Multiplexer-Based Array Multipliers

Presented by:

Kevin Biswas

April 1, 2005

Supervisor:

Dr. M. Ahmadi

University of Windsor
Electrical and Computer Engineering

Background

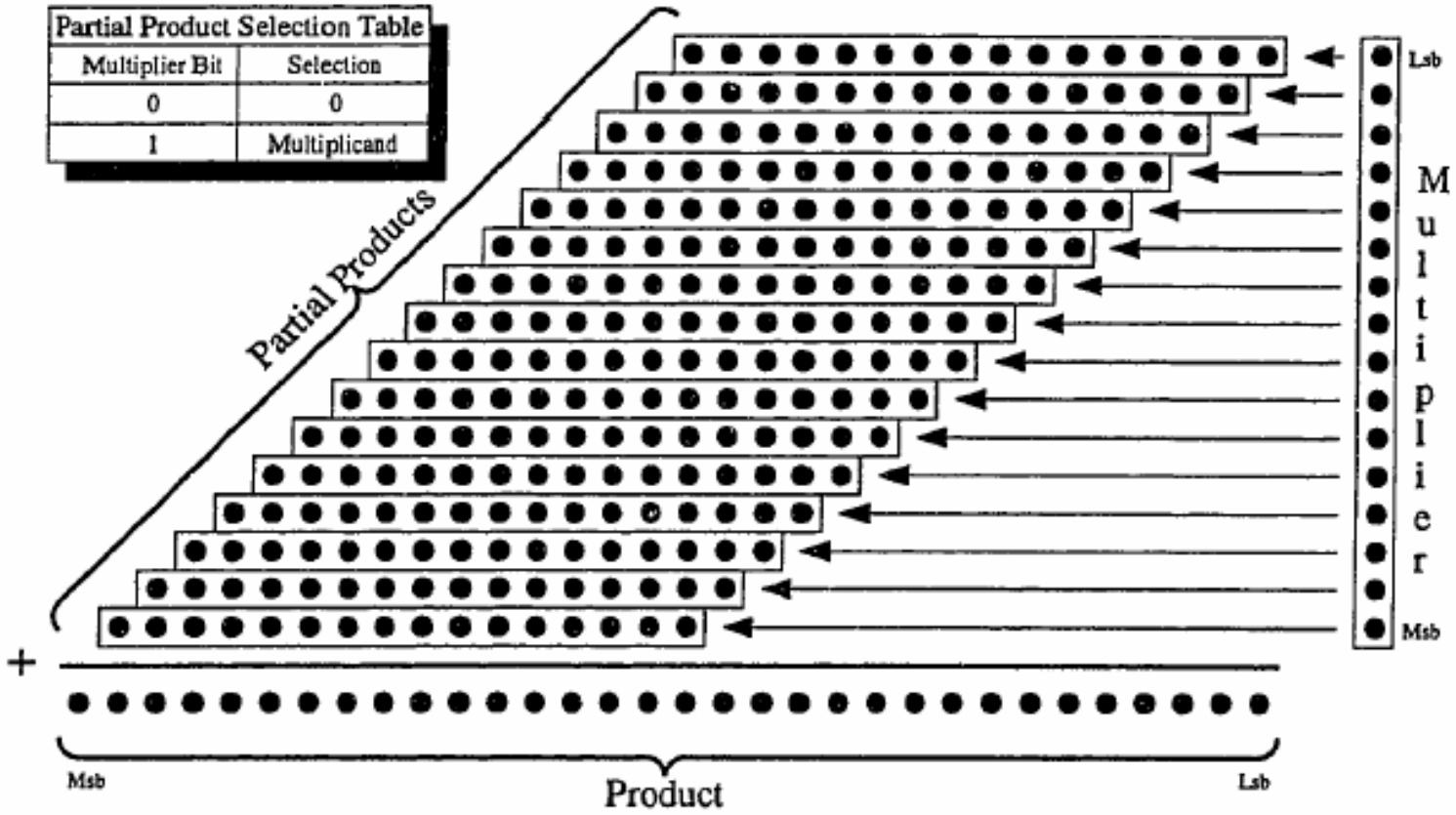
- In modern digital systems, the component responsible for handling arithmetic operations is known as the Arithmetic Logic Unit (ALU)
- These lie in the critical data path of the core data processing elements – CPU, DSP, and ASIC/FPGA processing and addressing ICs
- Performance of the system, in regards to numerical applications is directly related to the structure and design of the ALU which performs:
    - addition/subtraction
    - multiplication
    - shift/extension
    - exponentiation
    - many others

Background

- The most critical function carried out by ALU is multiplication
- Digital multiplication is not the most fundamentally complex operation, but is the most extensively used operation (especially in signal processing)
- Innumerable schemes have been proposed for realization of the operation

