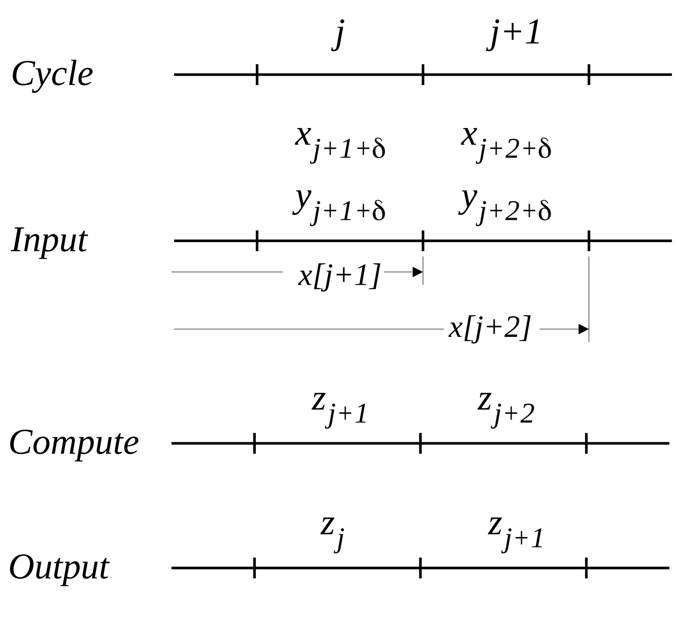
$$z_{j+1} = F(w[j], x[j], x_{j+1+\delta}, y[j], y_{j+1+\delta}, z[j])$$

online result

$$z[j+1] = (z[j], z_{j+1})$$

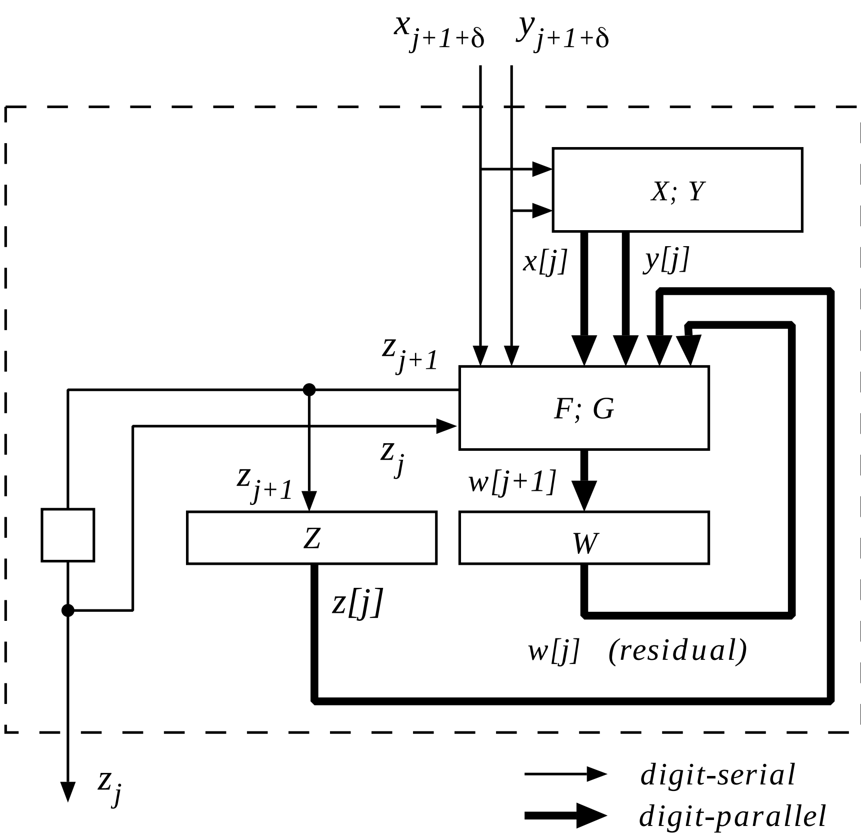
residual

$$w[j+1] = G(w[j], x[j], x_{j+1+\delta}, y[j], y_{j+1+\delta}, z[j], z_{j+1})$$



j
j+1
Cycle
$x_{j+1+\delta}$
$x_{j+2+\delta}$
$y_{j+1+\delta}$
$y_{j+2+\delta}$
Input
x[j+1]
x[j+2]
z_{j+1}
z_{j+2}
Compute
z_j
z_{j+1}
Output

(a)



$x_{j+1+\delta}$
$y_{j+1+\delta}$
X; Y
x[j]
y[j]
z_{j+1}
F; G
z_j
z_{j+1}
w[j+1]
Z
W
z[j]
w[j] (residual)
z_j
digit-serial
digit-parallel

(b)

Online delay (δ)

– critical parameter: determines the throughput (no pipelining)

Operation	LSDF	MSDF
Addition	0	2 ($r = 2$)
		1 ($r \geq 4$)
Multiplication	0	3 ($r = 2$)
		2 ($r = 4$)
(only MS half)	n	
Division	$2n$	4
Square root	$2n$	4
Max/min	n	0

GENERIC FORM OF EXECUTION AND IMPLEMENTATION.

- Execution: $n + \delta$ iterations of the recurrence, each one clock cycle
- Iterations (cycles) labeled from $-\delta$ to $n - 1$
- One digit of each input introduced during cycles $-\delta$ to $n - 1 - \delta$ and digits value 0 thereafter
- Result digits 0 for cycles $-\delta$ to -1 and z_1 is produced in cycle 0
- Result digit z_j is output in cycle j (one extra cycle to output z_n)

- The actions in cycle j:
  - Input $x_{j+1+\delta}$ and $y_{j+1+\delta}$.
  - Update $x[j+1] = (x[j], x_{j+1+\delta})$ and $y[j+1] = (y[j], y_{j+1+\delta})$ by appending the input digits.
  - Compute $v[j] = rw[j] + H_1$
  - Determine z_{j+1} using the selection function.
  - Update $z[j+1] = (z[j], z_{j+1+\delta})$ by appending the result digits.
  - Compute the next residual $w[j+1] = v[j] + H_2(z_{j+1})$
  - Output result digit z_j

IMPLEMENTATION

- Similar structure in all online algorithms:

  $\rightarrow$ all implemented with the same basic components, including

(i) Registers to store operands, results, and residual vectors;

(ii) Multiplication of vector by digit - trivial in radix 2;

(iii) Append units to append a new digit to a vector - concatenation;

(iv) Two-operand and multioperand redundant adders, such as signed digit adders, [3:2] carry-save adders and their generalization to [4:2] and [5:2] adders;

(v) Converters from redundant representations (i.e., signed digit and carry save) to conventional representations - on-the-fly converters (OTFC);

(vi) Carry-propagate adders of limited precision (3 to 6 bits) to produce estimates of the residual used in the digit-selection; and

(vii) Digit-selection schemes to obtain output digits - low precision.

Online vs. Conventional Arithmetic: Performance

- $z = E(\underline{\text{x}})$, z is scalar, $\underline{\text{x}}$ argument vector of n-digit elements, E a scalar expression

- Network of L levels of non-pipelined units to evaluate E

- Conventional arithmetic: units at level i wait for all units at level $i-1$ to finish

- Online arithmetic: unit at level i begins when $\delta + 1$ input digits are received

- Latency of a network of L levels of online units:

$$T_{OL} \leq [SUM_{i=1}^{L}(\delta_{max} + 1) + n]t_d$$

- Latency of a network of L levels of conventional units:

$$T_{CONV} \leq SUM_{i=1}^{L}(T_{imax} + t_{LOAD})$$

  – t_{imax} time of slowest operation at level i; t_{LOAD} time to transfer operands

SPEEDUP ANALYSIS

- Assume
  - $\delta_{imax} = 3$
  - $T_{imax} = cnt_d$, $c = 1$ for $T = O(n)$, $c = (log\ n)^2/n$ for $T = O(log^2 n)$
  - $t_{LOAD} = t_d$
  - Same number of units
  - Speedup

$$S = \frac{T_{CONV}}{T_{OL}} = \frac{L(cn+1)}{n+4L}$$

SPEEDUP ANALYSIS (cont)

