Design of a Parallel Adder Circuit for a Heavy Computing Environment and the Performance Analysis of Multiplication Algorithm

Jayanta Kumar Das
Applied Statistics Unit
Indian Statistical Institute
Kolkata-700108, India
dasjayantakumar89@gmail.com

Pabitra Pal Choudhury
Applied Statistics Unit
Indian Statistical Institute
Kolkata-700108, India
pabitrapalchoudhury@gmail.com

Sudhakar Sahoo
Computer Science Department
Institute of Mathematics and Applications
Bhubaneswar-751029, India
sudhakar.sahoo@gmail.com

***Abstract*—Firstly, this study proposed a new parallel adder circuit model in Carry Value Transformation (CVT)-Exclusive OR (XOR) paradigm. Secondly, an efficient multiplication algorithm is discussed along with its performance analysis on various inputs selection. Our design of proposed model for the addition of many integer pairs using parallel Cellular Automata Machines (CAMs) can perform the addition in a much better way with setting a preprocessing testing logic in it. CVT and XOR operations together can do the efficient addition of two non-negative integers for any bulk inputs using CAM. Multiplication is the repetitive addition process, which could be designed using recursive use of CAM. Our analysis up to 10 bits selection of all integer pairs suggest that the recursive use of CAM for multiplication becomes much faster in real life scenario for any types of inputs. Further exponential operation is highly needed for various fields of computer science which is also described in this paradigm.**

***Keywords*-Carry Value Transformation (CVT) and Exclusive OR (XOR) Operations, Adder Circuit, Multiplication, Performance Analysis.**

I. Introduction

In any bulk computational system a parallel architecture is more suitable than pipeline system to achieve the high speed. As parallel architecture consumes more area, cost is highly increasing. But in Nano technology system where less area is of prime importance, pipeline architecture is much better even if its speed is far less than the parallel architecture. An alternative route could be faster recursive circuit design which is our main focus of this study.

Various architectures are designed in VLSI for the arithmetic computation both in parallel as well as pipelined system, and also using recursive process [1-9]. Carry Value Transformations (CVT) (Special case of Integral Value Transformation (IVT) [10]) and Exclusive OR (XOR) are the two most important transformations operating on bits of strings [11, 12]. For the large scale cellular automata (CA) experiments, Cellular Automata Machines (CAMs) become very special compared to any kind of computing machine [13] and various CAMs are now available for all the research communities. Using CAM an adder circuit is proposed which can do better on using testing logic gate embedded in it [14]. Self time recursive parallel circuit [4] is designed based on CAM [14], can do better performance for the addition of two integers. This is because the CAM used here operates on clock cycle only (without any gate delays).

Some well-known multiplication algorithms [15-22] were discussed based on area and speed. In most of the cases the complexity is dependent on time delay to produce the final result. There are many real time systems like video decoder, encoder, and matrix multiplication etc. where system needs to work with high speeds. It can be seen that a single algorithm cannot satisfy all the criteria [9]. Appropriate algorithm has to be chosen based on the application. Further, exponentiation operation is used extensively in many fields including economics, biology, chemistry, physics, and computer science, with applications such as compound interest, population growth, chemical reaction kinetics, wave behavior, and public-key cryptography.

Various applications area in CVT-XOR paradigm are fractal formation [11, 12], both chaotic and regular pattern formation [12, 23], basis for development of theory like IVTs, multi-number CVT-XOR arithmetic operations [24] etc. Previously it has been shown that addition of two non-negative integers (Say X and Y) is exactly equal to their CVT and XOR operations sum i.e. X+Y=CVT (X, Y) + XOR (X, Y). And if recursively CVT and XOR operations are performed on two non-negative integers, then the maximum number of steps to get CVT=0 is $n + 1$ where n= MAX (X, Y) number of bits in binary [12]. It may be noted that the CVT-XOR transformation is better known as Carry-Save addition: the CVT and XOR values denote the carry vector and the pseudo-sum vector respectively [25]. The idea of delaying carry resolution is due to J. von Neumann and the number of iterations leading to the addition result is same as the longest carry chain [26]. So given any random integer pair, number of iterations leading to CVT=0 does not depend on the number of binary bits required to represent the pair. Rather it is interdependent on the bits comprising the pairs of various combinations.

