Available online at www.sciencedirect.com

ScienceDirect

Procedia Materials Science 10 (2015) 371 – 380

Procedia Materials Science

www.elsevier.com/locate/procedia

2nd International Conference on Nanomaterials and Technologies (CNT 2014)

Area Efficient Hybrid Parallel Prefix Adders

Poornima N[a],*,V S Kanchana Bhaaskaran[b]

[a]JSS Academy Of Technical Education, Bangalore, India.
[a,b]VIT University, Chennai, India.

Abstract

Addition is a timing critical operation in almost all modern processing units. The performance parameters such as the implementation area, the adder latency and the power dissipation decide the choice of adders for different applications. Hence, there is an extensive research attention towards designing higher speed and less complex adder architectures with lower power dissipation. Among the several adder topologies available, parallel-prefix adders are the most frequently employed as they offer many design choices for achieving area/power/delay efficiency and they also provide optimization of the trade-offs. This paper discusses the design and implementation of area-power optimized hybrid parallel-prefix Ling adder. The hybrid adder topology employed in this work uses Ladner-Fischer approach for even-indexed and Kogge-Stone structure for odd-indexed bits. The independent computation of carries for odd and even bits, directly leads to the reduction of fan-out of the prefix tree and thereby a reduced delay. The area efficiency is achieved by the computation of the real carries using modified Ling's equations. The proposed adders are implemented with word size of 16 bit and 32 bit based on modified Ling equations using 0.18µm CMOS technology. The synthesis results reveal that the proposed adders could achieve up to 24% and 35% saving of area-power product and power-delay product respectively, over the adders based on conventional Ling equations.

Keywords: **Parallel Prefix Adder; Ling recursion; Modified Ling recurrence; Hybrid tree structure; Area efficiency.**

1. Introduction

Binary addition is one of the most frequently used operations in the microprocessors, digital signal processors (DSP) and data-processing in application-specific integrated circuits (ASIC). Therefore, the binary adders

* Corresponding author. Tel.: +0-91-984 5988280;*E-mail address:* poorniman@jssateb.ac.in

doi:10.1016/j.mspro.2015.06.069

constitute the basic building blocks of the arithmetic and logic units (ALU), address generation units (AGU) and floating-point blocks, to name a few. The high performance wide adders typically use a parallel prefix tree to compute the group generate and the group propagate signals to compute the carries and the final sum bits. Hence, among the several adder designs presented in the literature, the Parallel Prefix Adder (PPA) designs are the most preferable for their higher speed of operation. In the last few decades, various addition algorithms have been proposed aiming at improving the computational efficiency of PPAs by optimizing one or more of the parameters, namely, speed, power, area and regularity of carry graphs.

Several parallel prefix adder topologies have been published in the literature, and they also present the comparisons among the parallel tree adders. Different design parameters such as the delay, fan-out, wiring complexity, regularity and the area required for implementation have been used to describe the comparative benefits of various adders. There are several techniques proposed for the carry computation in the parallel prefix trees. Sklansky (1960) proposes tree-prefix algorithms for adders, wherein a tree structure is used to compute the intermediate signals. Kogge and Stone (1973), in their scheme uses the recursive doubling property and the prefix trees were characterized by their minimum logic depth, regular structure, and unity fan-out and they are used when very high performance is needed. The main problem of the scheme is the large number of gates and the long lateral wires required between the consecutive stages, which increases power dissipation. The minimum depth prefix graph has been introduced by Ladner and Fischer (1980). As the longest lateral fanning wire extend from one node to n/2 other nodes, the capacitive fan-out load becomes larger for later levels in the carry graph. Additionally an appropriate number of buffering inverters are added to drive these large loads at the cost of slightly increased delay parameter. Brent and Kung (1982), propose the prefix-computation graph in an area-optimal way. Here, the lateral fan-out of each node is restricted to unity, similar to the Kogge-Stone graph, however, without using several long wires. In spite of the attractive topological structure, it incurs increased logical depth. Han and Carlson (1987) presented a new prefix tree, which is a hybrid of Brent-Kung and Kogge-Stone adders, with reduced number of computational nodes and slight increase in logical depth. Knowles (2001) has demonstrated how the various adder topologies influence the fan-out and the wiring density, thus, influencing the design decisions and yielding to better area/power trade-offs. Some adders, instead of being fully parallel, use sparseness to reduce the impact of lateral fan-out. Several sparse tree implementations have been published with sparseness of two and four. Sanu Mathew et al (2001) and S. Kao et al (2006) present the prefix tree with sparseness of two. Naffziger (1996) and Shimazaki et al (2004), implements 64 bit adders using sparse four trees to decrease the latency. Ling (1981) has proposed different carry generation equations where one propagate term is factored out to simplify the group generate function, thereby making the first level of prefix tree simpler, which further reduces the critical path delay. Dimitrakopoulos and Nikolos (2005) propose an approach which can save one logic level of implementation compared to the parallel-prefix structures proposed for the traditional definition of carry look-ahead equations and the structure reduces the fan-out requirements of the design.

