International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:4, 2009

Hybrid Prefix Adder Architecture for Minimizing the Power Delay Product

P.Ramanathan[1] and.P.T.Vanathi [2]

***Abstract*—**Parallel Prefix addition is a technique for improving the speed of binary addition. Due to continuing integrating intensity and the growing needs of portable devices, low-power and high-performance designs are of prime importance. The classical parallel prefix adder structures presented in the literature over the years optimize for logic depth, area, fan-out and interconnect count of logic circuits. In this paper, a new architecture for performing 8-bit, 16-bit and 32-bit Parallel Prefix addition is proposed. The proposed prefix adder structures is compared with several classical adders of same bit width in terms of power, delay and number of computational nodes. The results reveal that the proposed structures have the least power delay product when compared with its peer existing Prefix adder structures. Tanner EDA tool was used for simulating the adder designs in the TSMC 180 nm and TSMC 130 nm technologies.

***Keywords*—**Parallel Prefix Adder (PPA), Dot operator, Semi-Dot operator, Complementary Metal Oxide Semiconductor (CMOS), Odd-dot operator, Even-dot operator, Odd-semi-dot operator and Even-semi-dot operator.

I. Introduction

VLSI Integer adders find applications in Arithmetic and Logic units (ALU's), microprocessors and memory addressing units. Speed of the adder often decides the minimum clock cycle time in a microprocessor. The need for a Parallel Prefix adder is that it is primarily fast when compared with ripple carry adders. Parallel Prefix adders (PPA) are family of adders derived from the commonly known carry look ahead adders. These adders are best suited for adders with wider word lengths. PPA circuits use a tree network to reduce the latency to $O(\log_2 n)$ where 'n' represents the number of bits. A three step process is generally involved in the construction of a PPA. The first step involves the creation of generate, complementary kill and propagate signals for all the input operand bits.

$$G_i = A_i \bullet B_i \tag{1}$$

$$K_i = \overline{A_i + B_i} = \overline{A_i} \bullet \overline{B_i} \tag{2}$$

$$P_i = \overline{A_i \oplus B_i} \tag{3}$$

[1]P.Ramanathan is with the PSG College of Technology as a Senior Lecturer in Electronics and Communication Engineering Department, Peelamedu, Coimbatore – 641 004, Tamilnadu, India (corresponding author phone: +91-9443823221; fax: +91-422-2572477; e-mail: pramanathan_2000@yahoo.com).

[2]Dr.P.T.Vanathi., is with PSG College of Technology as a Assistant Professor in Electronics and Communication Engineering Department, Peelamedu, Coimbatore – 641 004, Tamilnadu, India. (Email : ptvani@yahoo.com).

Second step involves the generation of carry signals. In PPA, the dot operator '$\bullet$' and the semi-dot operator '$\circ$' are introduced. The dot operator '$\bullet$' is defined by the equation (4) and the semi-dot operator '$\circ$' is defined by the equation (5)

$$(g_i, \overline{k_i}) \bullet (g_{i-1}, \overline{k_{i-1}}) = (g_i, +\overline{k_i} g_{i-1}, \overline{k_i} \cdot \overline{k_{i-1}}) \tag{4}$$

$$(g_i, \overline{k_i}) \circ (g_{i-1}, \overline{k_{i-1}}) = (g_i, +\overline{k_i} g_{i-1}) \tag{5}$$

In the above equation, '$\bullet$' operator is applied on two pairs of bits $(g_i, \overline{k_i})$ and $(g_{i-1}, \overline{K_{i-1}})$. These bits represent generate and propagate signals used in addition. The output of the operator is a new pair of bits which is again combined using a dot operator '$\bullet$' or semi-dot operator '$\circ$' with another pairs of bits. This procedural use of dot operator '$\bullet$' and semi-dot operator '$\circ$' creates a prefix tree network which ultimately ends in the generation of all carry signals. In the final step, the sum bits of the adder are generated with the propagate signals of the operand bits and the preceding stage carry bit using a xor gate. The semi-dot operator '$\circ$' will be present as last computation node in each column of the prefix graph structures, where it is essential to compute only generate term, whose value is the carry generated from that bit to the succeeding bit.

