39th Asilomar Conference on Signals, Systems and Computers

TRUNCATION SCHEMES FOR RECURSIVE MULTIPLIERS

Pedram Mokrian, Kevin Biswas,
Huapeng Wu, Majid Ahmadi

Research Centre for Integrated Microsystems
Department of Electrical and Computer Engineering
University of Windsor
Windsor, Ontario, Canada

Outline

- Motivation & Objectives
- Recursive Digital Multipliers
- Existing Truncation Methods
- Proposed Truncation Schemes for Recursive Multipliers
- Simulation Results
- Conclusions

Motivation

- Constant word size is required throughout arithmetic operations, i.e. DSP applications
- Rounding circuitry can be complex
- Various truncations schemes have been presented for multipliers
  - Significant reduction in complexities
- Most existing schemes target array and tree multipliers
- The inherent structure of the digital recursive multiplier is very suitable for truncation

Objectives

- Error Analysis:
  - Eliminating a portion of the digital recursive multiplier
  - Proposed truncation schemes
- Gate Complexity Analysis

Recursive Digital Multipliers

- Recursive, or "divide and conquer" multiplication was proposed by Karatsuba and Ofman in 1962
- The Karatsuba-Ofman Algorithm (KOA) multiplies two long integers by executing multiplications and additions on their divided parts
- Fundamental principles of KOA is utilized in the recursive algorithm [Danysh and Swartzlander]

Recursive Digital Multipliers

- Mathematically, the recursive algorithm is established around the fact that any *2n* x *2n* bit multiplication can be carried out through four *n* x *n* bit sub-multiplications

- Considering two unsigned 2*n*-bit operands:

