DIGITAL ARITHMETIC

Miloš D. Ercegovac
Computer Science Department
University of California Los Angeles
and
Tomás Lang
Department of Electrical and Computer Engineering
University of California at Irvine

ABOUT THE BOOK

1. Overview [Chapter 1]
2. Two-Operand Addition [Chapter 2]
3. Multi-operand Addition [Chapter 3]
4. Multiplication [Chapter 4]
5. Division by Digit Recurrence [Chapter 5]
6. Square Root by Digit Recurrence [Chapter 6]
7. Reciprocal, Division, Reciprocal Square RToot, and Inverse Square Root by Iterative Approximation [Chapter 7]
8. Floating-point Representation, Algorithms, and Implementations [Chapter 8]
9. Digit-Serial Arithmetic [Chapter 9]
10. Function Evaluation [Chapter 10]
11. CORDIC Algorithm and Implementations [Chapter 11]

Chapter 1: Review of the Basic Number Representations and Arithmetic Algorithms

- General-purpose processors

  ◇ Main use: numerical computations

  ◇ Address calculations

- basic operations

- fixed point and floating point

- IEEE standard

- vector processors

- Special-purpose (application-specific) processors

- for numerically intensive applications

- single computation or classes of computations

Application-specific processor

- Areas of application:
  - signal processing
  - embedded systems
  - matrix computations
  - graphics, vision, multi-media
  - cryptography and security
  - robotics, instrumentation; others?
- Features:
  - better use of technology
  - improvement in speed, area, power
  - flexibility in
    - implementation; decomposition into modules
    - number systems and data formats; algorithms
- Need good design tools; difficult to change; FPGAs?

GENERAL-PURPOSE VS. APPLICATION-SPECIFIC

- Flexibility
- Matching specific applications
- Use of VLSI and special technology
- Use of hardware-level parallel processing
- Lower software overhead

ARITHMETIC PROCESSORS: USER'S VIEW

AP = (operands, operation, results, conditions, singularities)

- Numerical operands and results specified by
  - Set of numerical values $x \in N$ (finite and ordered set)
  - Range $V_{min} \leq x \leq V_{max}$
  - Precision
  - Number representation system (NRS)
- Set of operations: addition, subtraction, multiplication, division, ...
- Conditions: values of the results – zero, negative, etc.
- Singularities: Illegal results – overflow, underflow, Nan, etc.

LEVELS OF DESCRIPTION AND IMPLEMENTATION

- Numerical computations (applications)
- Algorithms
- Arithmetic operations

Operations	Operands-Result	Algorithm
Numerical function	Numbers	Numerical Algorithm
	$\downarrow$	$\downarrow$
$\downarrow$	*Number System*	*Arithmetic design*
	$\downarrow$	$\downarrow$
Digit-vector function	Digit vector	Digit-vector Algorithm
	$\downarrow$	$\downarrow$
$\downarrow$	*Digit Coding*	*RTL design*
	$\downarrow$	$\downarrow$
Bit-vector function	Bit vector	Bit-vector Algorithm
		$\downarrow$
		Logic Design

NUMBER REPRESENTATION SYSTEMS

value: $x \in N \Longrightarrow$NRS $\Longrightarrow X \in V$: digit vector

- Digit Vector

$$X = (x_a, \ldots, x_i, \ldots, x_b)$$

  – indexing:

  Leftward Zero Origin (LZ) (integers)

$$X = (x_{n-1}, x_{n-2}, \ldots, x_0)$$

  Rightward One Origin (RO) (fractions)

$$X = (x_1, x_2, \ldots, x_n)$$

  – Digit set D_i - set of values for digit x_i (usually consecutive integers)

NUMBER REPRESENTATION (cont.)

- Number of (unambiguously) representable numbers

$$|N| \leq \Pi |D_i|$$

- Number representation system

$$F : N \rightarrow V$$

- Choose NRS to
  - allow efficient computation(s)
  - suitable interface with other systems
- Different implementation-performance constraints

$\Longrightarrow$ variety of NRS

SOME CHARACTERISTICS OF NRS

a) Range: finite set of digit-vector values

b) Unambiguity: two numbers should not have same representation

$$\text{If } x \in N,\ y \in N,\ x \neq y \text{ then } F(x) \neq F(y)$$

c) Nonredundant/redundant

Redundant: $F^{-1}(X) = F^{-1}(Y)$

WEIGHTED NUMBER REPRESENTATION SYSTEMS

Integer x represented by digit vector $X = (x_{n-1} \ldots, x_0)$,

$$x = \sum_{i=0}^{n-1} x_i \cdot w_i$$

where

$$W = (w_{n-1}, \ldots, w_0) \; \textit{weight vector}$$

Define

$$R = (r_{n-1}, \ldots, r_0) \; \textit{radix vector}$$

so that

$$w_0 = 1 \; w_i = w_{i-1} r_{i-1}$$

WEIGHTED NUMBER REPRESENTATION (cont.)