The structure of the prefix network specifies the type of the PPA. The Prefix network described by Haiku Zhu, Chung-Kuan Cheng and Ronald Graham [1], has the minimal depth for a given 'n' bit adder. Optimal logarithmic adder structures with a fan-out of two for minimizing the area-delay product is presented by Matthew Ziegler and Mircea Stan [2]. The Sklansky adder [3] presents a minimum depth prefix network at the cost of increased fan-out for certain computation nodes. The algorithm invented by Kogge-Stone [4] has both optimal depth and low fan-out but produces massively complex circuit realizations and also account for large number of interconnects. Brent-Kung adder [5] has the merit of minimal number of computation nodes, which yields in reduced area but structure has maximum depth which yields slight increase in latency when compared with other structures. The Han-Carlson adder [6] combines Brent-Kung and Kogge-Stone structures to achieve a balance between logic depth and interconnect count. Knowles [7] presented a class of logarithmic adders with minimum depth by allowing the fan-out to grow. Ladner and Fischer [8] proposed a general method to construct a prefix network with slightly higher depth when compared with Sklansky topology but achieved

some merit by reducing the maximum fan-out for computation nodes in the critical path. Related work on PPA literature such as Ling adder [9], achieve improved performance gains by changing the equation of the dot operator '●'. Taxonomy of classical Prefix Parallel Adders based on fan-out, interconnect count and depth characteristics has been presented by Harris [10]. In this paper, a novel hybrid prefix adder structure for 8-bit, 16-bit and 32-bit has been proposed. The proposed structures have the least power delay product amongst all its peer one's.

II. EXISTING PARALLEL PREFIX ADDERS

Brent Kung adder is oriented towards simpler tree structure with a fewer computation nodes. Kogge-Stone adder possesses a regular layout and is preferred for high performance applications. Han-Carlson adder the reduces the hardware complexity when compared to that of a Kogge-Stone adder but at the cost of introducing a additional stage to its carry merge path. In Sklansky adder, the fan-out from the inputs to outputs along the critical path increases drastically which introduces latency in the circuit. Ladner-Fischer adder is an improved version of Sklansky adder, where the maximum fan-out is reduced. TABLE I. summarizes the data regarding the requirement of number of computation nodes and logic depth for various existing Parallel Prefix adders. Let 'n' be the word-length of the adder in terms of bits.

TABLE I
COMPARATIVE STUDY ON EXISTING PREFIX ADDERS

Adder Type	Number of Computation Nodes	Logic Depth
Brent-Kung	$[2*n-2-\log_2 n]$	$[(2*\log_2 n)-2]$
Kogge-Stone	$[(n*\log_2 n)-n+1]$	$\log_2 n$
Han-Carlson	$[\frac{n}{2}*(\log_2 n)]$	$[(\log_2 n)+1]$
Ladner-Fischer	$[\frac{n}{2}*(\log_2 n)]$	$[(\log_2 n)+1]$
Sklansky	$[\frac{n}{2}*(\log_2 n)]$	$\log_2 n$

III. PROPOSED PREFIX ADDER STRUCTURES

The Proposed 8-bit, 16-bit and 32-bit parallel prefix adder architectures are shown in Fig. 1, Fig. 2 and Fig. 3 respectively. The first stage in the architectures of the proposed prefix adder structures involve the creation of kill and complementary generate for individual operand bits using the equations (6) and (7)

$$K_i = \overline{A_i + B_i} = \overline{A}_i . \overline{B_i} \quad (6)$$

$$\overline{G_i} = (\overline{A_i \cdot B_i}) \quad (7)$$

In the above equations, A_i, B_i represent input operand bits for the adder, where 'i' varies from 0 to 7 for 8-bit, 0 to 15 for 16-bit and 0 to 31 for 32-bit adders respectively. The Propagate signal is derived using the generate and kill signals and is given by