- Minimum L so that $T_{OL} < T_{CONV}$

$$L_{min} = \lceil \frac{n}{cn-3} \rceil$$

Let $n = 32$ and $c = 25/32$, then $L_{min} = 2$

- Speedup for large L

$$S \rightarrow (cn+1)/\delta_{max}$$

Let $n = 32$ and $c = 25/32$, then $S_{max} < 9$

SPEEDUP ANALYSIS (cont)

- Evaluation of vector expressions
  - V vector operands, one vector result, each of M elements (n digits each)
  - Pipelined units: conventional with N stages; online array type, also pipelined
  - L network levels

c	n	N	M	S_p
25/32	32	4	100	4.9
1	32	4	100	6.2
1	32	8	1000	3.9
1	64	4	1000	15.0

- For large number of operands, $M \rightarrow \infty$, the speedup is

$$S_p = cn/N$$

- For large number of levels, $L \rightarrow \infty$, the speedup is

$$S_p = cn/4$$

- The additional speedup due to online arithmetic: 2 to 16 for typical precision

- What about cost? From analysis,

$$C_{OL} < C_{CONV} \text{ if } C_{OL-module} \leq 2 \times C_{CONV-module}$$

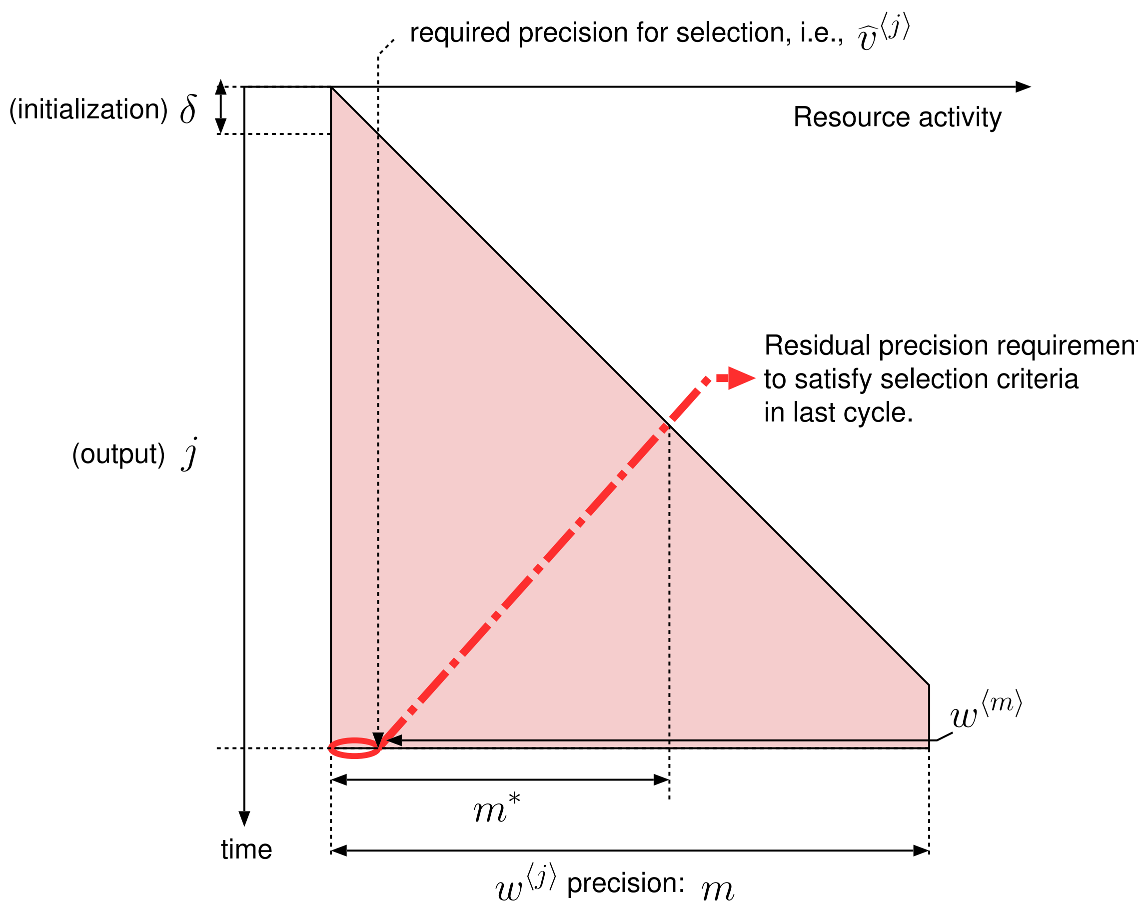
WHY IS LR ARITHMETIC USEFUL?

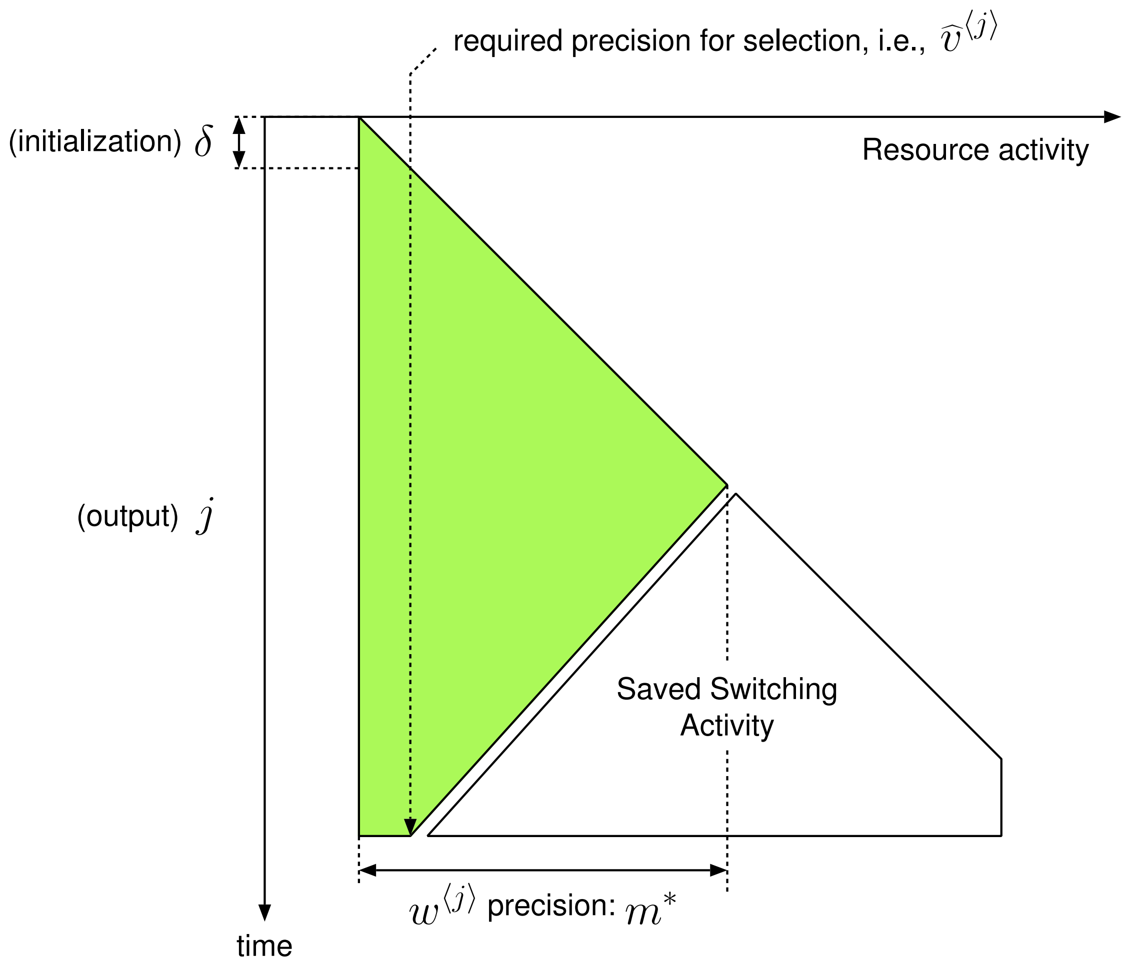
In LR mode, we can start an operation before finishing the previous one. Implications:

