VOL. 12, NO. 21, NOVEMBER 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DESIGN AND IMPLEMENTATION OF HYBRID CASCADED ENERGY EFFICIENT KOGGE STONE ADDER

J. Vignesh[1], E. Gajendran[2], Sameeruddin Khan[3], N. Balakumar[4] and S. R. Boselin Prabhu[5]
[1]Department of Computer Science and Engineering, Sreyas Institute of Engineering and Technology, Hyderabad, India
[2]Department of Computer Science and Engineering, Sree Dattha Group of Institutions, Hyderabad, India
[3]Sree Dattha Group of Institutions, Hyderabad-Telangana, India
[4]Department of Electrical and Electronics Engineering, Tamil Nadu College of Engineering, Coimbatore, India
[5]Department of Electronics and Communication Engineering VSB Engineering College, Tamil Nadu, India
E-Mail: eben4uever@gmail.com

ABSTRACT

Addition plays a vital role in all arithmetic operations. It is used widely in many VLSI systems such as application specific DSP architecture and microprocessor architecture. The energy efficient designs have gained more recent attention and for highly employed functional units, especially for the adders. The energy consumption of an adder depends on the circuit minimizing, the addition procedure, the recurrence structure and the wiring complexity. Weinberger and Ling are the two most commonly used binary addition algorithms that are used in adders. This paper gives the performance evaluation of addition algorithms on Kogge-Stone arrangement and have been observed to save energy by the proper selection of addition algorithms in 16-bit adders. The design of Kogge-stone adder has been carried out by using TANNER EDA tool. The efficiency ofthe adder design can be improved by prefix selection, the algorithm, computational sum and logic depth. When compared to Weinberger algorithm, the number of gates used is less in case of the proposed ling recurrence algorithm.

Keywords: arithmetic and logic unit, computer arithmetic logic, energy-efficient method, high-speed arithmetic, DSP architecture, microprocessor architecture.

1. INTRODUCTION

Binary addition is one of the greatest primitive and commonly used applications in computer arithmetic operation. A huge variety of recent algorithms and implementations have been formulated for binary arithmetic addition [1-3]. Parallel-prefix adder tree topology network such as Kogge-Stone adder [4], Sklansky adder [5], Brent-Kung adder [6], Han- Carlson adder [7], and Kogge-Stone adder using Ling adders [8, 9] can be used to realize higher system operating speeds with higher efficiency. Parallel prefix adders are suitable for very large scale integrated implementation, since they trust on the use of simple cells and sustain regular connections between them. The VLSI integer adders are critical elements in general purpose processor and digital signal processors, as they are employed in the design of ALU data paths. In nanometer range, it is very significant to develop different types of addition algorithms that provide high energy performance while reducing power consumption of the system.

The requirements of the adder are that, it should be initially fast and secondarily effectual in terms of power consumption, energy and chip area. For wide-ranging adders (N>16), the delay of carry look-ahead adders becomes subjected by the delay in passing the carry through the look- ahead stages. This delay can be decreased by looking ahead across the look-ahead blocks. In general, a multilevel tree of look-ahead adders could be constructed to achieve delay that grows with logN. Such adders are commonly referred as tree adders or parallel prefix adders. Many parallel prefix topology have been described in the literature, especially in the framework of addition.

The basic modules of adders can be designed in many ways. At second level, minimization can also be achieved by using specific logic families in the design. The energy and power consumption of a microprocessor adder depends on the circuit sizing, the addition algorithm, the recurrence arrangement and the wiring difficulty. In this paper, adder components have been designed, analyzed and compared with the existing techniques in deep submicron technology. Several alternates of the carry look-ahead equations, like Ling adders [9] have been offered that simplify the carry level computations and can lead to faster adder structures. A comparative survey on the above mentioned high-speed binary adders have been entailed to afford a list of energy-efficient circuit topology that would be applicable to any parallel prefix computation algorithms. By way of designing and implementation of high-speed adders, an improvement in energy and power performance have been perceived. The rest of the paper has been organized as follows. Section II describes about the detailed literature survey of existing adder circuits. Section III elaborates the architecture of VLSI addition algorithms. Section IV describes about the performance analysis of addition algorithms. Section V entails the design of VLSI adder circuits. Section VI describes about the simulation results and discussions. Finally the last section describes the conclusion.