For example, we take pairs (10, 5) and (1, 7) which are

978-1-5090-1560-3/17 $31.00 © 2017 IEEE
DOI 10.1109/IACC.2017.108

taking 1 and 4 iterations respectively to get CVT=0. So in this paradigm we have to fit a testing logic when CVT becomes zero. Further using parallel CAMs we can improve the time complexity (with regards to number of iterations) where we need some preprocessing task rather than testing. So parallel CAMs designed in this paradigm is also another agenda of this paper.

The remaining part of the manuscript is organized as follows: Section II proposed a new model of parallel CAMs based on preprocessing task and its performance analysis. In Section III, an efficient multiplication algorithm in CVT-XOR paradigm is discussed. The performance analysis of CAM circuit towards multiplication for the various integer pairs up to m=10 bit is shown in this section. Lastly, in Section IV we conclude the paper along with future research direction.

II. PROPOSED PARALLEL CAM MODEL AND ITS PERFORMANCE ANALYSIS

Previously a parallel CAM model is proposed for the addition of 8 integer pairs in parallel [24]. The designed model is set with 4 levels of CAMs. For n bit integer pairs worst case time complexity is $(n+1)+(n+2)+(n+3)+(n+4)$ i.e. $(4\times n)+10$. But the proposed model produce the final output as sum of many integers pairs computing their addition in parallel and propagation manner.

Here, we propose another model of parallel CAM designed with some different technique where our target is to give the output of addition of many integer pairs separately. We have seen in Section I that all the integer pairs do not require equal number of iterations. We have designed the CAM (shown in Fig. 1) in serially where first CAM (CAM-1) will perform the addition for those integer pairs which are taking single iteration (class-1), second CAM (CAM-2) will perform the addition for those integers pairs which are taking two iterations (Class-2) and so on. This is other than the testing logic used for each CAM. If all the integer pairs are inputted in the parallel CAM then all the outputs (addition results) will be obtained after n unit (n iteration) of time which is same as the time of CAM-n (assuming Class-n is non empty). Otherwise, the addition result will be obtained after k unit of time where k is the maximum class level for which the class is non-empty and all other classes from Class-$k+1$ to Class-n are empty. Clearly $1 \leq k \leq n$.

Performance Analysis of Parallel Model:

For an example say, we are trying to compute on using 16 CAMs and CAMs are executed for 16 units of time. During this time CAM-1 can perform the addition for maximum of 16 integer pairs as it takes 1 unit time for each pair, CAM-2 can perform the addition of 8 integer pairs and so on. Finally we obtain from first 8 CAMs 16+8+5+4+3+2+2+2 and last 8 CAMs can do only 1 integer pair each. Thus, total of 50 integer pairs addition are possible. Further, each CAM is doing the operation for next inputted integer pairs except few CAMs (CAM-1, CAM-2, CAM-4 and CAM-8). But using single CAM in serial we can do only 16 integer pairs addition in 16 units of time. Now if we increase the unit time to 32, we are able to do 100 integer pairs addition. Therefore with the increase of unit, more numbers of additions can be obtained. In this implementation, care has to be taken only for the initial selection of integer pairs based on their iteration numbers leading to zero.

III. MULTIPLICATION ALGORITHM USING CAM AND ITS PERFORMANCE ANALYSIS

In this section, a multiplication algorithm is proposed using the adder function and then the performance analysis of multiplication algorithm of two positive integers using recursive use of CAM is done up to m=10 bit for both distinct and random integer pairs.

Multiplication Algorithm:

Step 1: Take input of two positive integers M1 (multiplier) and M2 (multiplicand) of size n and m bit respectively.

Step 2: Set the size for M1 as $2 \times m + 1$ bits by padding the zeros in MSB and take two registers R1 and R2 of same size initially set to all zeros.