- Overlap data transmission and computation
- Limit data dependency: a constant delay independent of n

  $\rightarrow$ Suitable for arithmetic-intensive applications with data dependencies (recursive computations)
- Reduce width of interconnections

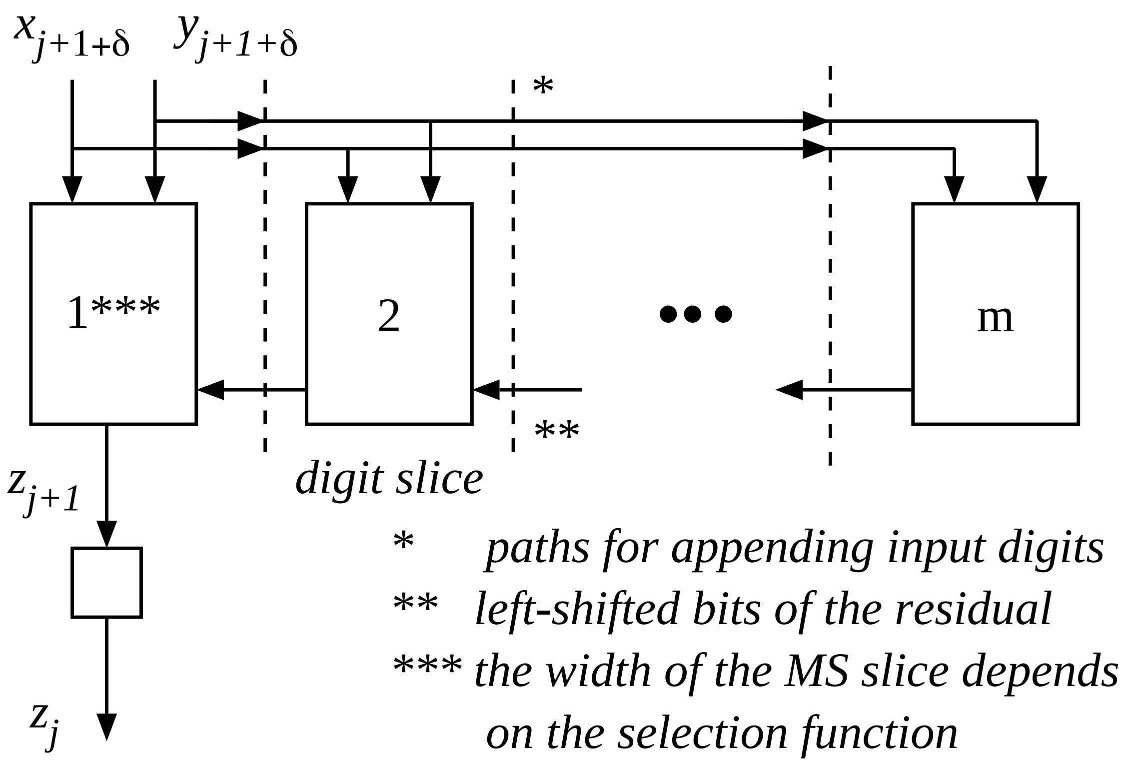
- Reduce signal activities - saving energy



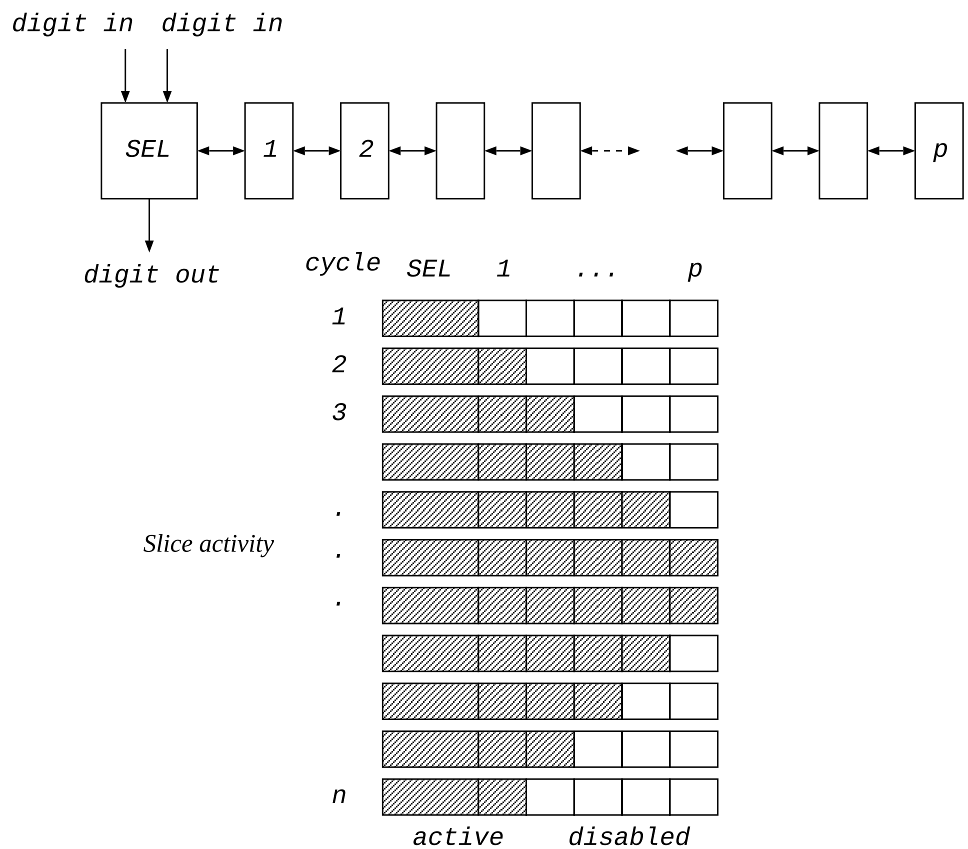

required precision for selection, i.e., $\widehat{v}^{\langle j \rangle}$
(initialization) δ
Resource activity
(output) j
Saved Switching Activity
$w^{\langle j \rangle}$ precision: m^*
time

- Easy support for variable precision computations:
  stop when sufficient accuracy achieved
- Zero-bias truncation instead of rounding: $1/2\ ulp$ error
- Expose parallelism between all operations: overlap always possible
- LR model provides a basis for composite (fused) operations
- Simplifies both partitioning into modules and their designs

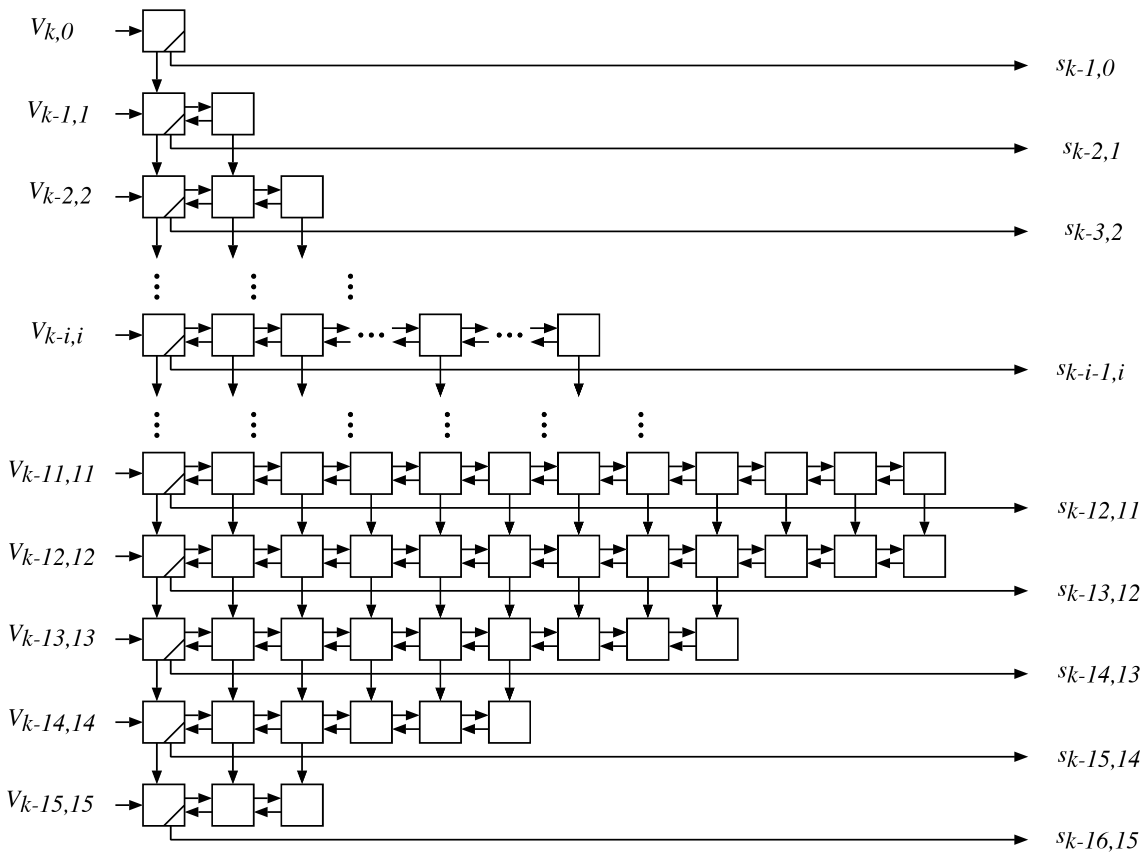
- Radix-r digit slice is the basic element $\rightarrow$ design effort largely independent of precision



