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], Shruti Jain[c]

[a]*School of Electronics & 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 & computation systems, multiplication is one of the prime operations. A multiplier is a key component in any kind of digital systems such as Multiply-Accumulate (MAC) unit, various FFT algorithms etc. The efficiency of a multiplier is mainly dependent upon the speed of operation & 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 proved 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, digital communication etc. As the operation of the multiplier can be segregated into two categories namely partial product generator & final sum using adder circuits, multiplication process requires more hardware resources & processing time in comparison to basic adder/subtractor circuit. In a simplified view, a multiplier requires AND gates (for partial product generation) & adder circuits (half adders & full adders) for addition of partial products to yield the final result. The figure 1 shows the simplified operation of a multiplier. As per the literature, various multiplier algorithms/architectures are proposed in the past such as booth encoder, Wallace tree adder, array multiplier, modified booth multiplier (as mentioned by Liao et.al [8]) etc. All these algorithms/architectures use different approaches to make the multiplier operation more efficient. For example, booth multiplier or modified booth multipliers are 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 [8], in the case of 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.

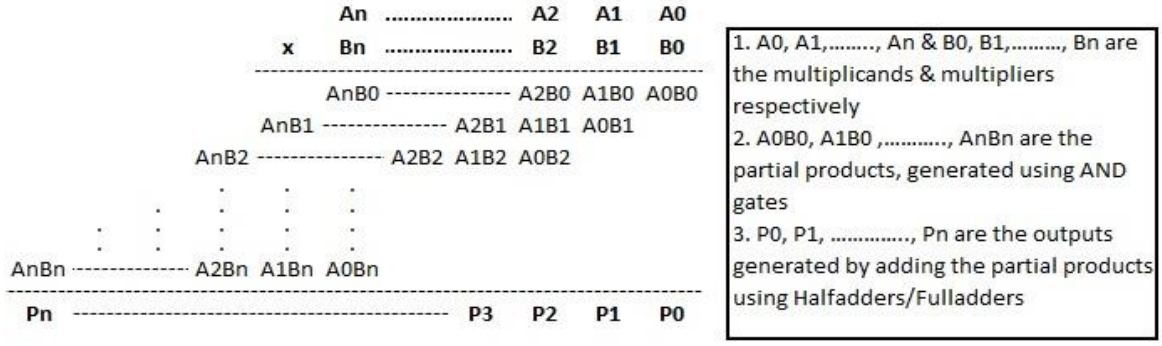

Figure 1. Basic multiplication operation

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

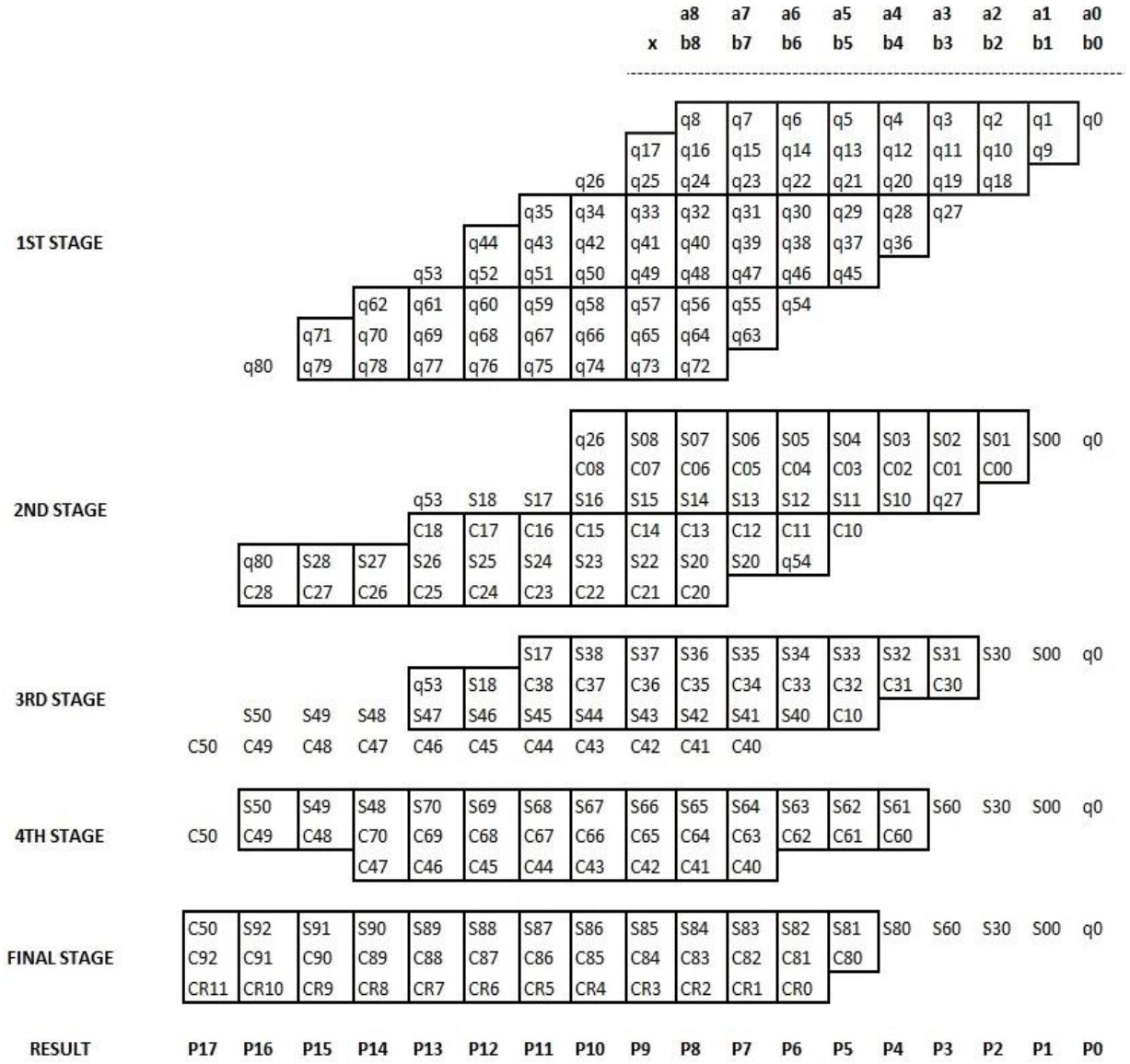

Figure 2. Wallace tree multiplier for 9 x 9 bit multiplication