Multiplicand: $A = A_H \times 2^n + A_L$

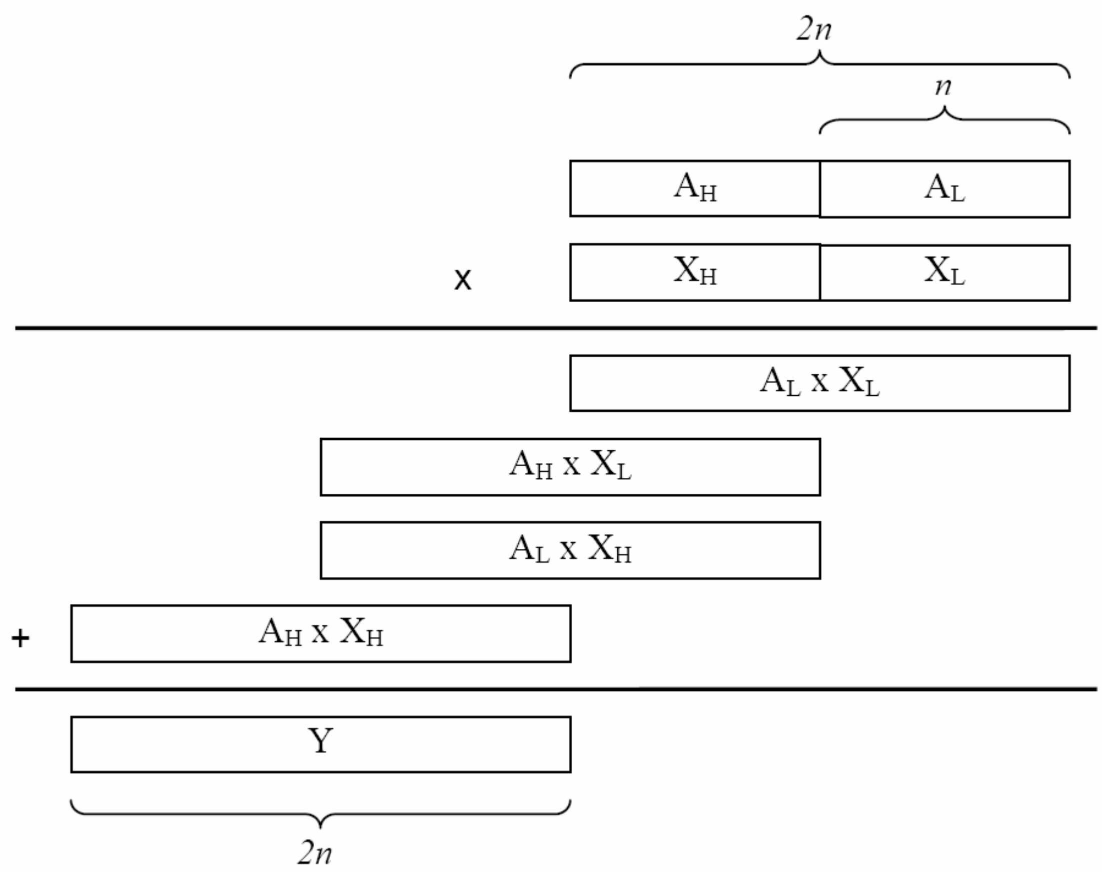
Multiplier: $X = X_H \times 2^n + X_L$

Product: $Y = A \cdot X = (A_H \times 2^n + A_L) \cdot (X_H \times 2^n + X_L)$

$$= A_H X_H \times 2^{2n} + (A_L X_H + A_H X_L) \times 2^n + A_L X_L$$

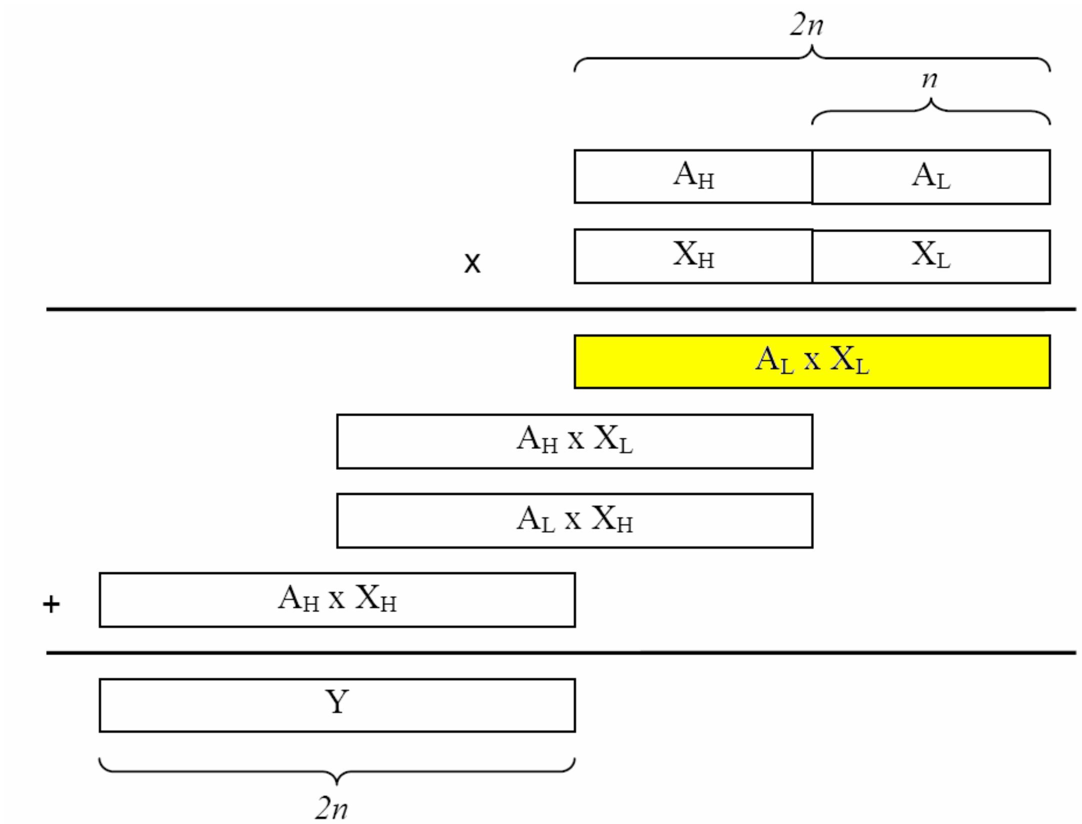
Recursive Digital Multipliers

- Block diagram representation for multiplication with a single level of recursion
- Y is the fixed-width rounded product of $2n$ bits



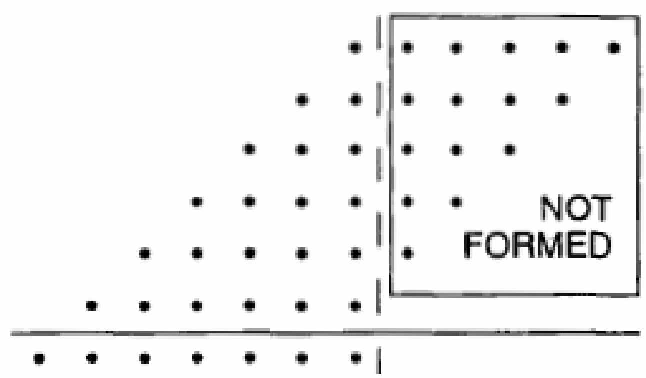
Recursive Digital Multipliers

- Of the 4 sub-multiples, the one contributing the least to the final product is $A_L \times X_L$
- Truncation schemes to be presented will target this particular component



Existing Truncation Methods

- Truncation schemes generally involve not generating the complete partial product matrix in a multiplication, and then applying a scheme to compensate for the error



- Correction schemes:
  - Constant Correction [Y. C. Lim]
  - Variable Correction [E. J. King and E. E. Swartzlander, Jr.]

