Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 4, April 2019, pp. 1190-1198
DOI: 10.23940/ijpe.19.04.p14.11901198

UCM: A Novel Approach for Delay Optimization

Rajkumar Sarma[a], Cherry Bhargava[a,*], Sandeep Dhariwal[b], and Shruti Jain[c]

[a]*School of Electronics and Electrical Engineering, Lovely Professional University, Phagwara, Punjab, 144411, India*
[b]*Alliance College of Engineering and Design, Alliance University, Bengaluru, Karnataka, 562106, India*
[c]*Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Himachal Pradesh, 173234, India*

Abstract

In the era of digital signal processing, such as graphics and computation systems, multiplication is one of the prime operations. A multiplier is a key component in any kind of digital system such as Multiply-Accumulate (MAC) unit, various FFT algorithms, etc. The efficiency of a multiplier is mainly dependent upon the speed of operation and power dissipation of the circuit along with the complexity level of the multiplier. This paper is based on Universal Compressor based Multiplier (UCM), which yields a high-speed operation with comparative power dissipation; hence, the enhanced performance is reported. The novel design of UCM is analyzed using Cadence Spectre tool in 90nm CMOS technology. Finally, the UCM is implemented using Nexys-4 Artix-7 FPGA board. The novel design of UCM has demonstrated significant improvement in terms of delay, which is explored in this paper.

Keywords: multiplier; compressor design; low power; high speed; Nexys-4 Artix-7 FPGA; Cadence Virtuoso; MAC unit; delay optimization

(Submitted on December 25, 2018; Revised on December 27, 2018; Accepted on April 15, 2019)

1. Introduction

Multiplication has a vast field of applications such as digital signal processing, multimedia systems, arithmetic operation, and digital communication. As the operation of the multiplier can be segregated into two categories, namely, partial product generator and final sum using adder circuits, the multiplication process requires more hardware resources and processing time in comparison to the basic adder/subtractor circuit. In a simplified view, a multiplier requires AND gates (for partial product generation) and adder circuits (half adders and full adders) for the addition of partial products to yield the final result. Figure 1 shows the simplified operation of a multiplier. Various multiplier algorithms/architectures have been proposed in the past, such as the booth encoder, Wallace tree adder, array multiplier, and modified booth multiplier (as mentioned by Liao et al. [1]). All these algorithms/architectures use different approaches to make the multiplier operation more efficient. For example, booth multipliers or modified booth multipliers are an algorithmic approach where the main focus is on reducing the total number of partial products. On the other hand, as explained by Liao et al. [1], in the case of the Wallace tree multiplier, the main focus is on the efficient addition of the partial products. Hence, a combination of both can provide a better result.

There are various multiplier circuits explained in literature that mainly focus on the issues of power consumption, delay of the multiplier circuit, and lesser area [2-18]. However, according to these studies, it was found that the area and the speed of operation are the two most conflicting design constraints. Hence, increasing the speed of operation enhances the area requirement. On the other hand, as the size of the transistor decreases, the area cannot become a major issue in today's digital systems. The power consumption and delay of a particular circuit depends upon the supply voltage (V_{DD}). A slight increment in the supply voltage increases the overall power consumption, and at the same time, it decreases the delay of the circuit. Hence, there is always a trade-off between power consumption and delay of a circuit. Therefore, the supply voltage plays an important role in designing a low power circuit, i.e., for a low power design, an optimized supply voltage needs to be chosen so that the output logic is valid and the power consumption is at a bare minimum with a comparable delay value. As per the literature survey, it was found that most of the multiplier design uses the Wallace tree multiplier as the basic

* Corresponding author.
E-mail address: cherry.bhargava@lpu.co.in

algorithm. In the majority of the cases, the basic Wallace tree multiplier algorithm was modified to get better results [1, 3-4, 8, 12-13]. The Wallace tree algorithm is the simplest way of designing multipliers with optimized delay/power consumption. In this paper, a high-speed multiplier architecture with a minimal value of supply voltage is proposed. In the implemented architecture, the supply voltage is minimized to reduce the power consumption of the circuit without compromising the speed of the multiplier circuit. The study mainly focuses on the optimization on the partial product addition. For partial product generation, the booth algorithm produces a better result than any other multiplication approaches. Secondly, as discussed above, the majority of the multipliers use a Wallace tree adder for partial product addition. Hence, an optimized and efficient partial product adder, which can replace the Wallace tree algorithm, can yield a better multiplier.