Basics of Digital Multiplication

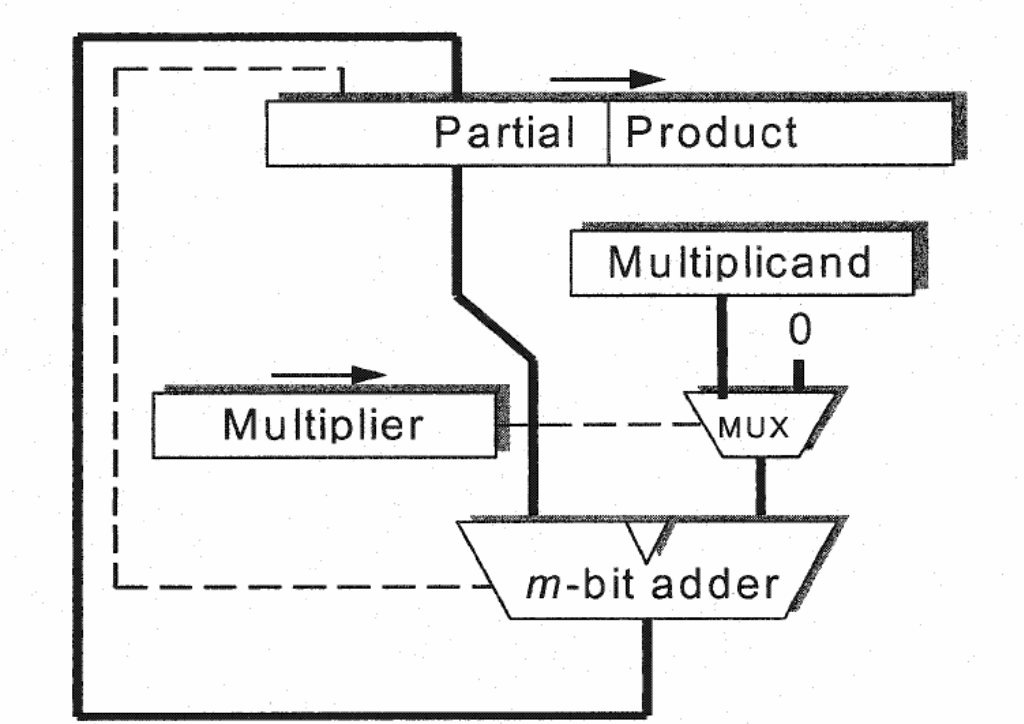
- Digital multiplication entails a sequence of additions carried out on partial products
- The method by which this partial product array is summed to give the final product is the key distinguishing factor amongst the numerous multiplication schemes



A 16-bit Partial Product Array *[Courtesy: Gary Bewick, Stanford]*

Multiplication Schemes

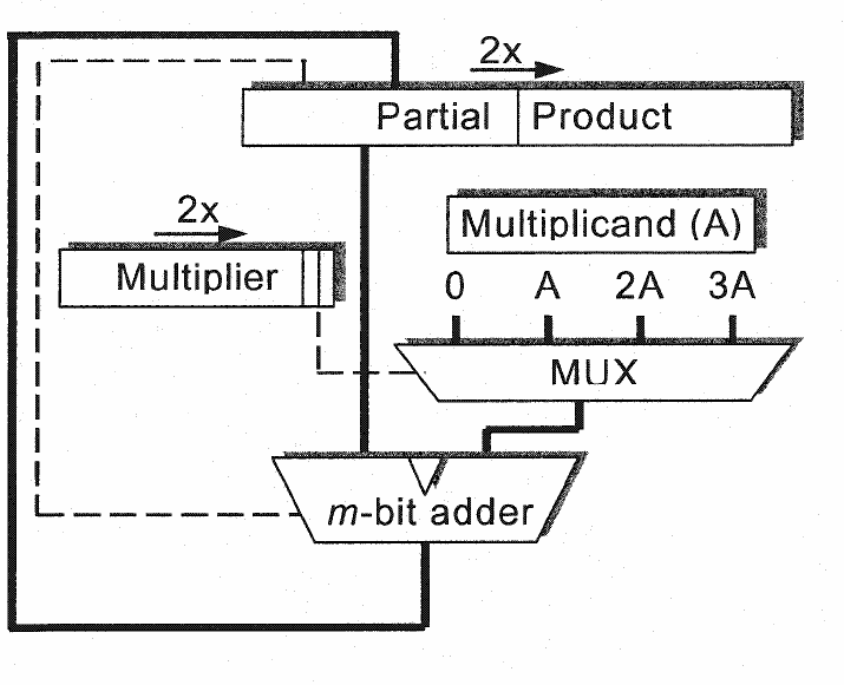
- ***Serial Multiplication (Shift-Add)***



*Standard **shift-add multiplier** implementation*
[Courtesy: Gary Bewick, Stanford]

Multiplication Schemes

- ***Serial Multiplication (High Radix Multipliers)***
  - Increased throughput by using more bits of the multiplier
  - Separate register required storing multiples of the multiplicand