$$P_i = \overline{G_i + K_i} \quad (8)$$

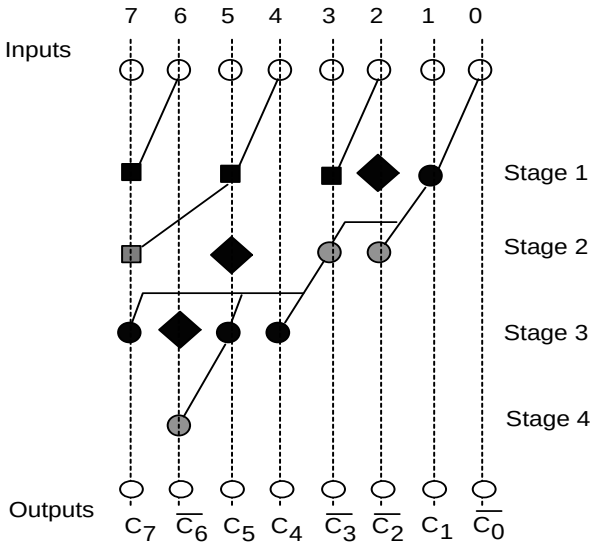

Fig. 1 Proposed 8-bit Parallel Prefix Adder

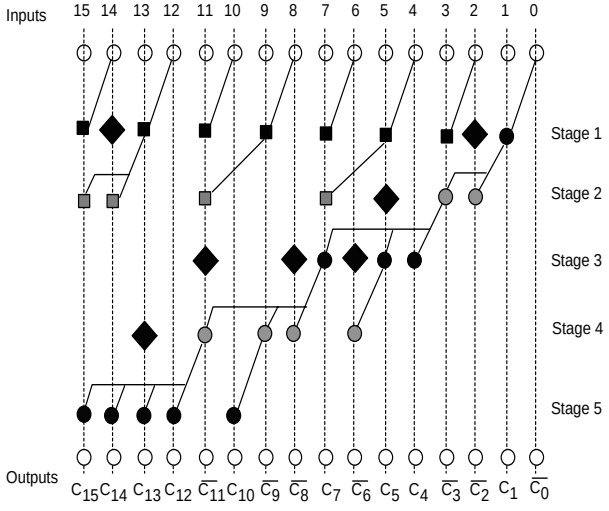

Fig. 2 Proposed 16-bit Parallel Prefix Adder

All the proposed designs are implemented using CMOS logic family. For deriving the carry signals in the second stage, this architecture introduces four different computation nodes for achieving improved performance. There are two cells designed for the dot operator.

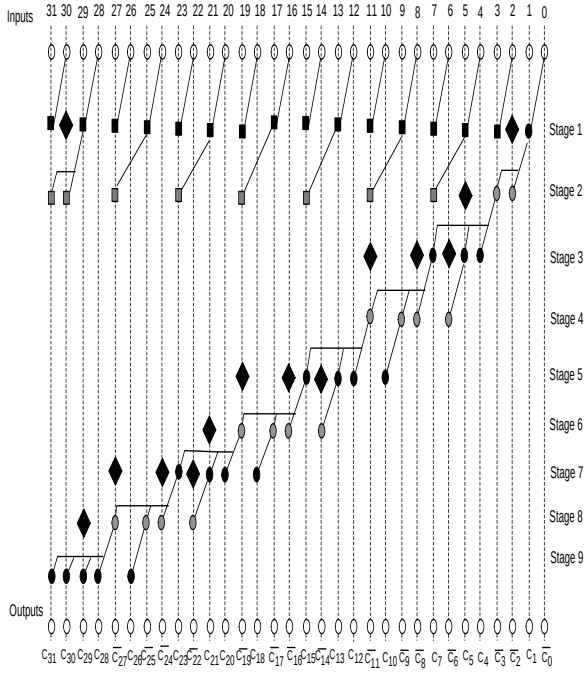

Fig. 3 Proposed 32-bit Parallel Prefix Adder

First cell for the dot operator named odd-dot represented by a '■', is defined by the equation (9)

$$(g,\bar{k}) = (\bar{g_i}, k_i) \blacksquare (\overline{g_{i-1}}, k_{i-1}) = (\overline{\bar{g_i} \cdot (k_i + \overline{g_{i-1}})},\ \overline{k_i + k_{i-1}}) \quad (9)$$