This paper discusses the design and implementation of hybrid prefix adder based on Ling equations, aiming at area reduction. The proposed hybrid adder employs two tree architectures, Kogge-Stone on odd-numbered bits and Ladner-Fischer on even-numbered bits. It uses modified Ling equations to reduce the complexity of the generate function at the first level. Furthermore, as the real carries for higher order bits are computed from the lower order Ling carries, a significant area saving has been achieved along with reduced delay and power. The area, delay and power parameters are computed and compared among the adders for two different word sizes to prove the area-power and power-delay efficiency of the proposed adders.

The rest of the paper is structured as follows: Section 2 revisits the basics of parallel-prefix addition and Ling adders are described in section 3. The proposed area-power efficient hybrid prefix adders based on modified Ling equations are presented in Section 4. Section 5 discusses the simulation and synthesis results that validate the area efficiency of the proposed adders. Finally, Section 6 concludes the paper.

2. Background

The delay of an adder depends on how fast the carry reaches each bit position. Hence, the major bottleneck in the design of binary addition is the carry chain which computes the carries. As the width of the input operand increases, the length of the carry chain and delay increases. To reduce the delay and to improve the performance, the parallel-prefix adders can be employed. The concept in parallel prefix adders is to compute a small group of intermediate prefixes and then find the large group prefixes, until all the carry bits are computed. Parallel prefix addition of the operands A and B of width n is done in three steps as shown below:

2.1. Pre-Processing stage

In this stage, Generate (g_i), Propagate (p_i) and half sum (d_i) signals for all the bits of the adder are calculated using the equations as given below.

$$g_i = a_i \cdot b_i \tag{1}$$
$$p_i = a_i + b_i \tag{2}$$
$$d_i = a_i \oplus b_i \tag{3}$$

Here $0 \le i \le n-1$, where n is the word length.

2.2. Prefix stage

In the prefix stage, the group level Generate and Propagate signals are computed using different prefix trees. The design factors which can influence the performance of prefix structures are defined below.
Radix or Valency: It identifies the number of carry bits merged at each logic stage and it also determines the number of stages required in the tree to compute all the required carry bits.
Logic Depth: It defines maximum number of logic stages in the carry graph from output to inputs.
Lateral Fan-out: The highest number of nodes driven by a node in a stage is the lateral fan-out of the stage.
Wiring Complexity: The maximum number of wires routed across the bit pitch between any two successive levels of the carry merge tree.
In the prefix tree, *Group Generate* and *Group Propagate* are calculated for the bits spanning from j to i using the equations expressed by

$$G_{i:j} = G_{i:k} + G_{k-1:j} \cdot P_{i:k} \tag{4}$$
$$P_{i:j} = P_{i:k} \cdot P_{k-1:j} \tag{5}$$

Here, $i > k > j$

The prefix operation can also be represented by an operator "$\circ$" which is associative and idempotent. The group generate and group propagate can be denoted by using the dot operator as given by $(g, p) \circ (g', p') = (g + p.g', p.p')$. Where the first output term $g + p.g'$ is the group generate and the second output term $p.p'$ is the group propagate.
Tree Sparseness: In a carry tree, if only some of the carries are computed instead of computing the carries at all the bit positions, then the resultant tree is sparse. Then, the sum will be calculated only with the available carries. Sparse tree architecture is used to separate the carry graph into critical and non-critical sections. The reason is to accelerate the critical path by pushing a part of the carry graph to a non-critical side path. Sparseness can be larger than two and can be applied to any carry tree.
Recurrence Equations: The set of logic equations used in the addition algorithms also influence the performance of adder. Two important recurrence equations which are frequently used are the Weinberger's recurrence and the Ling's recurrence. Weinberger recurrence uses the bitwise generate, bitwise propagate, group generate and group propagate signals as given by the Equations (1), (2), (4) and (5).These set of equations are used to compute the carries in parallel. The sizes of the group generate and group propagate signals can be arbitrary. Ling's recurrence equations simplifies the group generate function by stripping out one propagate term and re-introducing it in the last stage to compute the final sum bits, as described in Section 3.

2.3. Post-Processing stage

The sum bits are generated either by using simple XOR gates or by the use of conditional sum adders. In conditional sum adders, for each bit position, two tentative sums will be generated and the correct one will be selected when the relevant carry for that bit arrives.