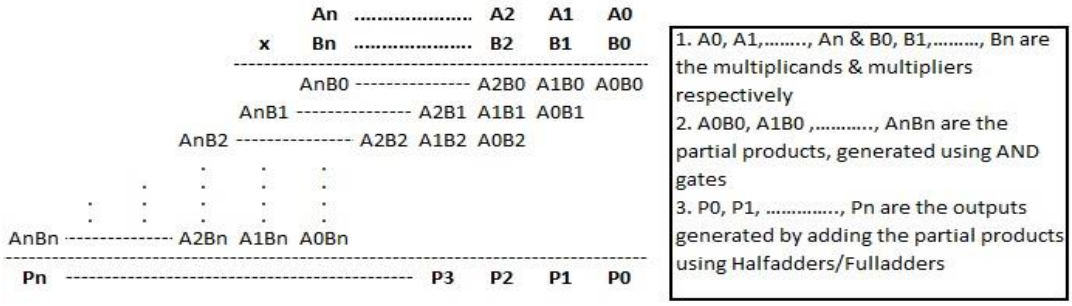

Figure 1. Basic multiplication operation

The following sections are discussed as follows: in section 2, Wallace tree multiplier architecture is presented; in section 3, the novel architecture is explained; in section 4, a detailed analysis of the UCM architecture is provided along with its realization on FPGA board; and in section 5, a detailed conclusion and future scopes of the UCM architecture are discussed.

2. Wallace Tree Multiplier Architecture

The conventional Wallace tree multiplier algorithm is divided into three stages:

- Stage 1: Partial product generation.
- Stage 2: Addition of partial products, which creates 'sum' and 'carry' separately.
- Stage 3: A final adder, which is generally a fast adder, to add the sum and carry together to yield the final result [9].

In stage 1, the partial products are generated by ANDing each multiplier bit with each multiplicand bit. It can be implemented either by using a conventional two input AND gate to find the partial product of each multiplicand and multiplier or by using an advanced booth multiplier to reduce the total number of partial products. With the help of the second order booth algorithm, the number of partial products is reduced to approximately half the bit width of the multiplier [3].

In stage 2, the partial products are added using half adder/full adder. To achieve this, the partial products with '*N*' rows are grouped together in sets of three rows each. Any rows that are not part of the group of three rows are transferred to the next level without any modification. In the groups of three rows, full adders are applied to the columns containing three partial products, and half adders are applied to the columns containing two partial products (in the groups of two rows) [15]. The columns with only one partial product are transferred to the next level without any modification. For the next level calculation, use the sum and carry output of the full adder/half adder of the previous level along with the remaining partial products. The same procedure is followed until there are only two rows left.

In stage 3, the remaining two rows are added either by using an n-bit Ripple Carry Adder (RCA) or by using a fast adder such as carry look-ahead adder, carry select adder, etc. Figure 2 elaborates on the operation of the Wallace tree multiplier algorithm in detail, where a0-a8 and b0-b8 represent the multiplicand and multiplier respectively, q0-q80 represents the partial products, S_{xx} and C_{xx} represent the sum and carry outputs of the half adder/full adder respectively, and CR_x represents the ripple carry at the final stage. Moreover, as shown in Figure 2, the rectangles with three variables represent the full adder and the rectangles with two variables represent the half adder.

3. Design Process of UCM Architecture

A universal *N*:1 bit compressor-based multiplier is proposed in this paper. The process flow of this research work is shown

in Figure 3. As shown in the figure, the novel architecture is designed in Cadence Virtuoso 90nm CMOS technology as well as in Verilog HDL. The power and delay analysis is carried out from virtuoso-based design, whereas the Verilog HDL program is used for FPGA prototyping.