Step 3: Set a pointer arrow p to LSB of M2

$for\ i = 1 : m - 1$

if LSB of M2 is 1

Copy the M1 into R1 slots

$else$

Set R1 slots as all zeros

end

Do the CVT and XOR operations on R1 and R2 respectively to get their sum on R2.

Left shift of M1 by padding 0 on LSB and set pointer arrow p for pointing to next left bit.

end

Step 4: Get the result, the multiplication of M1 by M2 from R2 register.

Illustration of Multiplication Algorithm: Multiplication of two non-negative integers can be seen as repetitive addition of one integer by another. Here we have used left shift technique and each left shift number becomes twice by padding zeros in the right most position. Towards left from the LSB position of M2, each time in multiplicand if bit 1 is found, we perform the left shift operation and add with the previous result using CAM but if 0 is found we perform the left shift of M1 and set to the CVT part (R1) all zeros. So, after CAM operation previous result in XOR part (R2) is unchanged. Here, we are performing the repetitive addition carefully on using CAM circuit. For Example, take two 4 bits integers, one Multiplier =1101 (13) and one Multiplicand = 1011 (11). Copy the Multiplier into M1 of size = $2 \times 4 + 1 = 9$ slots with 0's padded in MSB position and keep the Multiplicand in a 4 bit slot where