2. REVIEW OF ADDER CIRCUITS

2.1. Carry look ahead adder

A carry-look ahead adder (CLA) is a type of high speed adder used in digital logic circuits. A carry-look ahead adder increases the speed by decreasing the amount

of time required to determine carry bits in the system. It can be distinguished with the simpler but usually slower, ripple carry adder for which the carry bit is considered together with the sum bit, and each bit must wait until the previous carry has been calculated to begin computing its own result and convey bits.

The carry-look ahead adder determines one or more carry bits before the sum, which decreases the wait time to calculate the result of the larger value bits. Consider the n-bit addition of two numbers: A = a_{n-1}, a_{n-2}…..$a_0$and B =b_{n-1}, b_{n-2}. . .b_0 resulting in the sum, S = s_{n-1}, s_{n-2}……. s_0 and a carry, C_{out}. The first stage in carry-look ahead adder calculates the bit generate and bit propagate as follows,

$$g_i = a_i . b_i \text{ and } p_i = a_i + b_i \quad (1)$$

Where g_i is the bit generates and p_i is the bit propagate. These are then utilized to calculate the final sum and carry bits in the last stage as follows:

$$s_i = p_i \oplus c_i \text{ and } c_{i+1} = g_i + p_i . c_i \quad (2)$$

Where $\oplus$, + and represents exclusive, addition and multiply operations respectively. It is seen from equation (2) that, the first and last stages are inherently fast because they encompass only simple operations on signals local to each bit position. However, intermediary stages represent the long-distance propagation of carries, as a result of the performance of the adder merges on this part [10]. These intermediate stages can be used to determine group generate bit and group propagate bit to avoid waiting for a ripple, which in turn reduces the delay period. These group generate and propagates are given by,

$$P_{i:j} = P_{i:k} \cdot P_{k-1:j} \quad \text{and} \quad G_{i:j} = G_{i:k} + G_{k-1:j} . P_{i:k} \quad (3)$$

In this paper, Kogge-Stone implementation of Weinberger, Sparse-2 implementation of Weinberger, ling with merged first recurrence stage and bit-wise operations implementation of carry-look ahead adder have been carried out elaborately.

2.2 High-speed binary adder

A novel method is used to represent the new carry bit formation and propagation bit based on the perception of the complementing signal which was introduced in 1965.To study the impact of this complementing signal in performing binary addition and complementing signal look-ahead adder, one should estimate the formation of H_i and H_{i+1}as a function of adjacent bit pairs (i, i+1). Consider adding two binary numbers A and B together where, A = $a_0 2^n + a_1 2^{n-1} + a_2 2^{n-2}$+. . + $a_i 2^{n-i}$+ . . . +$a_n 2^0$and B = $b_0 2^n + b_1 2^{n-1} + b_2 2^{n-2}$ + . . . + $b_i 2^{n-i}$ +. . .+ $b_n 2^0$. The relation among the new carry (H_i, H_{i+1}) and the adjacent bit pairs (a_i, b_i, a_{i+1}, b_{i+1}) can be expressed all of these are generated by a_i, b_i or transferred through the low- order bits i+1, i+2, . . ., with the transmitting-enable switch to be ON. This signal or new carry can only be terminated when the inhibitor is ON (a_{i+1}+ b_{i+1}= 0).

2.3. Carry skip adder

The method is based on the detection and by-passing of the stages of a parallel prefix binary adder for the required addition (x+y), there happens the condition for carry- propagation. That is, the carry-signal is enabled to by-pass those steps of the carry paths for which,

$$x_i \neq y_i \quad (4)$$

Another criterion which does not make difference between propagated bit and generated bit conveys, but which is more efficient in the carry skip circuit is,

$$x_i = y_i \quad (5)$$

The circuit prescribed by Morgan and Jarvis, divides the adder into fixed groups both of six stages. The carry signal looking at the input of any group for which the condition is satisfied for each stage of the group, is transmitted to the next group through a special skip gate. At the same time, the carry is also allowed to propagate within the group to enter the determination of various sum bits.

