Implementation of 32-Bit Parallel Prefix Adders and Comparative Study forFastest Response

CHILAKA LYDIA[1], Dr. M. ASHOK KUMAR[2]

[1] P.G Scholor, PYDAH College of Engineering & Technology, Visakhapatnam, Andhra Pradesh,

[2] Associate Professor, PYDAH College of Engineering & Technology, Visakhapatnam, Andhra Pradesh.

Abstract— In this paper, we propose 32 bit Kogge-Stone, Spanning tree parallel prefix adders. In general N-bit adders like Ripple Carry Adders (slow adders compare to other adders), and Carry Look Ahead adders (area consuming adders) and Carry Save Adder are used in earlier days. But now the most Industries are using parallel prefix adders because of their advantages compare to other adders. Parallel prefix adders are faster and area efficient. Parallel prefix adder is a technique for increasing the speed in DSP processor while performing addition. We simulate and synthesis different types of 32-bit prefix adders using Xilinx ISE 14.7 tool. By using these synthesis results, we noted the performance parameters like number of area and delay. We compare these five adders in terms of LUTs (represents area) and delay values.

Keywords− prefix adder, carry operator, Kogge-Stone, Spanning Tree, Carry Save Adder, Carry Look Ahead Adder, Ripple Carry Adder.

I.INTRODUCTION

VLSI Integer adders find applications in Arithmetic and Logic Units (ALUs), 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 (PPA) is that it is primarily fast when compared with a ripple carry adder. PPA is a family of adders derived from the commonly known carry look ahead adders. These adders are suited for additions with wider word lengths. PPA circuits use a tree network to reduce the latency to 2 (log) O n where ‘n’ represents the number of bits. This chapter deals with the design proposal and implementation of new prefix adder architecture for 8-bit, 16-bit, 32-bit and 64-bit addition. The proposed architectures have the least number of computation nodes when compared with existing ones. This reduction in hardware of the proposed architectures helps to reap a benefit in the form of reduced power and power-delay product.

The Parallel prefix adder adders involve the execution of an operation in parallel. This is done by segmentation the operation in smaller pieces which are computed in parallel. The outcome of the operation depends on the initial inputs. Parallel Prefix Adder (PPA) is equivalent to carry look ahead adder (CLA).A Carry look ahead adder is a type of adder used in digital logic. CLA is designed to overcome the latency introduced by repelling effect of carry bits in RCA.A CLA improves speed by reducing carry bits. It calculates one or more carry bits before the sum, which reduces the wait time to calculate the result of larger bit value. CLA uses the concept of generating (G) and propagating (P) carries. The two differ in the way their carry generation block is implemented. Equations used to generate carry in CLA are given by:

$$Ci = Gi{-}1\ OR\ (Pi{-}1\ AND\ Ci{-}1\ )\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

The main advantage of PPA is the carry reduces the number of logic levels by essentially generating the carries in parallel.PPA fastest adder with focus on design time and is the choice for high performance adder in industry.