- Reduced energy wrt serial-parallel operation: not all slices need to be active all the time



- Overall throughput is determined by the delay of a digit computation: digit-level pipelining possible



- LR arithmetic requires redundant representations

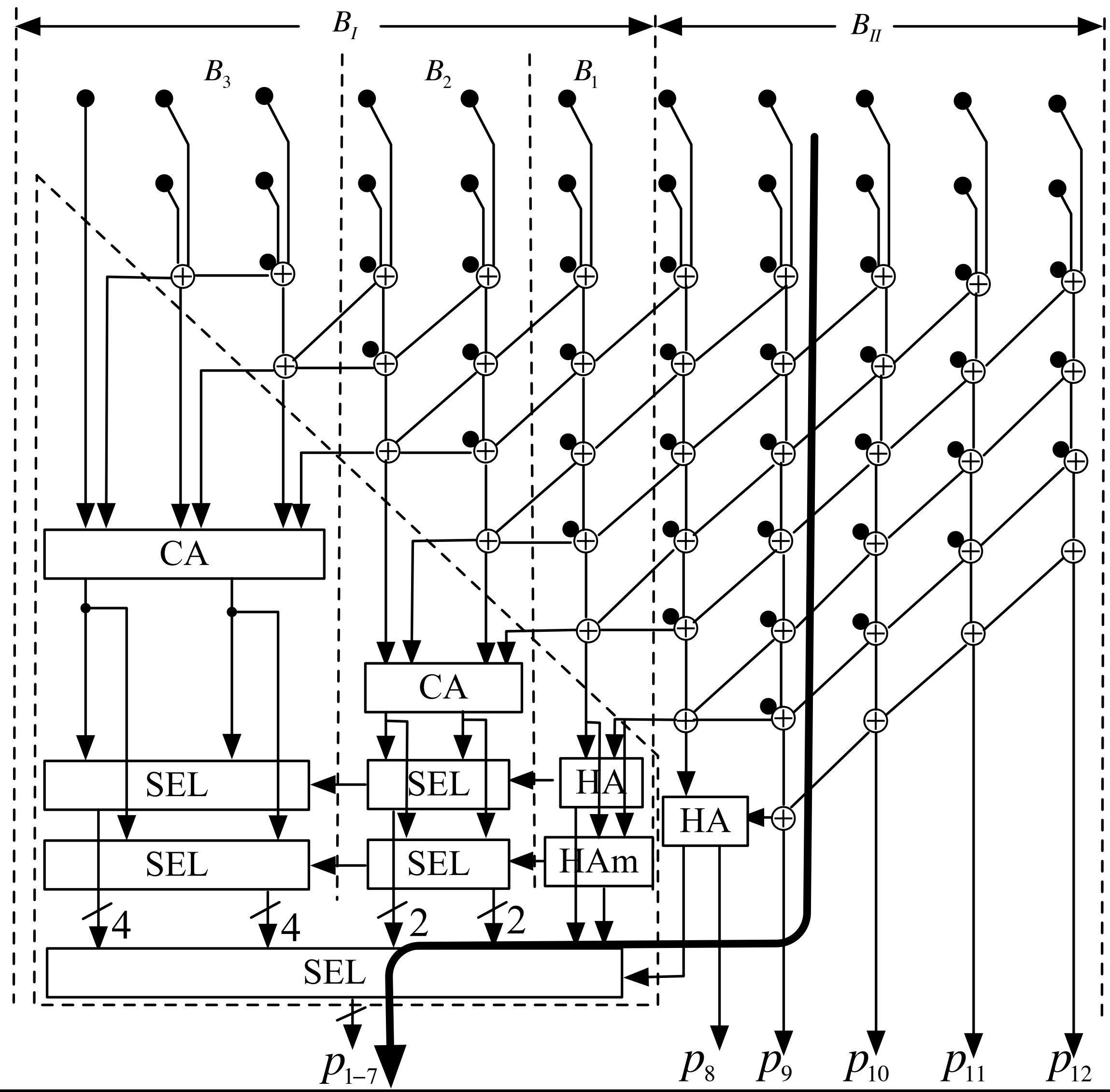
  – e.g., signed-digit sets {-1,0,1} for r=2, {-2, -1,0,1,2} for r=4, carry-save {0,1,2}, etc

  – higher cost/bit: 100% for binary, 50% for r=4, 20% for r=16, ... interestingly 0% for r=10

- On-the-fly conversion: Converts redundant LR (online) results into conventional representation without extra delay.

- Conventional digit-recurrence division, square root, function evaluators produce redundant results MSD first - compatible with LR arithmetic

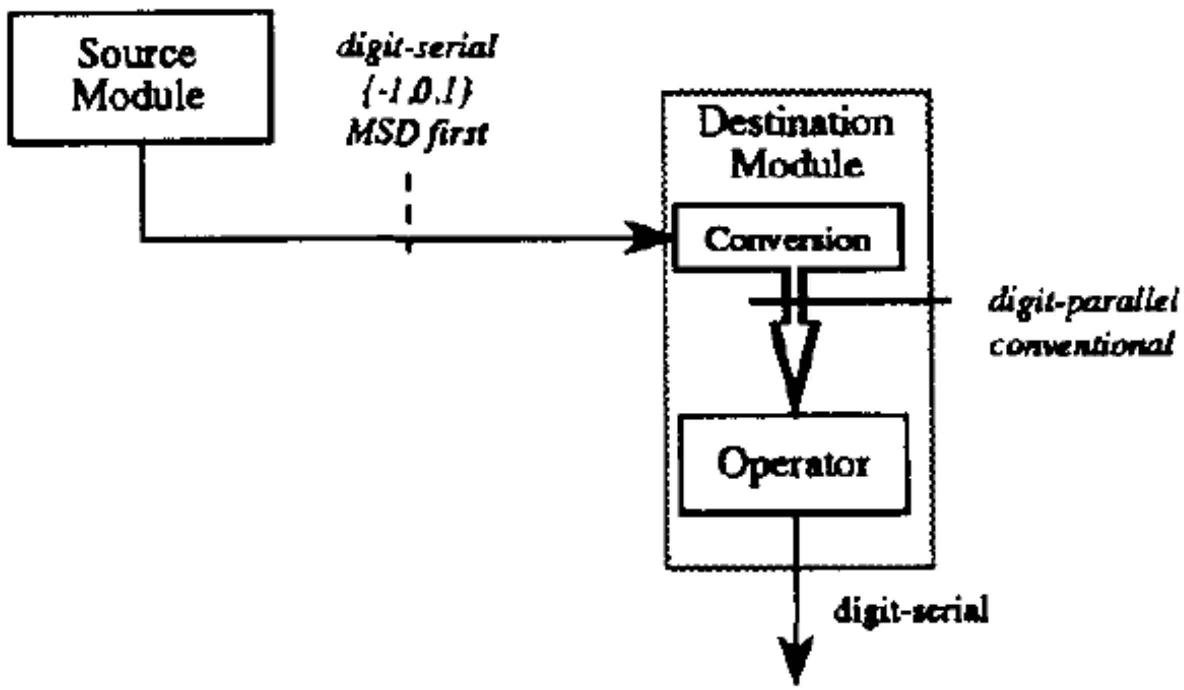
- In left-to-right carry-free (LRCF) multiplication (linear array):
  - MS and LS parts of the product obtained in parallel with the reduction of partial products
  - The final MS part of the product produced on-the-fly with carry-select adders
  - This eliminates the delay of the final addition, typically around 20-30% of the total delay



