Decimal Multiplication using compact BCD Multiplier

REKHA K. JAMES, SHAHANA T. K., K. POULOSE JACOB,
Cochin University of Science and Technology
Kochi, Kerala, India
{rekhajames, shahanatk, kpj}@cusat.ac.in

SREELA SASI
Gannon University
Erie, PA, USA
sasi001@gannon.edu

Abstract

Decimal multiplication is an integral part of financial, commercial, and internet-based computations. The basic building block of a decimal multiplier is a single digit multiplier. It accepts two Binary Coded Decimal (BCD) inputs and gives a product in the range [0, 81] represented by two BCD digits. A novel design for single digit decimal multiplication that reduces the critical path delay and area is proposed in this research. Out of the possible 256 combinations for the 8-bit input, only hundred combinations are valid BCD inputs. In the hundred valid combinations only four combinations require 4 × 4 multiplication, 64 combinations need 3 × 3 multiplication, and the remaining 32 combinations use either 3 × 4 or 4 × 3 multiplication. The proposed design makes use of this property. This design leads to more regular VLSI implementation, and does not require special registers for storing easy multiples. This is a fully parallel multiplier utilizing only combinational logic, and is extended to a Hex/Decimal multiplier that gives either a decimal output or a binary output. The accumulation of partial products generated using single digit multipliers is done by an array of multi-operand BCD adders for an (n-digit × n-digit) multiplication.

1: INTRODUCTION

Nowadays, decimal arithmetic is receiving significant attention in the financial, commercial, and internet-based applications. These applications often store data in decimal format. Currently, general purpose computers do decimal computations using binary arithmetic. But, a number of decimal numbers such as 0.2 cannot be represented precisely in binary. In this world of precision, such errors generated by conversion between decimal and binary formats are no more tolerable. Recently, support for decimal arithmetic has received increased attention due to the growing importance in financial analysis, banking, tax calculation, currency conversion, insurance, telephone billing and accounting which cannot tolerate such errors. This can be overcome by using a decimal arithmetic and logic unit (ALU). Decimal arithmetic operations are typically more complex, slower and occupy more area leading to more power and less speed when implemented in hardware. Hence, the major consideration while implementing decimal arithmetic is to enhance its speed and reduce area as much as possible. Due to the growing importance of decimal arithmetic, standard specifications are recently added to the draft revision of the IEEE 754 Standard for Floating-Point Arithmetic.

Decimal multipliers are typically implemented using an iterative approach because of their complexity. Usually, the entire multiplicand is multiplied by one multiplier digit to generate a partial product in each cycle. The partial product is added to an intermediate product register that holds the previously accumulated partial products. In an iterative decimal multiplier presented in [1], decimal partial products are generated by creating two partial products for each multiplier digit. Multiplying two n-digit Binary Coded Decimal (BCD) numbers requires n iterations, where all iterations consist of two binary carry-save additions and three decimal corrections. After n iterations, the carry and sum are added using a decimal carry-propagate adder to produce the final product. The multiplier presented in [2] generates the partial products by the retrieval of product of BCD digits from look-up tables. Several existing designs for decimal multiplication generate and store multiples of the multiplicand before partial product generation, and then use the multiplier digits to select the appropriate multiple as the partial product [3, 4]. The multiplier presented in [4] makes use of a secondary set of multiples generated using combinational logic. Iterative additions are performed in two pipeline stages, which allows for a higher frequency of operation. The latency of this multiplier is $(n + 4)$ cycles and a new multiplication can begin every $(n + 1)$ cycle. The multiplier in [5] stores intermediate product digits in a less restrictive, redundant format called the overloaded decimal representation that reduces the delay of the iterative portion of the multiplier. These approaches are based on either slow accumulation of easy multiples or costly retrieval of product of BCD digits from look-up tables. An alternative approach is to generate the partial product as needed. Generating the

978-1-4244-2315-6/08/$25.00 ©2008 IEEE.

partial products as needed is an ideal approach for three reasons as enumerated in [6]. The use of decimal digit-by-digit multipliers for partial product generation leads to less number of cycles, less wiring and do not require registers to store multiples of the multiplicand. The algorithm presented in [6] reduces the complexity of partial product generation by employing a recoding scheme to restrict the magnitude range of the operand digits. Further, by restricting the range of each digit in the partial product, the complexity of partial product accumulation is also significantly reduced. An integral building block of a decimal digit by digit multiplier is the single digit multiplier. The single digit multiplier in [7] uses a standard 4 × 4 unsigned binary multiplier that generates an 8-bit binary output which needs to be corrected to two BCD digits.