There are various multiplier circuits explained in the literature which mainly focuses on the issues of power consumption, delay of the multiplier circuit & lesser area [1,2,3,4,5,6,7,9,10,11,12,13,14,15,16,17,18]. But as per studies, it is found that, area & 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 day by day the size of the transistor is decreasing, the area cannot become a major issue in today's digital systems. The power consumption & delay of a particular circuit depends upon the supply voltage (V_{DD}). A slight increment in the supply voltage increases the overall power consumption & at the same time, it decreases the delay of the circuit. Hence there is always a trade-off between power consumption & 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 is needed to be chosen so that the output logic is valid & the power consumption is bare minimum with a comparable delay value. As per the literature survey, it is found that most of the multiplier design uses Wallace tree multiplier as the basic algorithm & in majority of the cases, the basic Wallace tree multiplier algorithm has been modified to get better results [2,3,7,8,12,13]. The reason for the same is that the Wallace tree algorithm is the simplest way of designing multiplier 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 with the speed of the multiplier circuit. The study mainly focusses on the optimization on the partial product addition. The reason behind the same is that, for partial product generation, booth algorithm produces a better result than any other multiplication approaches. Secondly as discussed above, majority of the multipliers use Wallace tree adder for partial

product addition. Hence, an optimized & efficient partial product adder, which can replace the Wallace tree algorithm, can yield a better multiplier.