Several design choices such as tree topologies, recurrence equations, full or sparse implementation and circuit design styles are available for selection while designing the parallel prefix adders. In this work, a hybrid tree topology, based on modified Ling's carry look-ahead equations is chosen and implemented. The effects of using the mixed tree structure and the variation of recurrence equations on area, power and delay parameters of the adders are analysed. The validation of the proposed design is made by implementation and comparison of hybrid adders based on conventional and modified Ling's equations.

3. Ling Adders

Ling (1981) has proposed different carry generation equations, in which one propagate term is factored out to simplify the group generate function, which in turn makes the first level of prefix tree simple. This further reduces the critical path delay. Ling equations exploit the property that the group generate term is much more complex than the group propagate term. Hence, by absolving some complexity of the generate term to propagate term both the functions will become balanced. Ling formulated a new generate function referred to as the pseudo carry or reduced generate $H_{n::0}$ which is less complex than the conventional group generate $G_{n::0}$ .The propagate term in Ling's formulation is referred to as the pseudo propagate. The reduced generate makes the first level of carry tree simpler. The factored propagate term will be combined with the simplified group generate function or the pseudo carry to compute the final sum. According to Ling (1981), the Ling pseudo carry can be given by

$$H_{i:j} = g_i + G_{i-1:j} \tag{6}$$

Then each Ling carry H_i can be expressed as

$$H_i = g_i + g_{i-1} + p_{i-1} \cdot g_{i-2} + \ldots\ldots + p_{i-1} \cdot p_{i-2} \cdots\cdots p_1 \cdot g_0 \tag{7}$$

By taking the logical AND of H_i with p_i the conventional carry c_i can be recreated using equation

$$c_i = p_i \cdot H_i \tag{8}$$

Final sum bits for each bit position i, can be computed by

$$S_i = d_i \oplus c_{i-1} = d_i \oplus (p_{i-1} \cdot H_{i-1}) \tag{9}$$

The sum computation can be transformed into

$$S_i = \overline{H_{i-1}}\, d_i + H_{i-1} (d_i \oplus p_{i-1}) \tag{10}$$

This can be implemented using a multiplexer that selects either d_i or $(d_i \oplus p_{i-1})$ based on the value of H_{i-1} .As the computation of d_i and $(d_i \oplus p_{i-1})$ are not falling in the critical path, the computation of Ling carries will not impede the performance of the adder. On the other hand, due to the reduced complexity of the Ling carries compared to the conventional carries, the computation time for the sum bits will be reduced.

4. Hybrid Prefix Adders Based On Ling And Modified Ling Equations

Ling adder employing the radix-2 at its first level of prefix tree will have the equations for the group generate and the group propagate signals computed from the adjacent bit pairs (a_i, b_i) and (a_{i-1}, b_{i-1}) as given by

$$G_{i:i-1} = a_i \cdot b_i + a_{i-1} \cdot b_{i-1} \tag{11}$$

$$P_{i:i-1} = (a_i + b_i).(a_{i-1} + b_{i-1}) \tag{12}$$

The two bit group generate and group propagate defined in Equations (11) and (12) can be defined as the intermediate generate G_i^* and the intermediate propagate P_i^* signals according to Dimitrakopoulos and Nikolos (2005). The intermediate generate G_i^* and intermediate propagate P_i^* signals can also be produced by combining the two consecutive bit generate and two consecutive bit propagate signals as expressed by

$$G_i^* = g_i + g_{i-1} \tag{13}$$

$$P_i^* = p_i \cdot p_{i-1} \tag{14}$$

Where, $0 \le i \le n-1$ with, $g_{-1} = p_{-1} = 0$ and $G_k^* = P_k^* = 0$ for $k < 0$.

For an 8-bit adder based on the Ling equations, by using Equation (6), the fourth Ling carry can be expressed as

$$H_4 = g_4 + g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0$$
$$= (g_4 + g_3) + p_3 \cdot p_2 \cdot (g_2 + g_1) + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot g_0 = G_{4:3} + P_{3:2} \cdot G_{2:1} + P_{3:2} \cdot P_{1:0} \cdot G_{0:-1}$$

$$H_4 = (G_{4:3}, P_{3:2}) \circ (G_{2:1}, P_{1:0}) \circ (G_{0:-1}, P_{-1:-2}) = (G_4^*, P_3^*) \circ (G_2^*, P_1^*) \circ (G_0^*, P_{-1}^*) \quad (15)$$