Existing Truncation Methods

- Constant Correction
    - A constant is added to the remaining columns of the partial products matrix based on average value of the bits which are not formed and expected value of rounding error

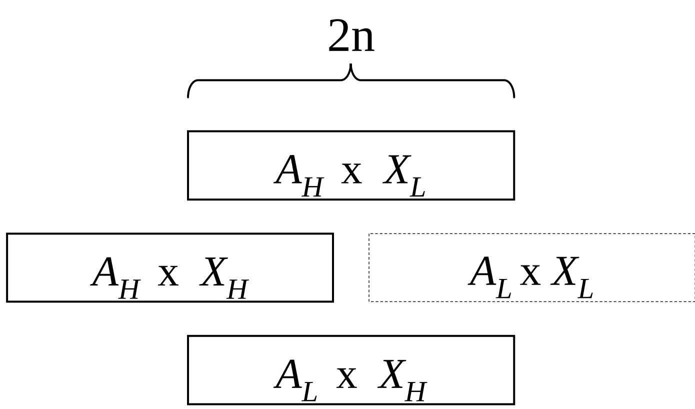
- Variable Correction
    - Correction value is data dependent:
        - If all elements are zeros in the most significant column not formed, there is no correction
        - If all elements are ones, then a maximum correction value is used

- Variable correction results in a lower variance in error
- Constant correction is efficiently implemented with tree multipliers and variable correction with array multipliers

Proposed Truncation Schemes for Digital Recursive Multipliers

- Three truncation schemes targeting recursive multipliers involving data-dependent correction terms are proposed

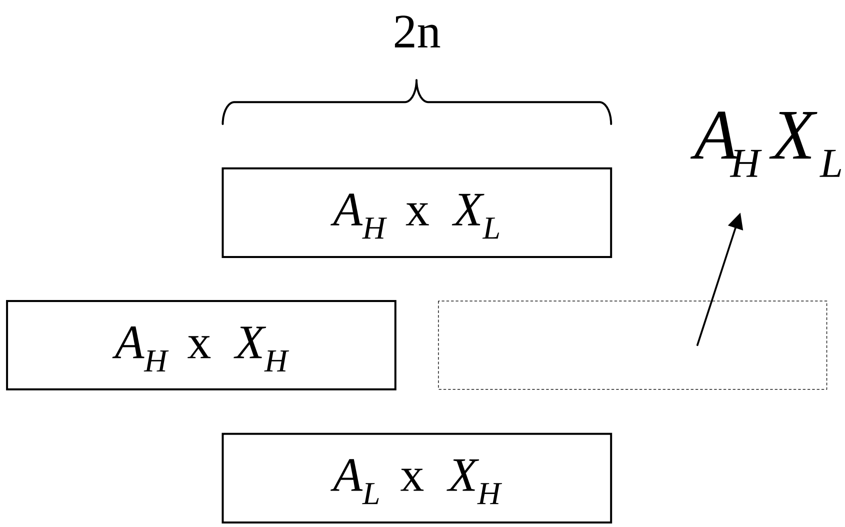
- Original recursive multiplier can be represented in this way:



- The component $A_L X_L$ will be truncated

Proposal #1 Truncation Scheme

- $A_H X_L$ or $A_L X_H$ is simply used to replace the truncated term $A_L X_L$:

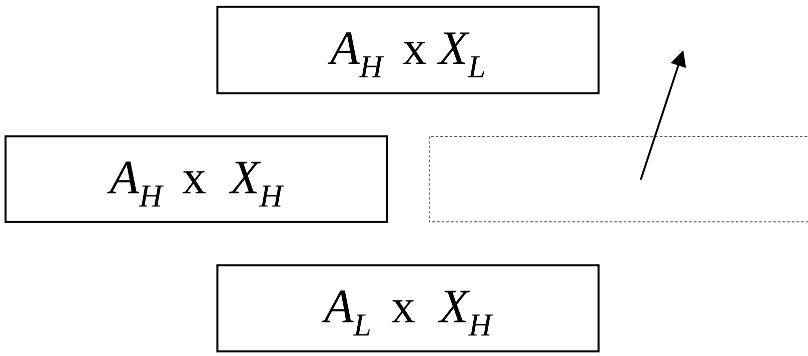


- This correction term is generated at no extra cost

Proposal #2 Truncation Scheme

- The average of the two blocks $A_H X_L$ and $A_L X_H$ replaces $A_L X_L$:

