ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

Development of Reconfigurable Optimal Decimal Multipliers: An Overview

Prashant R Indurkar[1], Sanjay V Dudul[2]

Associate Professor, Department of E & T, B D College of Engineering, Sevagram, Wardha, India[1]

Professor & Head, Department of Applied Electronics, SGB Amravati University, Amravati (M.S), India[2]

ABSTRACT*:* The realization *of* Multiplication operation is essential for most of the scientific applications as well as in commercial applications like banking, accounting, financial analysis, tax calculation, currency conversion, and insurance etc. Multiplier is a key building block and almost obligatory component in all such applications. That is why reconfigurable PLDs are equipped with dedicated embedded multipliers. The prerequisite of the multiplier implementation is that it should be primarily fast and secondarily efficient in terms of power consumption and chip area. The multiplication involves two basic operations viz. generation of partial products and their accumulation. To speed up the multiplication process, one can reduce the number of partial products to be generated and later, accelerating their accumulation. The design of multiplier depends upon the type, range and precision of data to be processed by the multiplier block viz. fixed point and floating point numbers represented in binary and decimal format.

Since, in electronic computers, binary data can be stored more efficiently and processed faster than decimal data, binary multiplication is suitable for scientific applications due to its mathematical properties and performance advantage. However, the decimal format is preferred for many non-scientific applications because of greater precision and decimal rounding. Thus, current financial, e-commerce and user-oriented applications make an intensive use of integer and fractional numbers represented in decimal radix.

This paper presents an overview of the development of reconfigurable decimal multipliers. The multiplication can be realized using sequential technique where partial products are generated sequentially and each newly generated product is added to previously accumulated partial product. The second method is a parallel multiplier in which partial products are generated in parallel and accumulated using a fast multi-operand adder. The third method is an array multiplier, where array of identical cells are used for generating new partial products; accumulating them simultaneously. In this method, no separate circuits are required for generation and accumulation of partial products which reduces execution time but increases hardware complexity. There are several designs of decimal multipliers which are implemented and synthesized by researchers with improvement in one or more design parameters viz. speed, power, and area. In this paper, the module functionality and performance issues of several designs of decimal multipliers are compared.

KEYWORDS*:* IEEE 754 2008 Standard, Decimal Multiplier, Floating Point, Fixed Point, Reconfigurable hardware

I. INTRODUCTION

The synthesis of circuits carrying out arithmetic operations has been a central activity, directly related to computer development. For the fulfillment of this activity, there is a need to describe the computer representations of the various number formats that humans use and the implementation of the basic mathematical operations employing various algorithms. These mathematical operations can be implemented in software or hardware or hardware-software co-design.

The primary physical characteristics of digital design are speed, area and power. For a given computing application implementation, several technologies are available along with a large range of tradeoffs between the various aspects of performance, unit cost, and non-recurring costs (including development effort). A software implementation, targeting off-the-shelf microprocessors, is easy to develop and reproduce but will not always provide the best performance. For cost or performance reasons, some applications will be implemented as application specific integrated

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

circuits (ASICs). An ASIC provides the best possible performance and may have a very low unit cost, at the expense of a very high development cost. An intermediate approach is the use of reconfigurable circuits, or field-programmable gate arrays (FPGAs).

When it comes to the representation of data / numbers in computers, one can use binary data or decimal data representation. Binary data and binary arithmetic is suitable for scientific applications due to its mathematical properties and performance advantage, since, in electronic computers, binary data can be stored more efficiently and processed faster than decimal data. But, now-a-days, decimal numbers permeate our global economy in commerce, trade, banking, and business, with the proliferation of monetary transactions and increasing demand for financial computations. People represent and exchange information in base 10 and perform calculations applying the rules of decimal arithmetic they are taught at school.

Since, the decimal format is preferred for many commercial applications like banking, accounting, financial analysis, tax calculation, currency conversion, and insurance etc., current financial, e-commerce and user-oriented applications make an intensive use of integer and fractional numbers represented in decimal radix.

General purpose microprocessors generally provide hardware support for binary-to-from-decimal conversions. Very recently, hardware support for decimal arithmetic in mainframe microprocessors was developed but limited to speed up some basic decimal integer arithmetic operations. Therefore, microprocessors had to manage decimal numbers using, mainly, their binary fixed and floating-point arithmetic units, introducing costly performance penalties due to conversion operations. Thus, decimal data is converted to a binary representation before being processed using binary arithmetic and the results are converted back to a decimal representation. For intensive decimal data workloads, these conversion operations contribute significantly to the total processing time.

When it comes to accuracy and errors, the binary floating point units may generate accuracy errors when processing the decimal data directly, since many decimal fractions cannot be represented exactly in binary (for example, 1/10 = 0.1 has not an exact binary representation). This is leading to the accumulation of large errors after several decimal operations. In many financial and monetary applications, the errors introduced by decimal to binary conversions are not tolerated and must be corrected since they may violate the legal requirements of accuracy.

