A NOVEL DOMINO LOGIC DESIGN FOR EMBEDDED APPLICATION

Dr.K.Sujatha
Associate Professor, Department of Computer science and Engineering,
Sri Krishna College of Engineering and Technology, Coimbatore, Tamilnadu, India.

Abstract

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 sizing, the addition algorithm, the recurrence structure and the wiring complexity. Weinberger and Ling are the two most commonly used binary addition algorithms that are used in adders. The addition algorithms have been examined on Kogge-Stone structure 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 of the adder design can be improved by prefix selection, algorithm, computational sum and logic depth. When Compared to Weinberger algorithm, the number of gates used is less in the proposed ling recurrence algorithm.

Keywords: Arithmetic and logic structures, computer arithmetic, energy-efficient design, 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 proposed 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 and the system high efficiency. Parallel prefix adders are suitable for VLSI 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 since they are employed in the design of arithmetic-Logic Units, floating-point arithmetic 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 efficient in terms of power consumption, energy and chip area. For wide-ranging adders ($N > 16$), the delay of carry look-ahead adders becomes dominated by the delay of passing the carry through the look-ahead stages. This delay can be decreased by looking ahead across the look-ahead blocks. In general, we can build a multilevel tree of look-ahead adders to achieve delay that grows with logN. Such adders are commonly referred to as tree adders or parallel prefix adders. Many parallel prefix topologies 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 structure and the wiring complexity. In this paper, adder components are designed, analyzed, and compared with the previous techniques in deep submicron technology. Several alternates of the carry look-ahead equations, like Ling carries [9], have been presented that simplify carry level computation and can lead to faster adder structures. Most high speed adders depend on the previous carry to generate the present sum. Ling adders [8, 9], on the other hand, make use of Ling carry and propagate bits, in order to calculate the sum bit. As a result, dependency on the previous bit addition is reduced. This paper describes a comparative survey on the above mentioned high-speed binary adders and to afford a list of energy-efficient circuit topology that will be applicable to any parallel prefix computation algorithms. By designing and implementing high-speed adders, we perceived that there is an improvement in energy and power performance.

2. ADDERS

2.1. Carry Look Ahead Adders

A carry-look ahead adder (CLA) is a one 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 calculating its own result and carry 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. The Kogge-Stone adder and Brent-Kung adder are examples of this type of adder. Consider the n-bit addition of two numbers: A = an−1, an−2. . . a0and B =bn−1, bn−2, . . . , b0 resulting in the sum,
S = sn−1, sn−2, . . . , s0 and a carry, Cout.

The first stage in carry-look ahead adder calculates the bit generates and bit propagate as follows:

$$gi = ai \cdot bi \qquad (1)$$
$$pi = ai + bi,$$

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

$$s_i = p_i \oplus c_i,$$
$$ci+1 = gi + pi \cdot ci, \qquad (2)$$

Where ·, + and ⊕ represent AND, OR, and XOR operations. It is seen from (2) that the first and last stages are inherently fast because they involve 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 hinges 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. This group generates and propagates are given by:

$$Pi:j = Pi:k \cdot Pk-1:j, \quad (3)$$
$$Gi:j = Gi:k + Gk-1:j \cdot Pi:k.$$

In this paper, we have used the Kogge- Stone implementation of Weinberger, Sparse-2 implementation of Weinberger, sparse-2 ling with merged first recurrence stage and bit-wise operations implementation of carry-look ahead adder.

2.2 High-speed binary adder

A new method is used to represent the new carry bit formation and propagation bit based on the concept 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 Hi and Hi+1, as a function of adjacent bit pairs (i, i + 1). Let us consider adding two binary numbers A and B together, where A = ao2n + al2n-1 + a22n-2+. . + at 2n-t+ . . .+ an20;B = b02n + b1 2n-1 + b22n-2 + . . . + bi2n-i +. . .+ bn20 . The relation among the new carry (Hi, Hi+,) and the adjacent bit pairs (ai, bi; ai+1, bi+1) can be expressed all of these are generated by ai, bi or transferred through the low- order bits, i + 1, i +2, . . ., with the transmitting-enable switch ON.

This signal or new carry can only be terminated when the inhibitor is ON (ai+l + bi+1, = 0).