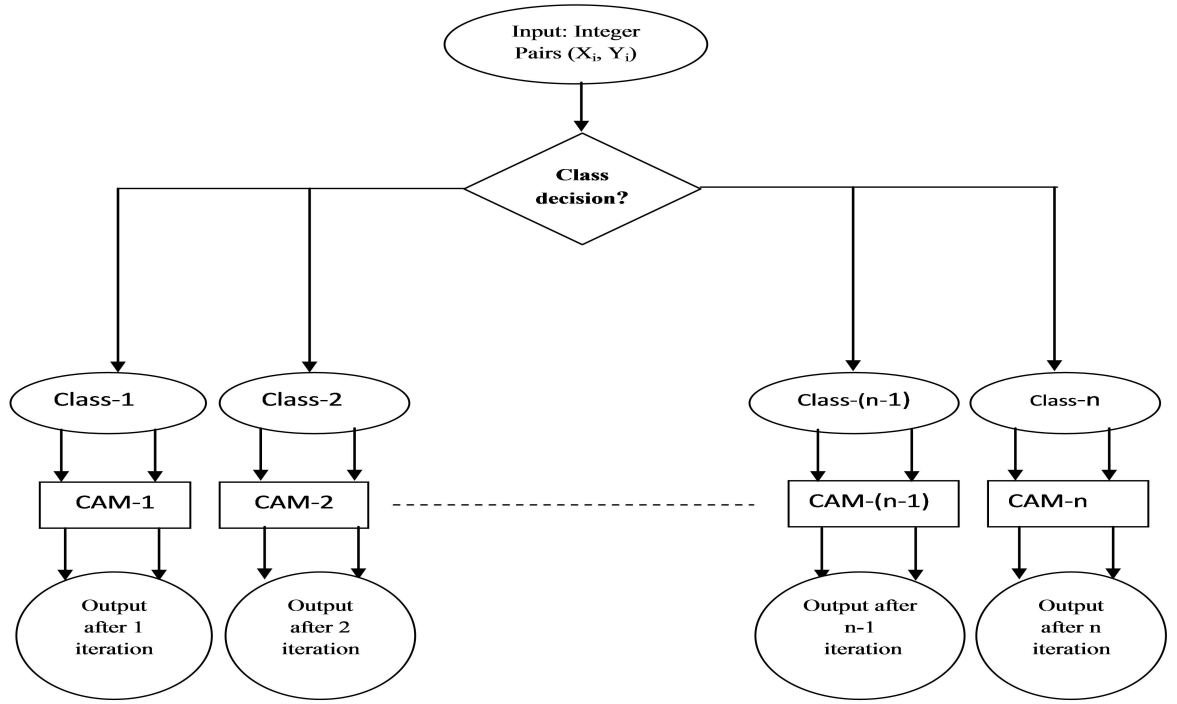

Figure 1 Proposed model of Parallel CAMs

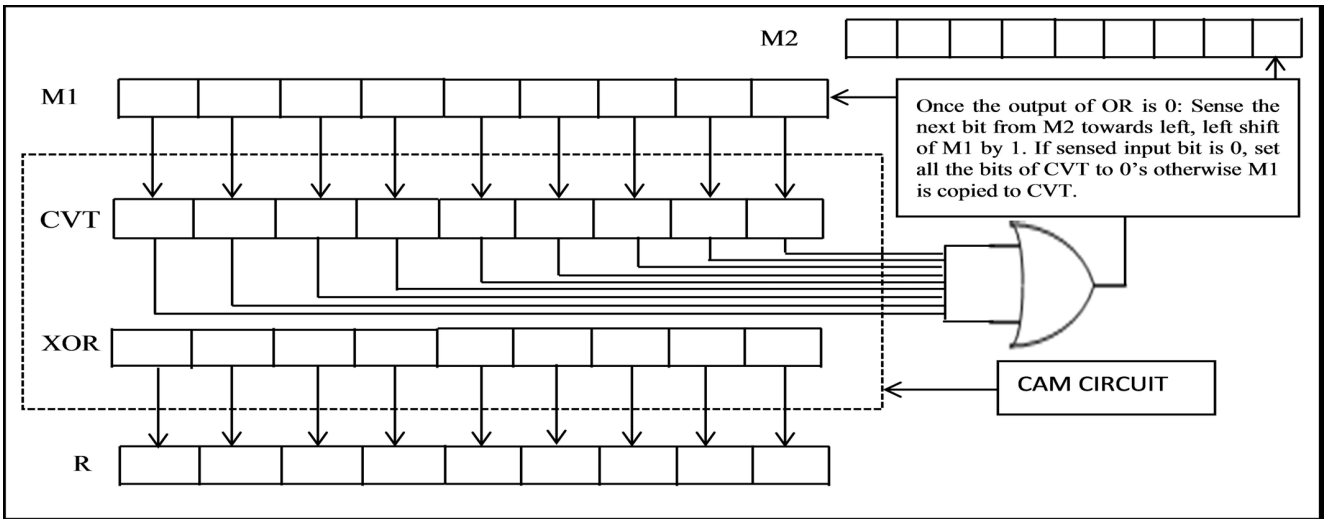

Figure 2 Proposed Block Diagram of Multiplication Algorithm for VLSI Implementation

READ head (pointer arrow) is pointing to the LSB position which in this case is 1. At $1st$ iteration, M1 is copied to R1. Then CAM is operated on R1 and R2 to get their sum which is stored in R2. In $2nd$ iteration, arrow pointing to M2 is shifted to left next bit position which is again 1. Left shift is performed on M1 with 0 padded in LSB and then M1 is copied to R1, then again CAM is operated on R1 and R2 (result of previous R1 and R2). Then in $3rd$ iteration, read head pointing to M2 which is 0, in this case M1 is left shifted, but R1 is set to all zeros and performed similar operation like $1st$ and $2nd$ iteration. After $4th$ iteration, the final result is stored in R2 which is the multiplication of M1 by M2. A proposed block diagram of the above Algorithm is shown in Fig. 2 where CVT and XOR are R1 and R2

respectively and R is holding the final result which is copied from R2.

Performance Analysis of Multiplication Algorithm:

Previously it has been seen that maximum number of iterations leading to either CVT=0 or XOR=0 is $n+1$ [11]. But in practical scenario, it is found that for many integer pairs to get either CVT=0 or XOR=0 is significantly less than the value of n. In the above multiplication, we have used the recursive CAM for multiplication algorithm to get intermediate addition result. So, the fastest multiplication can be achieved in this paradigm. For an example, m=3 bit integers, the multiplicand and multiplier bits are equal. As per the design of our algorithm, the pairs which are converging to CVT=0 in one iteration, two iterations, three iterations will take 1×3 iterations, 2×3 iterations, 3×3 iterations respectively for their multiplication and corresponding BAR graph is shown in Fig. 3. Therefore average iteration for the multiplication of all 3 bits distinct integer pairs is $((27\times3)+(21\times6)+(12\times9)+(4\times12))/64 = 5.67$.

