A Low Power High Performance Radix-4 Approximate Squaring Circuit

Satyendra R. Datla
Southern Methodist University
Dallas, Texas
sdatla@ti.com

Mitchell A. Thornton
Southern Methodist University
Dallas, Texas
mitch@lyle.smu.edu

David W. Matula
Southern Methodist University
Dallas, Texas
matula@lyle.smu.edu

Abstract

An implementation of a radix-4 approximate squaring circuit is described employing a new operand dual recoding technique. Approximate squaring circuits have numerous applications including use in computer graphics, digital radio modules, implementation of division and function approximation in ALU circuits. The theory of operation of the circuit is described including radix-4 operand dual recoding. Our recoding yields non negative partial squares and other features which simplify the design of the approximate squaring circuit. Results of the implementation in terms of delay, power, and area in both 130nm and 90nm technologies are presented and analyzed. The results show the circuit is power, area and performance efficient, yielding reduction factors by three or more when compared to a truncated multiplication approach using state-of-the-art logic synthesis tools. The radix-4 squaring circuit is also shown to be more efficient than a radix-2 state-of-the-art binary squaring circuit.

1.0 Introduction

Approximate squaring circuits have numerous applications as mentioned in [12-15,20] such as cryptography, computation of Euclidean distance among pixels for a graphics processor or in rectangular to polar conversions in several signal processing circuits where full precision results are not required. As indicated in [14,25], customized squaring modules do have important applications in digital signal processing. Specifically, in [6], a method is described where resolution can be increased during a graphics blend operation through the incorporation of a squaring operation implemented by a multiplier followed by a truncation circuit. Clearly the approach described in this paper allows for improvement in such application. In [7], a method for frame synchronization in a digital radio is described where a digital squaring circuit is integral to the process. Hardware transcendental function designs [9] have also employed approximate squaring circuits. These are just a few examples where high-performance approximate squaring circuits are desirable.

Often, designers implement a squaring operation using a multiplier circuit. Approximate multiplication has been investigated using a truncated multiplier [8]. The multiplier may utilize a radix-4 or radix-8 Booth recoding to reduce the size of the partial product array [1,2,3]. The squaring operation yields symmetry in the partial product array when compared to a standard multiplier. This property has been investigated to provide optimizations in multiplier design at the bit level [18,19]. The design focusing on a squaring circuit employing this symmetry was proposed in [21], and numerous studies optimizing binary squaring circuits appear in [12-15,20]. These designs primarily optimize by using hardwired bit product arrangements to reduce array sizes for efficient accumulation, mostly focusing on low precision. Since squaring is a unary operation, lookup tables have also been incorporated in proposed designs of squaring circuits [23,24]. Extension to the design of a radix-4 squaring circuit employing Booth recoding and "folding" of the partial products was introduced in [16], with further implementation optimization studies discussed in [10,11,17].

Booth recoded multipliers yield partial products whose formation requires the complexities of both sign extension and two's complementation. The Booth-folded recoded squarer in [16] reduces two's complementation to a straightforward one's complementation. Avoidance of two's complementation particularly simplifies the generation of integer squares modulo the integer word size, as further investigated in [22]. In this paper, we investigate implementations of a new radix-4 operand dual recoding method [5] for the squaring operation. The recoding yields non-negative partial squares avoiding need for sign extensions and furthermore yields a radix-16 reduced array of partial products. This recoding is particularly effective for the design of an approximate squaring circuit where the partial squares can be generated with shifts and one's complements using a few guard bits.

The paper is organized as follows. First, a summary of the distinctions between existing radix-2 and our radix-4 squaring methods is provided sufficient to make the paper self contained. Then our implementation of the proposed methods is elaborated and synthesis results are given. The results show a substantial advantage for the use of a specialized approximate squaring circuit compared to a truncated multiplier, with the radix-4 squarer a significant improvement over the radix-2 squarer[4].

2.0 Squaring Methodology

As introduced in [19,21], the squaring operation for an operand in binary can be realized by a modified partial product array of about one half the size of the full multiplier array without the need for the traditional Booth multiplier recoding. Specifically, for binary squaring of the normalized p-bit operand $x = 01.b_1b_2\ldots b_{p-1}$, the p^2 bit product terms of the multiplication array may be reduced to $p(p+1)/2$ terms employing $b_i^2 = b_i$ and $b_ib_j+b_jb_i = 2b_ib_j$ for $i<j$. This provides an array of just over half the size with a depth $\lceil (p+1)/2 \rceil$. Furthermore, the depth of the array can be reduced to $\lceil (p/2) \rceil$ by incorporating the half adder relation $b_ib_{i+1} + b_i = 2b_ib_{i+1} + b_i(\sim b_{i+1})$ in the array formation.