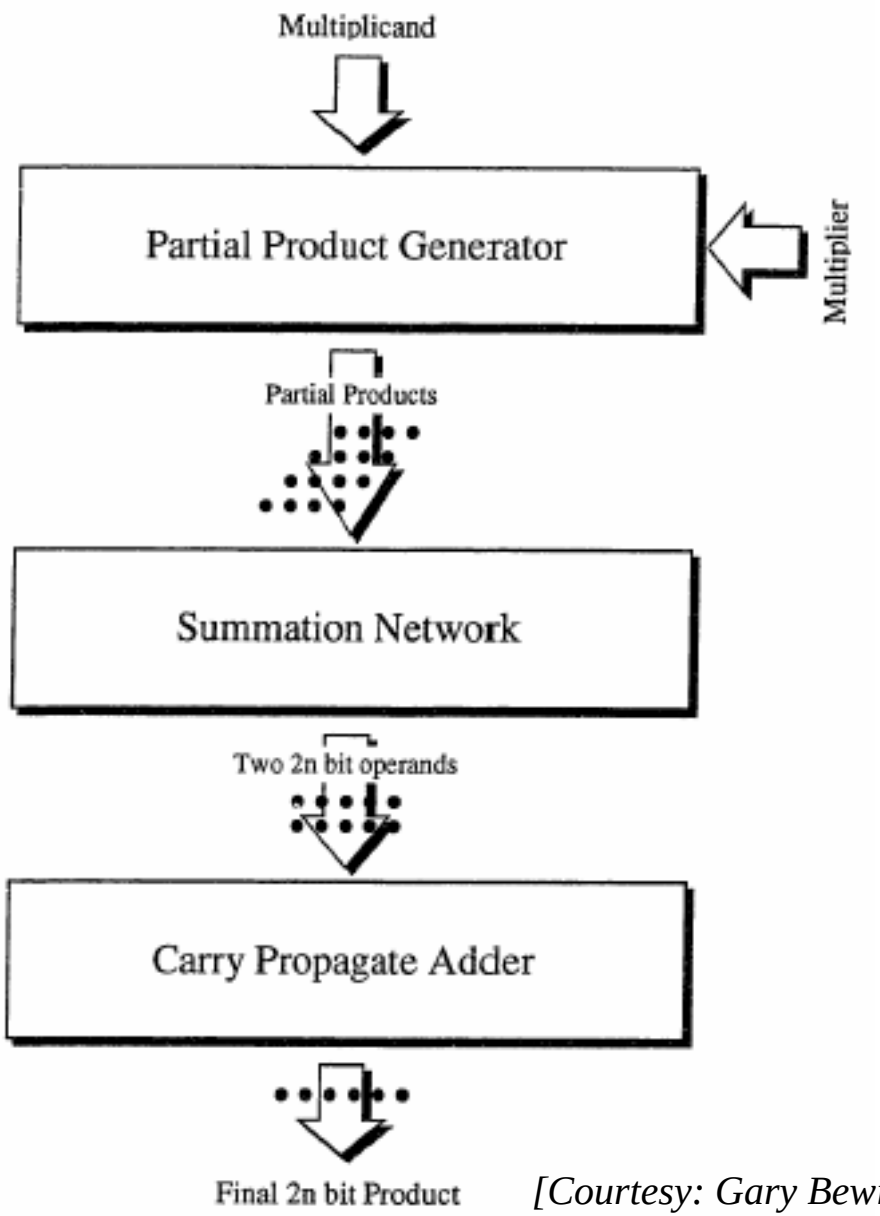


Shift-Add implementation of a basic ***radix-4 multiplier***

Multiplication Schemes

- ***Parallel Multipliers***
  - Serial multiplication implementations are primitive with simple architectures (used when there is a lack of a dedicated hardware multiplier)
  - Parallel implementations are used for high performance machines, where computation latency needs to be minimized
  - Two branches of parallel multipliers:
    - Column compression
    - Linear parallel
  - Partial products are generated simultaneously

Multiplication Schemes



[Courtesy: Gary Bewick, Stanford]

- ***Tree Multiplier***
  - Offers potential for multiplication in time *O(log n)*
  - Once partial product array is formed, bits are passed to reduction network
  - Here column-wise compression of the bits takes place, yielding two final partial products
  - Final product is obtained by addition of these two partial products
  - Considered to be irregular in form and does not permit efficient VLSI realization

Multiplication Schemes

- ***Array Multiplier***
  - Partial products are independently computed in parallel
  - Consider two binary numbers *A* and *B*, of *m* and *n* bits, respectively:

$$A = \sum_{i=0}^{m-1} A_i 2^i \qquad B = \sum_{j=0}^{n-1} B_j 2^j$$

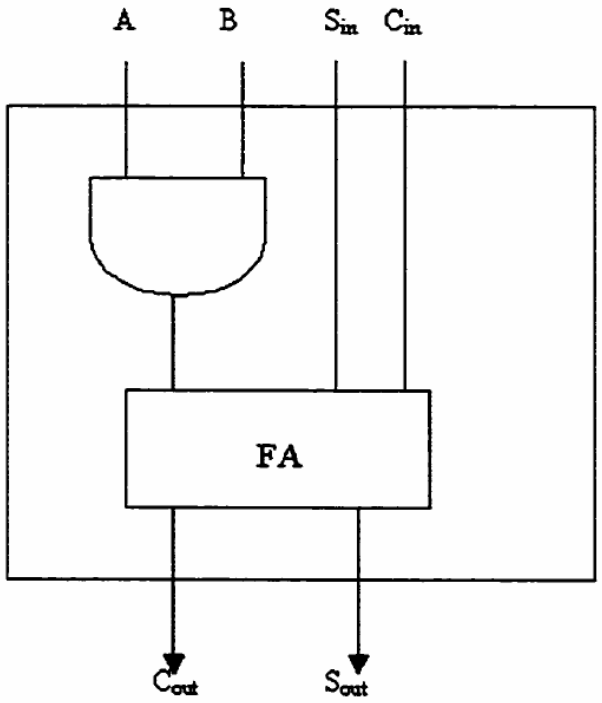
$$P = A \bullet B = \sum_{i=0}^{m-1} A_i 2^i \bullet \sum_{j=0}^{n-1} B_j 2^j \qquad P = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} A_i B_j 2^{i+j}$$