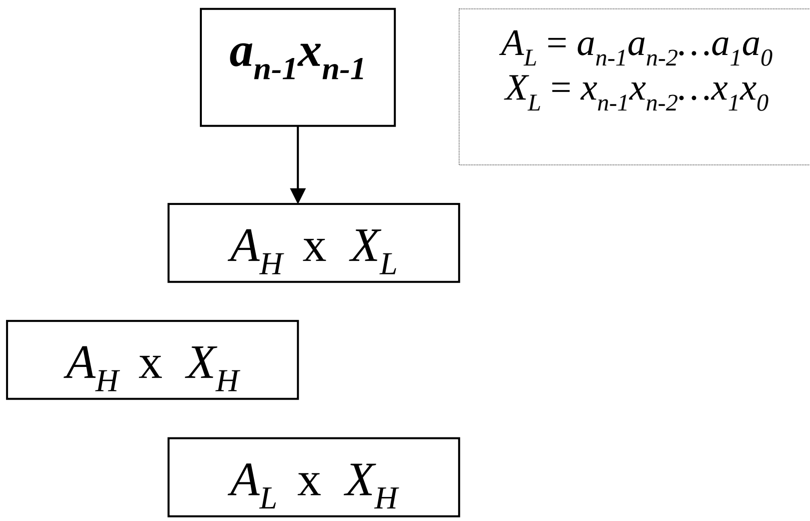
$$\frac{A_H X_L + A_L X_H}{2}$$



- High correlation between the truncated term and the replacement term

Proposal #3 Truncation Scheme

- Partial product bit $a_{n-1}x_{n-1}$ (MSB generated in $A_L X_L$) is added at the least significant bit position of block $A_H X_H$:



- A 1-bit correction term is generated with little extra cost
- Simplifies the partial products accumulation step

Simulation Results

- **Error Statistics**: Comparison with existing truncation techniques (6-bit recursive multiplier)

Multiplier Type	***Mean Error***	***Max. Positive Error***	***Max. Negative Error***	***Variance***
Truncation	-0.469	0.000	-0.984	0.086
True Rounding	0.000	0.500	-0.500	0.083
Constant Correction 1	0.4	1	-4	0.2
Constant Correction 2	-0.06	3	-2	0.2
Variable Correction	0.06	1.4	-0.9	0.1
Removal of $A_L X_L$	-0.191	0.500	-1.266	0.128
Proposal #1	-0.000	1.141	-1.156	0.128
Proposal #2	0.037	0.875	-0.906	0.109
Proposal #3	0.059	1.250	-0.828	0.173

Table 1. Error Statistics (2*n* = 6)

Simulation Results

- **Complexity Comparison**: Savings in number of gates
- It is assumed that base multiplier for the recursive structure is array multiplier
- Estimate one full adder as 12 gates, one half adder as 4 gates

2*n*	Original	Proposal #1		Proposal #2		Proposal #3	
	No. of Gates	No. of Gates	% Savings	No. of Gates	% Savings	No. of Gates	% Savings
8	812	712	12	852	-5	644	21
16	3196	2600	19	2884	10	2452	23
32	12956	10120	22	10692	18	9812	24
64	52444	40136	24	41284	21	39508	25

Table 2. Complexity Comparison

Simulation Results

- All three proposed schemes show that they have comparatively low errors
- In terms of complexity, hardware savings are close to 25% when *n* is large
- However, gate number is not always a good metric for overall performance
- For example, Proposal #2 (averaging of two components) involves an addition of 2 more rows to the partial product matrix, increasing the time required for partial products reduction step
- Further hardware reduction is expected if these correction schemes are applied to multi-level recursive multipliers

Multi-level Recursive Multiplier

- Maximum complexity savings after truncation of least significant sub-multiple:

Levels of Recursion	***Projected Maximum Complexity Savings (%)***
1	25.0
2	37.5
3	43.8

Conclusions

- Three reduced-hardware truncation schemes have been proposed, targeting the very regular composition of recursive multipliers
- Examination of hardware overhead and truncation error trade-off allows for the proper selection of a scheme
- These are the initial results of an on-going investigation on reduced-hardware truncations schemes with error correction schemes

Thank you for your attention!