When it comes to rounding, binary floating point arithmetic operations perform binary rounding instead of decimal rounding. Consequently, applications, such as financial, commercial, tax, and internet based applications, which are sensitive to representation and rounding errors often require decimal arithmetic.

II. RELATED WORK

The primary motivation for the development of decimal multiplication component is to enable users of computing systems to achieve results from decimal operations that are the same as if performed by hand, albeit tremendously faster. When performing a decimal operation on a system that does not support the storage of decimal numbers, adequate decimal data types, or decimal operations, the result is subject to representation error, conversion error, and rounding (or round off) error. Further, using a system designed for binary arithmetic to perform decimal arithmetic often leads to errors that are difficult to diagnose.

The rounding error would need to occur in software if hardware does not support decimal operations. This is because existing binary operations do not recognize the concept of decimal digits, and therefore, may improperly round the data. Further, it is generally preferable to round a decimal number to a specified number of digits. Having hardware support of decimal operations also has the benefit of significantly faster performance over software support.

III. REVIEW OF REPORTED RELATED PUBLISHED WORK

A. ANALYSIS OF REPORTED WORK FOR DECIMAL64 MULTIPLICATION

Sonia Gonzalez-Navarro et al. [45] presented the first complete design of a BID-based DFP multiplier, which provides correctly rounded results for multiplying IEEE 754-2008 decimal64 numbers. In this design, the number of digits to round off is estimated in parallel with the calculation of the intermediate product and a binary multiplier with carry-save feedback is employed. This approach allows the design to use a single binary multiplier for both significand multiplication and rounding. The design has variable latency to leverage the fact that multiplication results are not often

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

rounded. The design demonstrates that BID multiplication can be efficiently implemented in hardware with much lower latency than a software implementation. Optimizations decrease the BID multiplier's area and critical path delay.

The authors have modelled the optimized decimal64 BID (BID64) multiplier in RTL level Verilog and performed synthesis on a 64-bit, 5-stage pipelined version of the DFP multiplier for DPD-encoded operands (DPD64). The synthesis of the design is done using Synopsis Design Compiler TSMC's tcbn65gplus 65nm CMOS standard cell library with a segmented wire load model. In this technology, a fan-out-of-four (FO4) inverter's delay is 31.3 ps.

The Synthesis results for all DFP multipliers are shown in Table 1. The optimized design (BID64 OPT) has a lower delay and lower area-delay product than all the other multipliers. Although, BID64 ASIL has low area and latency, its worst case delay is very high. In comparison, BID64 PIPE decreases the worst case delay by 73 percent, but increases the area by 19 percent. Compared to BID64 PIPE, BID64 OPT has 2.3 percent less delay and 6.9 percent less area.

It is observed that the DPD64 always produces results in five cycles (latency). Although DPD64 has a throughput of one result per cycle, BID64 OPT can have similar throughput when results do not need to be rounded. However, when results need to be rounded, BID multiplier has a latency of 15 cycles and produces a new result once every 15 cycles. This is because the same data path is used to perform multiplication and rounding, and rounding requires four passes through the multiplier after the original significand multiplication.

Table 1
Synthesis Results of decimal64 multipliers

Design	Area A (μm^2)	Delay D (ns)	A x D (μm^2 x ns)
BID64 OPT [45]	55198	0.84 (27 FO4)	46366
DPD64 [15]	50369	1.04 (33 FO4)	52383
BID64 ASIL [48]	49729	3.24 (103 FO4)	161122
BID64 PIPE [48]	59274	0.86 (28 FO4)	50976
BID64 ASAP [49]	59100	0.81 (26 FO4)	47871

Brian Hickmann, Andrew Krioukov, Michael Schulte and Mark Erle, in their paper [15], presented a fully parallel decimal floating-point multiplier for 64-bit DFP number with significands encoded in the DPD format. The novelty of the design is that it is the first parallel decimal floating-point multiplier offering low latency and high throughput. The design is based on a previously published parallel fixed-point decimal multiplier which uses alternate decimal digit encodings to reduce area and delay. The fixed-point design is extended to support floating-point multiplication by adding several components including exponent generation, rounding, shifting, and exception handling. The Area and delay estimates presented in the paper show a significant latency and throughput improvement with a substantial increase in area as compared to IEEE P754 compliant sequential floating-point multiplier.