This paper presents a novel design for single digit decimal multiplication to reduce the critical path delay and area, which allows for a fast multiplier design. The accumulation of partial products generated using single digit multipliers is done by an array of multi-operand BCD adders for an (n-digit × n-digit) multiplication. This is a fully parallel multiplier utilizing only combinational logic, and can be extended for floating point multiplication of decimal digits.

The organization of the paper is as follows: Initially, a new approach for single digit multiplication is discussed. The design is then extended to a Hex/Decimal multiplier that gives either a decimal output or a binary output depending on the requirement. A decimal fixed point multiplier is then proposed using single digit decimal multipliers. Finally, the paper concludes by tabulating a comparison of area and delay analysis of the proposed design with the multiplier in [7] using logic synthesis tool Leonardo Spectrum from Mentor Graphics Corporation using ASIC Library.

2: BCD DIGIT MULTIPLICATION

A key component of a fixed-point multiplier is a single digit multiplier that multiplies an *n*-digit multiplicand, *A*, by an *n*-digit multiplier, B producing a *2n*-digit product, P. The single digit multiplier accepts two BCD inputs (A, B) which can take a value [0-9]. It realizes a function F(A, B), giving a product in the range [0, 81] represented by two BCD digits. There are one hundred possible combinations of inputs for multiplication, out of which only 4 combinations require 4 × 4 multiplication, 64 combinations need 3 × 3 multiplication, and the remaining 32 combinations use either 3 × 4 or 4 × 3 multiplication. The proposed design makes use of this property. The single digit multiplier consists of two parts: a binary multiplier that gives a binary product $p_{(7\text{-}0)}$, and a binary to BCD converter. Since the multiplier accepts only BCD inputs, the 4-bit inputs can be either $8(1000_2)$ or $9(1001_2)$. This restricts the binary product bits to $p_{(6\text{-}0)}$.

A. Binary Multiplier

The binary multiplier consists of a 3 × 3 multiplier, a 4 × 3 multiplier, a 3 × 4 multiplier, and a 4 × 4 multiplier. Figures 1, 2, 3 and 4 show the 3 × 3, 4 × 3, 3 × 4 and 4 × 4 multiplication for BCD inputs respectively. In 4 × 3 multiplication for BCD inputs, one of the inputs is either $8(1000_2)$ or $9(1001_2)$. So, the 4 × 3 multiplier or a 3 × 4 multiplier gets simplified to three 2-input AND gates. In 4 × 4 multiplication for BCD inputs, both inputs are either $8(1000_2)$ or $9(1001_2)$. So the 4 × 4 multiplier gets simplified to a half adder (a 2-input AND gate and a 2-input XOR gate) as seen in Figure 4.

			x_2	x_1	x_0	×
			y_2	y_1	y_0	
			x_2y_0	x_1y_0	x_0y_0	
		x_2y_1	x_1y_1	x_0y_1		
	x_2y_2	x_1y_2	x_0y_2			
	p_5	p_4	p_3	p_2	p_1	p_0

Figure. 1: 3 × 3 Multiplication

		1	0	0	x_0	×
			y_2	y_1	y_0	
		y_0	0	0	x_0y_0	
	y_1	0	0	x_0y_1		
y_2	0	0	x_0y_2			
y_2	y_1	y_0	x_0y_2	x_0y_1	x_0y_0	

Figure. 2: 4 × 3 Multiplication of BCD inputs

			x_2	x_1	x_0	×
		1	0	0	y_0	
			x_2y_0	x_1y_0	x_0y_0	
x_2	x_1	x_0				
x_2	x_1	x_0	x_2y_0	x_1y_0	x_0y_0	