The following sections are discussed as follows: in section 2, Wallace tree multiplier architecture is discussed, in section 3, the novel architecture has been explained, in section 4, a detailed analysis of the UCM architecture is discussed along with its realization on FPGA board & in section 5, a detailed conclusion & future scopes of the UCM architecture is 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' & `carry' separately
Stage 3: a final adder which is generally a fast adder to add the sum & 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 conventional two input AND gate to find the partial product of each multiplicand & multiplier or by using advanced booth multiplier to reduce the total number of partial products. With the help of 2nd order booth algorithm, the number of the partial product is reduced to half (approx.) the bit width of the multiplier [2].

In stage 2, the partial products are added using half adder/full adder. To achieve the same, 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 & 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 & carry output of the full adder/half adder of the previous level along with the remaining partial products. The same procedure is followed until & unless 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 the operation of Wallace tree multiplier algorithm in detail, where a0-a8 & b0-b8 are representing the multiplicand & multiplier respectively; q0-q80 is representing the partial products; S_{xx} & C_{xx} are representing the sum & carry outputs of half adder/full adder respectively & CR_x is representing the ripple carry at the final stage. Moreover, as shown in the figure 2, the rectangles with 3 variables represent full adder & the rectangles with 2 variables represent 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 the 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 & delay analysis are carried out from virtuoso-based design whereas Verilog HDL program is used for FPGA prototyping.