To test and analyze the proposed decimal floating-point multiplier, register transfer level models of the design were coded in Verilog, including RTL models of the fixed-point multiplier from. The Verilog models were synthesized using LSI Logic's gflxp 0.11 µm CMOS standard cell library and Synopsys Design Compiler Y-2006.06-SP1. In addition, the Synopsys's DesignWare IP library was used extensively to further improve the results. The design was synthesized for a large range of pipeline depths to explore tradeoffs in terms of latency, area, and delay. This synthesis was performed using Synopsys's auto-pipelining feature, which introduces a small amount of variability in the results. The fan-out-of-four (FO4) inverter delay values are calculated by dividing the critical path delay by 55 ps, which is the delay of an inverter driving four similar inverters in the cell library. The synthesis results are given in Table 2. The results show an 11-stage pipelined version of the presented design significantly outperforms a sequential IEEE P754 floating point multiplier in terms of both latency and throughput, while maintaining a similar clock frequency. The sequential design produces one multiplication result every 21 cycles in the normal case (25 cycle latency). An 11-stage version of design has similar critical path delay, significantly reduced latency, and one result per cycle throughput while incurring a substantial 371% increase in area.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

Table 2
Synthesis Results Parallel Multipliers

Design	Area A (μm^2)	Delay D (ps)
Sequential	237607	850
Combinational	651435	4410
1 stage	675488	4420
5 stages	736954	1310
10 stages	826495	880
11 stages	833845	850
12 stages	862729	820

Mark Erle, Michael Schulte and Brian J Hickmann, in their paper [14], presented the design of a decimal floating-point multiplier in which operands are stored in the decimal64 format with Densely Packed Decimal (DPD) encoding. The core of the multiplication algorithm is an iterative (sequential) scheme of partial product accumulation employing decimal carry-save addition to reduce the critical path delay. Novel features of the proposed multiplier include support for decimal floating-point numbers, on-the fly generation of the sticky bit, early estimation of the shift amount, and efficient decimal rounding, which does not exhibit rounding overflow. RTL models of 64-bit DFP multiplier and its predecessor decimal fixed-point multiplier were coded in Verilog. Both designs were synthesized using LSI Logic's gflxp 0.11um CMOS standard cell library and the Synopsys Design Compiler. The fan-out-of-four (FO4) inverter delay values are calculated by dividing the critical path delay by 55 ps. The synthesis results are given in Table 3.

Table 3
Synthesis Results Sequential Multipliers

Design	Area A (μm^2)	Delay D (ps)
DFP 64-bit DPD format	237607	850
Fixed Point	119653	810

Carlos Minchola and Gustavo Sutter, in their paper [30], reported the design and implementation of a hardware module to calculate the decimal floating-point (DFP) multiplication where the 64-bit operands are encoded in DPD format. For synthesis and implementation XST and Xilinx ISE 10.1 tool have been used respectively. The circuits were implemented in a Virtex-4 speed grade -12 using timing constraints. The synthesis results of DFP multiplier and binary multiplier are presented in Table 4.

Table 4
Results of DFP & Binary Multipliers

Multiplier	#slices	# LUT	T (ns)	# Cy	Delay (ns)
DFP multiplier					
Low latency	3494	5961	5.5	12	66.0
Low area	1222	1966	6.2	26	161.2
Binary multiplier	1381	2308	4.1	09	36.9

R.Raafat, et al., in their paper [19], proposed a fully parallel Decimal64 floating point (FP) multiplier compliant to IEEE Standard 754-2008 for floating point arithmetic. The multiplier possesses novel methods to target low latency. The multiplier reads significand, decoded from a densely-packed decimal (DPD) encoding into Binary Coded Decimal (BCD). The design uses a novel BCD–4221 recoding for decimal digits to improve the area and latency of the partial product generation and the partial product reduction tree. Several enhancements were introduced to the design; the final carry propagation adder is implemented using a fully parallel decimal adder with a Kogge-Stone prefix tree, the sticky bit is generated in parallel to the shifter to reduce the critical path delay. The multiplier is modeled using RTL VHDL

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

and then it is functionally verified using FPGEN test cases supplied by IBM. The proposed DFP multiplier has a low latency and a small area. The proposed floating point multiplier is hardware verified by integrating into NIOS II processor on Altera Cyclone II FPGA development kit. The combinational design shows about 17% improvement in the Area and Delay product over the parallel decimal floating point multiplier presented in [15]. The synthesis results are given in Table 5.

Table 5
AREA AND DELAY FOR MULTIPLIER DESIGNS

Hardware Design	Delay ns (FO4)	Area μm^2(Nand2)
Combinational	7.6 (84.4)	846848.31 (84861)
1 stage	7.53 (83.6)	869870 (87169)
5 stages	2.2 (24.4)	1160125 (118590)
10 stages	1.52 (16.8)	1293553.8 (146428)