For a state-of-the-art approximate radix-2 squarer we employ this reduced depth array design from [4] and truncate the lower order half of the array, except for a couple of columns of guard bits to tightly bound the approximate square. This optimized approximate radix-2 squarer array is of order about $1/4^{th}$ the size of a comparable full multiplier array, or equivalently about ½ the size of the truncated approximate multiplier.

Recently [5] an operand "dual recoded" radix-4 squaring method has been introduced which is particularly suited for approximate squaring. We adopt the method from [5] and perform various implementation studies. Before proceeding to performance comparison of this high-radix squarer, to make this presentation self contained we summarize the methodology and foundation for the new dual recoded radix-4 squarer.

For the squaring operation, the single operand assumes both the role of the multiplier and multiplicand. The high radix dual recoding recognizes these distinct asymmetric roles of the single operand. The dual recoding concurrently provides a "squarer" digit string in the high radix and a corresponding sequence of successively truncated "squarands" in binary form. The i^{th} squarer digit multiplies the i^{th} squarand in the i^{th} partial square generator, with the array of partial squares summed to generate the square. The following summary is taken from [5].

For the left-to-right leading digit dual recoding, the i^{th} squarand is determined only from bits of lesser or equal significance to the bits determining the i^{th} high radix squarer digit. The catalyst for characterizing the left-to-right higher radix dual recoding is the sequence of two's complement tails of the operand.

Definition: Given the p-bit normalized operand $x = 01.b_1b_2\ldots b_{p-1}$, the radix-4 two's complement tails of x are $t_0 = x = 1.b_1b_2\ldots b_{p-1}$ and $t_i = (\bar{b}_{2i-1}b_{2i}.b_{2i+1}\ldots b_{p-1})$ for $1 \le i \le \frac{p+1}{2}$, where $\bar{b}_{2i-1} = -b_{2i-1}$.

The two's complement tails are related to the Booth radix-4 representation.

Observation 1:

$x = \sum_{i=0}^{\left\lfloor \frac{p+1}{2} \right\rfloor} (t_i - t_{i+1} \bullet 4^{-1})4^{-i} = \sum_{i=0}^{\left\lfloor \frac{p+1}{2} \right\rfloor} d_i 4^{-i}$ where

$di = (t_i - t_{i+1} \bullet 4^{-1})$ is the i^{th} Booth radix-4 digit of x.

Proof: Note that:

$t_i - t_{i+1} \bullet 4^{-1} = (\bar{b}_{2i-1}\, b_{2i}.b_{2i+1}) - (0.\bar{b}_{2i+1})$ so

$t_i - t_{i+1} \bullet 4^{-1} = -2\, b_{2i+1} + b_{2i} + 2b_{2i+1} \bullet 2^{-1} = d_i$, and

$d_i \in \{-2,-1,0,1,2\}$ is recognized as the i^{th} Booth recoded radix-4 digit.

The squares of the two's complement tails are now shown to provide the foundation for our operand dual recoding.

Theorem 1: Let $q_i = t_i + t_{i+1} \bullet 4^{-1}$ for $0 \le i \le (p+1)/2$. Then

$x^2 = \sum_{i=0}^{\left\lfloor \frac{p+1}{2} \right\rfloor} d_i q_i 16^{-i}$. Furthermore, when d_i and q_i are both non zero, they have the same sign $(-1)^{b_{2i-1}}$, so then:

$x^2 = \sum_{i=0}^{\left\lfloor \frac{p+1}{2} \right\rfloor} |d_i||q_i| 16^{-i}$.

Proof: Note that $x^2 = \sum_{i=0}^{\left\lfloor \frac{p+1}{2} \right\rfloor} (t_i^{\,2} - t_{i+1}^{\,2} \bullet 16^{-1})16^{-i}$, and

$(t_i^{\,2} - t_{i+1}^{\,2} \bullet 16^{-1}) = (t_i - t_{i+1} \bullet 4^{-1})(t_i + t_{i+1}4^{-1}) = d_i q_i$

It is readily shown that: $d_i = (-1)^{b_{2i-1}} |d_i|$, and

$q_i = (-1)^{b_{2i-1}} |q_i|$, so $d_i q_i = |d_i||q_i|$ for

$0 \le i \le \left\lfloor \frac{p+1}{2} \right\rfloor$.