Figure. 3: 3 × 4 Multiplication of BCD inputs

The 3 × 3, 3 × 4 and 4 × 3 multipliers give a 6-bit binary product, while a 4 × 4 multiplier produces a 7-bit binary result. The binary to BCD converter, following the binary multiplier, will be a 7-input converter. The design can be

further simplified if conversion needs to be done only for 6-bit products. The 6-bit converter converts the binary output of the 3 × 3 multiplier, 4 × 3 multiplier or 3 × 4 multiplier outputs to its corresponding BCD. Instead of using a 7-bit binary to BCD converter, the 4 × 4 multiplier is designed to produce an 8-bit BCD output as shown in Figure 5. The 4 × 4 multiplier and the binary to BCD conversion circuit of its product now gets reduced to a 2-input AND, NAND, XOR and NOR gates.

$$
\begin{array}{cccccc}
 & & & 1 & 0 & 0 \; x_0 \quad \times \\
 & & & 1 & 0 & 0 \; y_0 \\
 & & & \multicolumn{3}{c}{\text{----------}} \\
 & & & y_0 & 0 & 0 \; x_0y_0 \\
1 & 0 & 0 & x_0 & & \\
\multicolumn{6}{c}{\text{----------}} \\
1 & 0 & x_0y_0 & x_0 \oplus y_0 & 0 & 0 \; x_0y_0
\end{array}
$$

Figure. 4: 4 × 4 Multiplication of BCD inputs

$$
\begin{array}{cccccc}
 & & & 1 & 0 & 0 \; x_0 \times \\
 & & & 1 & 0 & 0 \; y_0 \\
\multicolumn{6}{c}{\text{----------}} \\
x_0y_0 & (x_0y_0)' & (x_0y_0)' & x_0 \oplus y_0 & 0 & (x_0 + y_0)' \; x_0 \oplus y_0 \; x_0y_0
\end{array}
$$

Figure. 5: 4 × 4 Multiplication of BCD inputs generating 8-bit BCD output

B. 6-bit Binary to BCD Converter

Binary product can be converted to an equivalent BCD by a six-input, eight-output combinational logic. Although the general binary-to-BCD conversion is extensively addressed in the literature [8–10], a special, simpler and faster, binary-to-BCD converter depicted in [7] for a 6-bit input is used in this proposed design. The first row in Figure 6 shows the BCD weights. The weights of p_3, p_2, p_1 and p_0 are the same as the corresponding weights in the original binary number $p_{(5\text{-}0)}$. But, weights 16 and 32 of p_4 and p_5 have been decomposed to (10, 4, 2) and (20, 10, 2) respectively. The three overloaded decimal digits in the right four columns are added, by an overloaded decimal adder to get the overloaded sum ($D_3D_2D_1D_0$) and a carry. The carry is added to the two BCD digits ("0 0 p_5 p_4" and "0 0 0 p_5") in the left four columns leading to ($D_7D_6D_5D_4$). Finally, the overloaded binary result is corrected to the valid BCD form [5]. If the product was a 7-bit number then product term will have a p_6 bit with weight 64 which is to be decomposed into (40, 20, 4). This increases the depth of BCD addition required by one more level in lower digit level and in higher digit level. So by making use of a 6-bit converter the depth of addition is reduced.

The block diagram of the proposed single digit BCD multiplier is shown in Figure 7.

80	***40***	***20***	***10***	***8***	***4***	***2***	***1***
0	0	p_5	p_4	p_3	p_2	p_1	p_0
0	0	0	p_5	0	p_4	p_4	0
0	0	0	0	0	0	p_5	0
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure. 6: The principle of 6-bit binary to overloaded BCD conversion