$$P = \sum_{k=0}^{mn-1} P_k 2^k$$

- P_k is known as the partial product term, also called the summand

Multiplication Schemes

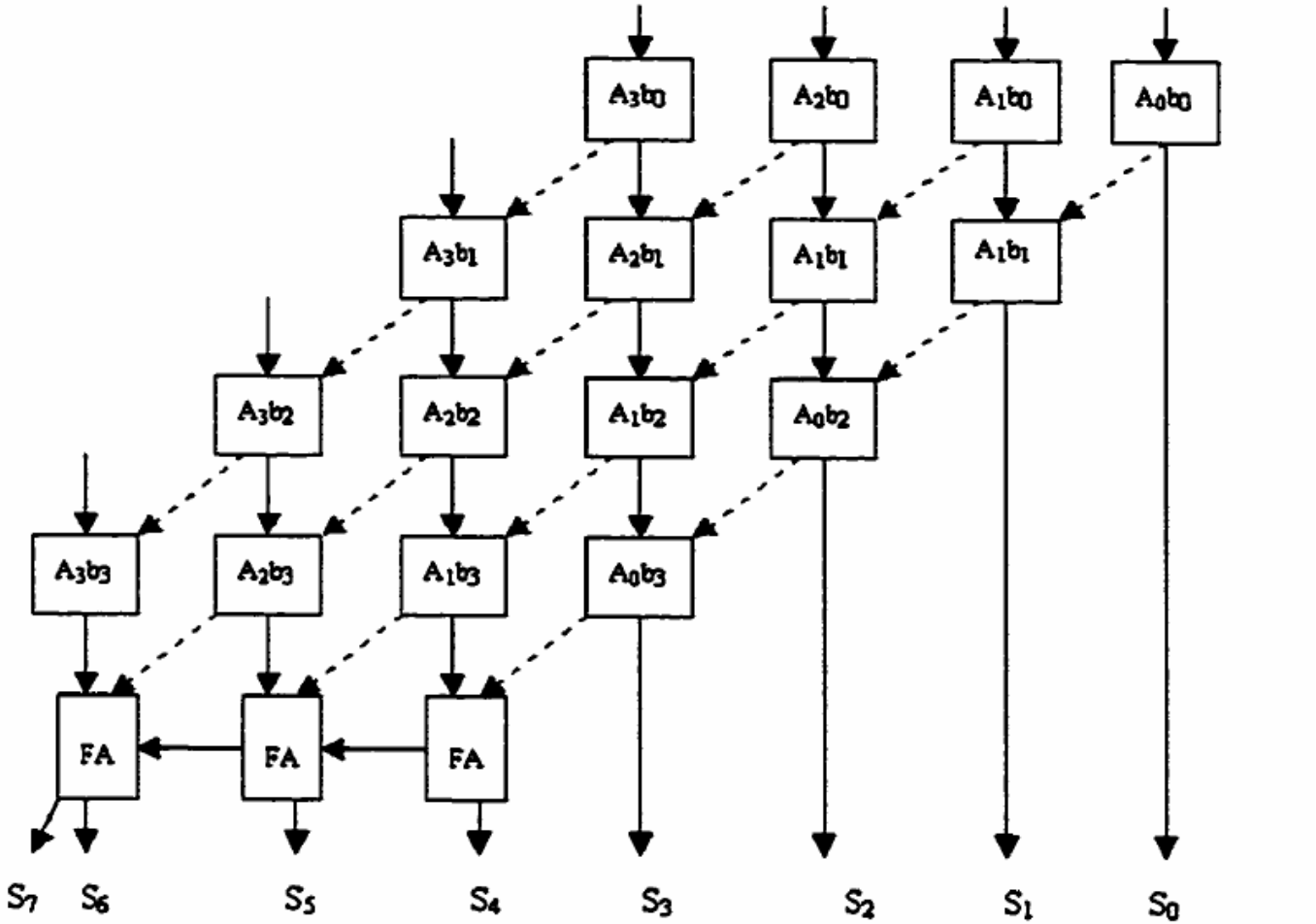
- ***Array Multiplier***
  - There are *mn* summands that are produced in parallel by a set of *mn* AND gates
  - *n x n* multiplier requires *n(n-2)* full adders, *n* half-adders and n^2 AND gates
  - Worst case delay would be $(2n+1)t_d$



- Basic cell of a parallel array multiplier

Multiplication Schemes

- ***Array structure of parallel multiplier***



[Courtesy: Karteek Chada, Texas A&M]

Multiplexer-Based Array Multiplier

- New multiplication algorithm based on a different mechanism has been proposed by Dr. Pekmestzi (National Technical University of Athens)
- Algorithm is symmetric because at each step one bit of the multiplier and one bit of the multiplicand are processed

- Mathematically, this method can be described:

Consider two positive integer numbers, X and Y:

$$X = x_{n-1}x_{n-2}\cdots x_0 = \sum_{j=0}^{n-1} x_j 2^j$$

$$Y = y_{n-1}y_{n-2}\cdots y_0 = \sum_{j=0}^{n-1} y_j 2^j$$

Also, define:

$$X_{n-1} = x_{n-2}x_{n-3}\cdots x_0 = \sum_{j=0}^{n-2} x_j 2^j \text{ and } X = X_{n-1} + 2^{n-1}x_{n-1}$$

$$Y_{n-1} = y_{n-2}y_{n-3}\cdots y_0 = \sum_{j=0}^{n-2} y_j 2^j \text{ and } Y - Y_{n-1} + 2^{n-1}y_{n-1}$$