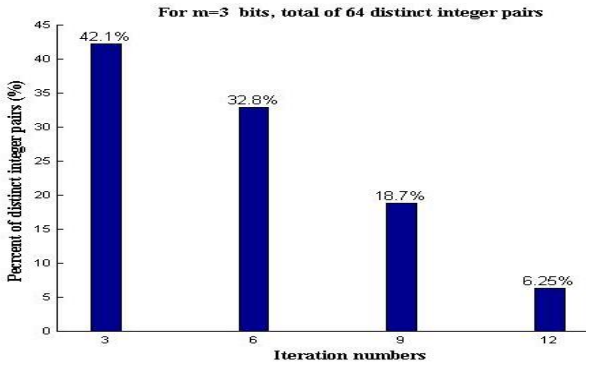

Figure 3 Graph showing the number of integer pairs in percentage wise (m=3 bit) to get the final multiplication result.

Similarly, we have generated up to m=10 bits for all the distinct integer pairs and corresponding iteration numbers to get their multiplication result in percentage wise is shown in TABLE I. Based on our analysis up to m=10, it can be easily seen that with the increase of bit number from m=2 to m=10 corresponding integer pairs take lesser number of iterations. Most of the cases more than 80% of integer pairs whose multiplications can be achieved in less than $(m\times m)/2$ iterations.

Further, random selection of integer pairs shows that iteration number leading to CVT=0 decrease significantly (randomly selecting number of integer pairs equal to all possible (2^m numbers) distinct of integer pairs for every m as shown in TABLE II. For random selection, we have used randi function in MATLAB version 2013a). Therefore corresponding multiplication result can also be achieved with lesser number of iterations. In Fig. 4 and Fig. 5 the average cases of iterations number for all integer pairs $(2 \leq m \leq 10)$ (verification by a computer program) with the bit number up to m=10 bit are shown, we found that the average iteration leading to multiplication result is under 35 and 25 iterations for all distinct and randomly selected integer pairs respectively. So, instead of getting multiplication result in $m\times m$ steps (worst case complexity), we can obtain the result significantly in fewer steps on the average for any bulk inputs scenario. But, on the other hand, it is increasing circuit overhead. From Fig. 2, it is quite clear that one will need n-input OR gate to implement the logic, where n is the multiplicand bits. Therefore transistors count will be huge while implementing it in VLSI. This cannot be allowed in efficient and reliable system designing. Parallel multiplier similar to Fig. 1 may be designed to tackle this problem.

Exponentiation is a mathematical operation, written as b^n, involving two numbers: the base b and the exponent n. When n is a positive integer, exponentiation corresponds to repeated multiplication of the base; that is b^n is the product of b multiplying n times. The multiplication algorithm can be used for the exponential operation to reduce time complexity in this paradigm.

IV. Discussion and Conclusion

We have thoroughly discussed the proposed parallel model for the addition of integer pairs. It is observed that we are able to reduce the significant number of steps to get addition result for bulk inputs. Based on this model one can easily design VLSI circuit. We have thoroughly discussed that the multiplication result is dependent on the additive property of CVT and XOR operations. On using convergence behaviour of integer pair, we have provided a simple multiplication algorithm. It is found that with the increase of bit numbers, success story (more numbers of pairs reaching the convergence in smaller iterations) towards the convergence behavior for getting CVT = 0 become more prominent, thereby observed multiplication result to be more and more effective with fewer steps. Further with the selection of random integer pairs, iteration numbers leading to get multiplication result can be reduced significantly. So in both the cases distinct as well as random selection of integer pairs, we can achieve the multiplication result in a faster time space for bulk inputs.

We have used the MATLAB software for the computation of various results which is very simple and provides some mathematical hypothesis. In future the research can be directed taking one real circuit and demonstrate its performance using standard EDA tool like cadence or synopsys. Then only our proposed algorithm can be compared with multiplication methods and circuits available with respect to their computation time, area overhead, power consumption etc. Although the circuit has been tested only up to 10 bits integers, it could be better for the circuits if the additional validation using larger data set can be provided.