2.4. Carry-select adder

The carry-select adder commonly consists of two ripple carry adders and a multiplexer. Adding two n-bit numbers with a carry-select adder is done with two adders in order to achieve the calculation twice, one time with the supposition of the carry being zero and the other assuming one. The number of bits in each carry select block can be uniform. In the even case, the optimum delay occurs for a blockof size. When a variable, the block size must have a delay, from addition inputs A and B to the carry out, equal to that of the multiplexer chain leading into it, so that the carry out is determined just in time.

2.5 Conditional-sum adder

A conditional sum adder is a recursive arrangement which is based on the carry-select adder. In case of conditional sum adder, the MUX level chooses between two n/2-bit inputs that are themselves made as conditional-sum adder. The bottom level of the tree consists of pairs of 2-bit adders plus 2 single-bit multiplexers. The conditional sum adder undergoes a very large fan-out of the intermediate carry outputs. The fan out could be high as n/2 on the previous level.

3. ADDITION ALGORITHMS

In this paper, mathematical analysis has been carried out for Weinberger adder and Ling adder [11-12].

3.1 Kogge-stone adders

The major difference between Kogge-Stone adders and other adders is higher performance. It calculates carry equivalent to every bit with the help of group generate bit and group propagate bit. In this adder the logic levels are given by $\log_2 N$, and the fan-out is 2.

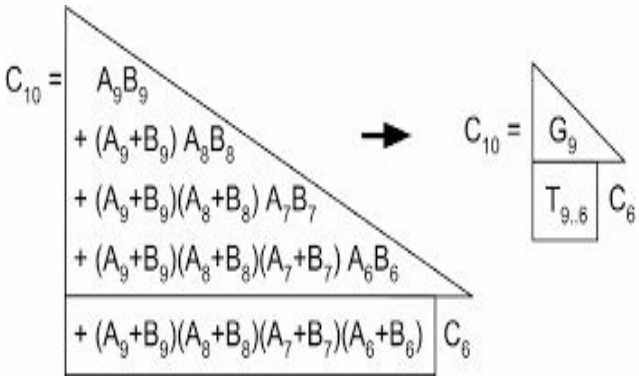

Figure-1. Weinberger's recurrence for addition.

3.2 Weinberger's recurrence for addition

Weinberger offered a general form for carry recurrence which is not restricted in group sizes and number of levels for carry computation. The conventional carry look-ahead adder is a definite case of this general carry recurrence. The sum and carry are defined and indexed as follows,

$$S_i = a_i \oplus b_i \oplus C_i \text{ and } C_{i+1} = a_i.b_i + (a_i + b_i)Ci \quad (6)$$

In Weinberger's recurrence method, the carry propagation speed has been improved by the usage of generate bit and propagate bit. Propagate can either be executed using an OR or XOR realizations. To differentiate the OR realization of propagate as transmit t, and the XOR realization as p, Weinberger's bit operations is described as,

$$g_i = a_i.b_i \text{ and } t_i = a_i + b_i \quad (7)$$

Substituting (7) into (6) results in,

$$C_{i+1} = g_i + t_i.C_i \quad (8)$$

Weinberger established that the recurrence applies to any prefix variation by the usage of group generate bit G and group transmit bitT. The computations of G and T are associative and idempotent, which permits for a wide range of recurrence tree structure possibilities for carry computation.