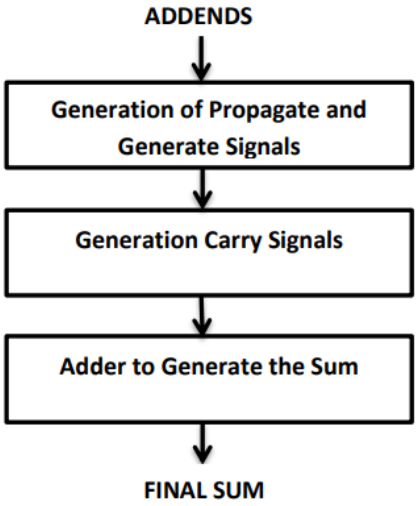

Figure 1: Architecture of Parallel Prefix Adder

II. LITERATURE REVIEW

The following are the few reference that I have gone through for the parallel prefix adders design, implementation and result. In [1],[2], and [3] Design and implementation of the different adders like RCA, CSA, CLA and KSA. In [4], give a brief working of design and implementation of8-bit Kogge Stone Adder (KSA) and its modified KSA design using cadence tool. In [5], Brief working of Parallel Prefix Adders and performance of six different parallel prefix adders design and implementation using different TSNC technology nodes, they are KSA, BKA, HCA, Sklansky Adder, LFA and Knowel Adder. In [6], Comparative analysis of different Kogge Stone Adders with Ripple Carry and Carry Skip Adders. In [7] and [8], Frontend design and implementation different adders like KSA and BKA. In [9], detail description and working about GDI (Gate Diffusion Input) technology, In [10], detail description and working of CMOS (Complementary Metal Oxide Semiconductor) Technology.

III. IMPLEMENTATION

3.1 Ripple Carry Adder Circuit (RCA):

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage. In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry in of that stage occurs. Propagation delays inside the logic circuitry is the reason behind this. Propagation delay is time elapsed between the application of an input and occurrence of the corresponding output.

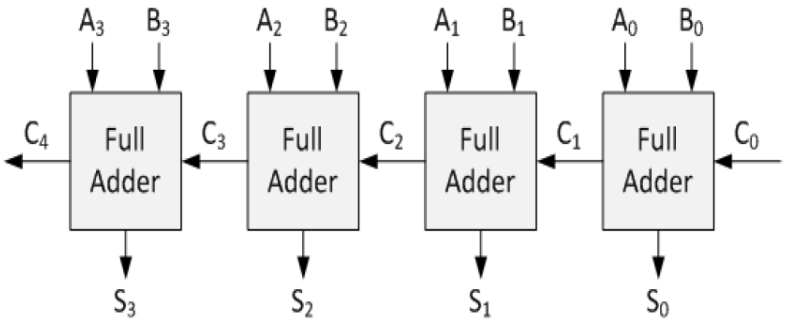

Figure 2: Ripple carry adder

3.2 Carry Lookahead adder (CLA):

Carry Lookahead adder is a fast parallel adder as it reduces the propagation delay by more complex hardware, hence it is costlier. In this design, the carry logic over fixed groups of bits of the adder is reduced to two-level logic, which is nothing but a transformation of the ripple carry design.

This method makes use of logic gates so as to look at the lower order bits of the augends and addend to see whether a higher order carry is to be generated or not. Let us discuss in detail

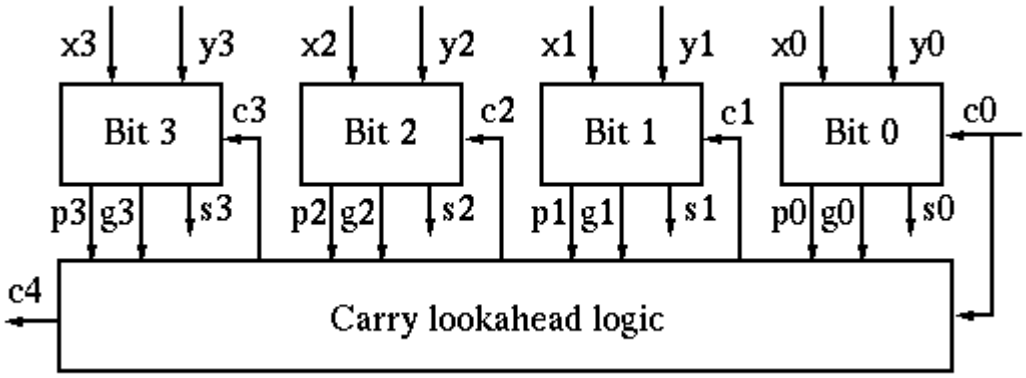

Figure 3: Carry Lookahead adder

3.3 Kogge Stone Adder (KSA):

KSA is a parallel Prefix Adder. It is considered as fastest and is widely used in industry for high performance arithmetic circuits. KSA employs the 3-stage structure of the CLA adder, the improvement is in the carry generation stage which is the most intensive one. In KSA carries are computed fast by computing the carries in parallel. This is often desirable to use an adder with good timing, area and efficiency trade off. The carry computation method leads to speed up the overall operation significantly. This reduces the area and increase the speed. As shown in Figure 1.3.1. In this paper, design and implementation of optimized 8 bit KSA is proposed.