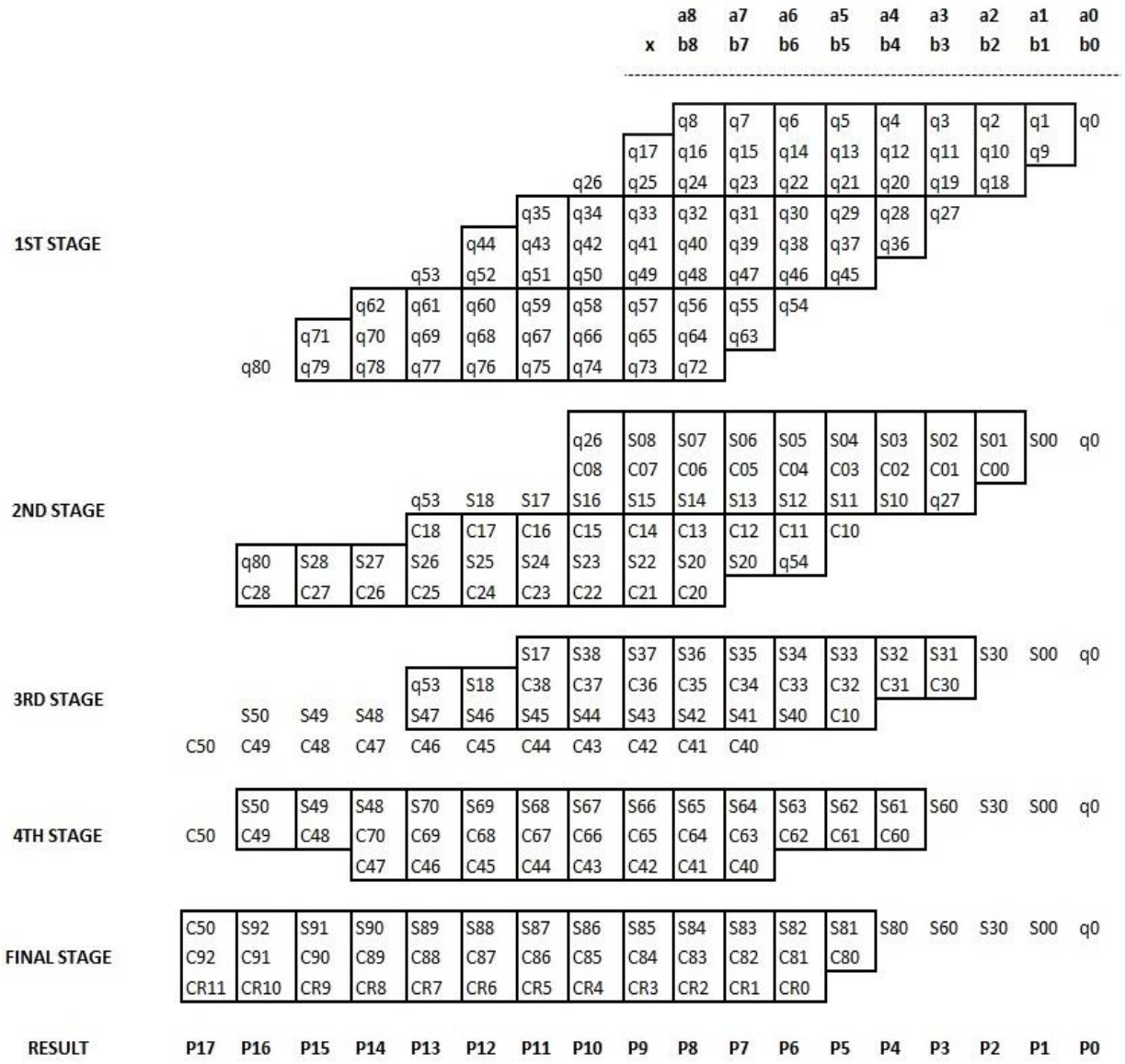

Figure 2. Wallace tree multiplier for 9x9 bit multiplication

4. UCM Architecture

Although the Wallace tree multiplier is much faster than the array multiplier [4], it requires a large number of adders. Secondly, the Wallace tree multiplier is highly irregular and complicated. Therefore, in order to overcome the irregular structure, several modified Wallace tree algorithms have been proposed in literature [1, 3-4, 6-9,12-14]. All these multiplier algorithms are based upon the Wallace tree. Hence, replacing the Wallace tree algorithm may further improve the result of the multiplier. Another important point here is, instead of using a traditional Wallace tree adder, compressor circuits such as 3:2 compressors or 4:2 compressors can be used for partial product addition. However, as there is a possibility of using the same compressor again and again for addition (like Wallace tree addition), the result would not be very effective. Keeping all these points in mind, the UCM architecture is designed as shown in Figure 4, where in the first stage, individual adders are used for adding partial products, and in the second, third, and final stage, the rectangles with three variables represent the full adder and the rectangles with two variables represent the half adder. The UCM architecture consists of three stages. Stage 1 and stage 3 remain the same for UCM architecture (compared to a Wallace tree). Whether it is partial product generation, addition of intermediate sums, or carry using fast adders, these can be chosen according to the requirements of

the designer. Hence, it is more important to replace the stage 2, i.e., the addition of partial products, which creates sum and carry separately.