3.3. Ling's transformation

Technology limitations on fan-in and wired-OR in ECL (transistor stack height in complementary metal oxide semiconductor) interested in the simplification of Weinberger's recurrence. Ling established a transformation which became capablein achieving this generalization by factoring transmit, t from carry and expressed as,

$$C_{i+1} = g_i + t_i.C_i \text{ and } C_{i+1} = t_i(g_i + C_i) \quad (9)$$

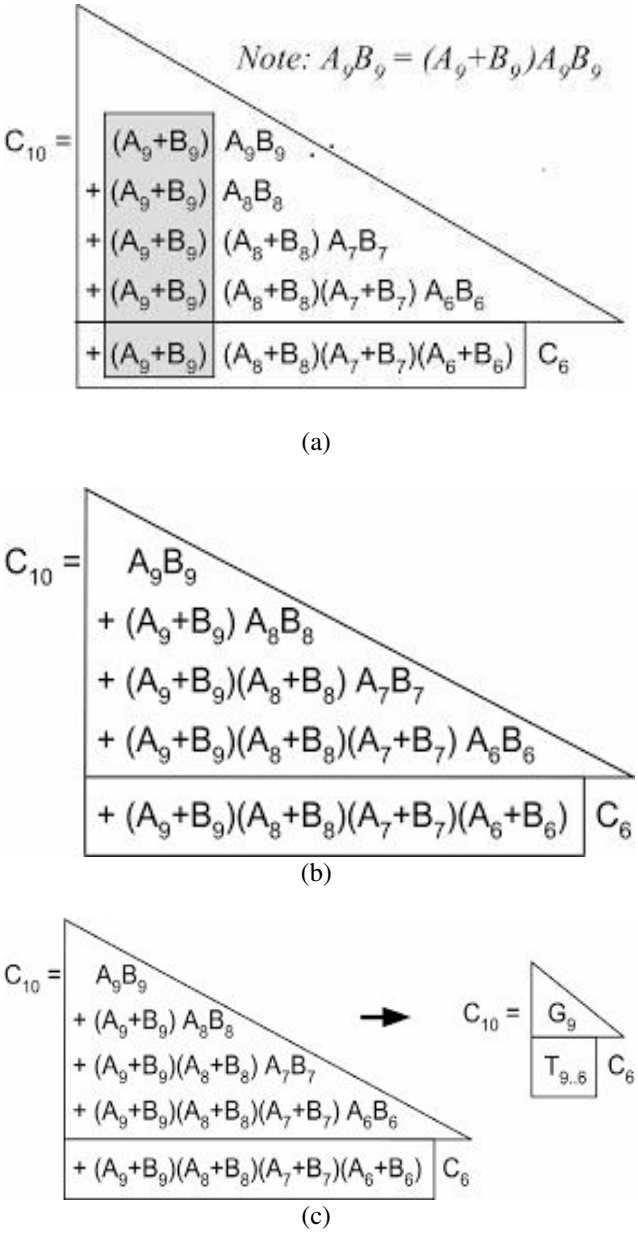

Figure-2. (a) Weinberger's recurrence (b) Intermediate form (c) Ling's transformation.

The overall transformation from Weinberger's recurrence to Ling's recurrence has been depicted in Figure-2. To create Ling's recurrence, this transformation is applied to C_6 which allows for a recurrence for C_{10} to be created using H and T as shown in Figure-3. For the recurrence, H_9 has one less term than G_9 in Weinberger's recurrence. To allow for recurrence $T_{8.6}$ is combined with t_5, resulting in the same number of terms as $T_{9.6}$ used in Weinberger's recurrence.

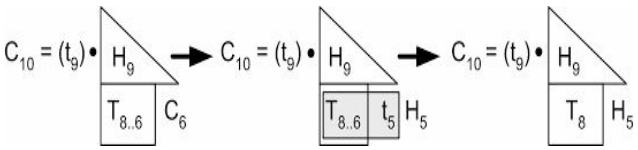

Figure-3. Ling's recurrence.