Mathematical Description of Multiplexer-Based Array Multiplication

Product of the numbers X and Y is therefore:

$$P = XY = \left\{2^{n-1}x_{n-1} + X_{n-1}\right\}\left\{2^{n-1}y_{n-1} + Y_{n-1}\right\}$$
$$= 2^{2n-2}x_{n-1}y_{n-1} + 2^{n-1}\left\{x_{n-1}Y_{n-1} + y_{n-1}X_{n-1}\right\} + X_{n-1}Y_{n-1}$$

Now define: $P_{n-1} = X_{n-1}Y_{n-1}$ and generally, $P_j = X_jY_j$

Where X_j and Y_j are the numbers formed by the j least significant bits of X and Y respectively. The product P_j can then be expressed using the following recursive equation:

$$P_j = X_jY_j$$
$$= 2^{2j-2}x_{j-1}y_{j-1} + 2^{j-1}\left\{x_{j-1}Y_{j-1} + y_{j-1}X_{j-1}\right\} + X_{j-1}Y_{j-1}$$
$$= 2^{2j-2}x_{j-1}y_{j-1} + 2^{j-1}\left\{x_{j-1}Y_{j-1} + y_{j-1}X_{j-1}\right\} + P_{j-1}.$$

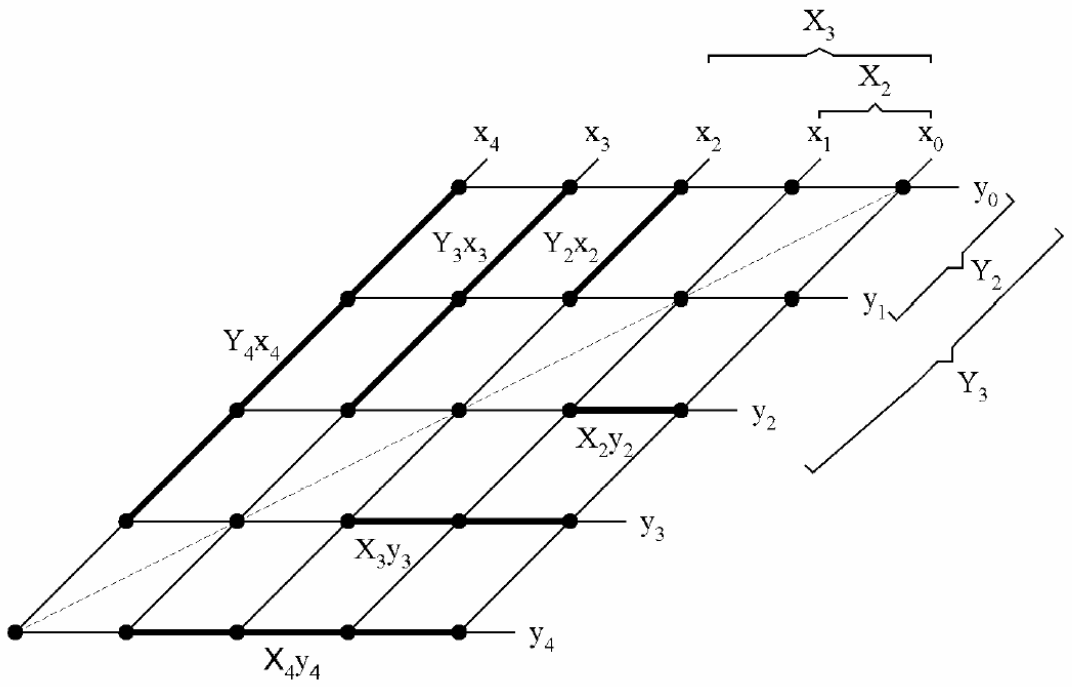
Mathematical Description of Multiplexer-Based Array Multiplication

P = XY can now be expressed with the following summations:

$$P = \sum_{j=0}^{n-1} x_j y_j 2^{2j} + \sum_{j=1}^{n-1} \{x_j Y_j + X_j y_j\} 2^j$$

Where, $Z_j = x_j Y_j + X_j y_j$ and therefore: $P = \sum_{j=0}^{n-1} x_j y_j 2^{2j} + \sum_{j=1}^{n-1} Z_j 2^j$

- Above equation can be illustrated in this schematic
- Grouping of partial products are distinguished by connecting them with solid lines
- Folding the array along the line of symmetry gives the final form of the algorithm



Mathematical Description of Multiplexer-Based Array Multiplication

- Looking at the truth table for Z_j, it can be seen that computation is only required when x_j and y_j are both equal to 1

Truth Table for Z_j

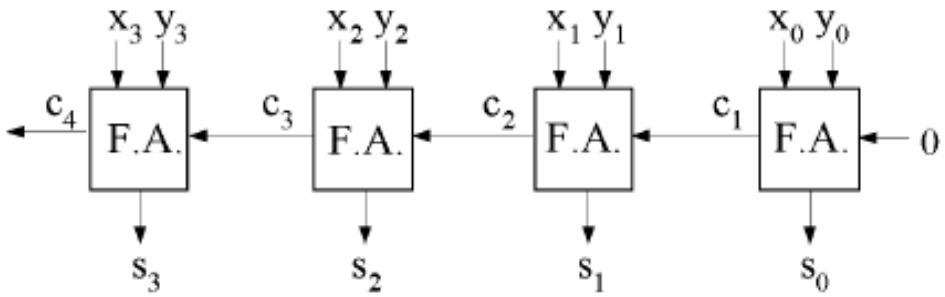
x_j	y_j	Z_j
0	0	0
0	1	X_j
1	0	Y_j
1	1	$X_j + Y_j = S_j$