B_I
B_{II}
B_3
B_2
B_1
CA
CA
SEL
SEL
SEL
SEL
SEL
HA
HAm
HA
4
4
2
2
p_{1-7}
p_8
p_9
p_{10}
p_{11}
p_{12}

- Reduction of intermodule communication bandwidth:
  - Approach A: Source module: compute result MSDF, convert on-the-fly, transmit in parallel to the destination module – full precision communication
  - Approach B: Source module: compute result MSDF, transmit serially to the destination module during computation; convert to parallel using OTFC in the destination module – serial communication

  Approaches A and B have the same total latency but vastly different communication cost.


Source Module
digit-serial
{-1,0,1}
MSD first
Destination Module
Conversion
digit-parallel
conventional
Operator
digit-serial

Composite (fused) Algorithms

- To reduce the overall online delay of a group of operations, several operations combined into a single *multi-operation online algorithm*

- Example: Sum of squares $x^2 + y^2 + z^2$; Inputs in [1/2,1), output in [1/4, 3).

- Online delay $\delta_{ss} = 0$ when the output digit is over-redundant vs. (3+2+2=7) of a network of separate online operators

- Example: Givens rotation operator $y = \frac{x}{(x^2+y^2)^{1/2}}$

- Off-diagonal operations overlapped with computation of rotation factors

SUM OF SQUARES

1. $[Initialize]$

   $w[0] = x[0] = y[0] = z[0] = 0$

2. $[Recurrence]$

   for $j = 0 \ldots n-1$

   $v[j] = 2w[j] + (2x[j] + x_j 2^{-j})x_j + (2y[j] + y_j 2^{-j})y_j + (2z[j] + z_j 2^{-j})z_j$

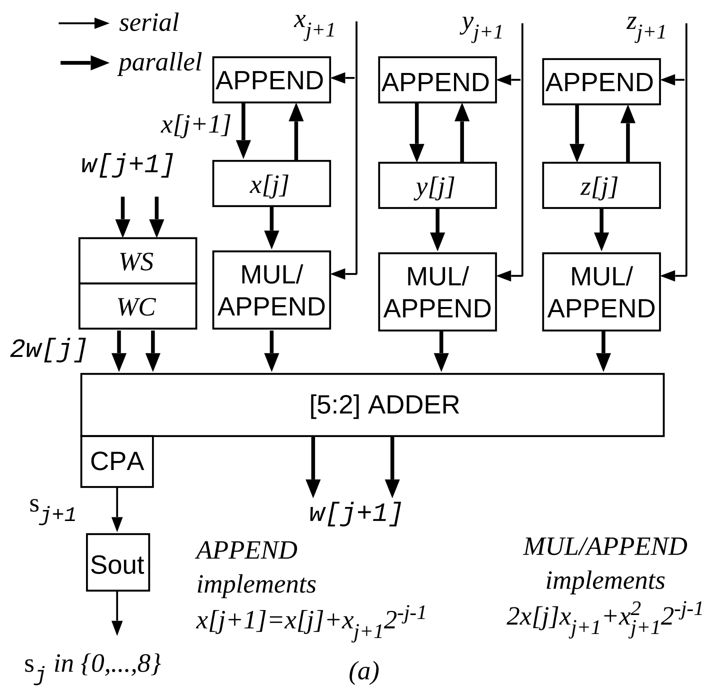
   $w[j+1] \leftarrow csfract(v[j])$

   $s_{j+1} \leftarrow csint(v[j])$

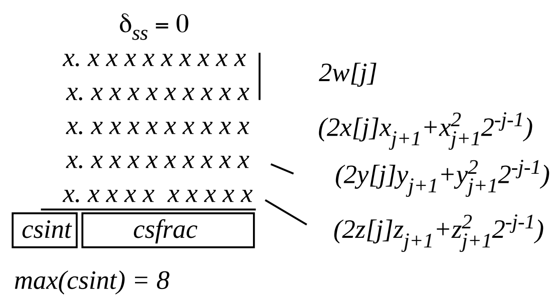
   $x[j+1] \leftarrow (x[j], x_{j+1});\ y[j+1] \leftarrow (y[j], y_{j+1});\ z[j+1] \leftarrow (z[j], z_{j+1})$

   $S_{out} \leftarrow s_{j+1}$

   end for



(a)



Note: the fractional portion of the 5-2 CSA produces at most three carries

(b)

Higher Order LR Arithmetic: the E-Method

Instead of evaluating an expression, find equivalent system of linear equations, and use online operations to solve it. Typically, the first component of the solution is equivalent to the value of the expression. Specifically,

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

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

$$f(x) \Rightarrow L : \mathbf{A} \cdot \mathbf{y} = \mathbf{b}$$

such that $y_1 = f(x)$.

2. Solve the system using LR digit-by-digit vector recurrences in m steps for m digit result.

Typical recurrence: $w2[j+1] = 2(w2[j] - d2_j - q_1 \cdot d1_j + x \cdot d3_j)$

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.

Notation

- Matrices and vectors of elements in boldface: the coefficient matrix **A** of order N; the solution vector **y** = $(y_1, \ldots, y_N)$; the right-hand side vector **b** = $(b_1, \ldots, b_N)$.

- The residual vector at step j;

$$\mathbf{w}[j] = (w1[j], \ldots, wN[j]) \tag{1}$$

- The result digit-vector at step j

$$\mathbf{d}[j] = (d1_j, \ldots, dN_j) \tag{2}$$

where digit $dk_j \in \{-1, 0, 1\}$ is the j-th digit of $y_k = SUM_{j=1}^{m} dk_j 2^{-j}$.

E-METHOD ALGORITHM

1. $[Initialize]$
   w$[0]$ = **b**; **d**$[0]$ = **0** ;
2. $[Recurrence]$
   for $j = 0 \ldots m-1$
   v$[j] = 2($**w**$[j]-$**Ad**$[j])$;
   d$[j+1] \leftarrow SEL(\mathbf{v}_{\mathbf{est}}[j])$;
   w$[j+1] \leftarrow$**v**$[j]$;
   $y_1[j+1] \leftarrow CONVERT(y_1[j], SEL(\mathbf{v}_{\mathbf{est}}[j])$
   end for
3. $[Result]$
   $y_1[m] \approx f(x)$

– Corresponds to SRT division in vector form where

- Residuals in redundant form, represented by the pseudo-sum WS and stored-carry WC bit-vectors.

- SEL is the digit selection function

$$dk_{j+1} = SEL(vk_{est}[j]) = \begin{cases} 1 & \textbf{if} \quad vk_{est}[j] \geq 0.5 \\ 0 & \textbf{if} \quad -0.5 \leq vk_est[j] \leq 0 \\ -1 & \textbf{if} \quad vk_{est}[j] \leq -1 \end{cases}$$

where $vk_{est}[j]$ is the estimate of $vk[j]$ truncated to one fractional bit.

Note that the multiplications in the term $\mathbf{A}\times\mathbf{d}[j]$ are implemented as digit-vector by digit multipliers - trivial in radices 2 and 4.

- Examples of mapping : a rational function $R_{2,3}(x)$ is mapped to the matrix/vector form:

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

Solving the system produces $\mathbf{y}$ such that:

$$y_1 = R_{3,2}(x) = \frac{p_2 x^2 + p_1 x + p_0}{q_3 x^3 + q_2 x^2 + q_1 x + 1}$$

– Note: no division used in solving L

Similarly, a polynomial $P_3(x)$ is mapped to the following system

$$\begin{bmatrix} 1 & -x & 0 & 0 \\ 0 & 1 & -x & 0 \\ 0 & 0 & 1 & -x \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

such that $y_1 = P_3(x) = p_3x^3 + p_2x^2 + p_1x + p_0$

EXAMPLE: EVALUATION OF A RATIONAL FUNCTION

- To evaluate the rational function $sinh(x) \approx R_{3,4}(x)$ to m bits, we iterate m times.