Ling's recurrence is performed on,

$$H_i : C_{i+1} = t_i H_i \text{ and } H_i = g_i + t_{i-1}.H_{i-1} \quad (10)$$

The main advantage of Ling's transformation is the compatibility with the prefix operator "." for recurrence of H_i and T_{i-1}.As a result, Ling's H and T have the same favourable properties as Weinberger's G and T when using the prefix operator ".". Ling's transformation decreases the difficulty of the recurrence by one term t_i in the first stage of the carry tree with the usage of H_{i-1} instead of C_i. However, the decrease in recurrence difficulty is achieved at the expense of increase in sum complexity.

$$S_i = A_i \oplus B_i \oplus (t_{i-1} H_{i-1}) \quad (11)$$

Improved sum complexity can be alleviated by using conditional logic. The summation can be implemented using a multiplexer with H_{i-1} as the select.

$$S_i = \begin{cases} A_i \oplus B_i & \text{if } H_{i-1} = 0 \\ A_i \oplus B_i \oplus t_{i-1} & \text{if } H_{i-1} = 1 \end{cases} \quad (12)$$

A multiplexer permits for sum to be computed with no improved complexity on the critical path compared to Weinberger's in the delay improvement achieved from decreasing the recurrence by one term.

4. ANALYSIS OF ADDITION ALGORITHMS

As mentioned earlier, Weinberger and Ling are the most commonly used addition algorithms in CMOS technology. Doran's transformations are not suitable for efficient CMOS realization. Doran demonstrations the recurrences that are more suitable in CMOS technology are Weinberger and Ling. Weinberger introduced the concept of generate bit and propagate bit signals to permit parallel prefix computation of carry signals to increase the system speed of addition. Ling transformation simplifies the carry recurrence of Weinberger at the cost of increased sum complexity when compared to Weinberger. Also, Ling recurrence algorithm gives better performance than Weinberger adder.

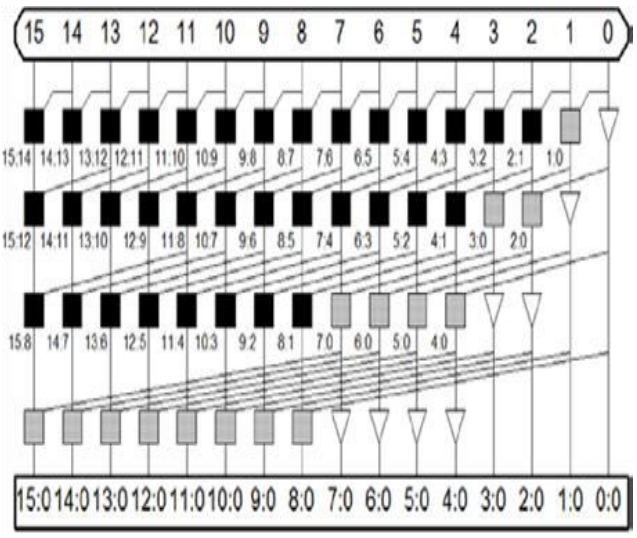

Figure-4. 16-bit kogge stone Weinberger's adder.

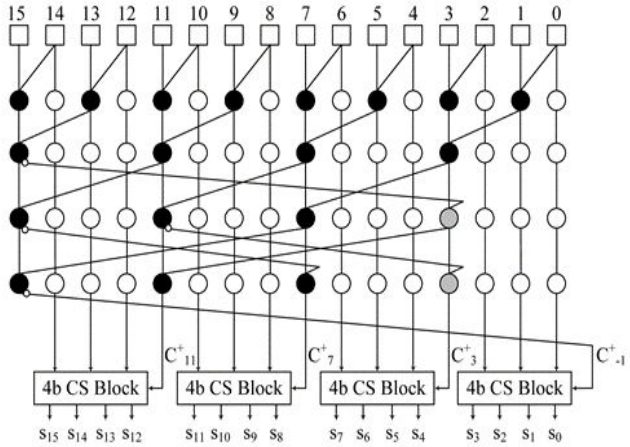

Figure-5. 16-bit kogge stone Ling adder.

Kogge-Stone (KS) adder is one of the well-known high performance and minimum depth parallel prefix adder arrangement, but it has high wiring complexity and energy consumption. This structure can be applied over Weinberger and Ling addition algorithms. The schematic articulation of 16-bit Kogge-Stone adder with Weinberger addition algorithm is depicted in Figure-4, and the schematic articulation of sparse-2 Ling adder with merged first recurrence stage is depicted in Figure-5. The logic gates in carry path for both addition algorithms are the same except the first carry combine stage. Ling recurrence uses NAND gate in the first carry combine stage that is OAI gate in Weinberger. Also, the internal network of gates are different for each other's. Additionally, the sum generation blocks include different logic gates and their structures are different. The number of internal nodes present in Ling adder is higher.

