ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Design and Estimation of delay, power and area for Parallel prefix adders

Divya Tejaswi Pirati
P.G. Scholar,
Department of Electronics & Communication Engineering,
VRS &YRN College of Engineering & Technology, Chirala. A.P

Sunil Dayakar Gundala
Assistant Professor,
Department of Electronics & Communication Engineering,
VRS &YRN College of Engineering & Technology, Chirala. A.P

Abstract:

Abstract—Parallel Prefix Adders have been established as the most efficient circuits for binary addition. The binary adder is the critical element in most digital circuit designs including digital signal processors and microprocessor data path units. The final carry is generated ahead to the generation of the sum which leads extensive research focused on reduction in circuit complexity and power consu mption of the adder. In VLSI implementation, parallel-prefix adders are known to have the best performance. This paper investigates four types of carry-tree adders (the Kogge-Stone, sparse Kogge-Stone, spanning tree, Brent kung Adder) and compare them to the simple Ripple Carry Adder and Carry Skip Adder. These designs of varied bit-widths are simulated using implemented on a Xilinx version Spartan 3E FPGA. These fast carry-chain carry-tree adders support the bit width up to 256. We report on the area requirements and reduction in circuit complexity for a variety of classical parallel prefix adder structures.

Key words *— parallel prefix adders; carry tree adders; FPGA; logic analyzer; delay; power.*

I.INTRODUCTION

In Digital Computer Design adder is an important component and it is used in multiple blocks of its architecture. In many Computers and in various classes of processor specialization, adders are not only used in Arithmetic Logic Units [6], but also used to calculate addresses and table indices. There exist multiple algorithms to carry on addition operation ranging from simple Ripple Carry Adders to complex CLA.

The basic operations involved in any Digital Signal Processing systems are Multiplication, Addition and Accumulation. Addition is an indispensible operation in any Digital, DSP or control system. Therefore fast and accurate operation of digital system depends on the performance of adders [6]. Hence improving the performance of adder is the main area of research in most digital circuits. Binary addition is a fundamental operation in most digital circuits. There are a variety of adders, each has certain performance. Each type of adder is selected depending on where the adder is to be used.

Adders are critically important elements in processor chips and they are used in floating-point arithmetic units, ALUs, memory addressing, program counter updating, Booth Multipliers, ALU Designing, multimedia and communication systems, Real-time signal processing like audio signal processing, video/image processing, or large capacity data processing etc. The requirements of the adder are that it is primarily fast and secondarily efficient in terms of power consumption. In VLSI implementations, parallel-prefix adders are known to have the best performance. In this paper, designing and implementing the tree-based adders on FPGAs are described.

Tree-based adder structures are implemented on FPGA and compared with the Ripple Carry Adder (RCA) and the Carry Skip Adder (CSA). Some conclusions and suggestions are made for improving FPGA designs to enable better tree-based adder performance. Parallel prefix (or tree prefix) adders provide a good theoretical basis to make a wide range of design trade-offs in terms of delay, area and power. Parallel Prefix Adders

(PPA) is designed by considering carry look adder as a base. Similar to a CLA they employ the 3-stage structure shown in Figure.1 CLA and a PPA differs in second stage. In second stage carry signal of binary addition is generated.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

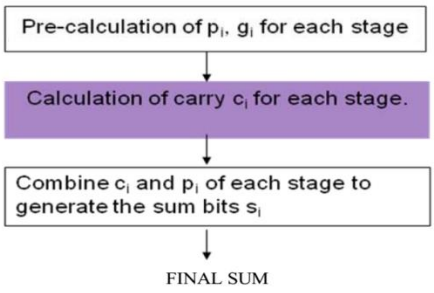

Figure.1 Stages of Binary Addition

Three stage structure of the carry look ahead and parallel prefix adder. In a PPA the prefix operator "o" is introduced and the carry signal generation is treated as a prefix problem.

II. RELATED WORK

We compared the design of the ripple carry adder with the carry-look ahead, carry-skip, and carry-select adders on the Xilinx 4000 series FPGAs. Only an optimized form of the carry-skip adder performed better than the ripple carry adder when the adder operands were above 56 bits. A study of adders implemented on the Xilinx Vertex II yielded similar results. The previous authors considered several parallel prefix adders implemented on a Xilinx Vertex 5 FPGA. It is found that the simple RCA adder is superior to the parallel prefix designs because the RCA can take advantage of the fast carry chain on the FPGA. This study focuses on carry-tree adders implemented on a Xilinx Spartan 3E FPGA. The distinctive contributions of this paper are two-fold. First, we consider tree-based adders and a hybrid form which combines a tree structure with a ripple-carry design. The Kogge-Stone adder is chosen as a representative of the former type and the sparse Kogge Stone and Brent Kung Adder is representative of the latter category. Second, this paper considers the practical issues involved in testing the adders and provides actual measurement data to compare with simulation results. The previous works cited above all rely upon the synthesis reports from the FPGA place and route software for their results.

A 16-bit Kogge-Stone adder is built from 16 generate and propagate (GP) blocks, 37 black cells (BC) blocks, 16 (GC) blocks, 16 sum blocks. Kogge-Stone prefix tree is one of the adders that use fewest logic levels. Gray cells are inserted similar to black cells except that the gray cells final output carry outs instead of intermediate G/P group. The reason of starting with Kogge-Stone prefix tree is that it is the easiest to build in terms of using a program concept. The Figure.2 shown below is 16-bit (a power of 2) prefix tree and it is not difficult to extend the structure to any width if the basics are strictly followed. The sparse Kogge-Stone adder consists of several smaller ripple carry adders (RCAs) on its lower half, a carry tree on its upper half. It terminates with RCAs. The number of carries generated is less in a sparse Kogge Stone adder compared to the regular Kogge-Stone adder. The functionality of the GP block, black cell and the gray cell remains exactly the same as in the regular Kogge-Stone adder. The sparse Kogge-Stone adder, this design terminates with a 4- bit RCA. As the FPGA uses a fast carry-chain for the RCA, it is interesting to compare the performance of this adder with the sparse Kogge-Stone and regular Kogge-Stone adders. The Figure.4 Shown below is the Block diagram of 16-Bit Sparse Kogge-Stone Adder.