Similarly H_5 can be expanded as,

$$H_5 = (G_5^*, P_4^*) \circ (G_3^*, P_2^*) \circ (G_1^*, P_0^*) \quad (16)$$

As seen from Equations (15) and (16), by using the intermediate generate and propagate pairs (G_i^*, P_{i-1}^*) and by treating separately the Ling carries of the even and the odd-indexed bit positions, the final pseudo carries can be derived. As the pseudo carries of the odd and the even-indexed bit positions are computed independently, the lateral fan-out of the prefix tree reduces which in turn reduces the delay. Since there is no interference between the prefix trees of the even and the odd bit positions, separate tree topologies can be used to further jointly optimize the delay, area and power. By induction, the Ling pseudo carries H_i and H_{i+1} of the consecutive even and odd bit positions *i* and *i+1*, respectively are given by

$$H_i = (G_i^*, P_{i-1}^*) \circ (G_{i-2}^*, P_{i-3}^*) \circ \ldots\ldots\ldots\ldots \circ (G_0^*, P_{-1}^*) \quad (17)$$

$$H_{i+1} = (G_{i+1}^*, P_i^*) \circ (G_{i-1}^*, P_{i-2}^*) \circ \ldots\ldots\ldots\ldots \circ (G_1^*, P_0^*) \quad (18)$$

Fig.1 shows 8 bit Hybrid parallel prefix adder based on the Ling recursion. In this adder structure, three stages are required for the computation of Ling carry bits. The black circular nodes and the black square nodes of Fig.1 are defined as shown in Fig.2 (a) and Fig.2 (b).Finally, the sum bits are produced in the fourth stage represented by the diamond nodes in Fig.1. The definition of the diamond node is depicted in Fig.2 (c).

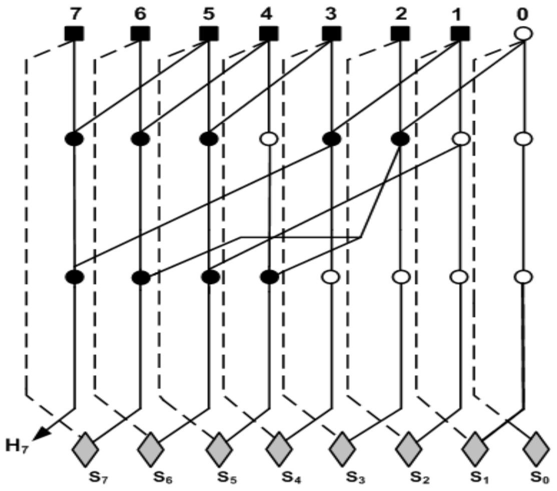


Fig. 1. 8 bit Hybrid prefix adder based on Ling equation.

Fig.3 and Fig.4 show the 16-bit and 32-bit hybrid adders based on Ling equations. Since each sum bit is produced with a multiplexor instead of a simple XOR gate, it demands more area than the conventional prefix adders. Furthermore, the most significant Ling carry generated does not represent the actual carry-out of the adder and the generation of the carry-out requires an additional AND gate. These drawbacks can be eliminated by generating the real carries instead of the pseudo carries. Hence, the diamond nodes of Fig.1can be replaced by simple XOR gates, which can reduce the implementation area.

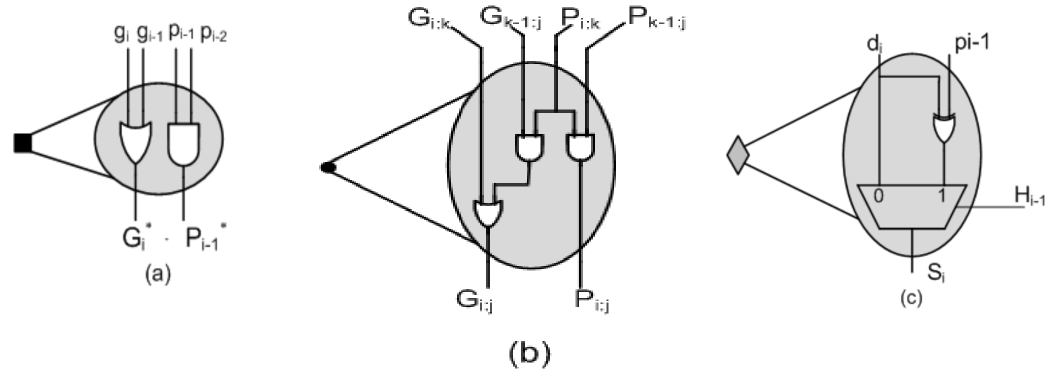
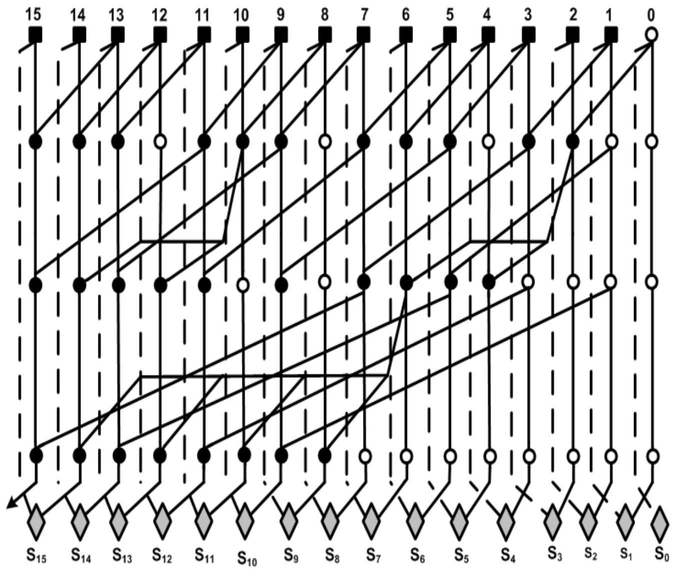