A. ANALYSIS OF REPORTED WORK FOR 16X16 DIGITS DECIMAL FIXED POINT ITERATIVE MULTIPLICATION

Mark A. Erle and Michael J. Schulte, in their paper [5], presented two novel designs for fixed-point decimal multiplication that utilize decimal carry-save addition to reduce the critical path delay. The decimal multipliers presented in this paper use dedicated hardware to operate at high frequencies with relatively low latencies. First, a multiplier that stores a reduced number of multiplicand multiples and uses decimal carry-save addition in the iterative portion of the design is presented. When multiplying two n-digit operands to produce a 2n-digit product, the multiplier design has a worst-case latency of n + 5 cycles and an initiation interval of n + 5 cycles, where *n* is the number of significant digits in the multiplier operand. Then, a second multiplier design is proposed with several notable improvements including fast generation of multiplicand multiples that do not need to be stored, the use of decimal (4:2) compressors, and a simplified decimal carry-propagate addition to produce the final product. When multiplying two n-digit operands to produce a 2n-digit product, the improved multiplier design has a worst-case latency of n + 4 cycles and an initiation interval of n + 1 cycles. The area & delay for 16 x16 digit iterative fixed point multiplier is given in Table 6.

Table 6
Area and Delay – Iterative Fixed Point multiplier

Latency (cycles)	**Throughput (Ops/cycle)**	**Area (Nand2)**	**Delay ps (FO4)**
20	1/17	18550	810 (14.7)

Robert D. Kenney, Michael J. Schulte and Mark A. Erle, in their paper [7], reported an iterative decimal multiplier, which is operated at high clock frequencies and scales well to large operand sizes. A new decimal representation for intermediate products, which allows for a very fast two stage iterative multiplier design, is proposed. Decimal multipliers are synthesized using a 0.11 micron CMOS standard cell library and operate at clock frequencies close to 2 GHz. The latency of the proposed design to multiply two n-digit BCD operands is (n + 8) cycles with a new multiplication able to begin every (n + 1) cycle. The synthesis result for 16 x 16 Iterative Fixed point multiplication are given in Table 7.

Table 7
Area and Delay – Iterative Fixed Point multiplier

Latency (cycles)	**Throughput (Ops/cycle)**	**Area (Nand2)**	**Delay FO4**
24	1/17	31550	12.7

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

Mark A. Erle, Eric M. Schwarz and Michael J. Schulte, in their paper [6], presented a novel design for fixed-point decimal multiplication that utilizes a simple recoding scheme to produce signed-magnitude representations of the operands thereby greatly simplifying the process of generating partial products for each multiplier digit. The partial products are generated sequentially using a digit-by-digit multiplier on a word by-digit basis, first in a signed-digit form with two digits per position, and then combined via a combinational circuit. As the signed-digit partial products are developed one at a time while traversing the recoded multiplier operand from the least significant digit to the most significant digit, each partial product is added along with the accumulated sum of previous partial products via a signed-digit adder. The implementation requires n+4 cycles. The synthesis result for 16 x 16 Iterative Fixed point multiplication are given in Table 8.

Table 8
Area and Delay – Iterative Fixed Point multiplier

Latency (cycles)	Throughput (Ops/cycle)	Area (Nand2)	Delay FO4
20	1/17	16000	16

B. ANALYSIS OF REPORTED WORK FOR 16X16 DIGITS DECIMAL FIXED POINT PARALLEL MULTIPLICATION

Alvaro Vazquez, Elisardo Antelo and Paolo Montuschi, in their paper [17], reported two novel architectures for parallel decimal multipliers, based on a new algorithm for decimal carry–save multioperand addition that uses a novel BCD–4221 recoding for decimal digits. It significantly improves the area and latency of the partial product reduction tree. Three schemes for fast and efficient generation of partial products in parallel are also presented. The recoding of the BCD–8421 multiplier operand into minimally redundant signed–digit radix–10, radix–4 and radix–5 representations using new recoders reduce the complexity of partial product generation. In addition, SD radix–4 and radix–5 recodings allow the reuse of a conventional parallel binary radix–4 multiplier to perform combined binary/decimal multiplications. Evaluation results show that the proposed architectures have interesting area–delay figures compared to conventional Booth radix–4 and radix–8 parallel binary multipliers and other representative alternatives for decimal multiplication. The synthesis results of decimal radix 10 parallel multiplier and combined binary/decimal parallel multiplier for 64 bit 16 decimal digit input operands are given in Table 9a.

Table 9a
Area and Delay – Parallel Fixed Point multiplier

Design	Latency FO4 (cycles)	Area μm^2 (Nand2)	Delay ps (FO4)
	54.4 (8)	369238 (60500)	840 (15.3)
Decimal Radix 10	72	40000	
Bin/Dec Radix 4/5	61/71	53500	

Tomas Lang and Alberto Nannarelli, in their paper [47], presented implementations of combinational (parallel) decimal Multiplication unit with simpler recoding of the operands which reduces the number of partial product pre-computations and uses counters to eliminate the need of the decimal equivalent of a 4:2 adder. The 16-digit radix-10 combinational multiplier has been synthesized using the STM 90 nm CMOS standard cells library and Synopsys Design Compiler. The synthesized unit has a critical path of 2.65 ns and an area of 0.3 mm2 equivalent to the area of about 68, 000 NAND-2 gates. The results show that the combinational decimal multiplier offers a good compromise between latency and area when compared to other decimal multiply units and to binary double-precision multipliers. The comparison of radix-4 and radix-10 multipliers is in Table 9b.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