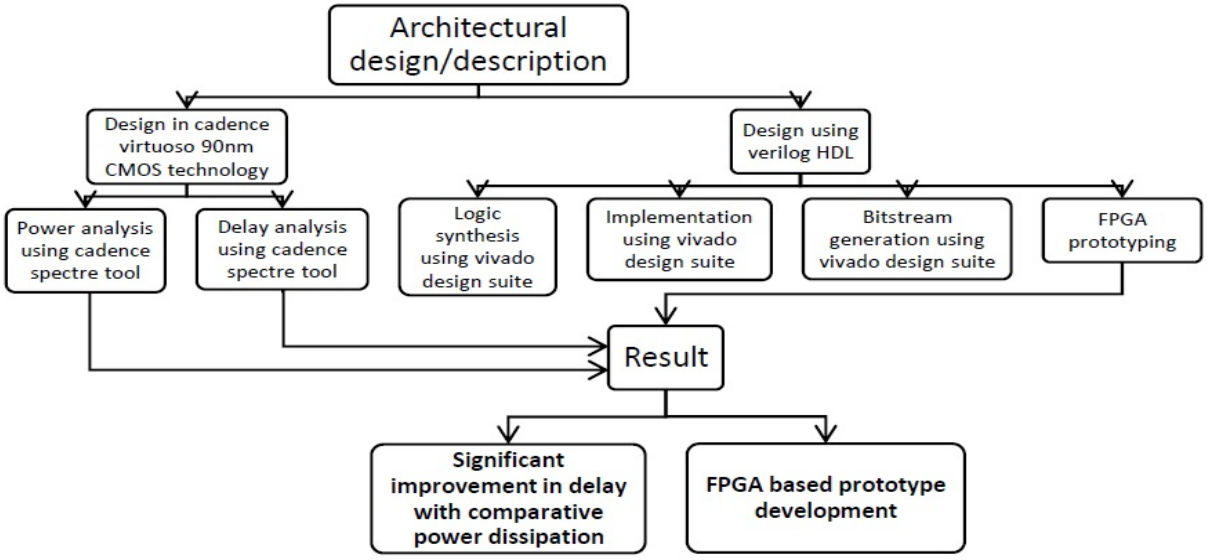

Figure 3. UCM architecture design process flow

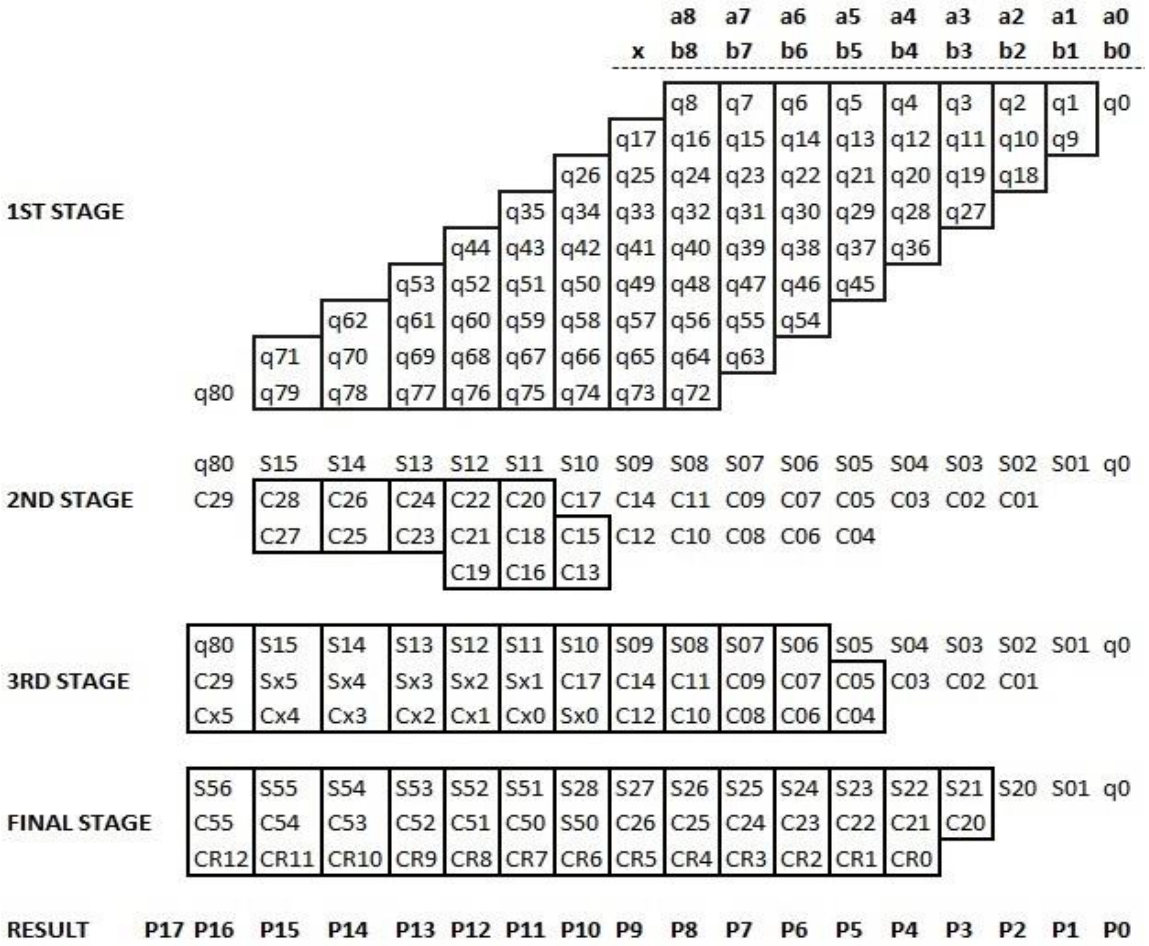

Figure 4. UCM architecture for 9x9 bit multiplication

4.1. Addition of Partial Products

While adding partial products, the partial products are aligned in such a way that the summation of bit location of multiplicand and multiplier are equal. The summation of bit location can be called the 'weight' of a particular partial product. For example, in Figure 4, 'q35', 'q43', 'q51', 'q59', 'q67', and 'q75' are aligned in a single column because the weight is eleven for all of the mentioned partial products, i.e., q35 = a8b3, q43 = a7b4, q51 = a6b5, etc. Thus, the summation of the bit location is either 8 + 3, 7 + 4, or 6 + 5, which is equal to 11 in all cases. Hence, for the addition of partial products, the alignment is very important. Once the partial products are aligned, the next step is to add all the partial product falling in that particular column. Firstly, the total number of stages and levels need to be identified. Each stage consists of an AND-XOR gate pair, and the total number of stages in one level is counted from top to bottom. The total number of stages in the first level is '*i*-1', where '*i*' is the total number of partial products to be added in a particular column.