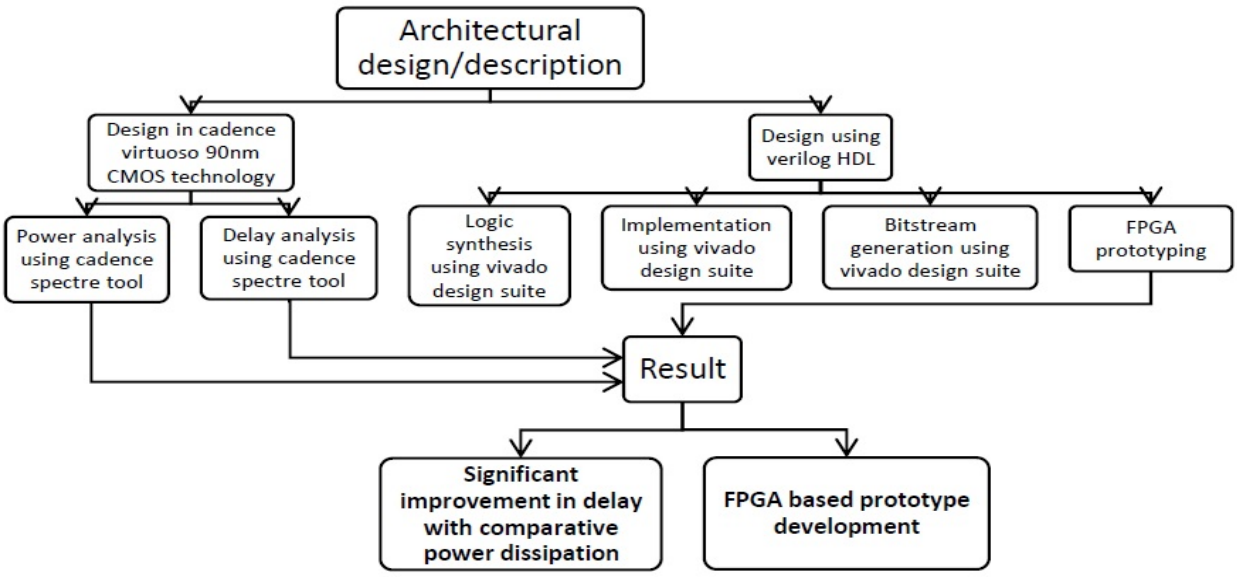

Figure 3. UCM architecture design process flow

4. UCM architecture

Although the Wallace tree multiplier is much faster than array multiplier [3], it requires a large number of adders. Secondly, Wallace tree multiplier is highly irregular & complicated. So, in order to overcome the irregular structure, several modified Wallace tree algorithms are proposed in the literature [2,3,5,6,7,8,9,12,13,14]. All these multiplier algorithms are based upon Wallace tree. Hence replacing the Wallace tree algorithm may further improve the result of the multiplier. Another important point here is, instead of using traditional Wallace tree adder, compressor circuits such as 3:2 compressors or 4:2 compressors etc. can be used for partial product addition. But as there is a possibility of using the same compressor again & again for doing addition (same like Wallace tree addition), the same wouldn't be much effective. Keeping all these points in mind, the UCM architecture is designed as shown in figure 4, where in the 1st stage, individual adders are used for adding partial products & in 2nd, 3rd & final stage, the rectangles with 3 variables represent full adder & the rectangles with 2 variables represent half adder. The UCM architecture consists of three stages. The stage 1 & stage 3 remains the same for UCM architecture (as that of Wallace tree), because whether it is partial product generation or addition of intermediate sum or carry using fast adder, these can be chosen according to the requirement of the designer. Hence, it is more important to replace the stage 2 i.e. addition of partial product which creates sum & carry separately.