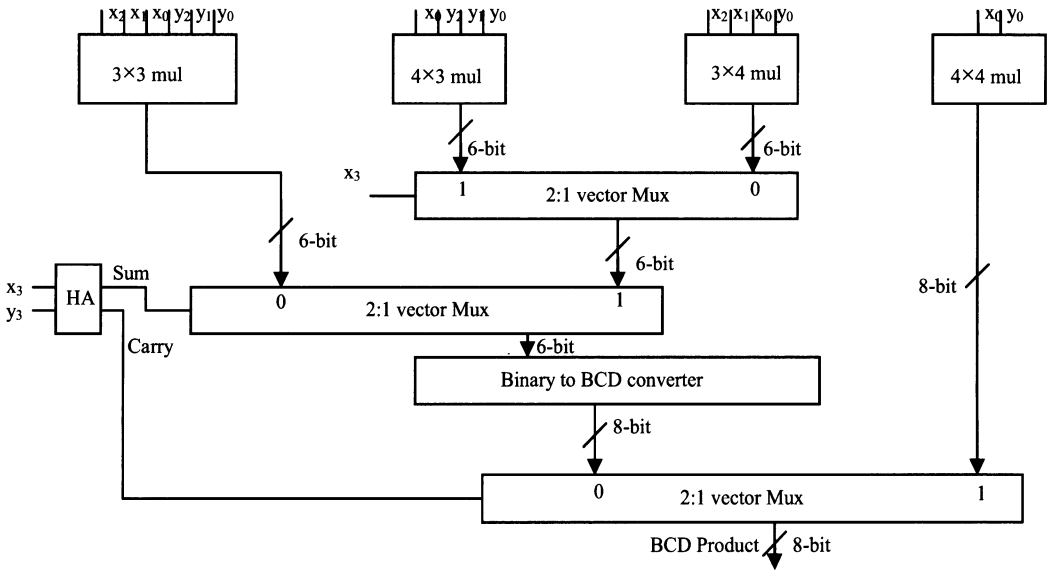

Figure. 7: Single digit BCD multiplier

The first 6-bit 2:1 multiplexer selects the 3 × 4 or 4 × 3 multiplier output depending on x_3 bit. Second 6-bit 2:1 multiplexer does the selection of 3 × 3 multiplier output or the output of the first multiplexer depending on the status of x_3 and y_3 bits. If x_3 and y_3 are different then the output of the first multiplexer is passed to the BCD converter, else the output of 3 × 3 multiplier is passed. After the 6-bit binary to BCD conversion the third multiplexer (8-bit 2:1 vector mux) selects the BCD converted output or the output of the 4 × 4 multiplier output (which gives an 8-bit BCD result) depending on x_3 and y_3 bits. If both are '1' then the 4 × 4 multiplier output is selected, else the BCD converter output is passed as the final product.

A comparison of the proposed design with the existing design in [7] in terms of area and critical path delay is done with the logic synthesis tool Leonardo Spectrum from Mentor Graphics Corporation using ASIC Library 0.18 micron, 1.8 V CMOS technology, and is tabulated in Table 1. The table shows that the proposed design has reduced area and delay compared to the existing one in [7].

Table 1. Comparison of area and delay of single digit BCD multiplier implementations

Type of Multiplier	Area (μm^2)	% reduction	Delay(ns)	% reduction
Proposed multiplier	**495**	**7%**	**7.81**	**16%**
Multiplier in [7]	532		9.26	

3: HEX / DECIMAL MULTIPLIER

The single digit decimal multiplier design is extended to a Hex/Decimal multiplier that gives either a decimal output or a binary output depending on the requirement. Digital Signal Processing (DSP) applications require binary multipliers while financial and commercial applications require decimal multipliers. A Hex multiplier accepts two 4-bit binary inputs and gives an 8-bit product. The single digit decimal multiplier has a 3 × 3 multiplier block. This can be extended to realize a 4 × 4 multiplier with some extra hardware. The terms marked in green in Figure 8 indicates the additional AND products that are required for a 4 × 4 multiplication compared to a 3 × 3 multiplication. Hence to realize a 4 × 4 multiplication only those terms which are marked in green need be added to the result product of a 3 × 3 multiplication as shown in Figure 9. The design for the single digit BCD multiplier is modified as shown in Figure 10 to make it a Hex/Decimal multiplier. The additional hardware required is seven AND gates, and an adder that adds three 4-bit numbers. The adder can be realized by five full adders and one half adder. The AND array works in parallel with the 3 × 3 multiplier and the adder works in parallel with the rest of the BCD multiplier circuit. Hence no additional delay is added up due to the additional hardware. Finally a 2:1 vector multiplexer selects one of the products (Hex/Decimal) depending on a control input.