On the other hand, the horizontal count of AND-XOR pair is the total number of levels required for the design. In a different angle, we can say that the total number of levels required in a design is the total number of AND-XOR pair required in the bottommost stages. Basically, it is the count of AND-XOR pairs from right to left. In each level, the total number of stages required will be decremented by one until it satisfies the formula: $2^n - 1 \geq i$.

Where '*i*' is the total number of the partial product to be added and '*n*' is the total number of levels required. '*i*' and '*n*' are integers starting from 1, 2, 3, ⋯, ∞. For example, to add three partial products in a column, the total number of levels will be: $2^n - 1 \geq 3$, so $n = 2$. Similarly, suppose $i = 8$, i.e., $2^n - 1 \geq 8$, so $n = 4$ and so on. The basic block diagram for K stages and L levels is shown in Figure 5. In Figure 5, A_0, A_1, A_2, up to A_K are the partial products; the term Y_0 is the sum, and Y_1, Y_2, Y_3, ⋯, Y_L are the carries. Therefore, in simple words, the algorithm shown in Figure 5 is a N-bit compressor circuit, which generates the sum of a particular column and single/multiple carries.

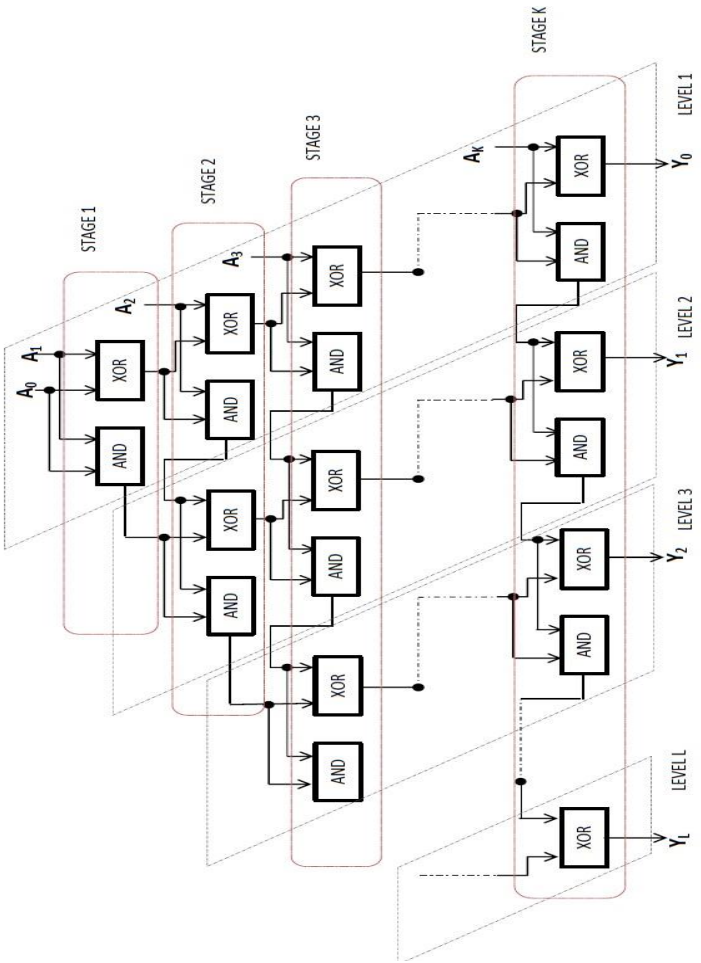

Figure 5. AND-XOR gate arrangement with K stages and L levels having A_0, A_1, A_2, ⋯, A_K partial products (with equal weights) for a particular column

4.2. Special Cases

- In the last level, instead of AND-XOR pair, only XOR gate is to be used.
- If $i = 2$, only one level is to be used to get the sum as well as carry. In this case, the output from the AND is the carry.
- For $i = 1$, the input itself is the output (sum), and there is no carry output.

It is very important to note that the output through the level 1 is the sum of the partial products present in a particular column and the outputs of rest of the levels, i.e., level 2 to level L are the corresponding carry bits. After getting the sum as well as carry bit of all columns, the next step is to add up the sum bits with the carry bit of the previous columns. Any of the efficient algorithms, such as the dada algorithm, Wallace tree algorithm, or even ripple carry adder, can be used as the number of rows is reduced substantially. A detailed design is shown in Figure 4.