By performing the two's complement when dictated by $b_{2i-1}=1$ and deleting the resulting sign, we obtain:

$$|q_i| = \begin{cases} b_{2i}.b_{2i+2}b_{2i+3}\ldots b_{p-1} & \text{for} \quad b_{2i-1}=0 \\ b'_{2i}.b'_{2i+2}b'_{2i+3}\ldots b'_{p-2}b'_{p-1} & \text{for} \quad b_{2i-1}=1 \end{cases}$$

where $b'_j = (1-b_j)$ for $1 \le j \le p-2$, and $b'_{p-1} = (2-b_{p-1})$.

In summary then our dual radix-4 recoding for squaring provides the sequence $|d_0|,|d_1|,\ldots,\left|d_{\frac{p-1}{2}}\right|$ of squarer digits where $d_i = -2b_{2i-1}+b_{2i}+b_{2i+1}$ is the i^{th} radix-4 Booth recoded digit for $0 \le i \le \frac{p-1}{2}$. The dual recoding concurrently provides the sequence $|q_0|,|q_1|,\ldots,\left|q_{\frac{p-1}{2}}\right|$ of **squarands**, with $|d_i||q_i|16^{-1}$ the i^{th} **partial square** for $0 \le i \le \frac{p-1}{2}$.

The radix-4 operand dual recoding for left-to-right squaring has a number of properties of considerable practical value for applications:

- The partial squares $|d_i||q_i|$ are all non-negative, so no sign extensions are needed.
- The partial squares are each scaled down by another power of 16, so an n-term sum provides an approximate square of about $4n$ bits of accuracy.
- The partial square generators are similar in design to Booth radix-4 partial product generators but simpler in two ways – no sign extensions are needed, and, on average, they are about half the size for the same precision.

For more details on the operand dual recoding see [5]. It is illustrative to consider a sample squaring operation as shown in the tables. Consider the 16-bit squaring operation with x normalized in the interval [½,1), in particular $x=0.1100010110001011_2$ in Example 1. This example uses the Radix-2 optimizations that were discussed before[4]. As can be seen, the array contains 8 rows and 95 terms including guard bits. To visualize the optimizations achieved with the proposed Radix-4 method, the array in Example 2 below can be referred to. This utilizes a radix-4 dual recoding and employs g=3 guard bits yielding a result that has a 1½ ulp lower bound on x^2. In comparison, the array has only 4 rows and 50 terms respectively.

																Guard bits		
1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0
		0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
				0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
						0	1	1	1	0	0	0	0	0	0	0	1	0
								0	0	0	0	0	0	0	0	0	0	0
											0	1	1	0	0	0	0	0
												0	0	1	0	1	0	0
														1	1	0		
1	0	0	1	1	0	0	0	0	1	1	0	1	1	1	1	0	0	1

Example1: 16-bit Truncated Radix-2 Squarer Array

														Guard bits			Squarer digits
1	0	0	0	1	0	1	1	0	0	0	1	0	1	1	0	0	0	0	2
				1	1	0	1	0	0	1	1	1	0	1	0	0	1	1	-2
									0	0	1	1	0	0	0	1	0	1	1
													1	0	1	1	1	0	-1
1	0	0	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	0	

Example 2: 16-bit Radix-4 Approximate Squarer Array

3.0 Implementation and Results

A block diagram of the squaring circuit is shown in Figure 1.