The second cell for the dot operator named even-dot represented by a '■', is defined by the equation (10)

$$(\bar{g},k) = (g_i, \bar{k_i}) \blacksquare (g_{i-1}, \overline{k_{i-1}}) = (\overline{g_i + \bar{k_i}\ g_{i-1}},\ \overline{\bar{k_i} \cdot \overline{k_{i-1}}}) \quad (10)$$

Similarly, there are two cells designed for the semi-dot operator. First cell for the semi-dot operator named odd-semi-dot represented by a '●', the second cell for the semi-dot operator named even-semi-dot represented by a '●', works are defined using equations (11) and (12) respectively.

$$(g) = (\bar{g_i}, k_i) \bullet (\overline{g_{i-1}}, k_{i-1}) = (\overline{\bar{g_i} \cdot (k_i + \overline{g_{i-1}})}) \quad (11)$$

$$(\bar{g}) = (g_i, \bar{k_i}) \bullet (g_{i-1}, \overline{k_{i-1}}) = (\overline{g_i + \bar{k_i}\ g_{i-1}}) \quad (12)$$

The stages with odd indexes use odd-dot and odd-semi-dot cells where as the stages with even indexes use even-dot and even-semi-dot cells.

CMOS logic family will implement only inverting functions. Thus cascading odd cells and even cells alternatively gives the benefit of elimination of two inverters between them, if a dot or a semi-dot computation node in an odd stage receives both of its input edges from any of the even stages and vice-versa. But it is essential to introduce two inverters in a path, if a dot or a semi-dot computation node in an even stage receives any of its edges from any of the even stages and vice-versa. From the prefix graph of the proposed structures, we observe that there are only few edges with a pair of inverters, to make $(\bar{g}, k)$ as $(g, \bar{k})$ or to make $(g, \bar{k})$ as $(\bar{g}, k)$ respectively. The pair of inverters in a path is represented by a '◆' in the prefix graph. By introducing two cells for dot operator and two cells for semi-dot operator, we have eliminated a large number of inverters. Due to inverter elimination in paths, the propagation delay in these paths has reduced. Further we achieve a benefit in power reduction, since these inverters if not eliminated, would have contributed to significant amount of power dissipation due to switching. The output of the odd-semi-dot cells gives the value of the carry signal in that corresponding bit position. The output of the even-semi-dot cell gives the complemented value of carry signal in that corresponding bit position. The final stage involves generation of sum bits from the derived Propagate signals of the individual operand bits and the carry bits generated in true form or complement form.

IV. Simulation environment

Simulation of various Parallel Prefix Adder designs were carried out with Tanner EDA tool using TSMC 180 nm and TSMC 130 nm technologies. All the Parallel Prefix Adder structures were implemented using CMOS logic family. The aspect ratio of the MOS transistor devices were chosen such that $\left(\frac{W}{L}\right)_n = 3 * \left(\frac{W}{L}\right)_p$. For TSMC 180 nm technology, threshold voltages of NMOS and PMOS transistors are around 0.3694 V and -0.3944 V respectively and the supply voltage was kept at 1.8 V.. For TSMC 130 nm technology, threshold voltages of NMOS and PMOS transistors are around 0.332 V and -0.3499 V respectively and the supply voltage was kept at 1.3 V. The parameters considered for comparison are power consumption, worst case delay and power-delay product. The various PPA structures were then compared with the number of computation nodes needed for circuit realizations.

V. Results and discussion

TABLE II, TABLE III and TABLE IV list out the structural characteristics for various 8-bit, 16-bit and 32-bit Parallel Prefix adders. From the tables it is observed that the proposed 8-bit, 16-bit and 32-bit Parallel Prefix adder has the least number of computation nodes amongst all other peer designs. Structure of the proposed 32-bit adder also reveals that the prefix tree builds along the main diagonal after the first two stages.