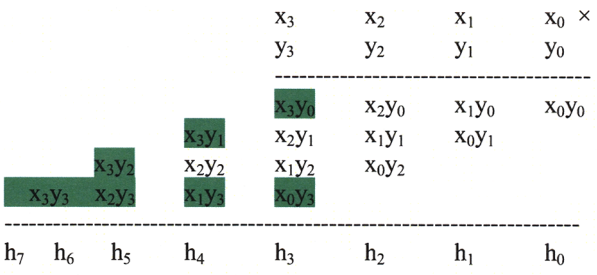

Figure. 8: 4 × 4 Multiplication

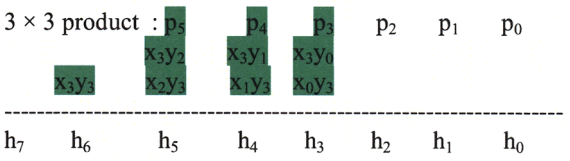

Figure. 9: Adder inputs for the 4 × 4 product

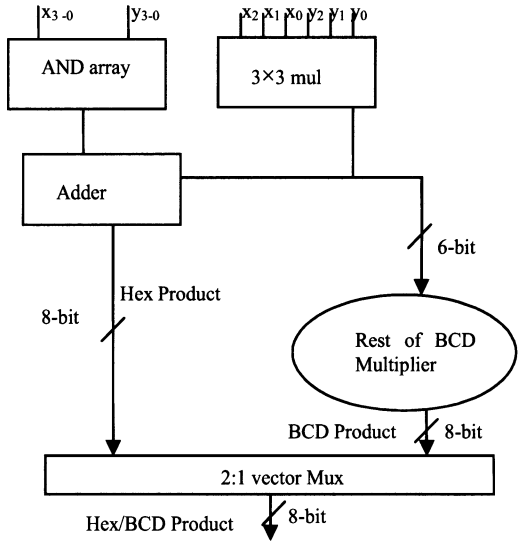

Figure. 10: Hex/Decimal multiplier

A comparison of the proposed Hex/Decimal multiplier design with one designed using the multiplier in [7] in terms of area and critical path delay is done with the logic synthesis tool Leonardo Spectrum from Mentor Graphics Corporation using ASIC Library 0.18 micron, 1.8 V CMOS technology. The comparison shows that the proposed design has a reduction in delay of 16.52% with an extra hardware of 17.24% compared to the Hex/Decimal multiplier designed using the multiplier in [7].

4: DECIMAL FIXED POINT MULTIPLICATION

The fixed point multiplier unit takes two n-digit operands, calculates n^2 partial products and returns their sum as a $2n$-digit integer. There are two main components in the fixed-point multiplier design: generation of partial products and reduction of partial products. In the first stage of the process, generation of partial products is done using n^2 single digit multipliers. After the generation of partial products they are reduced using multi-operand decimal adders to generate a $2n$-digit product. Many techniques have been developed to speed up the process of decimal addition. Direct decimal addition is one of the efficient techniques for two-operand decimal addition [11]. Erle and Schulte proposed a variant of direct decimal addition to produce intermediate results in a decimal carry-save format that can be used in an iterative decimal multiplier [12]. In another approach, proposed by Ohtsuki et al., a correction value of six is added to each digit of the first partial product using a binary carry-save adder [13]. Shirazi et al. proposed a technique for constant time decimal addition, called Redundant Binary Coded Decimal (RBCD) [14, 15]. Kenney and Schulte introduced three algorithms for performing fast decimal addition on multiple BCD operands: non-speculative tree, double correction speculation array and single correction speculation array [16]. The non-speculative tree algorithm that gives the minimum delay with same area of the three algorithms is the best suited for multi-operand decimal addition. In [17] a new scheme is proposed to obtain the sum of each decimal column via a network of carry-free adders and converting the sum into decimal format via a fast binary to decimal converter.

The block diagram of a fully parallel fixed point decimal multiplier is shown in Figure 11. This is a fully parallel multiplier that makes use of only combinational logic. The block is realized for a (7-digit × 7-digit) fixed point multiplication. This example is considered since it is an integral component of a 32-bit decimal floating point multiplication that has 7 significand digits. A (7-digit × 7-digit) multiplication results in 7 × 7 (49) partial products, each having 2 digits given by P_{ijL} and P_{ijH}. All partial product digits are generated using the single digit BCD multipliers. The first block is an array of single digit BCD multipliers that generates the 8-bit partial products P_{ijL} and P_{ijH}. The partial products are accumulated using an array of multi-operand BCD adders.