TABLE I PERCENTAGE OF NUMBER OF ALL DISTINCT INTEGER PAIRS FOR m=2 to m=10 BIT WITH REGARDS TO THEIR ITERATION NUMBER TO GET MULTIPLICATION RESULT

Bit(m)	Percentage of Distinct Integer Pairs (Iteration Number)										
2	56.2% (2)	31.2% (4)	12.5%(6)								
3	42.2% (3)	32.8% (6)	18.7% (9)	6.25% (12)							
4	31.6% (4)	32.4% (8)	23.4% (12)	9.37% (16)	3.12% (20)						
5	23.7% (5)	30.9% (10)	26.5% (15)	12.5% (20)	4.68% (25)	1.56% (30)					
6	17.7% (6)	28.8% (12)	28.7% (18)	15.2% (24)	6.25% (30)	2.34% (36)	0.78% (42)				
7	13.3% (7)	26.4% (14)	30.0% (21)	17.5% (28)	7.81% (35)	3.12% (42)	1.17% (49)	0.39% (56)			
8	10.0% (8)	23.9% (16)	30.8% (24)	19.6% (32)	9.27% (40)	3.90% (48)	1.56% (56)	0.58% (64)	0.19% (72)		
9	7.50% (9)	21.5% (18)	31.1% (27)	21.4% (36)	10.6% (45)	4.68% (54)	1.95% (63)	0.78% (72)	0.29% (81)	0.09% (90)	
10	5.63%(10)	19.1% (20)	30.9/% (30)	22.9% (40)	11.9% (50)	5.44% (60)	2.34% (70)	0.97% (80)	0.30%(90)	0.14%(100)	0.04% (110)

TABLE II PERCENTAGE OF NUMBER OF RANDOMLY SELECTED INTEGER PAIRS FOR m=2 to m=10 BIT WITH REGARDS TO THEIR ITERATION NUMBER TO GET MULTIPLICATION RESULT

Bit(m)	Percentage of Randomly Selected Integer Pairs (Iteration Number)					
2	62.5% (2)	37.5% (4)				
3	6.25% (3)	78.1% (6)	12.5% (9)	3.12 (12)		
4	51.9% (4)	42.9% (8)	4.29% (12)	0.78% (16)		
5	6.64% (5)	80.7% (10)	0.29% (15)	9.27% (20)	3.02% (25)	
6	0.87% (6)	91.1% (12)	5.59% (18)	1.97% (24)	0.36% (30)	
7	0.08% (7)	86.2% (14)	10.3% (21)	3.32% (28)	0.01% (35)	0.08% (42)
8	49.7% (8)	44.7% (16)	5.10% (24)	0.38% (32)	0.001% (40)	49.7% (48)
9	7.67% (9)	79.2% (18)	0.53 (27)	0.03% (36)	9.33% (45)	3.13% (54)
10	0.77%(10)	90.9% (20)	3.16% (30)	2.94% (40)	1.78% (50)	0.39% (60)