5. FPGA Prototyping and Implementation

In order to compare the implemented UCM with the Wallace tree multiplier and the array multiplier, the multipliers are designed on Cadence Virtuoso 90nm technology as well as Verilog HDL (for implementing it on Nexys-4 Artix-7 FPGA board). The result shows that the UCM is much more efficient in supply voltage as low as 600mV for a 5-bit as well as a 9-bit multiplier. The reason for implementing the 5-bit and 9-bit multiplier is to show the complexity and accuracy of the algorithm (for which odd number of inputs are taken), instead of 4-bit and 8-bit. Due to lowering the supply voltage, not only is the speed of operation improved in comparison with the Wallace tree algorithm, but also the power consumption has dropped substantially. The tabular comparison of UCM and the Wallace tree multiplier for 5-bit and 9-bit is shown in Tables 1 and 2 respectively. As there is always a trade-off between power and delay, the average power consumption of the UCM is slightly higher than that of the Wallace tree multiplier. For example, the average power (a total of static as well as dynamic) consumption of the UCM at a 600mV supply voltage and 5×5 bit operation is 20.32 μW, whereas for the Wallace tree multiplier, the result is recorded as 19.54 μW. Similarly, at 900mV for a 9×9 bit operation, the average power consumption for implemented UCM is 355.8 μW, whereas for the Wallace tree multiplier it is 299.9 μW.

Table 1. Delay comparison (in nanoseconds) of UCM vs Wallace tree multiplier in various supply voltages for 5×5 bit operation

	V_{DD}			
Multiplier	**0.6V**	**0.7V**	**0.8V**	**0.9V**
UCM	2.769	2.701	2.664	2.641
Wallace tree	2.789	2.717	2.677	2.652
Array multiplier	Invalid outputs			

Table 2. Delay comparison (in nanoseconds) of UCM vs Wallace tree multiplier in various supply voltages for 9×9 bit operation

	V_{DD}			
Multiplier	**0.6 V**	**0.7 V**	**0.8 V**	**0.9 V**
UCM	2.281	2.21	2.171	2.147
Wallace tree	2.401	2.298	2.241	2.205
Array multiplier	Invalid outputs			

At the same time, there is a significant improvement of delay for the implemented UCM in comparison to the Wallace tree. The irregular structure of the Wallace tree algorithm is the main cause for the lagging in delay. As per the Elmore formula, the wire delay is proportional to the square of its length, i.e.:

$$\tau_d = (R \times C \times L^2)/2$$

Where *R*, *C*, and *L* are the wire resistance, capacitance, and length respectively. Hence, an irregular structure with an increased length of wire can affect the speed of operation of the circuit. On the other hand, the array multiplier could not produce any result in such low supply voltages (below 1.0V), and therefore its power and delay analysis could not be performed.

The graphical representation of the delay analysis of 5×5 bit as well as 9×9 bit multipliers is shown in Figures 6 and 7. It is clear from the graphical analysis that in the 5×5 bit as well as 9×9 bit multiplication operation, the implemented UCM takes less time to pass the signal from the input to the output (critical path). As the supply voltage drops further, the

difference between the delay values of UCM and Wallace tree multiplier is significant, and it is much clearer in the 9×9 bit multiplier. For example, at a 600mV supply voltage and 9×9 bit multiplication, the difference in delay between the Wallace tree and implemented UCM is 120 Pico seconds. On the other hand, for 5×5 bit multiplication, the difference in delay between the two is 20 Pico seconds. Hence it can be summarized that as the multiplier size increases ($n \times n$ bit), the delay of the implemented UCM is significantly lower than the Wallace tree multiplier at an ultra-low supply voltage (as low as 600mV). Similarly, the FPGA implementation of the UCM on Nexys-4 Artix-7 FPGA board is shown in Figure 8. The FPGA realization is performed for 5 bits as well as 9 bits. The switches along with buttons are used as the 18-bit inputs, and the LEDs are used as 18-bit outputs for verification of the implemented UCM. For 9-bit multiplier realization, 213 out of 63400 (approximately 0.33%) LUTs are used as LOGIC units. Meanwhile, 36 input-output buffers (IOB) are used, out of which 18 are input buffers and 18 are output buffers. On the other hand, for 5-bit multiplier realization, 42 (approximately 0.06%) LUTs are used as LOGIC units and 20 input output buffers (IOB) are used. The total on-chip power for 9-bit as well as 5-bit UCM implementation is 40.62 mW, with a junction temperature of 25.2℃.