TABLE II
CHARACTERISTICS OF 8-BIT PARALLEL PREFIX ADDERS

Adder Type	Number of Computation Nodes		Logic Depth
	Dot	Semi-Dot	
Brent-Kung	4	7	4
Kogge-Stone	10	7	3
Han-Carlson	5	7	4
Ladner-Fischer	5	7	3
Sklansky	5	7	4
Proposed	4	7	4

TABLE III
CHARACTERISTICS OF 16--BIT PARALLEL PREFIX ADDERS

Adder Type	Number of Computation Nodes		Logic Depth
	Dot	Semi-Dot	
Brent-Kung	11	15	6
Kogge-Stone	34	15	4
Han-Carlson	17	15	5
Ladner-Fischer	17	15	4
Sklansky	17	15	5
Proposed	11	15	5

TABLE IV
CHARACTERISTICS OF 32-BIT PARALLEL PREFIX ADDERS

Adder Type	Number of Computation Nodes		Logic Depth
	Dot	Semi-Dot	
Brent-Kung	26	31	8
Kogge-Stone	98	31	5
Han-Carlson	33	31	6
Ladner-Fischer	33	31	6
Sklansky	33	31	5
Proposed	23	31	9

From the TABLES II, III and IV we infer that the proposed structures have the least number of Dot operator computation nodes. TABLE V, TABLE VI and TABLE VII list out the performance comparison of the various 8-bit, 16-bit and 32-bit Parallel Prefix adders in 180 nm technology. TABLE VIII, TABLE IX and TABLE X list out the performance comparison of various 8-bit, 16-bit and 32-bit Parallel Prefix adders in 130 nm technology.

TABLE V
PERFORMANCE COMPARISON OF 8-BIT PARALLEL PREFIX ADDERS USING 180 nm TECHNOLOGY

Adder Name	Average Power (μW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	51.14196	0.72	36.8222112
Kogge-Stone	59.99514	0.47	28.1977158
Han-Carlson	50.64183	0.64	32.4107712
Sklansky	52.14018	0.49	25.5486882
Ladner-Fischer	51.30227	0.57	29.2422939
Proposed	48.835	0.76	37.1146

TABLE VI
PERFORMANCE COMPARISON OF 16-BIT PARALLEL PREFIX ADDERS USING 180 nm TECHNOLOGY

Adder Name	Average Power (μW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	100.5192	1.08	111.800736
Kogge-Stone	140.2228	0.83	116.384924
Han-Carlson	109.24903	1.05	114.7114815
Sklansky	110.9823	1.02	113.201946
Ladner-Fischer	107.9553	1.06	114.432618
Proposed	97.84248	1.14	111.5404272

TABLE VII
PERFORMANCE COMPARISON OF 32-BIT PARALLEL PREFIX ADDERS USING 180 nm TECHNOLOGY

Adder Name	Average Power (μW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	216.0853	1.39	300.358567
Kogge-Stone	350.2907	0.92	322.267444
Han-Carlson	237.1478	1.25	296.43475
Sklansky	263.7364	1.11	292.747404
Ladner-Fischer	226.5094	1.23	278.606562
Proposed	204.2825	1.32	269.6529

TABLE VIII
PERFORMANCE COMPARISON OF 8-BIT PARALLEL PREFIX ADDERS USING 130 nm TECHNOLOGY

Adder Name	Average Power (µW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	14.37045	0.65	9.3407925
Kogge-Stone	17.42589	0.33	5.7505437
Han-Carlson	14.61578	0.45	6.577101
Sklansky	14.98214	0.33	4.9441062
Ladner-Fischer	14.65045	0.42	6.153189
Proposed	14.0836	0.67	9.436012