- The coefficients are obtained from rational function approximation of $sinh(x)$ in the interval $x \in [0, 1/6]$ with a relative error less than 10^{-13}.

- To satisfy the bounds and to have $a_{1,1} = 1$, the original coefficients are divided by q_0.We restrict the argument x to [0,1/8] and divide all normalized coefficients of P by 2 to make them $\leq 3/4$. This scaling requires one additional iteration. In illustrating the algorithm, we show only the first 12 steps producing the first 13 bits of the solution. The approximation has a relative error of 2^{-45} after 46 steps.

We illustrate the algorithm for $m = 12$. The normalized coefficients, rounded to 12 bits, are shown in hexadecimal:

$$
\begin{aligned}
p_3 &= 0.0d8 \\
p_2 &= 0.000 \\
p_1 &= 0.800 \\
p_0 &= 0.000 \\
q_4 &= 0.007 \\
q_3 &= 0.000 \\
q_2 &= -0.0fa \\
q_1 &= 0.000 \\
q_0 &= 1.000
\end{aligned}
$$

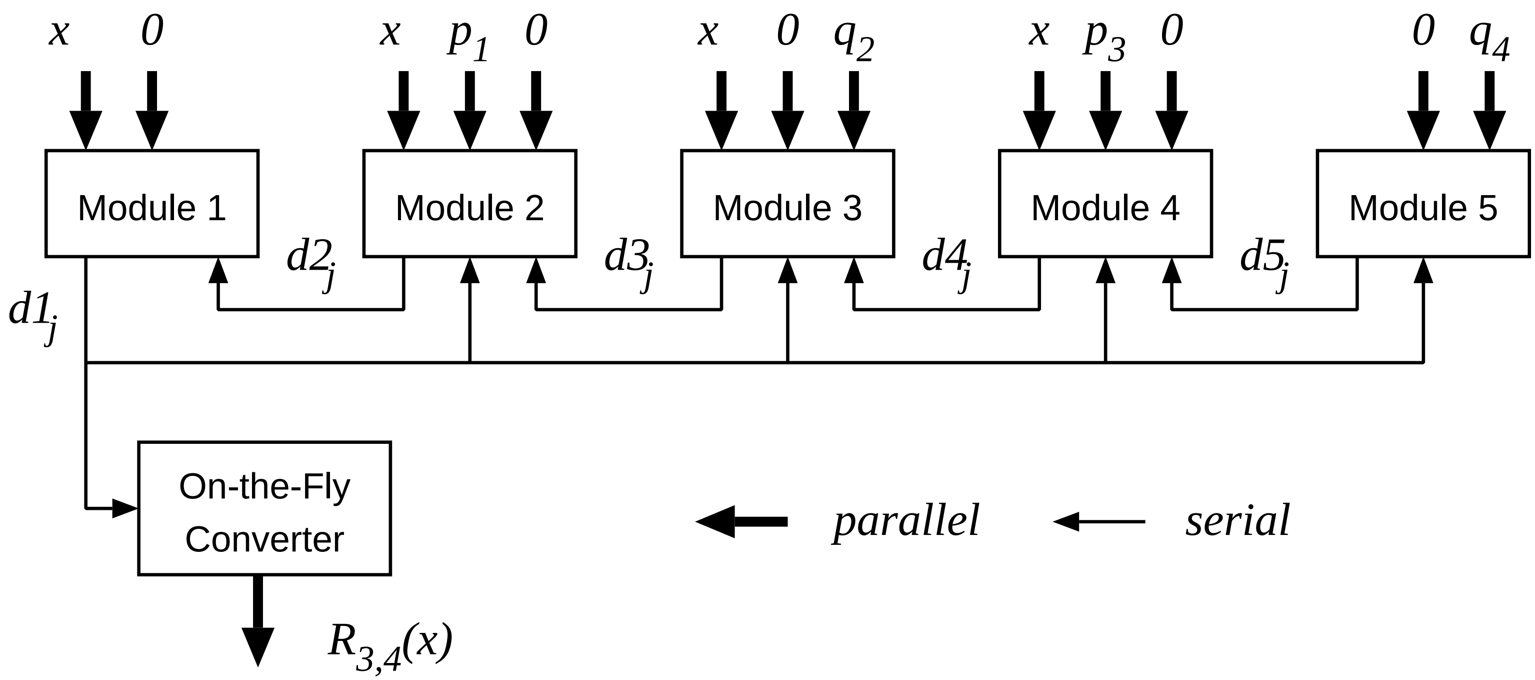
The recurrences are

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

The initial residuals are

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

THE NETWORK FOR EVALUATING RATIONAL FUNCTION



THE COMPUTATION TRACE

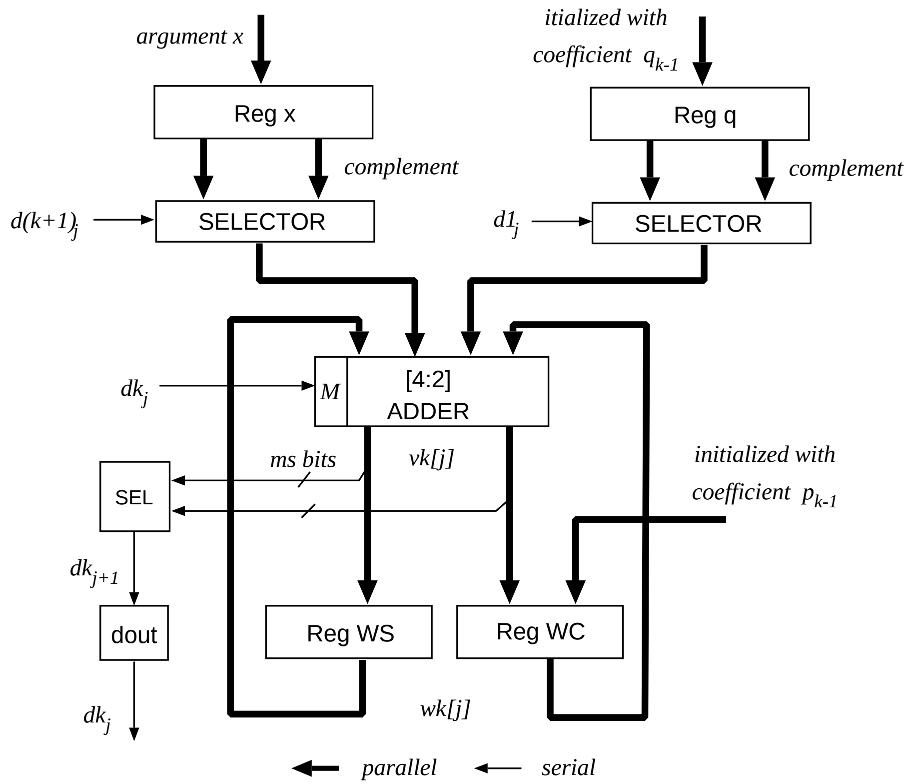
Evaluation of $sinh(0.10197)$ using rational approximation $R_{3,4}(x)$ and radix-2 E-method .

The error $|sinh(x) - y_1[13])| < 2^{-12}$. $y_1[13]$ is computed to compensate for the initial scaling of p coefficients by 2.