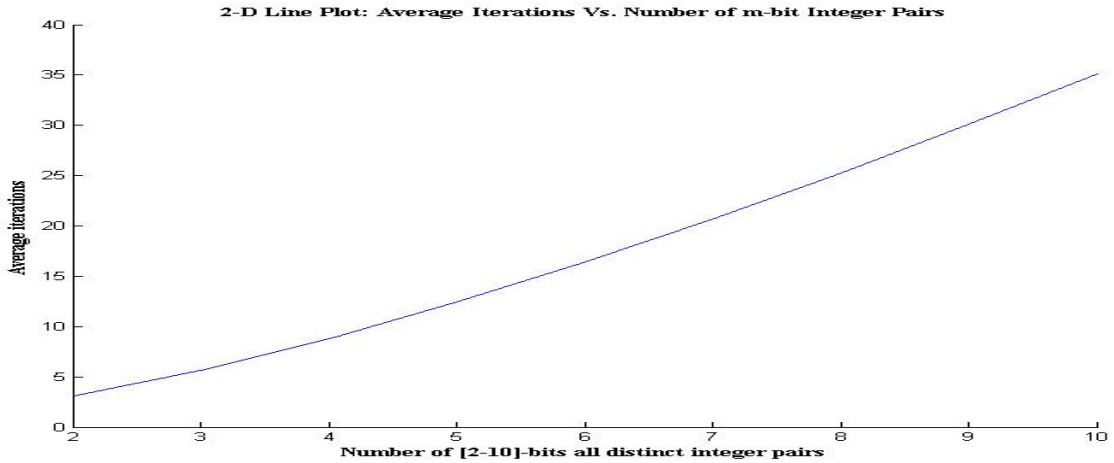

Figure 4 Average Iteration Graph for Distinct Integer Pairs

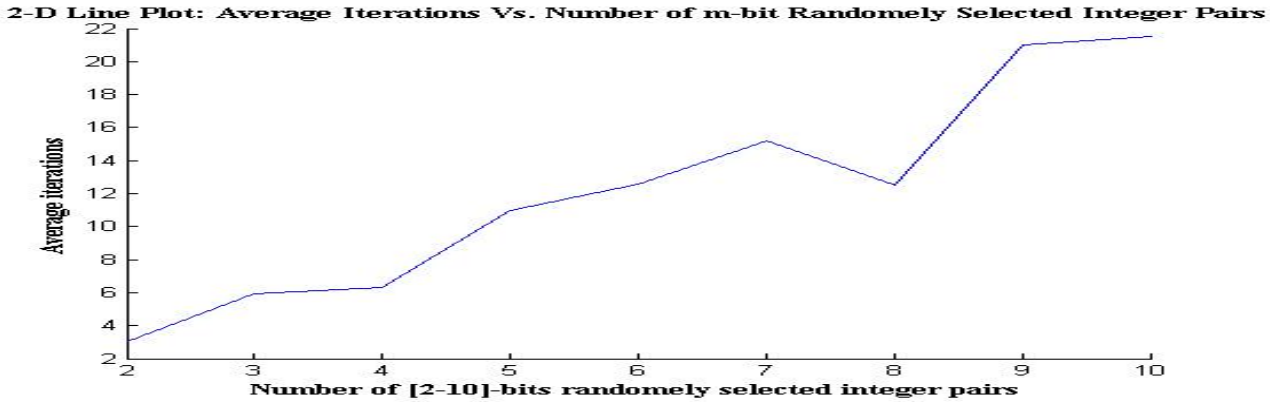

Figure 5 Average Iteration Graph for Randomly Selected Integer Pairs

ACKNOWLEDGMENT

The authors would like to thank to anonymus reviewers for their valuable comments to improve the quality of the paper.

REFERENCES

[1] J. C. Lo, *A Fast Binary Adder with Conditional Carry Generation*, IEEE Transactions on Computers, Vol. 46, No. 2, pp. 248-253, Feb. 1997.

[2] T. Lynch, E. E. Swartzlander Jr. *A Spanning Tree Carry Lookahead Adder*, IEEE Transactions on Computers, Vol. 41, No. 8, pp. 931-939, 1992.

[3] J. M. Dobson and G. M. Blair, *Fast twos complement VLSI adder design*, Electronics Letters, Vol. 31, No. 20, pp.17211722, 28th September 1995.

[4] M. Z. Rahman, L. Kleeman, and M. A. Habib, *Recursive Approach to the Design of a Parallel Self-Timed Adder*, IEEE transactions on very large scale integration (VLSI) systems, vol. 23, no. 1, January 2015.

[5] F.-C. Cheng, S. H. Unger, and M. Theobald, *Self-timed carry lookahead adders*, IEEE Trans. Comput., vol. 49, No. 7, pp. 659-672, Jul. 2000.

[6] H. Lee and G. E. Sobelman, *A New Low-Voltage Full Adder Circuit*,VLSI, 1997. Proceedings. Seventh Great Lakes Symposium on, DOI: 10.1109/GLSV.1997.580416

[7] R. Chawla, P. Kumar, P. Yadav, *Adder Circuit Design using Advanced Quantum Dot Cellular Automata (AQCA)*, National Conference on Recent Advances in Electronics & Computer Engineering, RAECE -2015, Feb.13-15, 2015.