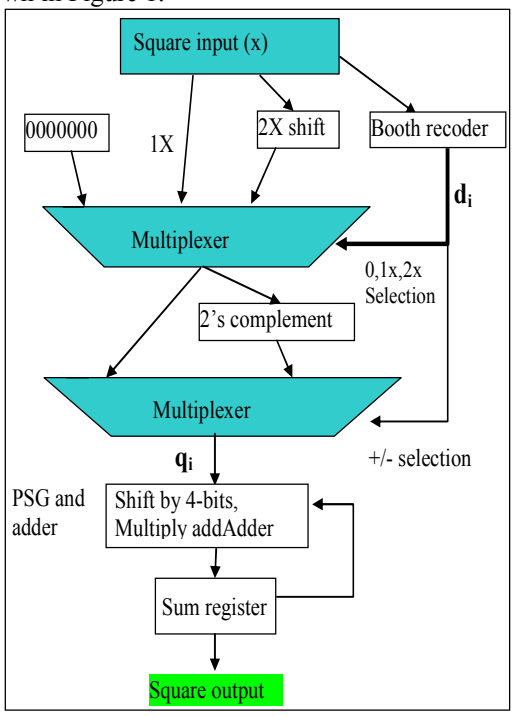

Figure 1: Block Diagram of Truncated Squaring Circuit

The Verilog™ HDL used to implement the circuits and the Synopsys Design Compiler™ is used to synthesize the circuit using both 130nm and 90nm cell libraries from Texas Instruments. Various squaring circuits were synthesized for operand sizes of n=12, 16, and 24 bits. In each of these cases additional guard bits of g=2, 2, and 3 respectively were included to ensure that the approximation is bounded by at most 2 ulps accuracy. The resulting circuits were also analyzed for maximum path delay and power dissipation. For comparison purposes, an n-bit operand truncated multiplier was also synthesized into the same cell libraries with an n-bit product and an appropriate number of guard bits as this type of circuit is commonly used for the generation of an approximate square.

Tables 1 and 2 contain the synthesis results in terms of path delay, power dissipation and area. We note that these results do not include delay due to routing however, since the multiplier adder array is more complicated than the reduced squaring adder arrays, the comparison of the truncated multiplier array to the squaring circuits is likely very conservative after including actual wire delays.

The results show that both the radix-2 and the radix-4 circuits yield a dramatic improvement in performance, power and area compared to the truncated multiplier circuit. The reductions range from a factor of two-to-three for delay, three-to-four for area and five-to-six for power.

Table 1: Synthesis Results using 130nm Cell Library

Circuit	Delay(ps)	Freq(Hz)	Dynamic Power (mW)	Leakage Power (nW)	Area (Gate count)
12-bit Truncated Multiplier	2160	4.63E+08	15.54	48.01	4001
16-bit Truncated Multiplier	2950	3.39E+08	41.18	105.65	8669
24-bit Truncated Multiplier	4690	2.13E+08	133.06	258.91	22140
12-bit Radix-2 Squarer	1030	9.71E+08	4.43	18.54	1659
16-bit Radix-2 Squarer	1180	8.47E+08	8.69	34.1	3480
24-bit Radix-2 Squarer	1740	5.75E+08	23.72	76.53	7140
12-bit Radix-4 Squarer	**860**	**1.16E+09**	**3.33**	**14.57**	**1310**
16-bit Radix-4 Squarer	**1030**	**9.71E+08**	**7.71**	**30.65**	**2826**
24-bit Radix-4 Squarer	**1620**	**6.17E+08**	**21.23**	**61.67**	**5790**

The radix-4 squarer performs better than the radix-2 squarer by about 10-to-20% in all metrics with the greatest reduction being in area. Figure 2 illustrates the comparisons of the radix-2 and radix-4 squarers in more detailed showing greater improvements as the size of the operand increases.

To summarize, the savings obtained with the approximate squaring circuit when compared with the equivalent sized radix-2 multiplier, Figure 2 contains comparison charts that illustrate the delay, power and area results of the 12-, 16-, and 24-bit operand approximate squaring circuits. It is clearly seen that the proposed circuit has significant savings and the efficiency of the proposed circuit gets better with increasing operand size.