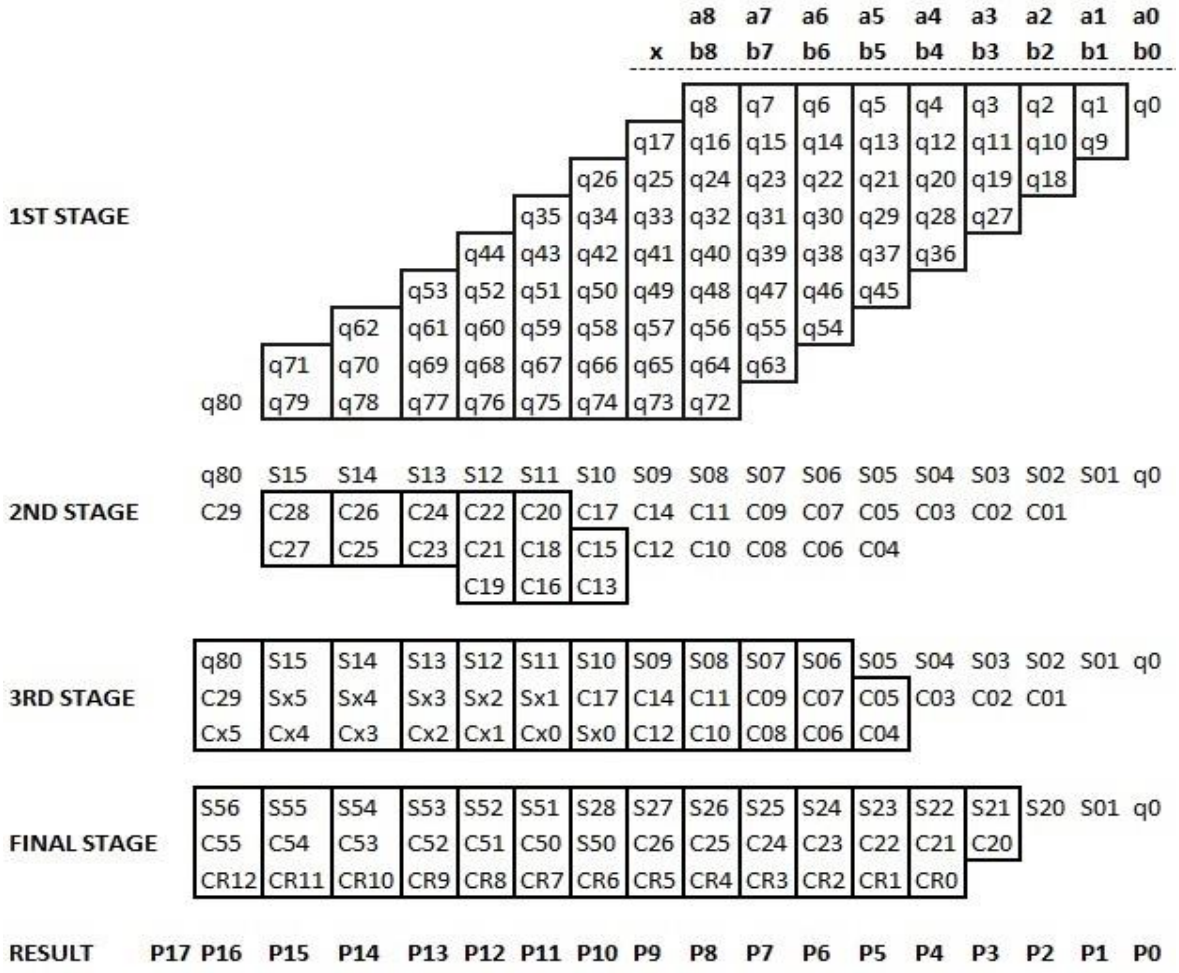

Figure 4. UCM architecture for 9 x 9 bit multiplication

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 & multiplier are equal. The summation of bit location can be called as `weight' of a particular partial product. For example, in the figure 4, `q35', `q43', `q51', `q59', `q67' & `q75' are aligned in a single column because of the reason that the weight is eleven for all of the mentioned partial products, i.e. q35=a8b3, q43=a7b4, q51=a6b5 etc. So, the summation of the bit location is either of 8+3 or 7+4 or 6+5, which is in all cases are equal to 11. Hence, for the addition of partial products, its 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. For adding a particular column firstly, the total number of stages & levels need to be identified. Each stage consists of an AND-XOR gate pair & 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.