Fig. 2.a)The node for computing intermediate generate and propagate, (b)prefix nodes for computing group generate and group propagate,(c) Sum computation node.



.

Fig. 3. 16 bit Hybrid parallel prefix adder based on Ling equations.

In Fig.1 the pseudo carries H_0 to H_3 are available after the second level. Then according to (8), the multiplication of respective propagate with these pseudo carries produces the real carries c_0 to c_3 which is achieved by taking logical AND of pseudo carries with propagate terms. The computations of real carries c_0 to c_3 are given by

$$p_0 \cdot H_0 = p_0 \cdot g_0 = c_0 \tag{19}$$
$$p_1 \cdot H_1 = p_1 .(g_1 + g_0) = g_1 + p_1 \cdot g_0 = c_1 \tag{20}$$
$$p_2 \cdot H_2 = p_2 .((g_2 + g_1) + p_1 \cdot p_0 \cdot g_0) = c_2 \tag{21}$$
$$p_3 \cdot H_3 = p_3 .((g_3 + g_2) + p_2 \cdot p_1 .(g_1 + g_0)) = c_3 \tag{22}$$

The pseudo carries after level 2 in Fig.1 are expressed as

$$H_4 = (G_4^*, P_3^*) \circ (G_2^*, P_1^*) \circ (G_0^*, P_{-1}^*) = G_{4:3} + P_{3:2} \cdot H_2 \tag{23}$$
$$H_5 = (G_5^*, P_4^*) \circ (G_3^*, P_2^*) \circ (G_1^*, P_0^*) = G_{5:2} + P_{4:1} \cdot H_1 \tag{24}$$
$$H_6 = (G_6^*, P_5^*) \circ (G_4^*, P_3^*) \circ (G_2^*, P_1^*) \circ (G_0^*, P_{-1}^*) = G_{6:3} + P_{5:2} \cdot H_2 \tag{25}$$
$$H_7 = (G_7^*, P_6^*) \circ (G_5^*, P_4^*) \circ (G_3^*, P_2^*) \circ (G_1^*, P_0^*) = G_{7:4} + P_{6:3} \cdot H_3 \tag{26}$$

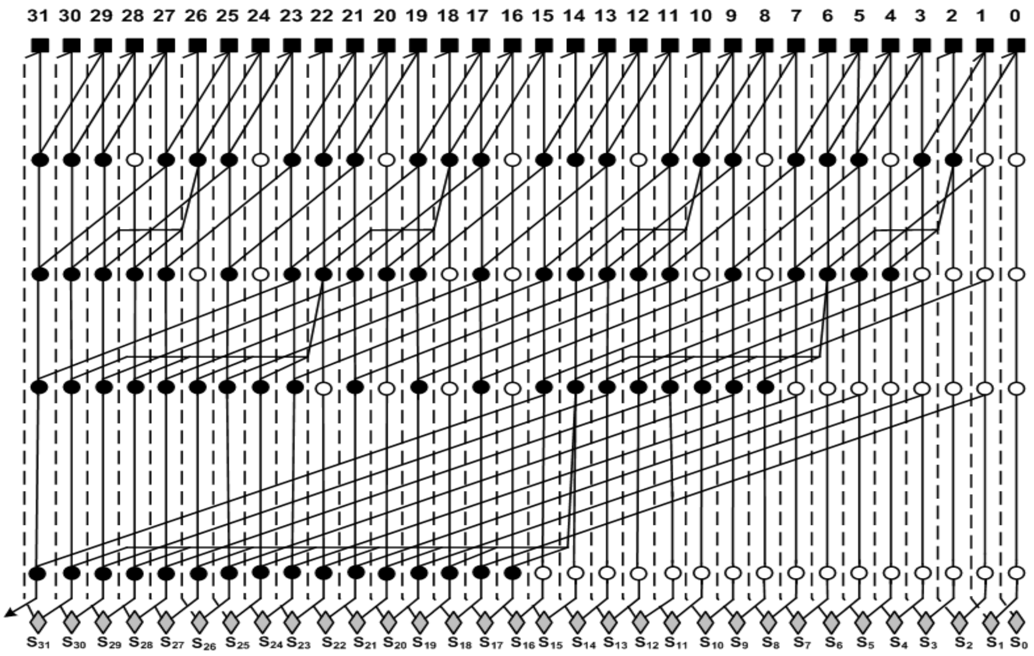

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
S31 S30 S29 S28 S27 S26 S25 S24 S23 S22 S21 S20 S19 S18 S17 S16 S15 S14 S13 S12 S11 S10 S9 S8 S7 S6 S5 S4 S3 S2 S1 S0

Fig. 4. 32 bit Hybrid parallel prefix adder based on Ling equations.