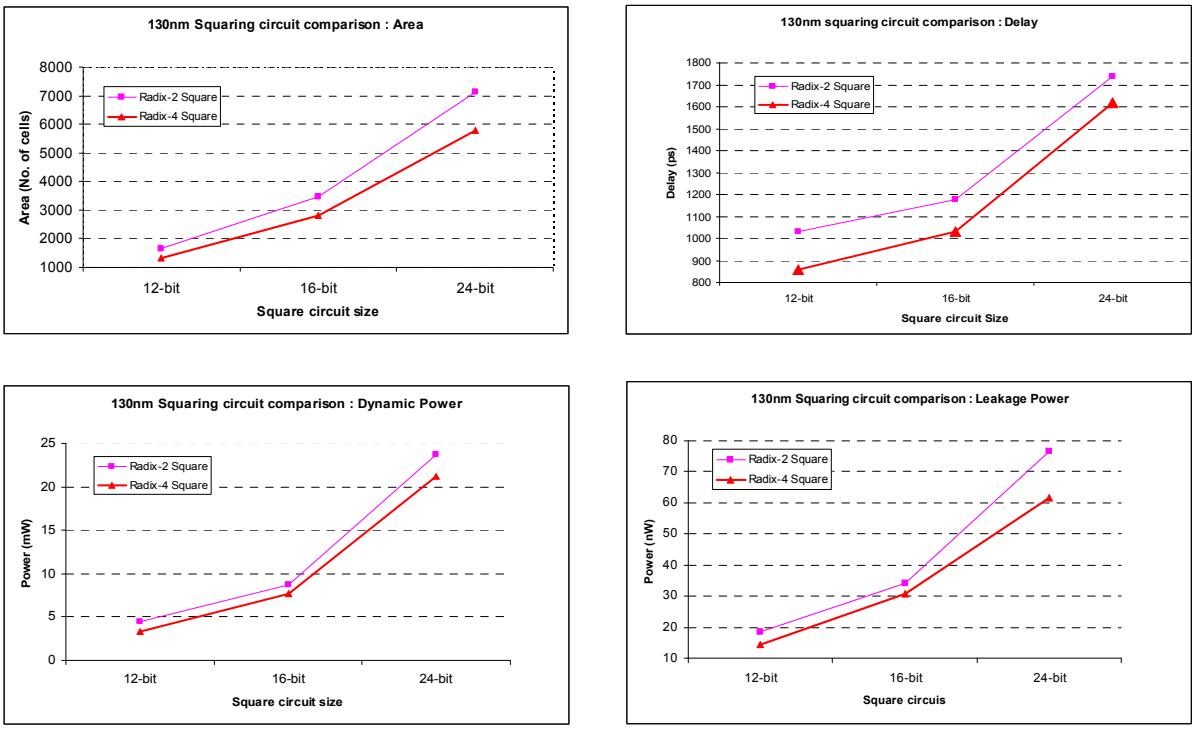

Figure 2. Metrics Comparison of Radix-4 & Radix-2 squaring circuits

Table 2: Synthesis Results using 90nm Cell Library

Circuit	Delay(ps)	Freq(Hz)	Dynamic Power (mW)	Leakage Power (nW)	Area (Gate count)
12-bit Truncated Multiplier	1700	5.88E+08	7.09	98.89	4119
16-bit Truncated Multiplier	2390	4.18E+08	17	188.54	7680
24-bit Truncated Multiplier	3380	2.96E+08	58.81	523.62	21112
12-bit Radix-2 Squarer	770	1.30E+09	2.16	46.3	1802
16-bit Radix-2 Squarer	930	1.08E+09	4	73.3	3262
24-bit Radix-2 Squarer	1390	7.19E+08	10.8	160.67	6783
12-bit Radix-4 Squarer	**650**	**1.54E+09**	**1.82**	**38.40**	**1522**
16-bit Radix-4 Squarer	**800**	**1.25E+09**	**3.03**	**59.61**	**2470**
24-bit Radix-4 Squarer	**1250**	**8.00E+08**	**9.97**	**145.47**	**5253**

4.0 Conclusion

A new approach for recoding radix-4 approximate squaring is investigated. The recoding avoids sign-extensions and generates a radix-16 reduced array of partial squares resulting in a low power and high performance approximate squaring circuit.

When compared to the truncated multiplier and the approach described in [4], this squaring circuit shows improvements in power, area, and delay when synthesized to standard cell libraries.

In the future, we plan to design other arithmetic circuits using the squaring circuit as a basis such as general multiplication and division circuits.

REFERENCES

[1] B. Parhami, **Computer Arithmetic Algorithms and Hardware Designs**, Oxford University Press, 2000, pp. 383-384.

[2] D. Knuth, **The Art of Computer Programming: Seminumerical Algorithms**, Addison Wesley, Vol. 2, 2nd Edition, 1981, pp: 441-466.

[3] M. Ercegovac and T. Lang, **Digital Arithmetic**. Morgan Kaufmann Publishers, 2004

[4] A. Pirson, J.-M. Bard, and M. Daoudi, "Squaring Circuit for Binary Numbers", *United States Patent*, No. 5,629,885, May 13, 1997.

[5] D. W. Matula, "Higher Radix Squaring Operations Employing Operand Dual Recoding", *Proc. IEEE 19th Symp. Computer Arithmetic*, June 2009.