5. DESIGN OF ADDERS

Adders have been applied with static CMOS, dynamic CMOS and CMOS compound domino logic families. Several approaches have been proposed to increase the energy efficiency: proper selection of circuit

family and prefix, reducing the number of logic gates without increasing gate count, reducing switching activity, reducing number of logic gates, load buffering and reducing the wiring complexity. Based on these approaches a high performance and energy efficient VLSI adders are constructed.

6. SIMULATION RESULTS AND DISCUSSIONS

In this paper, 16 bit Kogge Stone prefix-2 adder using Weinberger and ling recurrence algorithm has been designed in static, dynamic and domino CMOS logic. For high-performance dynamic adders, Ling adder shows a fundamental advantage in CMOS by reducing the complexity of the first stage of the recurrence tree.

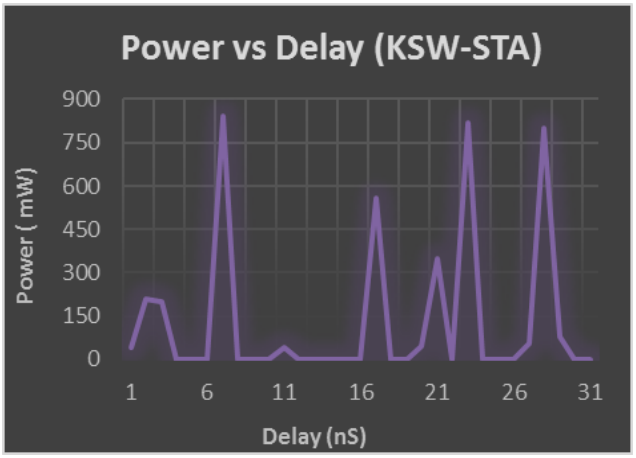

Figure-6. Power versus delay parameters (KSW-STA).

The designs which have the least number of stages are highly energy-efficient. Reduced energy consumption is a result of the decreased number of stages in the design, which permits for the same delay to be achieved while using a greater fan-out per stage. The energy reduction and performance enhancement of these designs is limited due to increased branching and gate complexity. The design of Kogge-stone adder has been carried out by using TANNER EDA tool and the design parameters are obtained for namely: power and delay. Figure 6 depicts the power versus delay parameter values obtained by using Weinberger structure in static mode of operation. It could be clearly observed that the power consumption of KSW-Static Mode of operation is 840 mW.

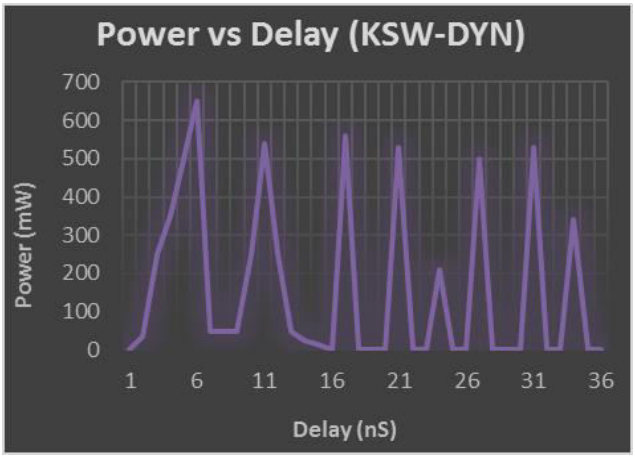

Figure-7. Power versus delay parameters (KSW-DYN).

Figure-7 shows the power versus delay parameter values that are obtained using Weinberger structure in dynamic mode of operation. It could be clearly observed that the power consumption of KSW-Dynamic Mode of operation is 650 mW. Dynamic mode of operation consumes less power when compared to static mode as well as domino mode.