In Equations (23) to (26), $G_{i:j}$ and $P_{i:j}$ are the group generate and group propagate signals from the bit positions j to i respectively. The computation of the real carries for the most significant group bits c_4 to c_7 requires $P_{i:0}$. However, in Equations (23)-(26), we have $P_{i-1:0}$ Hence, by modifying the level 3 prefix cells as depicted in Fig.5 the real carries can be produced. Then, the final sum bits can be computed by using the XOR gates. The overall delay for proposed 8-bit architecture for the carry computation is found to be six logic gates, which is one more than the same adder architecture based on Ling equations. Further, as XOR gates are used in place of the multiplexors for the sum computation, the area cost for the proposed method is less than that of the adders based on the Ling equations.

Fig.6 depicts proposed 16 bit adder based on modified Ling equations with the new prefix units in the last stage. The proposed method is extended to design the 32 bit hybrid parallel-prefix adder as shown in Fig. 7, where sixteen pentagonal nodes in level 4 produces real carries for higher group (i.e., for the bit-positions 16 to 31), and sixteen AND gates in level 4 computes real carries for the lower group (i.e., for the bit positions 0 to15). In these architectures, the nodes represented by star also computes the group generate and group propagate signals based on Equations (17) and (18).

5. Synthesis Results and Discussions

The proposed adders are coded using Verilog structural description and synthesized using 0.18μm CMOS technology. The hybrid adder architectures were implemented using both the conventional Ling recurrence equations and the modified Ling equations. Table 1 shows the delay comparison of the proposed adders with that of the adders based on the conventional Ling equations. Proposed adders can achieve delay saving up to 24.85%. The savings in delay is achieved due to the reduced fan out of the prefix tree which is due to the usage of separate prefix tree for odd and even bits. Table 3 depicts the power comparison of the proposed adders with the conventional hybrid Ling adders. The results show that the proposed adders are power efficient also.

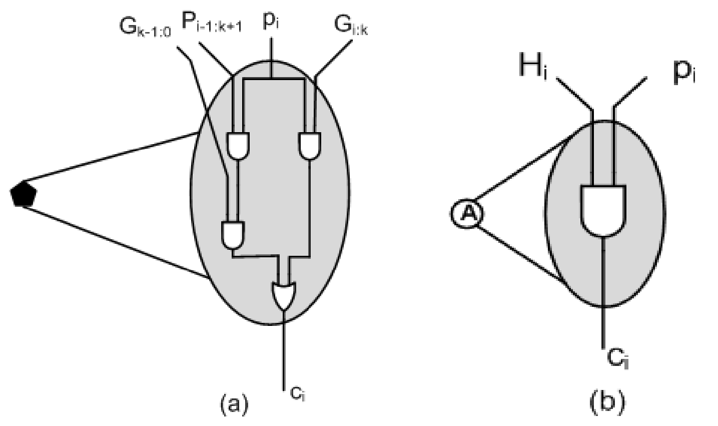


Fig. 5. Proposed modified prefix cells in the last stage(a) the nodes for bits n/2 to n-1 (b) the nodes for 0 to(n/2)-1.

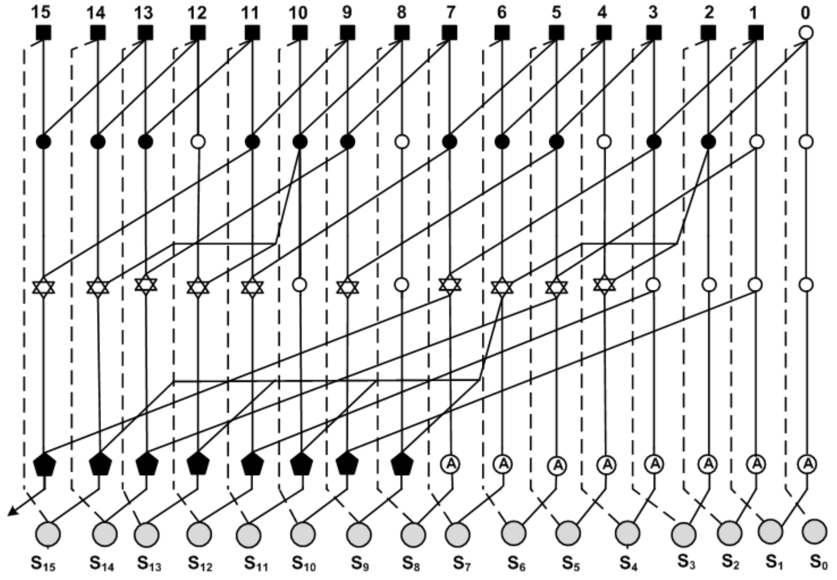


Fig.6. Proposed 16 bit Hybrid parallel prefix adder based on Modified Ling equations.