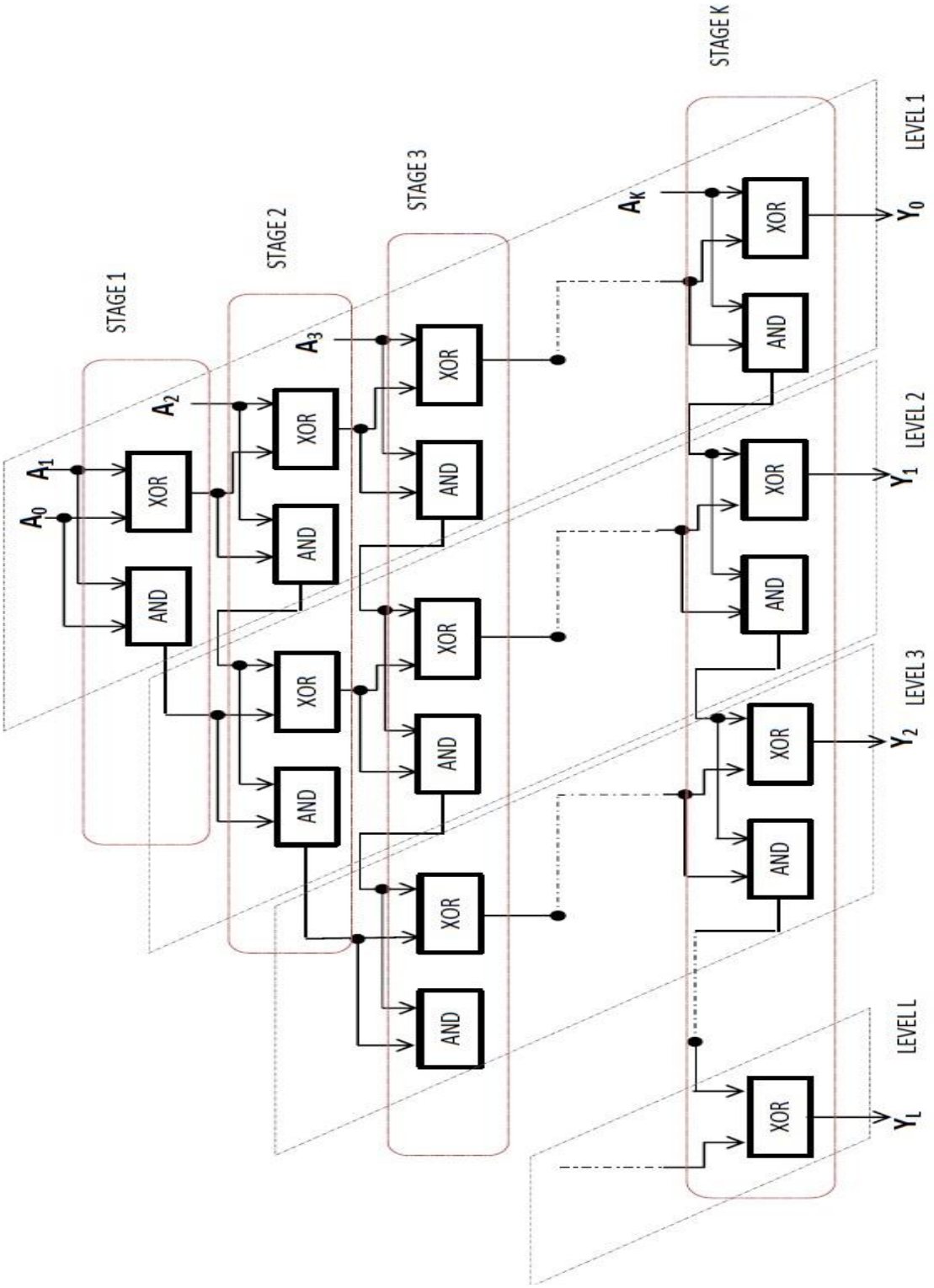

Figure 5. AND-XOR gate arrangement with K stages & L levels having A_0, A_1, A_2,....., A_K partial products (with equal weights) for a particular column

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 bottom most stages. Basically, it is the count of AND-XOR pair 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 & `n' is the total number of levels required. `i' & `n' are integers starting from 1, 2, 3, ..., ∞. For example, for adding 3 partial products in a column, the total number of levels will be: $2^n-1 \geq 3$, so n=2. Similarly, if suppose i=8, i.e. $2^n-1 \geq 8$, so n=4 & so on. The basic block diagram for K stages & 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 & Y_1, Y_2, Y_3,....., Y_L are the carries. Therefore, in simple words, the algorithm shown in the figure 5 is a N-bit compressor circuit which generates sum of a particular column & single/multiple carries.

4.2. Special cases

i. In the last level, instead of AND-XOR pair, only XOR gate is to be used.
ii. 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.
iii. For i=1, the input itself is the output (sum) & 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 & 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. For this any of the efficient algorithms such as dada algorithm, Wallace tree algorithm or even ripple carry adder can be used as the number of rows has reduced substantially. A detailed design is shown in figure 4.

5. FPGA prototyping & implementation

In order to compare the implemented UCM with Wallace tree multiplier & 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 5-bit as well as a 9-bit multiplier. The reason for implementing the 5-bit & 9-bit multiplier is to show the complexity & accuracy of the algorithm (for which odd number of inputs are taken), instead of 4-bit & 8-bit. Due to lowering the supply voltage not only the speed of operation is improved in comparison with the Wallace tree algorithm but the power consumption has dropped substantially. The tabular comparison of UCM & Wallace tree multiplier for 5 bit & 9 bit is shown in table 1 & table 2 respectively. As there is always a trade-off between power & delay, the average power consumption of the UCM is slightly higher than the Wallace tree multiplier. For example, the average power (a total of static as well as dynamic) consumption of the UCM at 600mV supply voltage & for 5 x 5-bit operation is 20.32uW whereas for Wallace tree multiplier, the same is recorded as 19.54uW. Similarly, at 900mV & for 9 x 9-bit operation, the average power consumption for implemented UCM is 355.8uW whereas for Wallace tree multiplier it is 299.9uW.

Table 1. Delay comparison (in nano second) of UCM Vs Wallace tree multiplier in various supply voltages for 5x5 bit operation

	V_{DD}			
Multiplier	**0.6 V**	**0.7 V**	**0.8 V**	**0.9 V**
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 nano second) of UCM Vs Wallace tree multiplier in various supply voltages for 9x9 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 Wallace tree. The irregular structure of Wallace tree algorithm is the main cause for the lagging in delay. As per Elmore formula, the wire delay is proportional to the square of its length, i.e.

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

Where R,C & L are the wire resistance, capacitance & length respectively. Hence with an irregular structure with 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) due to which its power & delay analysis could not be performed.