The multi-operand adder block for a (7-digit × 7-digit) fixed point multiplier is shown in Figure 12. The adder block consists of two 3-operand, 5-operand, 7-operand, 9-operand, 11-operand, 13-operand decimal adders. This is followed by High speed decimal adder to generate the final products (FP_i).

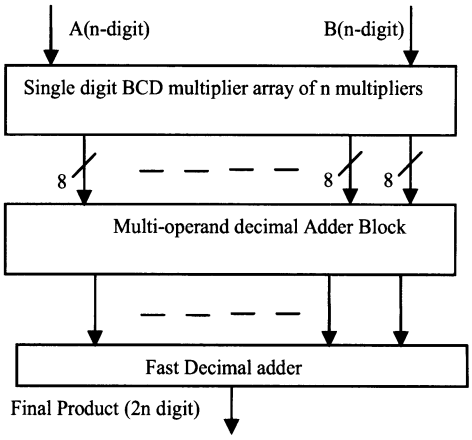

Figure. 11: Parallel Fixed Point Decimal Multiplier

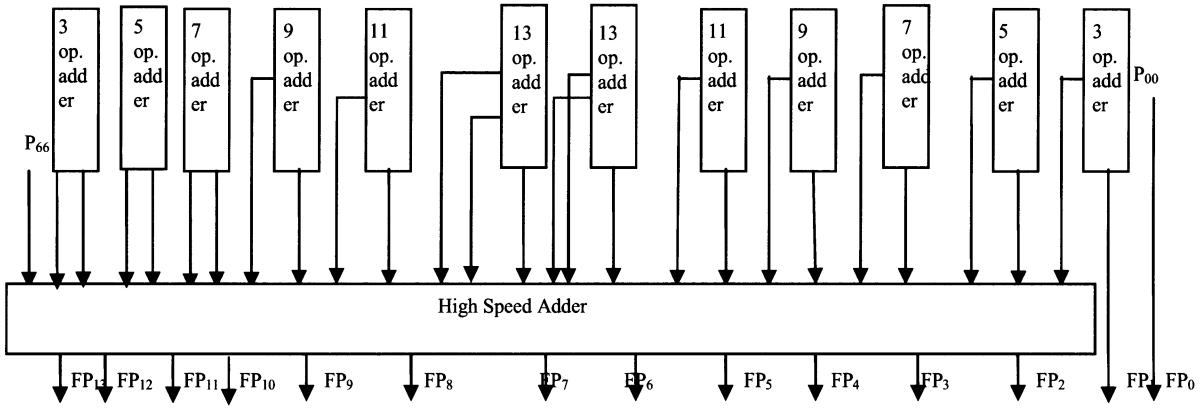

Figure. 12: Adder array for accumulating partial products

5: SYNTHESIS RESULTS

The proposed parallel decimal fixed point multiplier was coded for a (7-digit × 7-digit) multiplier in VHDL, and synthesized to evaluate the area and delay of the design. Synthesis was done using Leonardo Spectrum from Mentor Graphics Corporation with ASIC Library of 0.18 micron, 1.8 V CMOS technology. An area and delay breakdown for an approximate contribution of major components of the design shown in Figure 11 is given in Table 2.

Table 2. Area and Delay for different stages of Decimal Fixed Pont Multiplier (7-digit × 7-digit)

Component	Area		Delay	
	μm^2	%	ns	%
Single digit multiplier array	24255	79.33%	7.81	22.33%
BCD adder array	5358	17.5%	6.70	19.16%
Decimal adder	962	3.14%	20.46	58.5%
Fixed point Multiplier	30575	100%	34.97	100%

The proposed design differs from the iterative approach using easy multiples for partial product generation for hardware realization of BCD multipliers. The design in [7] uses a similar approach and so this is also synthesized in the same environment as the proposed multiplier. The proposed design of a single digit multiplier array block gives a reduction of 7% in area and 16% in delay compared to the multiplier in [7].