TABLE IX
PERFORMANCE COMPARISON OF 16-BIT PARALLEL PREFIX ADDERS USING 130 nm TECHNOLOGY

Adder Name	Average Power (µW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	29.10129	0.77	22.4079933
Kogge-Stone	41.31562	0.52	21.4841224
Han-Carlson	29.82515	0.67	19.9828505
Sklansky	32.68943	0.6	19.613658
Ladner-Fischer	29.65798	0.71	21.0571658
Proposed	23.955	0.79	18.92445

TABLE X
PERFORMANCE COMPARISON OF 32-BIT PARALLEL PREFIX ADDERS USING 130 nm TECHNOLOGY

Adder Name	Average Power (µW)	Delay (ns)	Power-Delay Product (X 10^{-15} Joules)
Brent-Kung	61.30189	1.02	62.5279278
Kogge-Stone	99.05527	0.65	64.3859255
Han-Carlson	68.65387	0.91	62.4750217
Sklansky	72.4006	0.82	59.368492
Ladner-Fischer	62.27339	0.93	57.9142527
Proposed	53.0858	0.97	51.493226

VI. Conclusion

It is observed that the proposed 16-bit and 32-bit prefix adders have the least value of power delay product when compared with its existing peer ones. The proposed design achieves about 20% to 40 % power savings when compared to Kogge-Stone adder. The proposed design attains an improvement of 5% to 14% power savings when compared with Brent-Kung adder. A benefit of 7% to 25% power saving is achieved in the proposed design when compared with Sklansky adder. A benefit of 5% to 22% improvement in power can be attained for the proposed adders when compared with Han-Carlson adder. It is inferred that the power saving gradually increases for the proposed design when the size of the prefix adder grows. It is also observed that the power delay product is minimal when the word length of the adder increases. The proposed design consumes least power with very less delay penalty when compared with its peer existing prefix adders. This significant improvement in power makes the proposed design more suited for ALU's, Multipliers and in Digital Signal Processors.

References

[1] Haikun Zhu, Chung-Kuan Cheng and Ronald Graham, "Constructing Zero Deficiency Parallel Prefix adder of Minimum Depth," Proceedings of 2005 Asia South Pacific Design Automation Conference, 2005, pp. 883–888.

[2] Matthew Ziegler, Mircea Stan, "Optimal Logarithmic Adder structures with a fan-out of two for minimizing area delay product," IEEE 2001.

[3] J. Sklansky, "Conditional Sum Addition Logic," IRE Transactions on Electronic computers, vol. EC-9, 1960, pp. 226-231.

[4] P.Kogge and H.Stone, "A Parallel Algorithm for the efficient solution of a general class of recurrence relations," IEEE Transactions on Computers, vol. C-22, no.8, August 1973, pp.786-793.

[5] R.Brent and H.Kung, "A Regular Layout for Parallel adders," IEEE Transaction on Computers, vol. C-31, no.3, March 1982, pp. 260-264.

[6] T. Han and D. Carlson, "Fast Area Efficient VLSI adders," Proceedings of the 8th Symposium on Computer Arithmetic, September 1987, pp.49-56.

[7] S.Knowles, "A Family of Adders", Proceeding of the 15th IEEE Symposium on Computer Arithmetic, June 2001, pp.277-281.

[8] R. Ladner and M. Fischer, "Parallel Prefix Computation," Journal of ACM, vol.27,no.4, October 1980, pp. 831-838.

[9] Giorgos Dimitrakopoulos and Dimitris Nikolos, "High Speed Parallel Prefix VLSI Ling adders," IEEE Transactions on Computers, vol.54, no.2, February 2005, pp. 225-231.

[10] David Harris, " A Taxonomy of Parallel Prefix Networks," Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers, 2003, pp.2213-2217.

P.Ramanathan is currently working as a Senior Lecturer in Electronics and Communication department, PSG College of Technology, Peelamedu, Coimbatore India.. He has about 10 years of teaching experience and is currently doing research in the area of VLSI Design. .

Dr.P.T.Vanathi is currently working as an Assistant Professor in Electronics and Communication department, PSG College of Technology, Coimbatore India.. She has about 20 years of teaching experience. Her research are Speech Signal Processing, Image Processing and VLSI Design. .