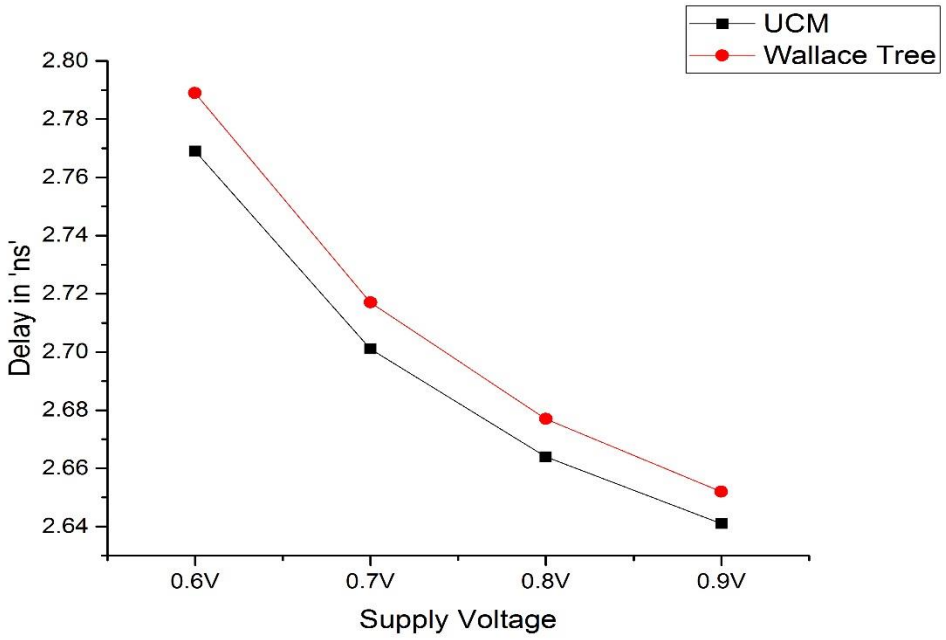

Figure 6. Graphical comparison of 5x5 bit UCM & 5x5 bit Wallace tree multiplier at voltages below 1V

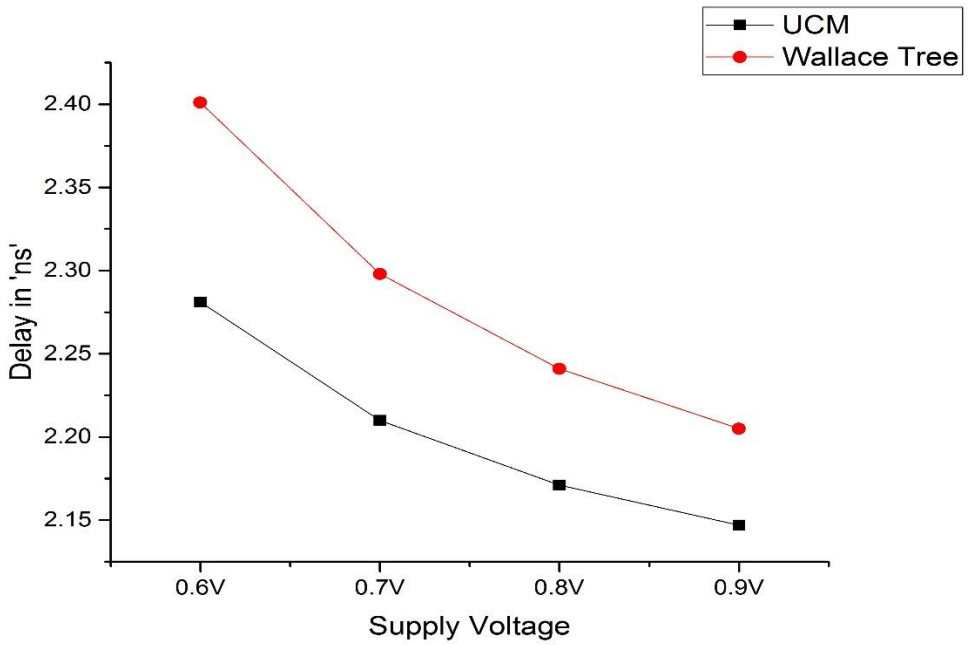

Figure 7. Graphical comparison of 9x9 bit UCM & 9x9 bit Wallace tree multiplier at voltages below 1V