[8] S. Z. Md Naziri1, R. C. Ismail1, A. Y. Md Shakaff, *Arithmetic Addition and Subtraction Function of Logarithmic Number System in Positive Region: An Investigation*, 2015 IEEE Student Conference on Research and Development (SCOReD) ,DOI: 10.1109/SCORED.2015.7449376.

[9] B. Kaur, V. Thakur, *Review of Booth Algorithm for Design of Multiplier*, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 4, 2014.

[10] Sk. S. Hassan, P. Pal Choudhury, B.K. Nayak, B.K., A. Ghosh, J. Banerjee, *Integral Value Transformations: A Class of Ane Discrete Dynamical Systems and an Application*, Journal of Advanced Research in Applied Mathematics, vol-7, issue-1, pp.62-73, 2015.

[11] P. Pal Choudhury, S. Sahoo, B. K. Nayak, *Theory of Carry Value Transformation and its Application in Fractal formation*, IEEE International Advance Computing Conference, 2008, DOI:10.1109/IADCC.2009.4809146.

[12] S. Pal, S. Sahoo, B. K. Nayak, *Properties of Carry Value Transformation*, International Journal of Mathematics and Mathematical Sciences, Volume 2012, Article ID 174372, 10 pages, 2012, http://dx.doi.org/10.1155/2012/174372.

[13] T. Toffoli, N. Margolis, *Cellular Automata Machines*, Cambridge, MA, MIT Press 1987.

[14] P. Pal Choudhury, S. Sahoo, M. Chakraborty, *Implementation of Basic Arithmetic Operations using Cellular Automata*, ICIT '08. International Conference, pp: 79-80, December 17-20, 2008, DOI: 10.1109/ICIT.2008.18.

[15] P. S. Kasat, D. S. Bilaye, H. V. Dixit, R. Balwaik, A. Jeyakumar, *Multiplication Algorithms for VLSI - A Review*, International Journal on Computer Science and Engineering (IJCSE), 2012.

[16] S. Samajder and P. Sarkar, *Fast Multiplication of the Algebraic Normal Forms of Two Boolean Functions*, Proceedings of Eighth International Workshop on Coding and Cryptography WCC 2013, April 15-19, 2013.

[17] A. D. Booth, *A signed binary multiplication technique*, The Quarterly Journal of Mechanics and Applied Mathematics, Volume IV, Issue 2, pp. 236-240, 1951.

[18] C. B. Boyer, (revised by Merzbach, Uta C.), *History of Mathematics*, John Wiley and Sons, Inc., ISBN 0-471-54397-7, 1991.

[19] A. Karatsuba and Y. Ofman, *Multiplication of Many Digital Numbers by Automatic Computers*. Proceedings of the USSR Academy of Sciences, Vol 145, No 2, pp. 293294, 1962.

[20] A. L. Toom, *The complexity of a scheme of functional elements realizing the multiplication of integers*, published in Soviet Math (translations of Dokl. Adad. Nauk. SSSR), 4, No. 3, 1963.

[21] S. A. Cook, *On the Minimum Computation Time of Functions*, PhD thesis, Harvard University Department of Mathematics, 1966.

[22] C. S. Wallace, *A suggestion for a fast multiplier*, IEEE Trans. on Electronic Comp., EC-13(1): 14-17, 1964.

[23] P. Pal Choudhury, Sk. S. Hassan, S. Sahoo, B. K. Nayak, *Act of CVT and EVT in the Formation of Number Theoretic Fractals*, International Journal of Computational Cognition (http://www.ijcc.us), VOL. 9, NO. 1, March 2011.

[24] J. K. Das, P. Pal Choudhury and S. Sahoo, *Multi-Number CVT-XOR Arithmetic Operations In Any Base System And Its Signicant Properties*, 2016 IEEE 6th International Conference on Advanced Computing, 2016, DOI 10.1109/IACC.2016.147.

[25] M. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann, San Francisco, 2004.

[26] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd edition, Oxford University Press, New York, 2010.