2.3. Carry Skip

Conference on New Digital Computer Techniques, Morgan and Jarvis elaborate a binary skip circuit. The method is based on the detection and by-passing of those stages of a parallel prefix binary adder in which, during a given addition (X+Y), there exists the condition for carry- propagation. That is, the carry-signal is enabled to by-pass those stages of the carry circuits for which

$$xi \neq yi \qquad (4)$$

An another criterion which does not differentiate between propagated and generated carries but which is more efficient in the carry skip circuit is

$$xi \cup yi \qquad (5)$$

The circuit prescribed by Morgan and Jarvis and outlined in block diagram divides the adder into fixed groups, each of six stages. The carry signal appearing at the input of any group for which 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 permit determination of the 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 assumption of the carry being zero and the other assuming one. After the two results are determined, the correct sum, as well as the correct carry, is then selected with the multiplexer once the correct carry is known. The number of bits in each carry select block can be uniform, or variable. In the uniform case, the optimal delay occurs for a block of size. When a variable, the block size should 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 structure based on the carry-select adder. In the 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 suffers from a very large fan-out of the intermediate carry outputs. The fan out can be as high as n/2 on the last level, where drives all multiplexers from to.

3. ADDITION ALGORITHMS

In this paper, mathematical analysis has been given for Weinberger adder and Ling adders.

3.1. Kogge-Stone Adders

The main difference between Kogge-Stone adders and other adders is its high performance. It calculates carries corresponding to every bit with the help of group generate bit and group propagate bit. In this adder the logic levels are given by log2N, and fan-out is 2.

3.2. Weinberger's Recurrence for Addition

Weinberger offered a general form for carry recurrence which was not limited in group sizes and number of levels for carry computation. The conventional carry-look- ahead (CLA) adder is a specific case of this general carry recurrence. The sum and carry are defined and indexed as follows.

$$Si = ai \oplus bi \oplus Ci$$
$$Ci+1 = ai.bi + (ai + bi)Ci \qquad (6)$$

In Weinberger's recurrence, the carry propagation speed has been improved through the use of generate bit and propagate bit. Propagate can either be executed using an OR or an XOR. To differentiate them we refer to the OR realization of propagate as transmit, t, and the XOR realization as, p. We define Weinberger's bit operations as:

$$gi = ai.bi$$
$$ti = ai + bi$$

Substituting into (6) obtains:

$$Ci+1 = gi + ti.Ci$$

Weinberger established that the recurrence applies to any prefix variation through the use of group generate, G, and group transmit, T

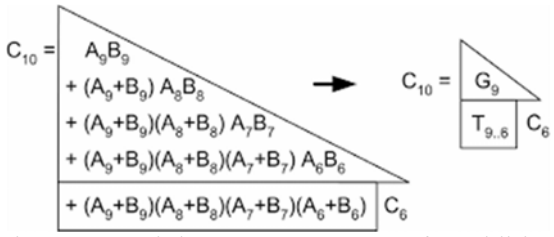


Figure 1. Weinberger's recurrence for addition

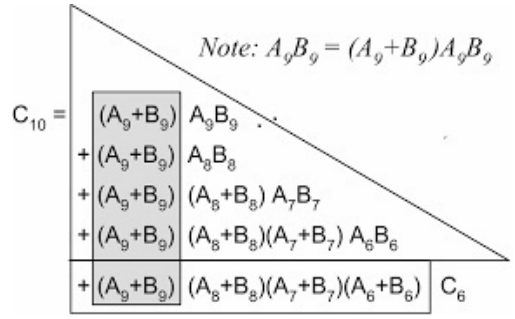
The computations of G and T are associative and idempotent, which allows for a wide range of recurrence tree structure possibilities for the carry computation.

3.3. Ling's Transformation

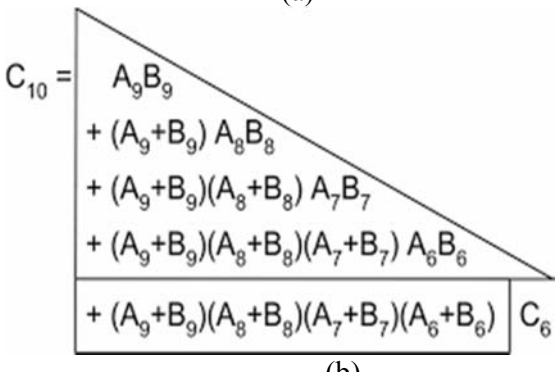
Technology limitations on fan-in and wired-OR in ECL (transistor stack height in CMOS) motivated simplifying Weinberger's recurrence. Ling established a transformation which was capable to achieve this simplification by factoring transmit, t, from carry.