Table 9b
Area and Delay – Parallel Fixed Point multiplier

Arch	Latency (FO4)	Area (Nand2)	Throughput (Ops/cycle)
Radix-4	31.1	34000	1
Radix-10	61.16	68000	1

L.Dadda and A.Nannarelli, in their paper [08], discussed the problem of adding the partial products in the combinational decimal multiplier. This addition is done with a tree of decimal carry-save adders [47] and using proposed multi-operand decimal addition, where the sum of each column of the partial product array is obtained first in binary form and then converted to decimal. The synthesis is done using Synopsys Design Compiler and the STM library of standard cells.With a 90 nm technology, total delay (latency) is reduced from 2.65 ns (58.9 FO4) to 2.51 ns (55.8 FO4) by using multi operand decimal addition, while the area is 297,256 μm^2 in this scheme and 300,000 μm^2 in the scheme where decimal carry save adders were used. The number logic gates required are 68000 (Area – Nand2). The result is shown in Table 10.

Table 10
Area and Delay – Parallel Fixed Point multiplier

Arch	Latency (FO4)	Area (Nand2)	Throughput (Ops/cycle)
Radix-10	55.8	68000	1

S.Gorgin and G.Jaberipur, in their paper [18], discussed about a method for carry-free addition of decimal numbers, where each equally weighted decimal digit pair of the two operands is partitioned into two weighted bit-sets. The arithmetic values of these bit-sets are evaluated, in parallel, for fast computation of the transfer digit and interim sum. In this fully redundant adder, both operands and sum are redundant decimal numbers with overloaded decimal digit set. This adder is shown to improve upon the latest high performance similar works and outperform all the previous alike adders. This adder and its restricted-input varieties are shown to efficiently fit in the design of a parallel decimal multiplier. The two-to-one partial product reduction ratio that is attained via the adder has lead to a VLSI-friendly recursive partial product reduction tree. Two alternative architectures, reduction schemes, for decimal multipliers are presented; one is slower, but area-improved, and the other one, relatively fast, consumes more area, but is delay-improved. However, both the designs are faster in comparison with previously reported parallel decimal multipliers w.r.t. area and latency comparisons. The comparison result after synthesizing the multipliers using Synopsis Design Compiler using TSMC 0.13 µm standard process show that latencies of proposed multipliers are given below in Table 11.

Table 11
AREA AND DELAY Comparison

Hardware Design	Delay (FO4)	Area (Nand2)
Proposed (Fast)	46.88	48625
Proposed (Slow)	51.06	39120

G. Jaberipur and A. Kaivani, in their paper [10], discussed about the design of fast parallel decimal multiplication circuits, by presenting two solutions with alternative designs. In order to improve the speed of parallel decimal multiplication, a new PPG method is presented and, fine-tune the PPR method of one of the full solutions and the final addition scheme of the other; thus, assembling a new full solution. The Logical Effort analysis and 0.13 µm synthesis show at least 13% speed advantage, but at a cost of at most 36% additional area consumption. The performance comparison is given in Table 12.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

Table 12
AREA AND DELAY Comparison

Hardware Design	Delay (FO4)	Area (Nand2)
Proposed [10]	53.53	79636
Lang et al. [47]	61.16	68000
Castellanos et al. [13]	60.47	53645

Alvaro Vazquez, Elisardo Antelo and Paolo Montuschi, in their paper [11], describes the architectures of two parallel decimal multipliers in which the parallel generation of partial products is performed using signed-digit radix-10 or radix-5 recodings of the multiplier and a simplified set of multiplicand multiples. The reduction of partial products is implemented in a tree structure based on a decimal multi operand carry-save addition algorithm that uses unconventional (non BCD) decimal-coded number systems. The evaluation results for 16-digit operands show that the proposed architectures have interesting area-delay figures compared to conventional Booth radix-4 and radix-8 parallel binary multipliers and outperform the figures of previous alternatives for decimal multiplication. The area-delay values measured for the SD radix-10 multiplier and SD radix-5 multipliers are given in Table 13.

Table 13
Area and Delay – Parallel Fixed Point multiplier

Arch	**Latency (FO4)**	**Area (Nand2)**	**Throughput (Ops/cycle)**
Radix-5	47.8	50900	1
Radix-10	48.4	44400	1

Liu Han and Seok-Bum Ko, in their paper [43], described a parallel decimal multiplication algorithm with three Components, which are a partial product generation, a partial product reduction, and a final digit-set conversion. First, a redundant number system is applied to recode not only the multiplier, but also multiples of the multiplicand in signed-digit (SD) numbers. Furthermore, a multi-operand SD addition algorithm is presented to reduce the partial product array. Finally, a digit-set conversion algorithm with a hybrid prefix network to decrease the number of the logic gates on the critical path is discussed. The synthesized results under 90 nm technology, after timing delay analysis, is given in Table 14.