6: CONCLUSION

This paper proposed a novel single digit BCD multiplier cell that can be used in multi-digit BCD multiplier circuits. It is demonstrated that this design gains a 7% savings in the area and 16% savings in delay compared to the existing design of [7]. This design leads to more regular VLSI implementation, and does not require special registers for storing easy multiples. The design was validated using (7-digit × 7-digit) fixed point decimal multiplication that is required for a 32-bit floating point decimal multiplication. The synthesized design has a latency of 34.97 ns and can be pipelined to increase throughput. The design for a single digit decimal multiplier is extended to a Hex/Decimal single digit multiplier that gives either a decimal output or a binary output depending on the requirement. Digital Signal Processing (DSP) applications require binary multipliers while financial and commercial applications require decimal multipliers. The comparison shows that the proposed design of Hex/Decimal multiplier has a reduction in delay of 16.52% with an extra hardware of 17.24% compared to the Hex/Decimal multiplier designed using the multiplier in [7].

7: REFERENCES

[1] T. Ohtsuki, et al., "Apparatus for Decimal Multiplication," U.S. Patent, June 1987, #4,677,583

[2] Ueda, T.: 'Decimal multiplying assembly and multiply module'.U.S. Patent 5379245, January 1995

[3] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, "The IBM z900 Decimal Arithmetic Unit," in *Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1335–1339, November 2001

[4] M. A. Erle and M. J. Schulte, "Decimal Multiplication Via Carry-Save Addition," IEEE 14th International Conference on Application-specific Systems, Architectures and Processors, pp. 348-358, June 2003

[5] R. D. Kennedy, M. J. Schulte and M. A. Erle, "A High-Frequency Decimal Multiplier," IEEE 14th International IEEE international conference on Computer Design (ICCD'04), pp. 22-29, Oct 2004

[6] Erle, M.A. Schwarz, E.M. Schulte, M.J, "Decimal multiplication with efficient partial product generation**", 17th IEEE Symposium on** Computer Arithmetic, 2005. ARITH-17 2005. pp. 21- 28, June 2005

[7] Jaberipur, G.; Kaivani, A, "**Binary-coded decimal digit multipliers",** Computers & Digital Techniques, IET Volume 1, Issue 4, July 2007 pp. 377 – 381

[8] Schmookler, M.: 'High-speed binary-to-decimal conversion', IEEE Trans. Comput., 1968, 17, (5), pp. 506–508

[9] Rhyne, V.T.: 'Serial binary-to-decimal and decimal-to-binary conversion', IEEE Trans. Comput., 1970, 19, (9), pp. 808–812

[10] Arazi, B., and Naccache, D.: 'Binary-to-decimal conversion based on the 282 1 by 5', Electron. Lett., 1992, 28, (23), pp. 2151–2152

[11] M. Schmookler and A. Weinberger, "High Speed Decimal Addition," IEEE Trans. Computers, vol. 20, no. 8, pp. 862-867, Aug. 1971

[12] M.A. Erle and M.J. Schulte, "Decimal Multiplication via Carry-Save Addition," Proc. IEEE 14th Int'l Conf. Application-Specific Systems, Architectures, and Processors, pp. 348-358, June 2003.

[13] T. Ohtsuki et al., "Apparatus for Decimal Multiplication," US Patent #4,677,583, June 1987.

[14] B. Shirazi, D.Y. Yun, and C.N. Zhang, "RBCD: Redundant Binary Coded Decimal Adder," IEE Proc.—Part E, vol. 136, no. 2, Mar. 1989.

[15] B. Shirazi, D.Y. Yun, and C.N. Zhang, "VLSI Designs for Redundant Binary-Coded Decimal Addition," Proc. Seventh Ann. Int'l Conf. Computers and Comm., pp. 52-56, Mar. 1988.

[16] R. D. Kenney and M. J. Schulte, 'High-Speed Multi-operand Decimal Adders' IEEE Transactions on Computers, vol. 54, No. 8, Aug 2005, pp. 953-963

[17] L. Dadda, 'Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach', IEEE Transactions on Computers, Vol. 56, No. 9, Sept 2007