Table 2 shows the area comparisons for proposed hybrid adder with the hybrid adder based on the conventional Ling equations. As can be observed from Table 2, proposed adder can save up to 10% of area. The reduction in area is achieved since the sum bits are computed using only the XOR gates without using the multiplexers. In addition to this fact, the hybrid tree structure employed also contributes to the reduced gate count, thereby improving upon the

Table. 1. Comparison of Delay (in pS) of proposed adders with adders based on Ling equations

Number of bits, n	Adder based on Conventional Ling recurrence	Adder based on modified Ling recurrence	Savings
16	986	741	24.85%
32	1286	999	22.31%

area cost factor. The savings obtained in the area-power and power-delay products for the 16-bit and 32-bit adders are plotted as shown in Fig. 8.As can be seen from the plots, with the proposed method a nominal savings of 23.5% and 35% can be achieved for area-power and power-delay products respectively.

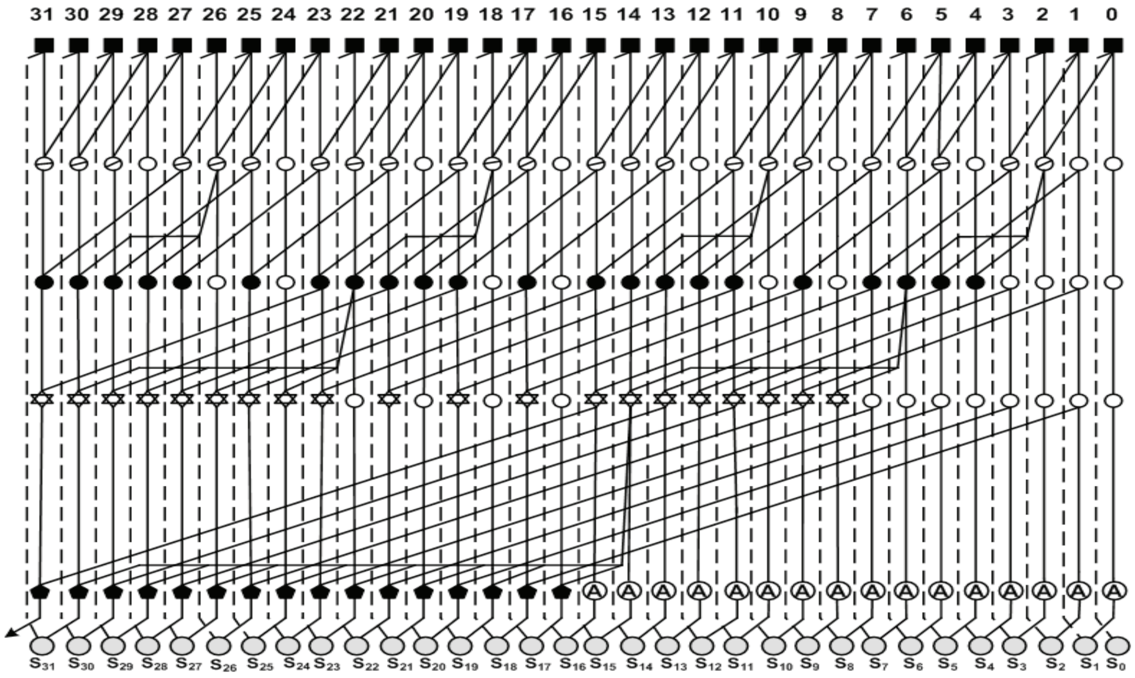

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
A
S_{31} S_{30} S_{29} S_{28} S_{27} S_{26} S_{25} S_{24} S_{23} S_{22} S_{21} S_{20} S_{19} S_{18} S_{17} S_{16} S_{15} S_{14} S_{13} S_{12} S_{11} S_{10} S_9 S_8 S_7 S_6 S_5 S_4 S_3 S_2 S_1 S_0

Fig. 7. Proposed 32 bit Hybrid parallel prefix adder based on modified Ling equations