$$Ci+1 = gi + ti.Ci$$
$$Ci+1 = ti\ (gi + Ci)$$

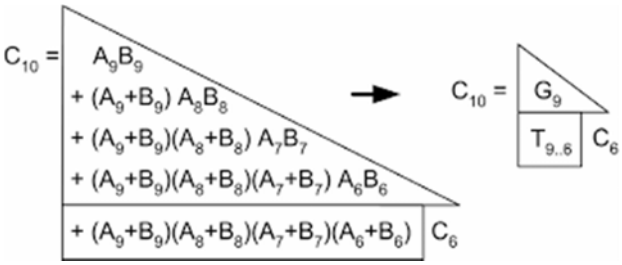
The transformation from Weinberger's recurrence to Ling's is shown in Fig. 2.



(a)



(b)



(c)

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

To create Ling's recurrence, this transformation is applied to C6 which allows for a recurrence for C10 to be created using H and T as shown in Fig. 3. For the recurrence, H9 has one less term than G9 in Weinberger's recurrence. To allow for recurrence T8.6 is combined with t5, resulting in the same number of terms as T9.6 used in Weinberger's recurrence.

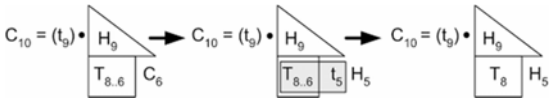


Figure 3. Ling's recurrence

Ling's recurrence is performed on

$$Hi: Ci+1 = ti\ Hi$$
$$Hi = gi + ti{-}1\ .\ Hi{-}1$$

The group recurrence relation for H and T permits for parallel prefix computation: (H+ and T+ denote the next logic level) Hi +=Hi+ Ti-1. Hi-1 and Ti−1+= Ti-1 Ti-2. An advantage of Ling's transformation is compatibility with the prefix operator "•" for the recurrence of Hi and Ti-1. As a result, Ling's H and T have the same favorable properties as Weinberger's G and T when using the prefix operator "•". Ling's transformation decreases the difficulty of the recurrence by one term ti in the first stage of the carry tree through the use of Hi-1 instead of Ci. However the decrease in recurrence difficulty is achieved at the expense of the increase in sum complexity.

$$Si = Ai \oplus Bi \oplus (t_{i-1} H_{i-1})$$

The improved sum complexity can be alleviated through the use of conditional logic. The summation can be implemented using a multiplexer, with Hi-1 as the select.

$$Si = \begin{cases} Ai \oplus Bi & \text{if } Hi-1 = 0 \\ Ai \oplus Bi \oplus ti\ -1 & \text{if } Hi-1 = 1 \end{cases}$$

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. In 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 an increase in sum complexity as compared to Weinberger. Also, Ling recurrence algorithm may give better performance than Weinberger adder. For accurate energy comparison of those well-known recurrence algorithms, the switching activity of internal nodes needs to be considered.

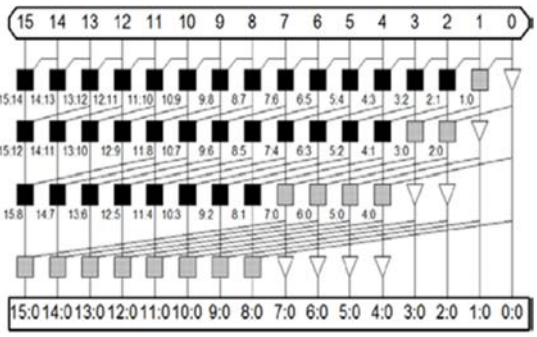


Figure 4: 16-bit kogge stone Weinberger's adder

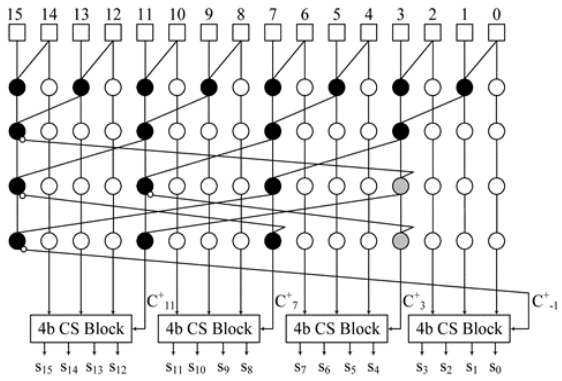


Figure 5: 16-bit kogge stone Ling adder

Kogge-Stone (KS) adder is the one of the well-known high performance and minimum depth parallel prefix adder structure, but it has high wiring complexity and energy consumption. This structure can be applied by Weinberger and Ling addition algorithms and the schematic of 16-bit Kogge-Stone adder with Weinberger addition algorithm is shown in Figure 4 and the schematic of sparse-2 Ling adder with merged first recurrence stage is shown 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 connections of gates are different from each other. In addition to them, sum generation blocks include different logic gates and their complexities are different. The number of internal nodes is higher in Ling adder.