- Fixed-radix NRS

$$r_i = r$$

Then $w_i = r^i$ so that

$$x = \sum_{i=0}^{n-1} x_i r^i$$

- Canonical digit set

$$D_i = \{0, 1, 2, \ldots, |r_i| - 1\}$$

- Conventional number system

  – Fixed radix positive
  – Canonical digit set

NON-CONVENTIONAL FIXED-RADIX SYSTEM

- Negative radix

  $r = -2$, $x = \Sigma_{i=0}^{n-1} x_i (-2)^i$

  1011 = (-8) + 0 + (-2) + 1 = -9

  0111 = 0 + 4 + (-2) + 1 = 3

- Complex radix

  $r = 2j$, $j = \sqrt{-1}$, $x_i \in \{0, 1, 2, 3\}$ (Knuth's quarter imaginary NRS)

  W: -8j -4 +2j +1

  1231 $\Longrightarrow 1 \times (-8j) + 2 \times (-4) + 3 \times (2j) + 1 \times 1 = -7 - 2j$

- Non-canonical digit set, $r = 2$, {-1,0,1} or {0,1,2}

  Example: radix 4 $D = \{-3, -2, -1, 0, 1, 2, 3\}$

  $x = 27$ represented by $(1, 2, 3)$ or $(2, -2, 3)$

REDUNDANT NRS

- Fixed radix r
- Non-canonical digit set

$$D = \{-a, -a+1, \ldots, -1, 0, 1, \ldots, b-1, b\}$$

- Symmetric if $a = b$
- Redundant $a + b + 1 > r \quad (a, b \le r - 1)$
  - "standard" $a, b \le r - 1$
  - over-redundant $a, b > r - 1$
  - Redundancy factor

$$\rho \ = \ \frac{a}{r-1}, \quad \rho > \frac{1}{2}$$

EXAMPLES OF REDUNDANT DIGIT SETS

r	a	Digit set	ρ	Comment
2	1	{ -1, 0, 1}	1	minimally/maximally redundant
4	2	{-2, -1, 0, 1, 2}	2/3	minimally redundant
4	3	{-3, -2, . . . , 2, 3}	1	maximally redundant
4	4	{-4, . . . , 4}	4/3	over-redundant
9	4	{-4, . . . , 4}	1/2	non-redundant
10	5	{-5, . . . , 5}	5/9	minimally redundant
10	6	{-6, . . . , 6}	2/3	redundant
10	9	{-9, . . . , 9}	1	maximally redundant
10	13	{-13, . . . , 13}	13/9	over-redundant

MIXED-RADIX NUMBER SYSTEM

- $r_i \neq r_j$
- Example: Representation of time $R = (31, 24, 60, 60)$
- Example: Factorial number system

$$\begin{aligned} r_i &= i+2, \quad i = 0, \ldots, n-1 \\ R &= (n+1, n, \ldots, 3, 2) \\ w_i &= (i+1)! \end{aligned}$$

Canonical digit set

Integers in range $0 \leq x \leq (n+1)! - 1$

NON-WEIGHTED NUMBER SYSTEMS: RESIDUE (RNS)

- Base vector B of moduli m_i

$$B = (m_{n-1}, m_{n-2}, \ldots, m_0)$$

  m_i positive integers and pairwise relatively prime

- Integer x is represented by vector

$$X = (x_{n-1}, x_{n-2}, \ldots, x_0)$$

  where $x_i = x \ mod \ m_i$

- Represents uniquely integers in the range

$$0 \leq x < \Pi_{i=0}^{n-1} m_i$$

(more later)

REPRESENTATION OF SIGNED INTEGERS

A. Directly in the number representation

Examples: Signed-Digit Number System

B. With an extra symbol: Sign-and-magnitude

C. Additional mapping on positive integers

$$\begin{array}{c}\text{Signed integers } x\\ \downarrow\\ \text{Positive integers } x_R\\ \downarrow\\ \text{Digit-vectors } X\end{array}$$