Table 14
Area and Delay – Parallel Fixed Point multiplier

Latency (FO4)	**Area (Nand2)**	**Throughput (Ops/cycle)**
43.1	49900	1

IV. OVERALL PERFORMANCE COMPARISON OF THE REPORTED WORK

The overall performance comparison of the reported work on the design of decimal multipliers based on IEEE 754 2008 decimal64 representation is as given below in the Table no.15.

Table 15
Overall Comparison – decimal64 based multipliers

Hardware Design	Encoding	Area (μm2)	Latency (cycles)
S.G-Navarro et al. [45]	BID	55198	5/15
B J Hickmann et al. [15]	DPD	50369	5
S.G-Navarro et al. [48]	BID	49729	3/8
C. Tsen [49]	BID	59100	5/15

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

The overall performance comparison of the reported work on the design of 16 x 16 digits decimal fixed point parallel multipliers is as given below in the Table no.16. The throughput is 1 as the number of cycles required is 1.

Table 16
Overall Comparison – 16 x 16 digit decimal fixed point parallel multipliers

Hardware Design	Area (Nand2)	Latency (FO4)
Liu Han et al. [43]	49900	43.1
T. Lang [47]	68000	58.9
L. Dadda [08]	68000	55.8
	49700	46.5
	79600	53.5
	49500	48.1
	60500	54.4

V. CONCLUSION

The hardware research into decimal formats and operations is motivated by projections that current software decimal solutions may be inadequate for the performance demands of future decimal applications. The Multiplication is an important and frequent operation in decimal applications, so it is compelling to investigate it as a component of a decimal hardware implementation based on IEEE 754 2008 standard. The standard includes specifications for decimal formats and operations. In this paper, we have reviewed and investigated the hardware suitability of these formats for reconfigurable decimal hardware implementation with respect to primary physical characteristics of a digital design viz. speed, power and area.

It is observed that the researchers have been targeting this problem for a long time. There exist several enhancements in the design of decimal multipliers for better design parameters but there is no indication that this trend of improvement will stop in future years. It is, most likely, possible to optimize the design further, and proposed future work aims towards it.

REFERENCES

[1] M. F. Cowlishaw, "Decimal Floating-Point Algorism for Computers," Proc. of the 16th IEEE Symposium on Computer Arithmetic, pp. 104-111, June 2003.
[2] IEEE Standards Committee, 754-2008 IEEE Standard for Floating-Point Arithmetic, (http://ieeexplore.IEEE.org/servlet/opac?punumber=4610933), pp. 1-58, Aug. 2008, DOI: 10.1109/IEEESTD.2008.4610935.
[3] M. Cornea et al., "A software implementation of the IEEE 754R decimal floating-point arithmetic using the binary encoding format," IEEE Trans. on Computers, vol. 58, No. 2, pp. 148-162, 2009.
[4] L-K. Wang et al., "A survey of hardware designs for decimal arithmetic," IBM Journal of Research and Development, vol. 54, No. 2, pp. 8:1-8:15, 2010.
[5] M. A. Erle and M. J. Schulte, "Decimal Multiplication Via Carry-Save Addition," IEEE Int. Conf. on Application Specific systems, Architectures, and Processors, pp. 348-358, June 2003.
[6] M. A. Erle, E. M. Schwarz, and M. J. Schulte, "Decimal Multiplication with efficient partial product generation," Proc. 17th IEEE Symp. on Computer Arithmetic, pp. 21-28, 2005.
[7] R. D. Kenney, M. J. Schulte, and M. A. Erle, "A high-frequency decimal multiplier," Proc. IEEE Int. Conf. Comput. Des.: VLSI in Comput. and Processors, pp. 26-29, 2004.
[8] L. Dadda and A. Nannarelli, "A Variant of a Radix-10 Combinational Multiplier," IEEE Int. Symp. in Circuits and Systems, ISCAS 2008, pp. 3370-3373, May 2008.
[9] L. Dadda, "Multioperand Parallel Decimal Adder: a mixed Binary and BCD Approach," IEEE Trans. on Computers, vol. 56, pp. 1320-1328, Oct. 2007.
[10] G. Jaberipur and A. Kaivani, "Improving the Speed of Parallel Decimal Multiplication," IEEE Trans. on Computers, vol. 58, No.11, pp. 1539-1552, Nov. 2009.
[11] A. V´azquez, E. Antelo, and P. Montuschi, "Improved Design of High-Performance Parallel Decimal Multipliers," IEEE Trans. on Computers, vol. 59, No. 5, pp. 679-693, May 2010.
[12] I. D. Castellanos and J. E. Stine, "Decimal partial product generation architectures," Proc. 51st Midwest Symp. on Circuits and Systems, pp. 962-965, Aug. 2008.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