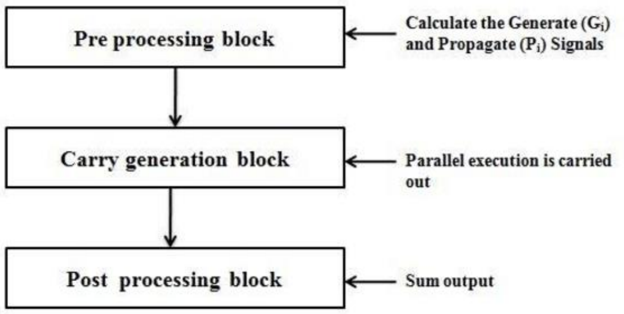

Figure 4: Architecture of Kogge Stone parallel prefix adder.

i. Pre-Processing Block:

This is the initial stage of the Parallel Prefix Adder, it's used to generate the Propagated signal and generated signal and this signal are computed for the given inputs by using following equations.

Pi = Ai XOR Bi………………………………(1)

Gi = Ai AND Bi……………………………….(2)

ii. Carry Generation Block:

Carry generated stage is a most important block in this adder design. It consists of two components such as Block Cell and Gray Cell. Block Cell is used to produce the Generated signal and Propagated signal, needed to the calculation of the next stage. Gray Cell is used to produce only Generated signal and this signal utilized or needed in the calculation of the Sum in next stage.

Black Cell: The black cell operator receives 2 set of generate and propagate signals (Gi, Pi) and (Gj, Pj) compute one set of generate and propagate signals (G, P).

G = Gi OR (Pi AND Pj)…………………………….. (3)

P = Pi AND Pj……………………………………….(4)

Gray Cell: The Gray operator receives two set of generate and propagate signals (Gi, Pi) and (Gj, Pj) compute one set of generate signals (G).

G = Gi OR (Pi AND Pj)……………………………….(5)

iii. Post Processing Block:

This is the final stage of the adder; Sum and Carry are the final outcome of the adder.

Si = Pi XORCi-1………………………………………..(6)

3.4 Spanning Tree Adder (SPA):

There are many algorithms to compute a spanning tree for a connected graph. The first is an example of a

a. vertex-centric algorithm:

1. Pick an arbitrary node and mark it as being in the tree.
2. Repeat until all nodes are marked as in the tree:

(a) Pick an arbitrary node u in the tree with an edge e to a node w not in the tree.

Add e to the spanning tree and mark w as in the tree.

We iterate n-1 times in Step 2, because there are n-1 vertices that have to be added to the tree. The efficiency of the algorithm is determined by how efficiently we can find a qualifying w.

b. The second algorithm is edge-centric:

1. Start with the collection of singleton trees, each with exactly one node.

2. As long as we have more than one tree, connect two trees together with an edge

in the graph.

This second algorithm also performs n steps, because it has to add n - 1edges to the trees until we have a spanning tree. Its efficiency is determined by how quickly we can tell if an edge would connect two trees or would connect two nodes already in the same tree, a question we come back to in the next lecture. Let's try this algorithm on our first graph, considering edges in the listed order: (AB, BC, CD, AE, BE, CE)

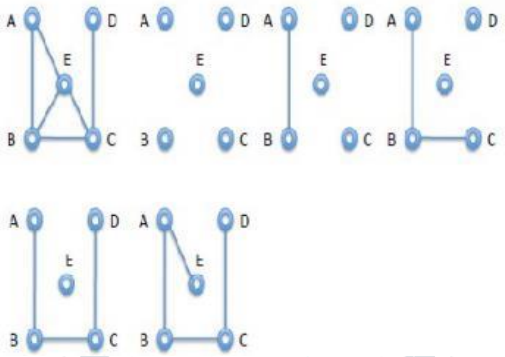

Figure 5: Spanning Tree parallel prefix adder

The first graph is the given graph the completely disconnected graph is the starting point for this algorithm. At the bottom right we have computed the spanning tree, which we know because we have added n - 1 = 4 edges. If we tried to continue, the next edge BE could not be added because it does not connect two trees, and neither can CE. The spanning tree is complete.