Examples:

- True-and-Complement (TC):
  - 2's complement
  - 1s' complement
- Biased representation

TRUE-AND-COMPLEMENT SYSTEM

- $-k \leq x \leq k$ signed integer (implicit value)
- x_R positive integer (representation value)
- C – complementation constant
- Mapping $x_R = x \ mod \ C$
- Unambiguous if $k < C/2$
- Equivalent to

$$x_R = \begin{cases} x & \textbf{if } x \geq 0 \\ C - |x| & \textbf{if } x < 0 \end{cases}$$

- Converse mapping

$$x = \begin{cases} x_R & \textbf{if } x_R < C/2 \\ x_R - C & \textbf{if } x_R > C/2 \end{cases}$$

NUMBER REPRESENTATION (cont.)

- x_R represented in any number system
- In fixed-radix system two common choices:
  - 2's complement: $C = r^n$ (Range-complement system)
  - 1s' complement: $C = r^n - 1$ (Diminished-radix-complement system)

BIASED REPRESENTATION

- B – bias
- $-k \leq x \leq k$
- $x_R = x + B$
- $B \geq k$

TYPES OF ARITHMETIC ALGORITHMS

- Bottom-up development

  Primitives

  + Addition/subtraction
  + Multioperand addition
  + Arithmetic shifts
  + Multiplication by digit
  + Result-digit selection (PLA)
  + Table look-up
  + Multiplication

- Algorithms

+ Composition of primitives

TYPES (cont.)

- (Digit) Recurrences (continued sums)

◇ Residual recurrence: $R[i+1] = f(R[i], X, Y, Z[i], z_{i+1})$

Uses: Add/sub, single-position shifts, multiplication by digit

◇ Output digit selection: $z_{i+1} = g(R[i], X, Y, Z[i])$

(keep $R[i+1]$ bounded)

Uses: Comparisons, PLA

◇ Result recurrence

$$Z[i+1] = Z[i] + z_{i+1}r^{i+1} \text{ (continued sum)}$$

Uses: Concatenation

- Examples:

◇ multiplication $R[i+1] = \frac{1}{r}(R[i] + X \cdot r^n y_i)$

◇ division $R[i+1] = rR[i] - q_{i+1}Y \quad q_{i+1} = g(R[i+1], Y)$

Types of algorithms

- Continued product recurrences

  $R[i+1] = f(R[i], X, Y, Z[i], z_{z+1})$

  Uses: Add/sub, variable shifts, mult. by digit

  $z_{i+1} = g(R[i], X, Y, Z[i])(\text{keep } R[i+1] \text{ bounded})$

  Uses: Comparisons, PLA

  $Z[i+1] = Z[i](1 + z_{i+1}r^{-(i+1)}) \quad (\text{continued product})$

  Uses: Variable shift, addition

- Example:

◇ division

$$
\begin{aligned}
R[i+1] &= rR[i](1 + q_{i+1}r^{-(i+1)}) + q_{i+1} \\
q_{i+1} &= g(R[i], Y) \\
Q[i+1] &= Q[i](1 + q_{i+1}r^{-(i+1)})
\end{aligned}
$$

TYPES (cont.)

- Iterative Approximations

  $Z[i+1] = f(Z[i], X, Y) \;\; \text{until } g(Z(i)) < \varepsilon$

  Example:

  ◇ reciprocal

  $Z[i+1] = Z[i](2 - Z[i]X)$

- Polynomial Approximations

  $z = a_0 + a_1 x + a_2 x^2 + \cdots$

PERFORMANCE

- Measures
  - + Execution time
  - + Throughput
- Improving speed

a) Arithmetic level

+ Reducing number of steps
  Example: higher radix
  Example: combinational instead of sequential
+ Reducing time of step
  Example: carry-save adder instead of carry-propagate
+ Overlap steps (concurrency/pipelining)
  Example: multiple generation and addition (in mult.)
  Example: simultaneous additions (in mult.)

b) Implementation level

+ Reduce number of logic levels

POWER AND COST

- Measures
  - + Packaging
  - + Interconnection complexity
  - + Number of pins
  - + Number of chips and types of chips
  - + Number of gates and types of gates
  - + Area
  - + Design cost; verification and testing cost
  - + Power dissipation
  - + Power consumption
- Reduction of cost