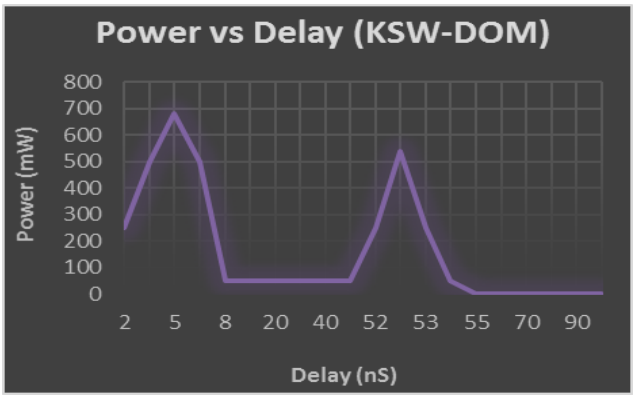

Figure-8. Power versus delay parameters (KSW-DOM).

Figure-8 shows the power versus delay parameter values that are obtained using Weinberger structure in domino mode of operation. It could be clearly observed that the power consumption of KSW-Dynamic Mode of operation is 680 mW. Thereby, domino mode of operation consumes less power when compared to KSW-Static Mode of operation but increased power consumption when compared to dynamic mode of operation.

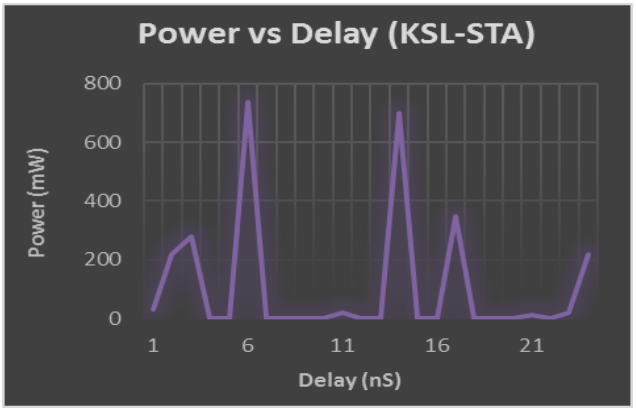

Figure-9. Power versus delay parameters (KSL- STA).

Figure-10. Power versus delay parameters (KSL-DYN).

Figure-9 depicts the power versus delay parameter values that is obtained using Ling structure in static mode of operation. It could be clearly seen that the power consumption of KSL-Static Mode of operation is 740 mW. As such, KSL-Static mode of operation consumes more power when compared to KSL-Dynamic Mode of operation and KSL-Domino mode of operation.

Figure-10 shows the power versus delay parameter values that is obtained using Ling structure in dynamic mode of operation. It could be observed that the power consumption of KSL-Dynamic Mode of operation is 540 mW, which is less when compared to both KSL-Static Mode of operation and KSL-Domino mode of operation.

Figure-11. Power versus delay parameters (KSL-DOM).

Figure-11 shows the power versus delay parameter values that is obtained using Ling structure in domino mode of operation. It could be observed that the power consumption of KSL-Domino Mode of operation is 560 mW, which is less when compared to KSL-Static Mode of operation but higher when compared to KSL-Dynamic mode of operation.

Table-1. Tabulated values of power of KSW and KSL.

Adder type	**Power** (mW)
KSW-STATIC	840
KSW-DYNAMIC	650
KSW-DOMINO	680
KSL-STATIC	740
KSL-DYNAMIC	540
KSL-DOMINO	560

The simulated values of KS-Weinberger adder and KS-Ling adderare compared and the results are shown in table 1. From the results it could be clearly seen that, KSW-Static adder consumes 840 mW power which is very much higher when compared to the power consumption by all other adders. Hence, KSW-Dynamic adder shows 29.23% reduced power consumption when compared to KSW-Static adder, 4.61% reduced power consumption when compared to KSW-Domino adder and 13.84% reduced power consumption when compared to KSL-Static adder. KSW-Domino adder shows 23.52% reduced power consumption when compared to KSW-Static adder and 8.82% reduced power consumption when compared to KSL-Static adder. KSL-Static adder shows 13.51% reduced power consumption when compared to KSW-Static adder. KSL-Domino adder shows 50% reduced power consumption when compared to KSW-Static adder, 16.07% reduced power consumption when compared to KSW-Dynamic adder, 21.42% reduced power consumption when compared to KSW-Domino adder and 32.14% reduced power consumption when compared to KSL-Static adder. KSL-Dynamic adder shows 55.55% reduced power consumption when compared to KSW-Static adder, 20.37% reduced power consumption when