- At each step, j, only s_j and c_{j+1} are new. The rest of the bits of S_j have been formed in the previous j-1 steps according to the relation:

$$S_j = S_{j-1} + x_{j-1} + y_{j-1}$$

Mathematical Description of Multiplexer-Based Array Multiplication

- S_j can be generated using a carry-propagate adder consisting of full-adders (FA):

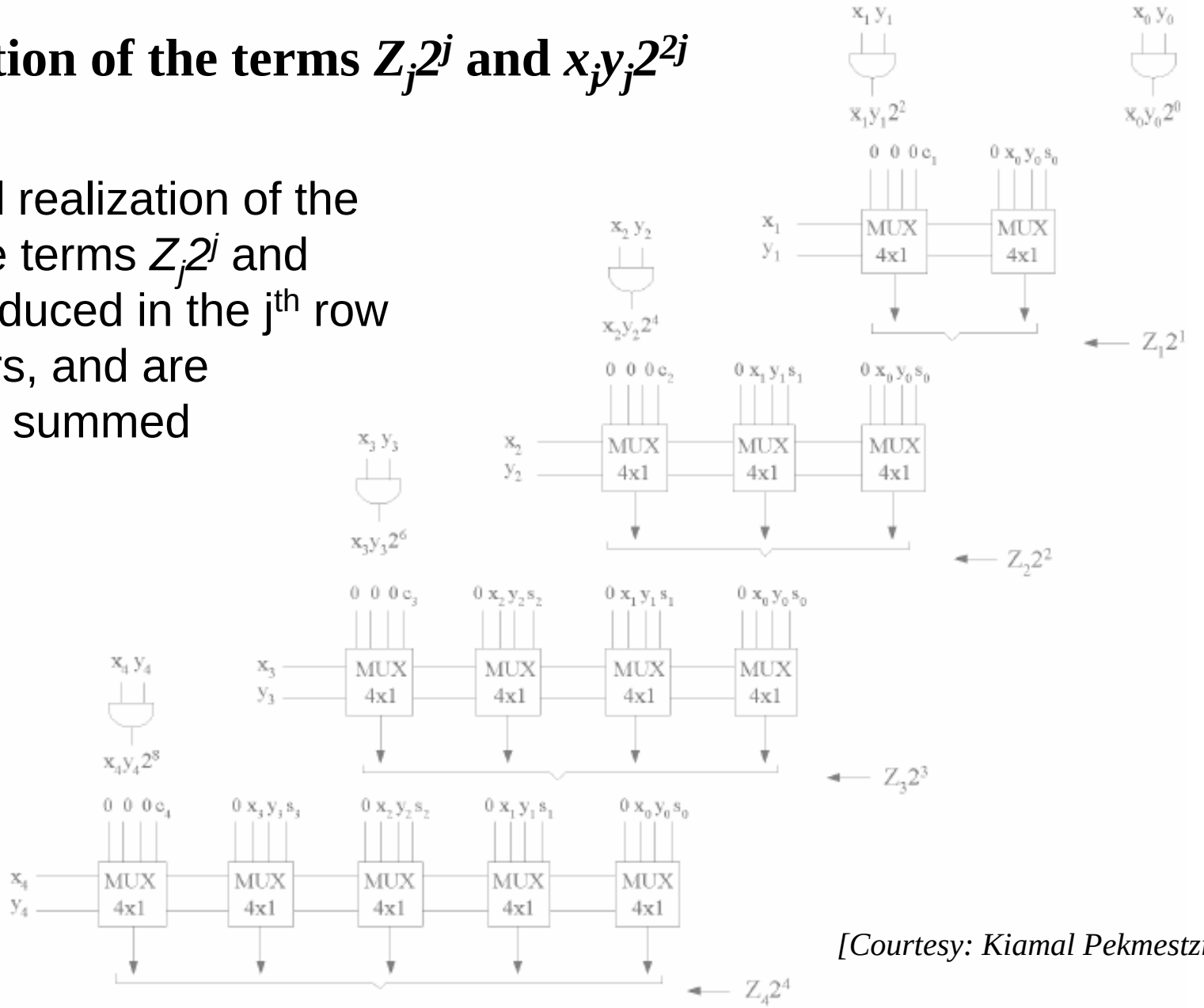


- S_j is found to be: $S_j = c_j s_{j-1} s_{j-2} \cdots s_0$
- Based on the truth table shown before, each term of the partial products Z_j can be selected accordingly
- This operation can be realized using a 4x1 multiplexer with bits x_j and y_j being the selection bits