[13] I. D. Castellanos and J. E. Stine, "Compressor Trees for Decimal Partial Product Reduction," Proc. 18th ACM Great Lakes Symp. on VLSI, pp. 107-110, May 2008.
[14] M. A. Erle, M. J. Schulte, and B. J. Hickmann, "Decimal floating point multiplication via carry-save addition," Proc. 18th IEEE Symp. on Computer Arithmetic, pp. 25-27, 2007.
[15] B. J. Hickmann, A. Krioukov, M. J. Schulte, and M. A. Erle, "A parallel IEEE P754 decimal floating-point multiplier," Proc. IEEE Int. Conf. Comput. Des., pp. 296-303, 2007.
[16] M. A. Erle, B. J. Hickmann, and M. A. Schulte, "Decimal Floating-Point Multiplication," IEEE Trans. on Computers, vol. 58, No. 7, pp. 902-916, July 2009.
[17] A. Vazquez, E. Antelo, and P. Montuschi, "A New Family of High-Performance Parallel Decimal Multipliers," Proc. 18th IEEE Symp. on Computer Arithmetic, pp. 195-204, June 2007.
[18] S. Gorgin and G. Jaberipur, "A fully redundant decimal adder and its application in parallel decimal multipliers," Microelectronics Journal, vol. 40, No. 10, Oct. 2009.
[19] R. Raafat et al., "A decimal fully parallel and pipelined floating point multiplier," in Forty-Second Asilomar Conference on Signals, Systems, and Computers, Asilomar, California, USA, Oct. 2008.
[20] R.D. Kenney and M.J. Schulte, "High-Speed Multi-operand Decimal Adders," IEEE Trans. Computers, vol. 54, no. 8, pp. 953-963, Aug. 2005.
[21] G. Jaberipur and A. Kaivani, "Binary-Coded Decimal Digit Multipliers," IET Computers & Digital Techniques, vol. 1, no. 4, pp. 377-381, July 2007.
[22] M.A. Erle, M.J. Schulte, and J.M. Linebarger, "Potential Speedup Using Decimal Floating-Point Hardware," Proc. 36th Asilomar Conf. Signals, Systems and Computers (ACSSC '02), vol. 2, pp. 1073-1077, Nov. 2002.
[23] M.F. Cowlishaw, "Densely Packed Decimal Encoding," IEE Proc.—Computers and Digital Techniques, vol. 149, no. 3, pp. 102-104, May 2002.
[24] G. Even and P.-M. Seidel, "A Comparison of Three Rounding Algorithms for IEEE Floating-Point Multiplication," IEEE Trans. Computers, vol. 49, no. 7, pp. 638-650, July 2000.
[25] N.T. Quach, N. Takagi, and M.J. Flynn, "Systematic IEEE Rounding Method for High-Speed Floating-Point Multipliers," IEEE Trans. VLSI Systems, vol. 12, no. 5, pp. 511-521, May 2004.
[26] A.Y. Duale, M.H. Decker, H.-G. Zipperer, M. Aharoni, and T.J. Bohizic, "Decimal Floating-Point in z9: An Implementation and Testing Perspective," IBM J. Research and Development, http://www.research.ibm.com/journal/rd/511/duale.html, 2007.
[27] W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," IEEE Transactions on Computers, vol. 49, no. 7, pp. 692–701, Jul. 2000.
[28] P.M. Seidal, L. McFearin, D. Matula, "Secondary Radix Recodings for Higher Radix Multipliers," IEEE Transactions on Computers, vol. 54, no.2, 2005
[29] P. Karlstrom, A. Ehliar, D. Liu, "High Performance, Low Latency FPGA based Floating point Adder and Multiplier Units in a Virtex 4," 24th Norchip Conference, 2006.
[30] C. Minchola, G. Sutter, "A FPGA IEEE 754 2008 Decimal64 Floating Point Multiplier," IEEE Conference ReConFig, 2009.
[31] M. Baesler, S.-O.Voigt, T. Teufel, "An IEEE 754-2008 Decimal Parallel and Pipelined FPGA Floating-Point Multiplier," International Conference on FPGAs FPL, 2010.
[32] C. Tsen, S. Gonzalez-Navarro, M. J. Schulte, L. Compton, "Hardware Designs for Binary Decimal-Based Rounding," IEEE Transactions on Computers, Vol. 60, No.5, 2011.
[33] H.A.H. Fahmy, T.El Deeb, M.Y. Hassan, Y. Farouk, R.R.Eissa,"Decimal Floating-Point for future Processors," IEEE Microelectronics (ICM), 2010.
[34] V.S. Dimitrov, K.U. Jarvinen, J. Adikari,"Area-Efficient Multipliers Based on Multiple-Radix Representations," IEEE Transactions on Computers, vol. 60, no.2, 2011.
[35] R.K.James, K.P.Jacob, S.Sasi, "Decimal Floating Point Multiplication using RPS Algorithms," International Conference on VLSI, Communications & Instrumentation, 2011.
[36] M.Al-Ashrafy, A.Salem, W.Anis, "An Efficient Implementation of Floating-Point Multiplier," IEEE, 2011.
[37] S. Emami, M. Dorrigiv, G. Jaberipur, G., "Radix-10 Addition with Radix-1000 Encodings of Decimal Operands," 16th CSI International Conference on Computer Architecture and Digital Systems, CADS'12, 2012.
[38] A. Jain, B. Dash, A.K. Panda, M Suresh, "FPGA Design of a Fast 32-bit Floating Point Multiplier," International Conference on Devices, Circuits and Systems, ICDCS'12, 2012.
[39] O. D.Al-Khaleel, N. H.Tulic, K. M.Mhaidat, "FPGA Implementation of Binary Coded Decimal Digit Adders and Multipliers," 8th International Symposium on Mechatronics & its Applications, ISMA'12, 2012.
[40] E. Carlos, G. Minchola, "Implementation of a Fully Pipelined BCD Multiplier in FPGA," 8th Southern Conference on Programmable Logic, SPL'12, 2012.
[41] S. Gorgin and G. Jaberipur, "A Fully Redundant Decimal Adder and Its Application in Parallel Decimal Multipliers," Microelectronics J., vol. 40, no. 10, pp. 1471-1481, Oct. 2009.
[42] M.Huang, K.Gaj, T.El-Ghazawi, "New Hardware Architectures for Montgomery Modular Multiplication Algorithm," IEEE Transactions on Computers Vol.60, No.7, 2011.
[43] L.Han, S.-B.Ko, "High Speed Parallel Decimal Multiplication with Redundant Internal Encodings," IEEE Transactions on Computers, Vol. 62 No.5, May 2013.
[44] D.Chen, L.-Han, Y. Choi, S.-B.Ko, "Improved Decimal Floating-Point Logarithmic Converter Based on Selection by Rounding," IEEE Transactions on Computers, Vol.61, No.5, 2012.
[45] S.G.Navarro, C.Tsen, M. J.Schulte, "Binary Integer Decimal-based Floating Point Multiplication," IEEE Transactions on Computers, Vol. 62 No. 7, July 2013.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Issue 12, December 2016