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

In this project, 16 bit Kogge Stone prefix-2 adder using Weinberger and ling recurrence algorithm is designed in static, dynamic and domino CMOS logic. For high-performance dynamic adders Ling shows a fundamental advantage in CMOS by reducing the complexity of the first stage of the recurrence tree. The designs which have the least number of stages are the most energy-efficient. The reduced energy is a result of the decreased number of stages in the design, which allows for the same delay to be attained while using a greater fan-out per stage. The energy reduction and performance enhancement of these designs is limited due to the 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 namely: Power and Delay.

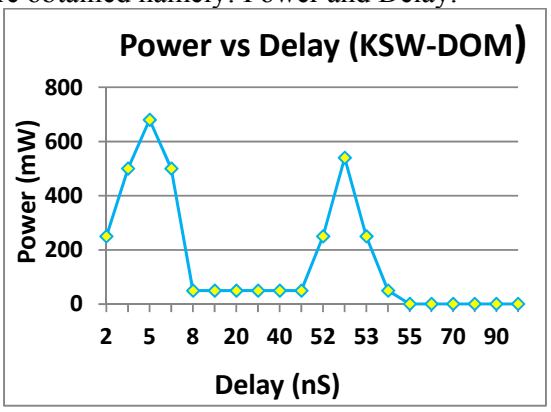


Figure 4: Power vs delay parameters (KSW-DOM)

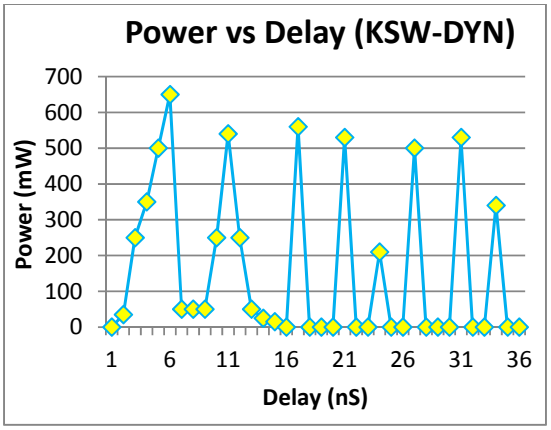


Figure 5 power vs delay parameters (KSW-DYN)

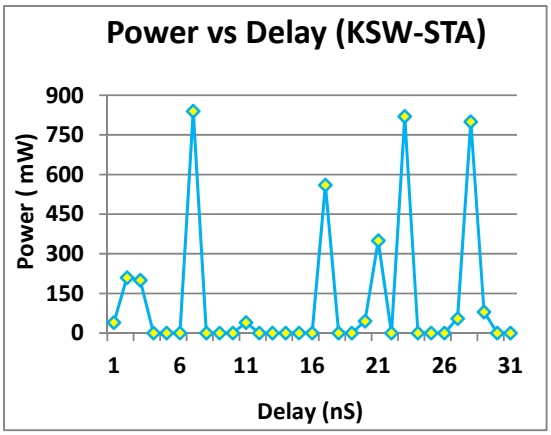


Figure 6 power vs delay parameters (KSW-STA)

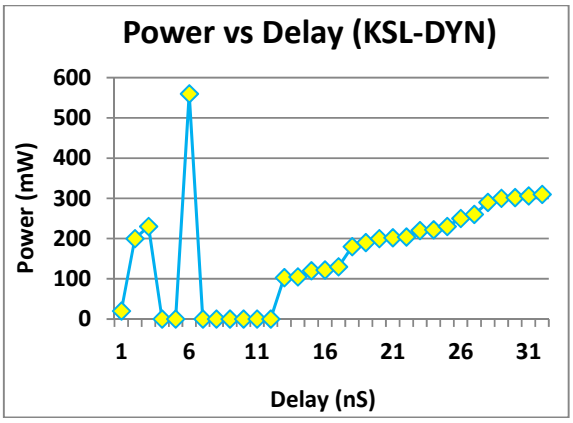


Figure 8 power vs delay parameters (KSL-DYN)