Parallel Realization of Multiplexer-Based Multiplier

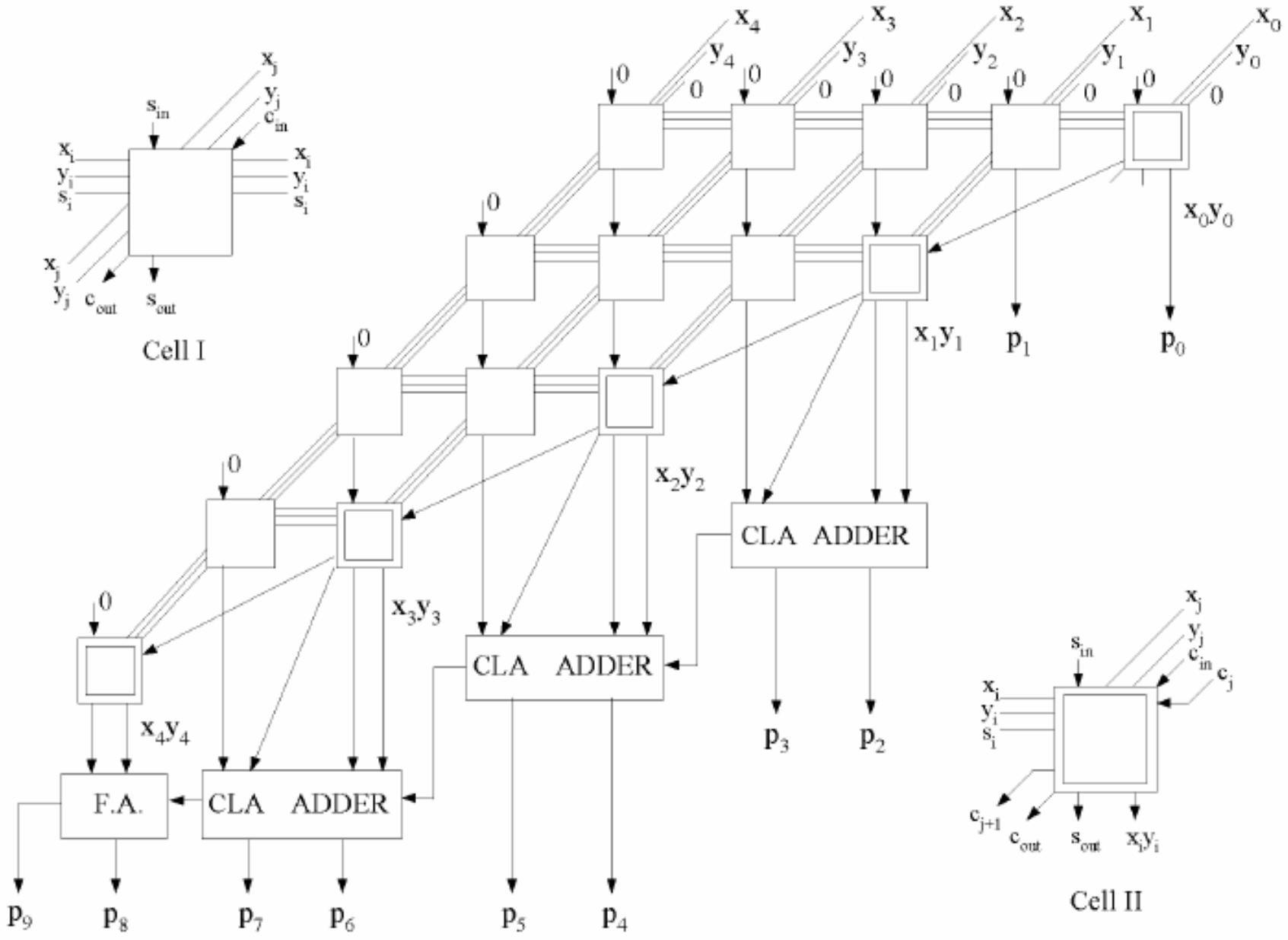
- **Implementation of the terms $Z_j 2^j$ and $x_j y_j 2^{2j}$**

- In the parallel realization of the algorithm, the terms $Z_j 2^j$ and $x_j y_j 2^{2j}$ are produced in the j[th] row of multiplexers, and are subsequently summed



[Courtesy: Kiamal Pekmestzi]

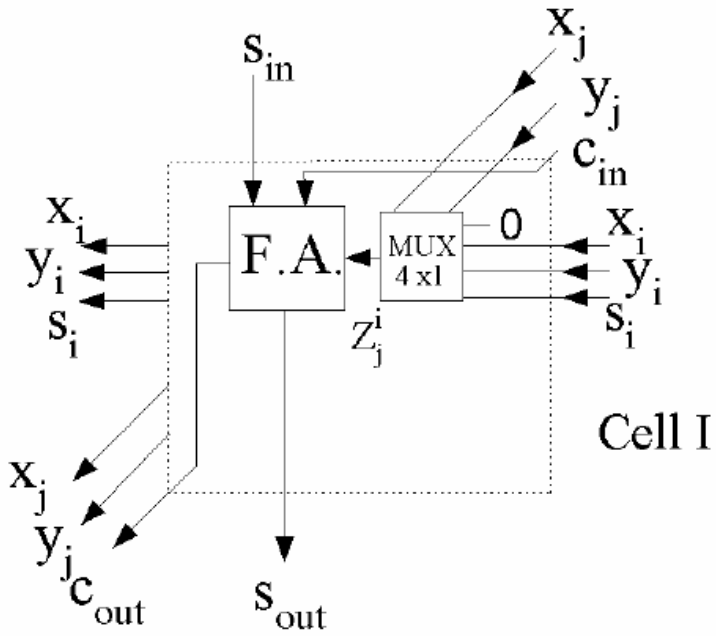
Parallel Realization of Multiplexer-Based Multiplier



[Courtesy: Kiamal Pekmestzi]