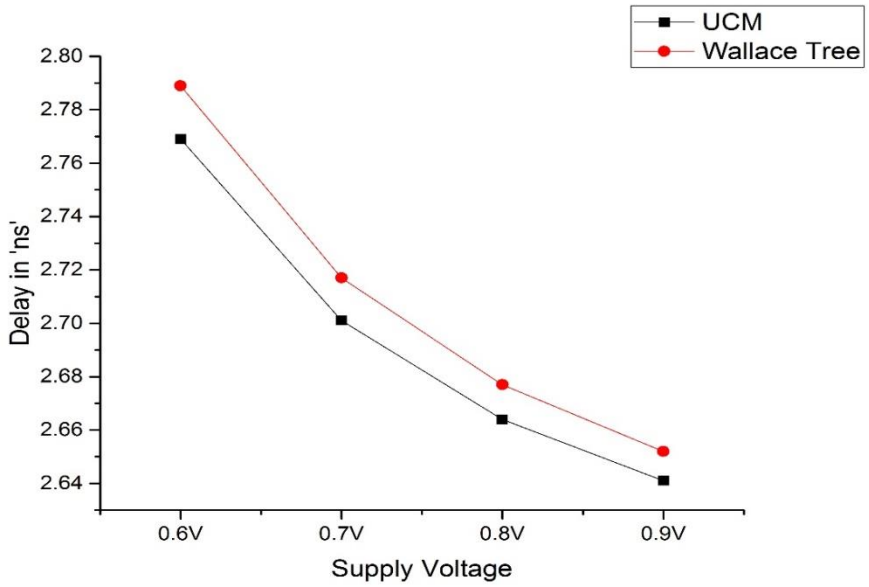

Figure 6. Graphical comparison of 5×5 bit UCM and 5××5 bit Wallace tree multiplier at voltages below 1V

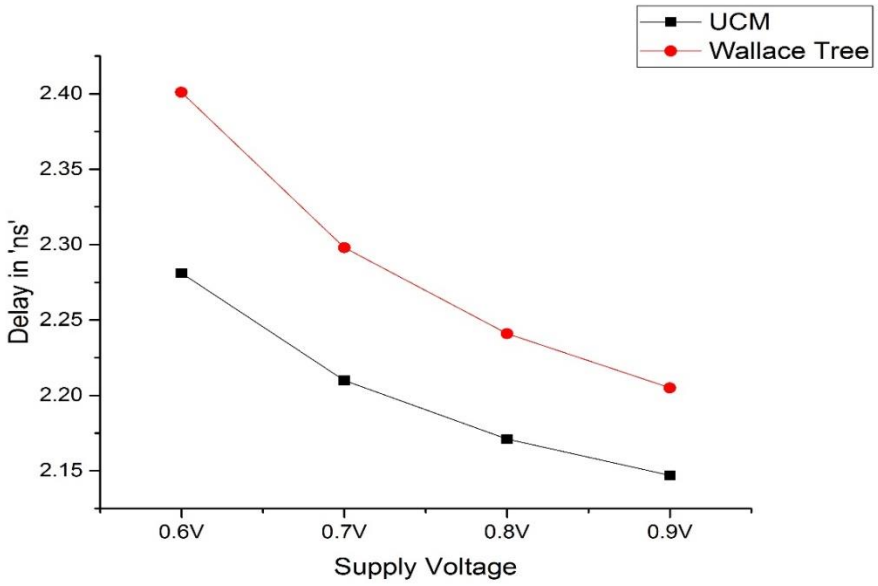

Figure 7. Graphical comparison of 9×9 bit UCM and 9×9 bit Wallace tree multiplier at voltages below 1V

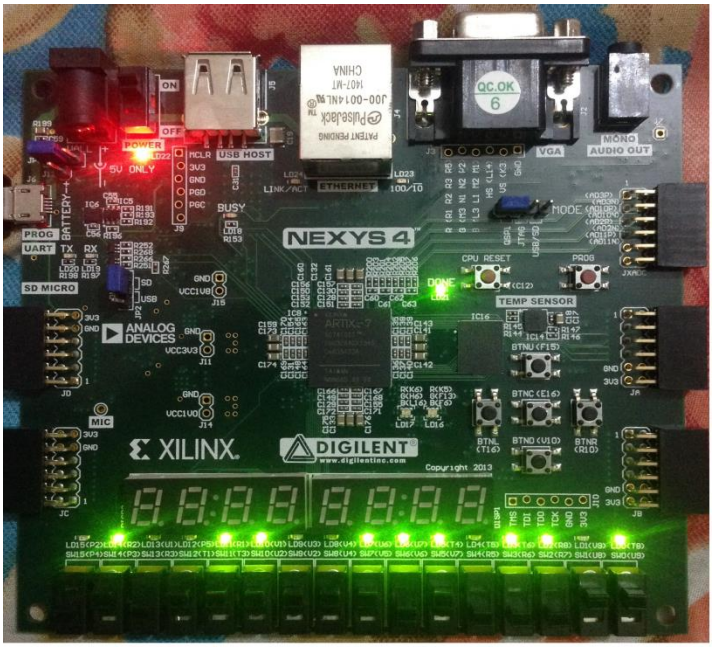

Figure 8. FPGA realization of the 9×9 UCM

6. Conclusions

This paper has proposed a high-speed multiplier based on a novel *N*-bit compressor algorithm, which is efficient in supply voltage as low as 600mV. The UCM can act as a base model for the implementation of high-end multiplier design. The main motive of the UCM design is to provide a high speed of operation at a low supply voltage. The overall design is implemented using GPDK90 technology library in Cadence Virtuoso. The delay and power analysis is carried out using Cadence Spectre tool. It is always a challenge to design a high-speed multiplier in low supply voltage, as the lowering of the supply voltage may degrade the output strength but reduce the power consumption significantly. In comparison to the Wallace tree multiplier, the UCM reduces the delay by 0.72% and 5% for the 5×5 bit and 9×9 bit operations respectively. This signifies that there could be further improvement in delay if the UCM is applied to higher order multiplication. The UCM architecture is suitable for the design of the Multiply and Accumulate (MAC) unit, as MAC operation high-speed multiplication is always desired.