The graphical representation of the delay analysis of 5x5 bit as well as 9x9 bit multipliers is shown in figure 6 & figure 7. It is clear from the graphical analysis that in 5x5 bit as well as 9x9 bit multiplication operation, the implemented UCM takes lesser time to pass the signal from input to the output (critical path). As the supply voltage drops further, the difference between the delay values of UCM & Wallace tree multiplier is significant & it is much clear in 9x9 bit multiplier. For example, at 600mV supply voltage & 9x9 bit multiplication, the difference in delay between Wallace tree & implemented UCM is 120

Pico seconds on the other hand, for 5x5 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 x n bit) the delay of the implemented UCM is significantly low than the Wallace tree multiplier at 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 done for 5 bits as well as 9 bits. The switches along with buttons are used as the 18-bit inputs & 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; whereas 36 input-output buffers (IOB) are used out of which 18 are input buffers & 18 are output buffers. On the other hand, for 5-bit multiplier realization, 42 (approximately 0.06%) LUTs are used as LOGIC units & 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 junction temperature as 25.2^{o} C.

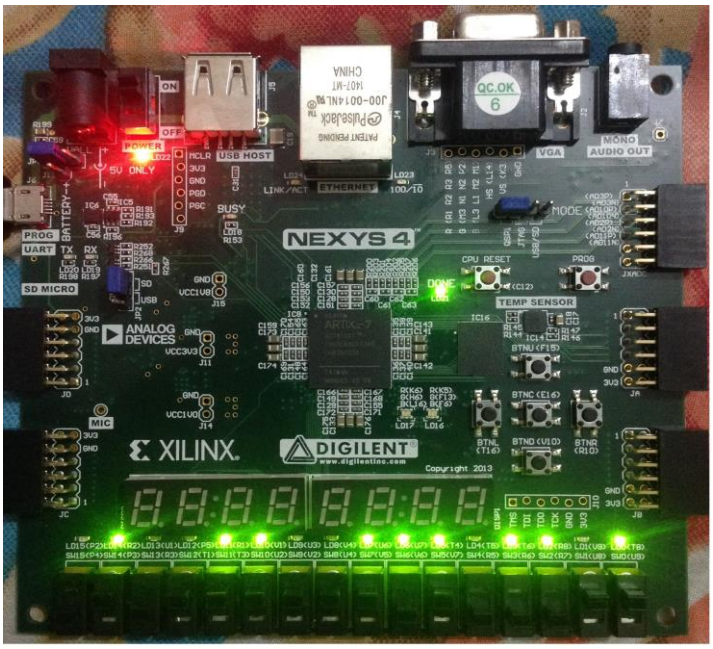

Figure 8. FPGA realization of the 9x9 UCM

6. Conclusion

This paper has proposed a high-speed multiplier based on a novel N-bit compressor algorithm, which is much efficient in supply voltage as low as 600mV. The UCM can act as a base model for 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 whole design is implemented using GPDK90 technology library in Cadence Virtuoso. The delay & power analysis is carried out using Cadence Spectre tool. It is always a challenge for the designers to design a high-speed multiplier in low supply voltage, as the lowering of the supply voltage may degrade the output strength but it reduces the power consumption significantly. In comparison to Wallace tree multiplier, the UCM has reduced the delay by 0.72% & 5% for 5X5 bit & 9X9 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 will be much suitable for the design of Multiply & Accumulate (MAC) unit, as for a MAC operation high-speed multiplication is always desired.

References

1. 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.
2. 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.
3. K. B. Jaiswal, N. Kumar, P. Seshadri, and L. G, "Low Power Wallace Tree Multiplier Using Modified Full Adder," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2015.
4. 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.

5. 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.
6. 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)*, Chennai, 2013.
7. 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.
8. 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.
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 & Electronics (PRIMEASIA)*, BITS Pilani, Hyderabad, 2012.
14. B. M. Reddy, H. N. Sheshagiri, B. R. Vijaykumar, and S. S., "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," *Electronlcs Letters*, pp. 752-754, 01 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 & Education*, 2009.