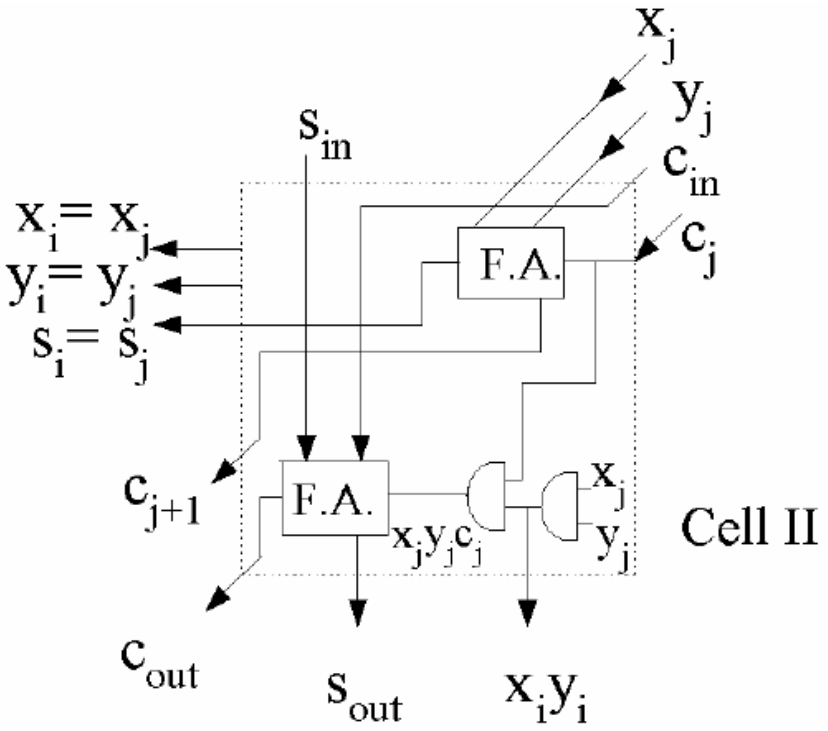
Parallel Realization of Multiplexer-Based Multiplier

- **Detail of Cell Types**
  - Cell I consists of:
    - 4 x 1 multiplexer
    - Full-Adder
    - Input bits x_j and y_j are broadcast to n-1-i cells of first type a the i^{th} horizontal row of the array
    - As well, they are broadcast to j+1 diagonally placed cells of the array



Parallel Realization of Multiplexer-Based Multiplier

- **Detail of Cell Types**
  - Cell II consists of:
    - Full-Adder which produces new bits $s_j=s_i$ and c_{j+1}
    - Bits s_i along with x_j and y_j are broadcast to all first type cells of the i^{th} row of the array
    - Bits c_{j+1} are propagated to the next second type cells
    - Bottom circuit completes addition of the term Z_j by summing the last bit of this term while at the same time generates x_jy_j which is required in computing the final product



Multiplexer-Based Array Multiplier

- Small complexity 2-bit carry-lookahead adders are used in the addition of the outputs of the right boundary cells
- Overall the total multiplier circuit consists of:
  - $\frac{n(n-1)}{2}+2n+1$ full-adders
  - $\frac{n(n-1)}{2}$ multiplexers
  - $2n$ AND gates
  - n-2 two-bit carry-lookahead (CLA) adders
- Multiplexer-based array multiplication scheme yields almost the same complexity with arrays based on the more common Modified Booth's algorithm while requiring a considerably smaller number of gates or transistors compared to other arrays

Circuit Comparison

- Exact gate number:
  - $8.5n^2+16.5n-22$ (Multiplexer-Based)
  - $10n^2+23n+13$ (Modified Booth)
- Transistor number:
  - $21n^2+37n-99$ (Multiplexer-Based)
  - $21n^2+52n+31$ (Modified Booth)
- Theoretical operation delay:
  - $T = (n+1)t_{FA}$ (Multiplexer-Based)
  - $T = nt_{FA}/2 + nt_{FA}$ (Modified Booth)
- Literature shows that multiplexer-based array multiplier outperforms modified Booth multiplier in both speed and power dissipation up to 26%!

Conclusions

- Multiplexer-based multiplication algorithm compares favourably to the most common array multipliers
- While having a circuit complexity similar to other array multipliers, multiplexer-based multipliers outperform them in terms of speed and power dissipation
- Implementation of multiplexer-based multipliers with faster and lower power consumption full-adder circuits could further improve performance
- The array structure of the multiplexer-based multiplier permits efficient VLSI implementation. However, the “zig-zag” shape could most likely be modified to a more rectangular form which would contribute to significant area-savings.

References

- Kiamal Z. Pekmestzi, “Multiplexer-Based Array Multipliers”, IEEE Transactions on Computers, Vol. 48, No. 1, January 1999.
- Pedram Mokrian, “A Reconfigurable Digital Multiplier Architecture” A Master’s Thesis, Department of Electrical and Computer Engineering, University of Windsor, April 2003.
- Gary W. Bewick, “Fast Multiplication: Algorithms and Implementation”, A Ph.D Dissertation, Stanford University, 1994.
- D. Radhakrishnan, “Low Voltage CMOS Full Adder Cells,” Electronics Letters, Vol. 35, pp. 1792-94, October 1999.
- Anant Yogeshwar Chitari, “VLSI Architecture for a 16-bit Multiply-Accumulator (MAC) Operating in Multiplication Time”, A Master’s Thesis, Department of Electrical Engineering, Texas A&M University – Kingsville, May 2000.
- Behrooz Parhami, “Computer Arithmetic: Algorithms and Hardware Designs”, Oxford University Press, New York, 2000.