[46] A.Aswal, P.M.Ganesh, G.N.S.Prasanna, "On Basic financial Decimal Operations on Binary Machines," IEEE Transactions on Computers, Vol.61, No.8, 2012.
[47] T. Lang and A. Nannarelli, "A Radix-10 Combinational Multiplier," Proc. of 40th Asilomar Conference on Signals, Systems, and Computers, pp. 313–317, Nov. 2006.
[48] S.G. Navarro, C. Tsen, and M.J. Schulte, "A Binary Integer Decimal-Based Multiplier for Decimal Floating-Point Arithmetic," Proc. 41st Asilomar Conf. Signals, Systems, and Computers, pp. 164-170, 2007.
[49] C. Tsen, S.G. Navarro, M.J. Schulte, B. Hickmann, and K. Compton, "A Combined Decimal and Binary Floating-Point Multiplier," Proc. IEEE 20th International Conference Application-Specific Systems, Architectures, and Processors, pp. 8-15, 2009.
[50] L. Eisen, J.W. Ward III, H.-. Tast, N. Mading, J. Leenstra, S.M. Mueller, C. Jacobi, J. Preiss, E.M. Schwarz, and S.R. Carlough, "IBM POWER6 Accelerators: VMX and DFU," IBM J. Research and Development, vol. 51, pp. 663-683, Nov. 2007.
[51] C.H. Webb, "IBM z10: The Next-Generation Mainframe Microprocessor," IEEE Micro, vol. 28, no. 2, pp. 19-29, Mar./Apr. 2008.
[52] P.T.P. Tang, "BID- Binary-Integer Decimal Encoding for Decimal Floating Point," Intel Corporation, technical report, http://754r.ucbtest.org/issues/decimal/bid_rationale.pdf, July 2005.
[53] M.F. Cowlishaw, "The *decNumber C* Library, v3.61. IBM", http://speleotrove.com/decimal/decnumber.pdf, 2011.
[54] Sun Microsystems. BigDecimal class, java 2 platform standard edition 5.0, API specification. http://java.sun.com/j2se/1.5.0/docs/api/java/math/BigDecimal.html, 2011.
[55] Intel Corporation, "Intel Decimal Floating-Point Math Library," http://software.intel.com/en-us/articles/intel-decimal-floatingpoint-math-library/, 2011.