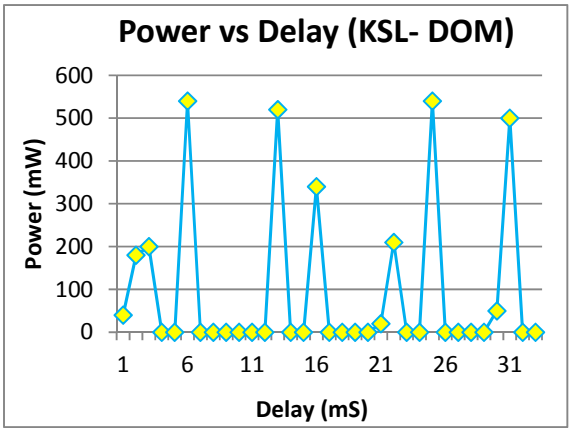


Figure 9 power vs delay parameters (KSL-DOM)

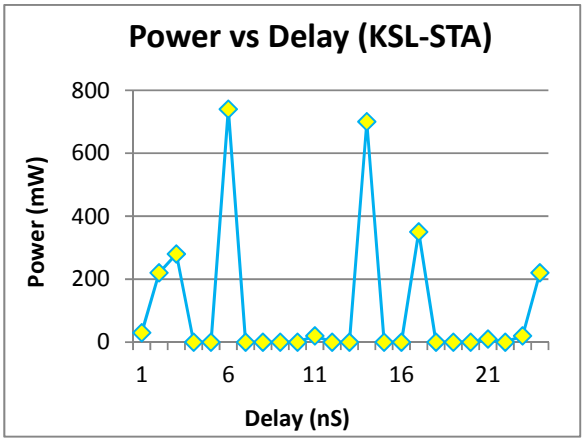


Figure 10 power vs delay parameters (KSL-STA)

All results are obtained by characterizing the delay of logic gates in nm CMOS technologies using the typical process corners. The delay of each logic gate is characterized under worst case single input switching conditions.

7. CONCLUSION

To increase the speed of computation of the arithmetic unit, Kogge Stone adder has been used. The present work comprises of the design of a low power and high efficiency VLSI adder in Static and dynamic and domino CMOS logic using Weinberger and ling recurrence algorithm. The power and delay of Kogge-Stone adder has been analyzed for Weinberger and Ling recurrence algorithm. By applying power-delay tradeoff on various levels, the simulation results show that Static kogge- stone ling adder has 11.9 % power reduction and 17.8 % delay improvement over Static Weinberger adder and the Dynamic KSA Ling adder has 13.8 % power reduction and 13.1% delay improvement over dynamic Weinberger adder and the Domino KSA Ling adder has 17.6 % power reduction and 13.1% delay improvement over Domino Weinberger adder. Energy efficient design may be applied to 16 bit Static and dynamic and domino CMOS Han-CARLSON adder using Ling recurrence algorithm. Across different nanometer technology nodes energy efficiency may be calculated in terms of energy and delay.

REFERENCES

[1] I. Koren, Computer Arithmetic Algorithms, A. K. Peters, 2002.

[2] B. Parhami, Computer Arithmetic Algorithms and Hardware Designs, Oxford University Press, 2000.

[3] M. Ergecovac and T. Lang, Digital Arithmetic, Morgan- Kauffman, 2003.

[4] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," IEEE Transactions on Computers, vol. C-22, no. 8, pp. 786–793, 1973.

[5] J. Sklansky, "Conditional-sum addition logic," IRE Transactions on Electronic Computers, vol. 9, pp. 226–231, 1960.

[6] R. P. Brent and H. T. Kung, "A Regular Layout for Parallel Adders," IEEE Transactions on Computers, vol. C-31, no. 3, pp. 260–264, 1982.

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

[8] H. Ling, "High-speed binary adder," IBM Journal of Research and Development, vol. 25, pp. 156–166, 1981.

[9] A. Baliga and D. Yagain, "Design of High speed adders using CMOS and Transmission gates in Submicron Technology: a Comparative Study," in Proceedings of the 4th International Conference on Emerging Trends in Engineering and Technology (ICETET '11), pp. 284–289, November 2011.

[10] S. Knowles, "A family of adders," in Proceedings of the 15th IEEE Symposium on Computer Arithmetic, pp. 277–281, June 2001.