The evaluation of $R_{3,4}(x)$ for $x = 0.000110100001$ with 12-bit precision, showing non-redundant next residual $v1$ (for simplicity). Other residuals are not shown.

j	$v1[j]$	$d1_{j+1}$	$d2_{j+1}$	$d3_{j+1}$	$d4_{j+1}$	$d5_{j+1}$	$y_1[j+1]*$
0	0.000000000000	0	1	0	0	0	0.000000000000
1	0.001101000010	0	0	0	0	0	0.000000000000
2	0.011010000100	0	0	0	0	0	0.000000000000
3	0.110100001000	1	0	0	1	0	0.001000000000
4	-0.010111110000	0	0	0	0	0	0.001000000000
5	-0.101111100000	-1	0	1	-1	0	0.000110000000
6	0.100001000000	1	0	-1	1	0	0.000111000000
7	-0.111110000000	-1	0	0	-1	0	0.000110100000
8	0.000100000000	0	0	0	1	0	0.000110100000
9	0.001000000000	0	1	1	0	0	0.000110100000
10	0.011101000010	0	0	-1	0	0	0.000110100100
11	0.111010000100	1	-1	1	0	0	0.000110100010
12	-0.011000111010	0	1	0	0	-1	0.000110100010

IMPLEMENTATION



SEL block produces estimate and performs selection

M block performs subtraction of dk_j

(register control signals not shown)

Example: FUSED EXPRESSION LR EVALUATION

$$h = \frac{a(f+gc)+e(1+cd)}{1+ab+cd}$$

mapped to

$$\begin{bmatrix} 1 & -a & 0 \\ b & 1 & -c \\ 0 & d & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} e \\ f \\ g \end{bmatrix}$$

Solving the system produces $\mathbf{y}$ such that:

$$y_1 = h$$

- Conventional:
  - Cost: 5 multiplications, 3 additions, 1 division; full interconnect
  - Time to evaluate h: $2t_{mult} + t_{add} + t_{div}$;
- Online:
  - Cost: 3 eqv. serial-parallel (SP) multiplications; serial interconnect
  - Time: m iterations for m-bit result i.e., time of a single SP multiplication

LR iterative E-method vs 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

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

This step requires $(n-1)$ full precision multiplications and $(n-1)$ full precision additions per row per step

The E-method solves L by iterating

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

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 $(n-1)$ redundant additions

– A significant reduction in cost, delay and energy

LR ARITHMETIC: SUMMARY OF FEATURES

+ Digit-in/digit-out, left-to-right model of computation (after online delay of δ steps, small integer)

+ Overlaps communication and computation

+ Exposes massive digit-level concurrency via overlap, masks serial nature of individual operations

+ Handles well deep data-dependent expressions and recursive algorithms: full result not needed to start next operation

+ Data-dependency penalty: online delay δ – it does not matter if linear or nonlinear systems

+ Inherently variable precision; can stop any time; truncated result with unbiased error

+ Modular design with digit-serial input/output between modules: economy of design effort

+ No time penalty for operand alignment in FLPT addition

+ Online algorithms implementation similar to implementation of digit-recurrence algorithms

+ Algorithms and implementations developed for most of basic arithmetic operations and for certain composite operations

+ Larger set of operations possible than with LSDF approach

+ Higher level operators possible: e.g., E-method for solving systems of linear equations

On the other hand:

- Requires redundant representations – higher cost for lower radix than conventional implementations

- Single operations are serial (serial-parallel equivalent)

- Online delay sensitive to MSD cancellation: rare event

- More complex design for mult, div, sqrt than conventional designs

LR ARITHMETIC - POTENTIAL USES

- 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 design 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, e.g., IIR filters, root finders.

- In convolutional neural networks, multilayer perceptrons, and in backpropagation

- Reconfigurable architectures - fused operations, minimal interconnect, and variable-precision

- Low energy arithmetic - minimal signal activity

Bibliography - Ercegovac and collaborators

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

[Selected papers authored or coauthored by M.D.Ercegovac and his students; covers Online Arithmetic and the E-Method]

PhD and MS Theses: at UCLA and U of Illinois

- M.D Ercegovac, *Radix-16 Evaluation of Certain Elementary Functions*, MS Thesis, University of Illinois Urbana-Champaign, 1972.
- M.D Ercegovac, *A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer*, PhD Dissertation, U of I TR-750, 1975.
- M.M. Takata, *A Design of Modular Arithmetic Unit for Polynomial and Rational Function Evaluation*, MS Thesis, CSD UCLA, June 1978.

- V.G. Oklobdzija, *An On-Line Higher Radix Square Rooting Algorithm*, MS Thesis, CSD UCLA, June 1978.
- O. Watanuki, *Floating-Point On-Line Arithmetic For Highly Concurrent Digit-Serial Computation: Application to Mesh Problems*, PhD Thesis, CSD UCLA,1981
- A. Gorji-Sinaki, *Error-Coded Algorithms For On-Line Arithmetic*, PhD Thesis, CSD UCLA, 1981.
- D. M. Tullsen, *A Very Large Scale Integration Implementation of an On-Line Arithmetic Unit*, MS Thesis, June 1986, Report No. CSD-860094.
- R. Brackert, *Design and Implementation of A High Speed Recursive Digital Filter Using On-Line Arithmetic*, (co-chair with A.N. Wilson, Jr.), PhD Thesis, CSD-EE UCLA1989
- P. K.-G. Tu, *On-Line Arithmetic Algorithms for Efficient Implementation*, PhD Thesis, CSD UCLA,1990
- J. S. Fernando, *Design Alternatives for Recursive Digital Filters Using On-Line Arithmetic*, PhD Thesis, CSD UCLA, 1993.
- M. E. Louie, *Variable Precision Arithmetic with Lookup Table Based Field Programmable Gate Arrays*, PhD Thesis, CSD UCLA, 1994.

- R.D. McIlhenny, *Complex Number On-line Arithmetic for Reconfigurable Hardware: Algorithms, Implementations, and Applications*, PhD Thesis, CSD UCLA, 2002.
- A.F. Tenca, *Variable Long-Precision Arithmetic (VLPA) for Reconfigurable Architectures*, PhD Thesis, CSD UCLA, 1998.

Journal papers

- 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.
- D. Wang and M.D. Ercegovac, A Radix-16 Combined Complex Division/Square Root Unit with Operand Prescaling, *IEEE Trans. Computers*, 61(9):1243-1255, 2012.
- M.D. Ercegovac and J.-M. Muller, An Efficient Method for Evaluating Complex Polynomials. *Journal of Signal Processing Systems*, Volume 58, Issue 1, Page 17, Springer 2010, also published online http://www.springerlink.com/content/5582844402n0t2x1/

- D. Lau, A. Schneider, M.D. Ercegovac, and J.A. Villasenor, FPGA-based library for on-line signal processing. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 28(1-2):129-43, Kluwer Academic Publishers, May-June 2001.
- J.S. Fernando and M.D. Ercegovac, A Method of Eliminating Oscillations in High-Speed Recursive Digital Filters. *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, 44(10):861-864, 1997.
- J.S. Fernando and M.D. Ercegovac. Conventional and on-line arithmetic designs for high-speed recursive digital filters. *J. of VLSI Signal Processing*, 7:189–197, 1994.
- M.D. Ercegovac and T. Lang. On-the-fly rounding. *IEEE Trans. Comput.*, Vol. 41(12):1497–1503, Dec. 1992.
- P.K.-G. Tu and M.D. Ercegovac. Gate array implementation of on-line algorithms for floating-point operations. *J. of VLSI Signal Processing*, (3):307–317, 1991.
- M.D. Ercegovac and T. Lang. Fast multiplication without carry-propagate addition. *IEEE Trans. Comput.*, C-39(11):1385–1390, November 1990.
- M.D. Ercegovac and T. Lang. On-the-fly conversion of redundant into conventional

representations. *IEEE Trans. Comput.*, Vol. C-36(7):895–897, July 1987.

- M.D. Ercegovac. A general hardware-oriented method for evaluation of functions and computations in a digital computer. *IEEE Trans. Comput.*, C-26(7):667–680, July 1977.
- K.S. Trivedi and M.D. Ercegovac. On-line algorithms for division and multiplication. *IEEE Trans. Comput.*, C-26(7):681–687, July 1977.
- M.D. Ercegovac. Radix-16 evaluation of certain elementary functions. *IEEE Trans. Comput.*, Vol. C-22(6):561–566, June 1973.