IV. SIMULATION AND RESULTS

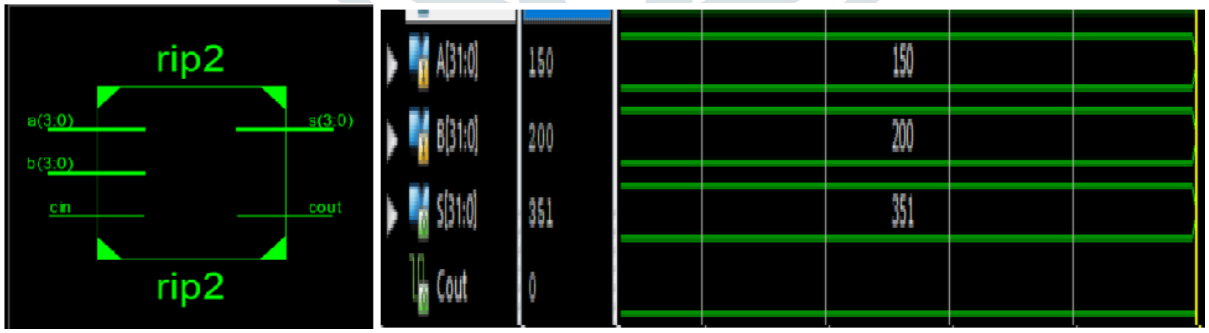

Figure 6: Ripple Carry Adder RTL and Output

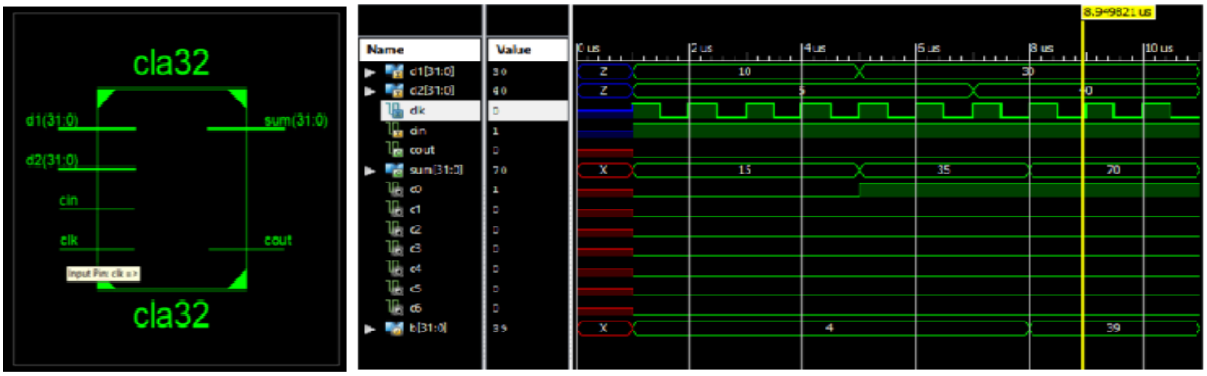

Figure 7: Carry Lookahead Adder RTL and Output

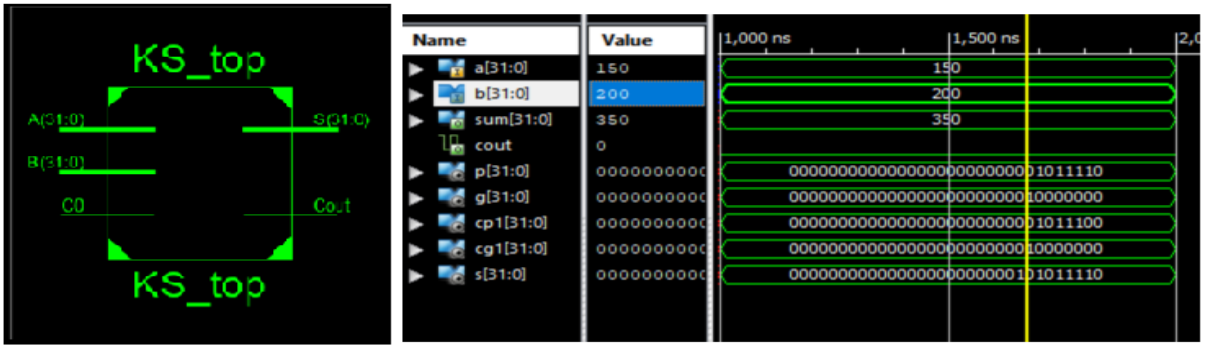

Figure 8: Kogge Stone adder RTL and Output

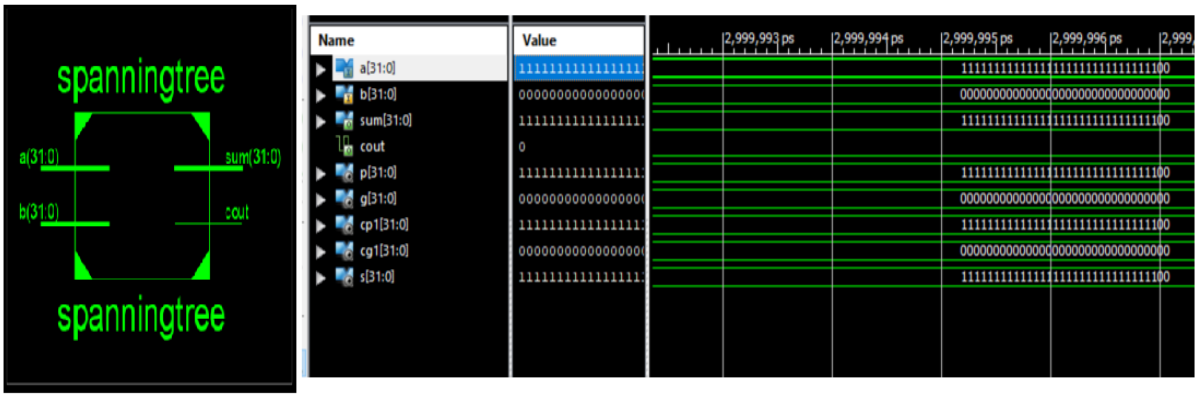

Figure 9: Kogge Stone adder RTL and Output

Table I. 32 Bit Adders Device utilization cell

SL. NO.	NAME of ADDER	NO. of SLICES	4INPUT LUT's	IO's	BONDED IOB'S
1	RCA	106	180	14	16
2	CLA	119	160	99	99
3	KSA	89	157	98	98
4	SPA	27	49	97	97

Table II. 32 Bit Adders Device utilization cell

SL. NO.	NAME of ADDER	LOGIC DELAY (ns)	ROUTE DELAY (ns)	TOTAL DELAY (ns)
1	RCA	**35.050 (65.7%)**	**22.847 (34.3%)**	**57.897**
2	CLA	**30.050 (65.7%)**	**14.847 (34.3%)**	**44.897**
3	KSA	**13.642 (64.0%)**	**7.684 (36.0%)**	**21.326**
4	SPA	**19.854 (63.8%)**	**11.274 (36.2%)**	**31.128**