[6] S.L. Augustine, D.C. Buhler, and B.G. Prouty, "Computer Graphics System with Improved Blending", *United States Patent*, No. 5,896,136, April 20, 1999.

[7] J. Junell and K. Mikko, "Frame Synchronization in a Device Receiving Digital Radio Transmissions, *European Patent Application"*, 95108198.3, May 30, 1995.

[8] K. E. Wires, M. J. Schulte, and J. E. Stine, "Combined IEEE Compliant and Truncated Floating Point Multipliers for Reduced Power Dissipation," in Proceedings of the *IEEE International Conference on Computer Design*, Austin, TX, pp. 497-500, September, 2001.

[9] P. M. Farmwald, "High Bandwidth Evaluation of Elementary Functions," *Proc. IEEE 5th Symp. Computer Arithmetic*, pp. 139-142. 1981.

[10] M. Ercegovac, "Left-to-Right Squarer with Overlapped LS and MS parts", *Conference Record of the 37th Asilomar Conference on Signals, Systems and Computers*, Volume 2, pp.1451-1455. November 2003

[11] A.G.H. Strollo and D. De Caro, "Booth Folding Encoding for High Performance Squarer Circuits" *IEEE Trans. Circuits and Systems-I1 Analog and Digital Signal Processing,* 50(5):250-254, 2003.

[12] Y.Yu Fengqi and A. N.Willson,"Multirate digital squarer architectures," in *Proc. 8th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS 2001)*, Malta, Sept. 2–5, 2001, pp. 177–180.

[13] R. K. Kolagotla and W. R. Griescbach, "VLSI implementation of a 350 MHz 0.35 m 8 bit merged squarer," *Electron. Lett.*, vol. 34, no. 1, pp. 47–48, Jan. 1998.

[14] J. Pihl and E. J. Aas, "A multiplier and squarer generator for high performance DSP applications," in *Proc. IEEE 39th Midwest Symp. on Circuits and Systems*, 1996, pp. 109–112.

[15] J.-T. Jae-tack Yoo, K. F. Kent F. Smith, and G. Ganesh Gopalakrishnan, "A fast parallel squarer based on divide-and-conquer," *IEEE J. Solid-State Circuits*, vol. 32, pp. 909–912, June 1997.

[16] D. De Caro and A. G. M. Strollo, "Parallel squarer using Booth-folding technique," *Electron. Lett.*, vol. 37, no. 6, pp. 346–347, Mar. 2001.

[17] A. G. M. Strollo, E. Napoli, and D. De Caro, "New design of squarer circuits using Booth encoding and Folding techniques," in *Proc. 8th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS 2001)*, Malta, Sept. 2–5, 2001, pp. 193–196.

[18] H.Ling,"High-speed computer multi-plication using a multiple-bit decoding algorithm," *IEEE Trans. Computers*, C-19, Aug. 1970, pp. 706-709.

[19] T.C. Chen, "A Binary Multiplication Based on Squaring." *IEEE Trans. Computers*, C-20:678-80, 1971.

[20] K.E. Wires, M.J. Schulte, L.P. Marquette: and P.I. Balzola. "Combined Unsigned and Two's Complement Squarers", *Conference Record of the 31st Asilomar Conference on Signals, Systems and Computers*, Volume 2, pp. 1215-1219, 1999.

[21] L. Dadda, "Squarers for binary numbers in serial form", *Proc. IEEE 7th Symp. Computer Arithmetic*, 1985.

[22] J. Moore, D.W. Matula, M.L. Thornton, "A Low Power Radix-4 Dual Recoded Integer Squaring Implementation For Use in Design of Application Specific Arithmetic Circuits", *Conference Record of the 42nd Asilomar Conference on Signals, Systems and Computers,* October 2008.

[23] E.G. Walters III, J.S. Schlessman, and M.J. Schulte, "Combined Unsigned and Two's Complement Hybrid Squarers", *Conference Record of the 35th Asilomar Conference on Signals, Systems & Computers*, pp. 861-866, November 2001.

[24] C.L. Wey and M.D. Shieh. "Design of a High-Speed Square Generator", *IEEE Trans. Computers*, vol. 47, no. 9, pp. 1021-1026, 1998.

[25] A. Liddicoat and M. J. Flynn, "Parallel Square and Cube Computations", *Conference Record of the 34th Asilomar Conference on Signals, Systems & Computers,* 2000.