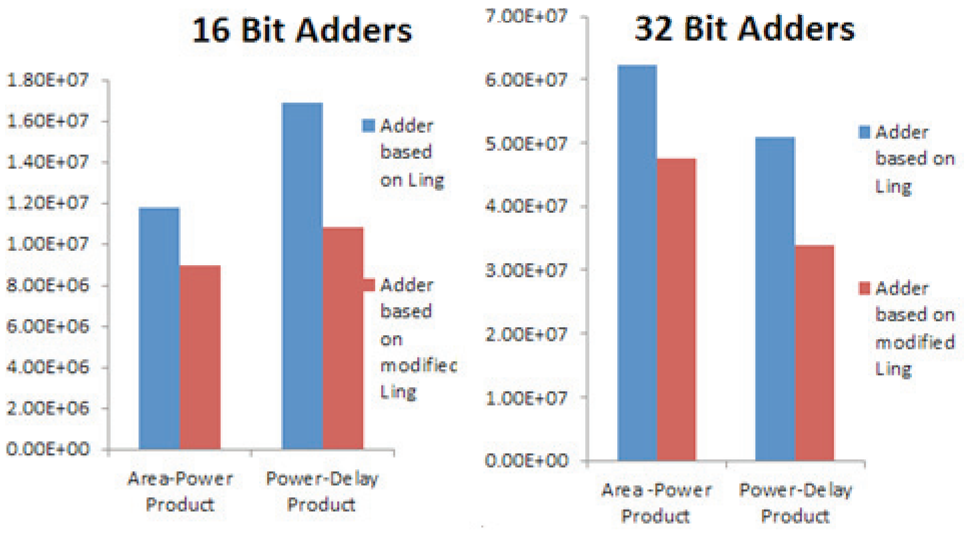


Fig. 8. Comparison of Area-Power product and Power-Delay product for 16 bit adders and 32 bit adders.

Table. 2 .Comparison of Area(in μm^2) of Proposed adders with adders based on Ling equations.

Number of bits, n	Adder based on Conventional Ling recurrence	Adder based on modified Ling recurrence	Savings
16	686	620	9.5%
32	1570	1409	10.25%

Table.3 .Comparison of Power (in nW) of Proposed adders with adders based on Ling equations

Number of bits, n	Adder based on Conventional Ling recurrence	Adder based on modified Ling recurrence	Savings
16	17154.773	14542.914	15.22%
32	39636.755	33776.734	14.78%

6. Conclusions

In this paper, we have proposed a novel technique for area efficient realization of parallel-prefix Ling adders using modified parallel-prefix units. Synthesis results demonstrate significant saving of the silicon area while comparing with the existing Ling adders. Reduction of 9.5% and 10.25% for the 16 and 32-bit adders respectively were realized. The proposed method in addition to the reduced area, incurs less delay and power dissipation. The synthesis results reveal that the proposed adders could achieve up to 24 % and 35% saving of area-power product and power-delay product respectively against the conventional Ling adders.

In the proposed adder, the complexity of the generate term is reduced only in the first level of prefix tree. By reducing the complexity at all levels of carry graph, further optimization of the performance parameters can be achieved, which will be the extended work of the paper.

References

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

Dimitrakopoulos, G.; Nikolos, D., 2005. High-speed parallel-prefix VLSI Ling adders, *Computers, IEEE Transactions* on , vol.54, no.2, pp.225-231.

Han.T and Carlson.D., 1987. Fast area-efficient VLSI adders, in *Proceedings of IEEE Symposium on Computer Arithmetic*, pp.49–56.

Kao.S, Zlatanovici.R, and Nikolic´.B.,2006. A 240 ps 64b carry-lookahead adder in 90 nm CMOS, *IEEE ISSCC Dig. Tech. Papers,* pp. 438–439.

Knowles, S., 2001.A family of adders, *Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium*, vol., no., pp.277,281.

Kogge.P.M. and Stone.H.S.,1973. 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.

Ladner.R.E. and Fischer.M.J., 1980. Parallel prefix computation,*Journal of the ACM*, vol. 27, no. 4, pp. 831–838.

Ling.H., 1981.High speed binary adder,*IBM J. Res. Develop.*, vol. 25, no.3, pp. 156–166.

Mathew.S., Krishnamurthy.R., Anders.M., Rios.R., Mistry.K and Soumyanath.K., 2001.Sub-500 ps 64b ALUs in 0.18 m SOI/bulk CMOS: Design and scaling trends, *IEEE ISSCC Dig. Tech. Papers*, pp. 318–319.

Naffziger.S, 1966. A sub-nanosecond 0.5 m 64b adder design, *IEEEISSCC Dig. Tech. Papers*, pp. 210–211.

Shimazaki.Y.,Zlatanovici.R.and Nikolic´.B., 2004. A shared-well dualsupply-voltage 64-bit ALU,*IEEE J. Solid-State Circuits*, vol. 39, pp.494–500.

Sklansky.J., 1960. Conditional-sum addition logic,*IRE Transactionson Electronic Computers*, vol. 9, pp. 226–231.