References

1. M. Liao, C. Su, C. Chang, and A. C. Wu, "A Carry-Select-Adder Optimization Technique for High-Performance Booth-Encoded Wallace-Tree Multipliers," *IEEE International Symposium on Circuits and Systems*, ISCAS 2002, 2002
2. D. Guevorkian, A. Launiainen, V. Lappalainen, P. Liuha, and K. Punkka, "A Method for Designing High-Radix Multiplier-based Processing Units for Multimedia Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, No. 5, pp. 716-725, 2005
3. N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, "A 600-MHz 54 54-bit Multiplier with Rectangular-Styled Wallace Tree," *IEEE Journal of Solid-State Circuits*, Vol. 36, No. 2, pp. 249-257, 2001
4. K. B. Jaiswal, N. Kumar, P. Seshadri, and G. Lakshminarayanan, "Low Power Wallace Tree Multiplier using Modified Full Adder," in *Proceedings of the 3rd International Conference on Signal Processing*, *Communication and Networking* (*ICSCN*), 2015
5. I. Kataeva, H. Engseth, and A. Kidiyarova-Shevchenko, "Scalable Matrix Multiplication With Hybrid CMOS-RSFQ Digital Signal Processor," *IEEE Transactions on Applied Superconductivity*, Vol. 17, No. 2, pp. 486-489, 2007
6. S. Khan, S. Kakde, and Y. Suryawanshi, "VLSI Implementation of Reduced Complexity Wallace Multiplier using Energy Efficient CMOS Full Adder," in *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research*, 2013
7. R. D. Kshirsagar, E. V. Aishwarya, A. S. Vishwanath, and P. Jayakrishnan, "Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture," in *Proceedings of the International Conference on Communication and Green Computing Conservation of Energy* (*ICGCE*), 2013
8. T. Y. Kuo and J. S. Wang, "A Low-Voltage Latch-Adder based Tree Multiplier," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Seattle, WA, 2008
9. X. V. Luu, T. T. Hoang, T. T. Bui, and A. V. Dinh-Duc, "A High-Speed Unsigned 32-bit Multiplier based on Booth encoder and Wallace-tree Modifications," in *Proceedings of the International Conference on Advanced Technologies for*

Communications (*ATC'14*), 2014
10. M. Nachtigal, H. Thapliyal, and N. Ranganathan, "Design of a Reversible Single Precision Floating Point Multiplier based on Operand Decomposition," in *Proceedings of the 10th IEEE conference on Nanotechnology*, Kintex, Korea, 2010
11. T. Onomi, K. Yanagisawa, M. Seki, and K. Nakajima, "Phase-Mode Pipelined Parallel Multiplier," *IEEE Transactions on Applied Superconductivity*, Vol. 11, No. 1, pp. 541-544, 2001
12. C. Paradhasaradhi, M. Prashanthi, and N. Vivek, "Modified Wallace Tree Multiplier using Efficient Square-Root Carry Select Adder," in *Proceedings of the International Conference on Green Computing Communication and Electrical Engineering* (*ICGCCEE*), Coimbatore, 2014
13. M. J. Rao and S. Dubey, "A High Speed and Area Efficient Booth Recoded Wallace Tree Multiplier for fast Arithmetic Circuits," in *Proceedings of the Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics* (*PRIMEASIA*), BITS Pilani, Hyderabad, 2012
14. B. M. Reddy, H. N. Sheshagiri, B. R. Vijaykumar, and S. Shanthala, "Implementation of Low Power 8-bit Multiplier using Gate Diffusion Input Logic," in *Proceedings of the 17th IEEE International Conference on Computational Science and Engineering*, 2014
15. A. K. Singh, B. P. De, and S. Maity, "Design and Comparison of Multipliers using Different Logic Styles," *International Journal of Soft Computing and Engineering* (*IJSCE*), Vol. 2, No. 2, pp. 374-379, 2012
16. L. Sousa, "Algorithm for Modulo (2^n+1) Multiplication," *Electronics Letters*, pp. 752-754, May 2013
17. C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, pp. 14-17, 1964
18. Q. Yi and H. Jing, "An Improved Design Method for Multi-bits Reused Booth Multiplier," in *Proceedings of the 4th International Conference on Computer Science and Education*, 2009