compared to KSW-Dynamic adder, 25.92% reduced power consumption when compared to KSW-Domino adder, 37.03% reduced power consumption when compared to KSL-Static adder and 3.70% reduced power consumption when compared to KSL-Domino adder. Among all these adder circuits, KSL-Dynamic adder shows reduced power consumption because of the novel concepts embedded within this particular adder.

7. CONCLUSIONS

To increase the computational speed of the arithmetic unit, Kogge Stone adder has been employed. This paper gives a detailed evaluation of the design of low power and highly efficient VLSI adders in static, dynamic and domino CMOS logic using Weinberger and Ling recurrence structure. The power consumption of Kogge-Stone adder has been analyzed for Weinberger and Ling recurrence structure. Simulation result clearly shows that KSW-Static adder consumes more power when compared to all type of adders. Also, KSW-Dynamic adder consumes less power when compared to KSW-Static adder, KSW-Domino adder and KSL-Static adder. Thereby, KSW-Domino adder consumes less power when compared to KSW-Static adder and KSL-Static adder. The proposed KSL-Static adder consumes less power when compared to KSW-Static adder. KSL-Domino adder shows reduced power consumption when compared to the power consumption of all type of adders, except KSL-Dynamic adder. KSL-Dynamic adder shows 55.55% reduced power consumption when compared to KSW-Static adder, 20.37% reduced power consumption when compared to KSW-Dynamic adder, 25.92% reduced power consumption when compared to KSW-Domino adder, 37.03% reduced power consumption when compared to KSL-Static adder and 3.70% reduced power consumption when compared to KSL-Domino adder. Thus, KSL-Dynamic adder gives better performance (i.e., reduced power consumption) when compared to all other adders, asthese circuit employs less number of logic gates when compared to the existing adders.

REFERENCES

[1] I. Koren. 2002. Computer arithmetic algorithms. A.K. Peters, Limited.

[2] B. Parhami. 2000. Computer arithmetic algorithms and hardware designs. Oxford University Press.

[3] M. Ergecovac and T. Lang. 2003. Digital arithmetic, morgan- kauffman.

[4] P.M. Kogge and H.S. Stone. 1973. A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Transactions on Computers. 22(8): 786-793.

[5] J. Sklansky. 1960. Conditional-sum addition logic. IRE Transactions on Electronic Computers. 9: 226-231.

[6] R. P. Brent and H.T. Kung. 1982. A regular layout for parallel adders. IEEE Transactions on Computers. 31(3): 260-264.

[7] T. Han and D. Carlson. 1987. Fast area efficient VLSI adders. Proceedings of IEEE Symposium on Computer Arithmetic. pp. 49-56.

[8] H. Ling. 1981. High-speed binary adder. IBM Journal of Research and Development. 25: 156-166.

[9] A. Baliga and D. Yagain. 2011. Design of high speed adders using CMOS and transmission gates in submicron technology: a comparative study. Proceedings of the 4th International Conference on Emerging Trends in Engineering and Technology. pp. 284-289.

[10] S. Knowles. 2001. A family of adders. Proceedings of the 15th IEEE Symposium on Computer Arithmetic. pp. 277-281.

[11] B.R. Zeydel, T. Kluter and V.G. Oklobdzija. 2005. Efficient mapping of addition recurrence algorithms in CMOS. 17th IEEE Symposiumon Computer Arithmetic.

[12] D. Baran, M. Aktan, H. Karimiyan and V.G. Oklobdzija. 2009. Exploration of switching activity behavior of addition algorithms. IEEE International Midwest Symposium on Circuits and Systems (MWSCAS-2009).