V. CONCLUSIONS

Novel hybrid PPA architectures for 8-bit, 16-bit, 32-bit is proposed that offers minimal power dissipation and least power-delay product. The architectures are realized using three Schemes. Scheme I provides least delay and least power-delay product for all the PPA architectures. Scheme II offers moderate power and delay for all the PPA architectures. Scheme III provides least power dissipation in all the PPA architectures. The proposed PPA in Scheme I achieves 3% to 7% power savings and 15% to 35% improvement in speed compared with Spanning Tree adder which has nearly same logic depth and number of computation nodes. The proposed PPA architecture in Scheme I attains 30% to 45% power savings at the expense of 5% to 10% delay penalty compared to Kogge-Stone adder which has maximum computation nodes. The proposed PPA in Scheme II achieves 3% to 15% power savings and 15% to 28% improvement in performance when compared to Spanning Tree adder. The proposed PPA architecture in Scheme II attains 35% to 52% power savings at the expense of 10% to 27% delay penalty compared to Kogge-Stone adder. The Proposed PPA in Scheme III offer 6% to 8% power savings and 10% to 25% improvement in critical path delay compared to a Spanning Tree. The proposed PPA architecture when compared to Kogge-Stone adder in Scheme III offers 45% to 55% power reduction at the expense of 17% to 25% delay penalty.

REFERENCES

[I] J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hili, 1984.

[2] Information Technology-Coding of Moving Picture and Associated Autio, MPEG-2 Draft International Standard, ISO/IEC 13818-1,2,3, 1994.

[3] JPEG 2000 Part I Finall19l Draft. ISO/IEC JTClISC29 WG 1.

[4] O. L. MacSorley, "High speed arithmetic in binary computers," Proc. IRE. vol. 49, pp. 67-91, Jan. 1961.

[5] S. Waser and M. 1. Flynn, Introduction to Arithmetic for Digital Systems Designers. New York: Holt, Rinehart and Winston, 1982.

[6] A. R. Omondi, Computer Arithmetic Systems. Englewood Cliffs, NJ:

[7] A. D. Booth, "A signed binary multiplication technique," Quart. J. Math., vol. IV, pp. 236-240,1952.

[8] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron Comput., vol. EC-13,n o. I, pp. 14-17, Feb. 1964.

[9] A. R. Cooper, "Parallel architecture modified Booth multiplier," Proc. Inst. Electr. Eng. G, vol. 135, pp. 125-128, 1988.

[10] N. R. Shanbag and P. Juneja, "Parallel implementation of a 4 X 4-bit multiplier using modified Booth's algorithm," IEEE J. Solid-State Circuits. vol. 23,n o. 4, pp. 1010-1013, Aug. 1988.

[11] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 X 54 regular structured tree multiplier," IEEE J. Solid-State Circuits. vol. 27, no. 9, pp. 1229-1236, Sep. 1992.

[12] 1. Fadavi-Ardekani, "M N Booth encoded multiplier generator using optimized Wallace trees," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. l,no. 2,pp. 120-125,Jun. 1993.

[13] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54 54 multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, vol. 30, no. 3, pp. 251-257, Mar. 1995.

[14] A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed-point IIR digital filters," J. Circuits, Syst., Comput., vol. 7, no. 3, pp. 191-209, 1997.

[15] A. Tawfik, F. Elguibaly, M. N. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-speed area-efficient inner-product processor," Can. J. Electr. Comput. Eng., vol. 19, pp. 187-191, 1994.

[16] F. Elguibaly and A. Rayhan, "Overf10w handling in inner-product processors," in Proc. IEEE Pacific Rim Con] Commun..Comput.. Signal Process .. Aug. 1997, pp. 117-120.

[17] F. Elguibaly, "A fast parallel multiplier-accumulator using the modified Booth algorithm," IEEE Trans. Circuits Syst., vol. 27,n o. 9, pp. 902-908, Sep. 2000.

[18] A. Fayed and M. Bayoumi, "A merged multiplier-accumulator for high speed signal processing applications," Proc. ICASSP, vol. 3, pp. 3212-3215, 2002.

[19] P. Zicari, S. Perri, P. Corsonello, and G. Cocorullo, "An optimized adder accumulator for high speed MACs," Proc. ASICON 2005. vol. 2, pp. 757-760,2005.

[20] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," IEEE J. Solid-State Circuits, vol. 25,n o. 2, pp. 584-594, Feb. 1990.