Conference papers

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

- W. Yan and M. D. Ercegovac, Radix-4 Energy Efficient Carry-Free Truncated Multiplier, *Proc. 50th Asilomar Conference on Signals, Systems and Computers*, 2016.
- M.D. Ercegovac and L. Meng, Low-power Radix-4 Quotient Generator, *Proc. 48th Asilomar Conference on Signals, Systems and Computers*, 2014.
- H. Parta, M.D. Ercegovac and S. Pamarti, RF Digital Predistorter Implementation using Polynomial Optimization, IEEE 57th International Midwest Symposium on Circuits and Systems, 2014.
- M.D. Ercegovac, On Approximate Arithmetic, *Proc. 47th Asilomar Conference on Signals, Systems and Computers*, 2013.
- N. Brisebarre, S. Chevillard, M. D. Ercegovac, J.-M. Muller and S. Torres. An Efficient Method for Evaluating Polynomial and Rational Function Approximations. *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 233-238, July 2008.
- P. Dormiani and M.D. Ercegovac, ISA Extensions for Online Floating-Point Addition. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, Vol. 6697, 12 pps., 2007.

- M.D. Ercegovac and J.-M. Muller, Complex Multiply-Add and Other Related Operators. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, Vol. 6697, 12 pps., 2007.
- M.D. Ercegovac and J.-M. Muller, A Hardware-Oriented Method for Evaluating Complex Polynomials. *IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 122-127, 2007.
- M.D. Ercegovac and J.-M. Muller, Arithmetic Processor for Solving Tridiagonal Systems of Linear Equations. *Proc. 40th Asilomar Conference on Signals, Systems and Computers*, pp. 337-340, 2006.
- P. Dormiani and M.D. Ercegovac, Interconnection Scheme for Networks of Online Modules. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, pp. 631308-1:12, 2006.
- R. McIlhenny and M.D. Ercegovac, On the Design of an On-line Complex Householder Transform, *Proc. 40th Asilomar Conference on Signals, Systems and Computers*, pp. 318-322, 2006.
- P. Dormiani, D. Omoto, P. Adharapurapu, and M.D. Ercegovac, A Design of Online Scheme for Evaluation of Multinomials. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, 12 pps., 2005.

- P. Adharapurapu and M.D. Ercegovac, A Linear-System Operator Based Scheme for Evaluation of Multinomials. *Proc. 17th IEEE Symposium on Computer Arithmetic*, pp. 249-256, 2005.
- P. Adharapurapu and M.D. Ercegovac, A Composite Arithmetic Scheme for Evaluation of Multinomials. *Proc. 38th Asilomar Conference on Signals, Systems and Computers*, pp. 1889-1893, 2004.
- R. McIlhenny and M.D. Ercegovac, On the Design of an On-Line Complex FIR Filter. *Proc. 38th Asilomar Conference on Signals, Systems and Computers*, pp. 478-482, 2004.
- M. D. Ercegovac, Left-to-right squarer with overlapped LS and MS parts. In *Proc. 37th Asilomar Conference on Signals, Systems and Computers*, pp. 1451-1455, 2003.
- Z. Huang and M.D. Ercegovac, FPGA Implementation of Pipelined On-line Scheme for 3-D Vector Normalization, *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 61-70, 2001.
- M.D. Ercegovac. Left-to-Right Carry-Free Scheme for Computing $ab + cd$. *Proc. 34th Asilomar Conference on Signals, Systems and Computers*, pp.1330-3, 2000.

- A.F. Tenca and M.D. Ercegovac. On the Design of High-Radix On-Line Division for Long Precision. In *Proc. 14th IEEE Symposium on Computer Arithmetic*, pages 59–66, 1999.
- A.F. Tenca, M.D. Ercegovac and M. Louie. Fast On-Line Multiplication Using LSA Organization. *Proc. SPIE on Image Processing Architectures, Digital Signal Processing*, volume 3807, 1999.
- A.F. Tenca and M.D. Ercegovac. A Variable Long-Precision Arithmetic Unit Design for Reconfigurable Coprocessor Architectures. *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 216-225, 1998.
- A.F. Tenca and M.D. Ercegovac. A High-Radix Multiplier Design for Variable Long-Precision Computations. *31st Asilomar Conference on Signals, Systems and Computers*, pages 1173-1177, 1997.
- A.F. Tenca and M.D. Ercegovac. Design of High-Radix Digit-Slices for On-Line Computations. *Proc. SPIE on High-Speed Computing, Digital Signal Processing, and Filtering using Reconfigurable Logic*, volume 2914, pages 14–25, 1996.
- M. Louie and M.D. Ercegovac. On digit-recurrence division implementations for field programmable gate arrays. *Proc. 11th IEEE Symposium on Computer Arithmetic*, pages 202–209, 1993.

- M. Louie and M.D. Ercegovac. A digit-recurrence square root implementation for field programmable gate arrays. In *Proc. IEEE Workshop on FPGAs for Custom Computing Machines*, pages 178–183, 1993.
- M. Louie and M. Ercegovac. Mapping division algorithms to field programmable gate arrays. *Proc. 26th Asilomar Conference on Signals, Systems, and Computers*, 1992.
- M.D. Ercegovac. On-line arithmetic for recurrence problems. *Proc. SPIE, Vol.1566, Advanced Signal Processing Algorithms, Architectures, and Implementations II*, pages 263–274, 1991.
- P.K. Tu and M.D. Ercegovac. Application of on-line arithmetic algorithms to the SVD computation: Preliminary results. *Proc. 10th IEEE Arithmetic Symposium*, pages 246–255, 1991.
- P.K. Tu and M.D. Ercegovac. Gate array implementation of on-line algorithms for floating-point operations. *Proc. 24th Asilomar Conference on Signals Circuits and Computers*, 1990.
- M.D. Ercegovac and T. Lang. Most-significant-digit-first and on-line arithmetic approaches for the design of recursive filters. *23rd Asilomar Conference on Signals, Systems and Computers*, pages 7–11, 1989.

- P. Tu and M.D. Ercegovac. Design of on-line division unit. *Proc. 9th IEEE Symposium on Computer Arithmetic*, pages 42–49, 1989.
- R.H. Brackert, M.D. Ercegovac, and A. Willson. Design of an on-line multiply-add module for recursive digital filters. *Proc. 9th IEEE Symposium on Computer Arithmetic*, pages 34–41, 1989.
- M.D. Ercegovac and T. Lang. On-line arithmetic for DSP applications. *Proc. 32nd IEEE Midwest Symposium on Circuits and Systems*, 1989.
- R.H. Brackert, A.N. Willson, and M.D. Ercegovac. Recursive filter using on-line arithmetic. *Proc. IEEE International Symposium on Circuits and Systems*, pages 1552–1556, 1989.
- M.D. Ercegovac and T. Lang. On-line schemes for computing rotation angles for SVDs. *Proc. SPIE on Advanced Signal Processing Algorithms, Architectures, and Implementations*, San Diego, pages 160-169, 1987.
- M.D. Ercegovac and T. Lang. On-line scheme for computing rotation factors. *Proc. 8th IEEE Symposium on Computer Arithmetic*, pages, 196-203, 1987.
- P. Tu and M.D. Ercegovac. A radix-4 on-line division algorithm. In *8th IEEE Symposium on Computer Arithmetic*, pages, 1-8, 1987.

- D. Tullsen and M.D. Ercegovac. Design and implementation of an on-line algorithm. In *Proc. SPIE Conference on Real-Time Signal Processing*, pages 92-99, San Diego, August 1986.
- M.D. Ercegovac. On-line arithmetic: An overview. *SPIE Vol. 495 Real-Time Signal Processing VII*, pages 86–93, 1984.
- A.L. Grnarov and M.D. Ercegovac. On-line multiplicative normalization. *Proceedings of the 6-th IEEE Symposium on Computer Arithmetic*, Aarhus, Denmark, pages 151-155, 1983.
- O. Watanuki and M.D. Ercegovac. Floating-point on-line arithmetic: Algorithms. *Proc. 5th IEEE Symposium on Computer Arithmetic*, pages 81–86, 1981.
- A. Gorji-Sinaki and M.D. Ercegovac. Design of a digit-slice on-line arithmetic unit. *Proc. 5th IEEE Symposium on Computer Arithmetic*, pages 72–80, 1981.
- C.S. Raghavendra and M.D. Ercegovac. A simulator for on-line arithmetic. *Proc. 5th IEEE Symposium on Computer Arithmetic*, pages 72–80, 1981.