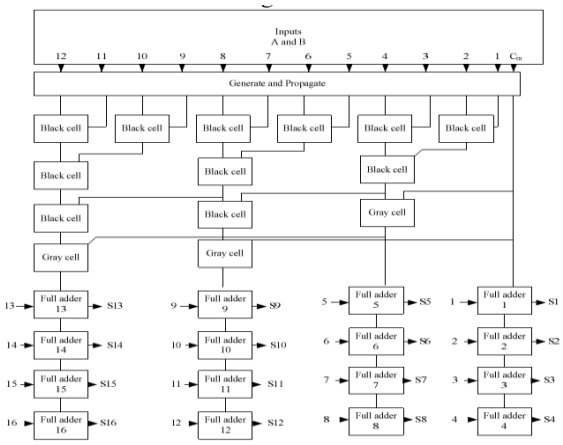

Fig.2. 16 bit sparse kogge-Stone adder

The 16 bit SKA uses black cells and gray cells as well as full adder blocks too. This adder computes the carries using the BC's and GC's and terminates with 4 bit RCA's. Totally it uses 16 full adders. The 16 bit SKA is shown in figure 2. In this adder, first the input bits (a, b) are converted as propagate and generate (p, g). Then propagate and generate terms are given to BC's and GC's. The carries are propagated in advance using these cells. Later these are given to full adder blocks. Another PPA is known as STA is also tested [6]. Like the SKA, this adder also terminates with a

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

RCA. It also uses the BC's and GC's and full adder blocks like SKA's but the difference is the interconnection between them [7].The 16 bit STA is shown in the below figure 3.

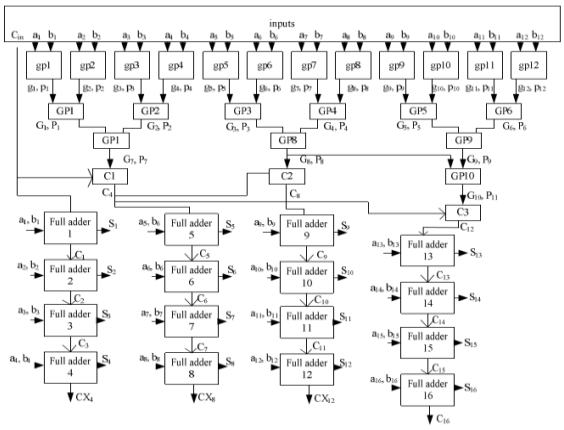

Fig.3. 16 bit spanning tree adder

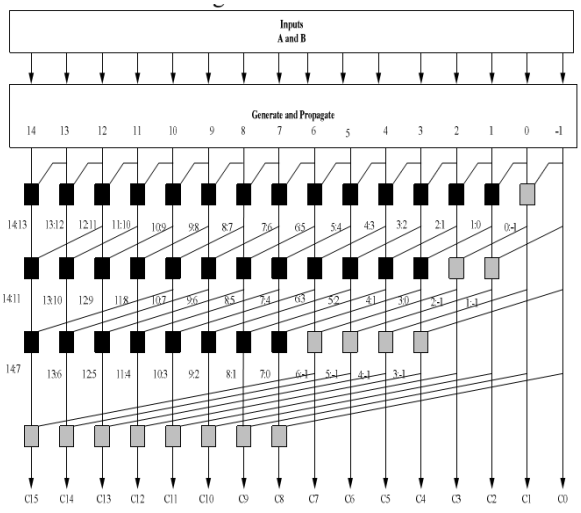

Fig.4. 16 bit kogge stone adder

Another carry tree known as BKA which also uses BC's and GC's but less than the KSA. So it takes less area to implement than KSA. The 16 bit BKA uses 14 BC's and 11 GC's but kogge stone uses 36 BC's and 15 GC's. So BKA has less architecture and occupies less area than KSA. The 16 bit BKA is shown in the below figure 5.

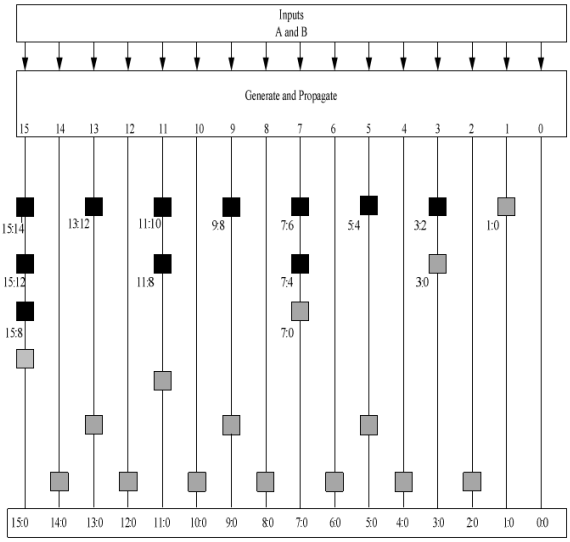

Fig.5. 16 bit Brent Kung adder

BKA occupies less area than the other 3 adders called SKA, KSA, and STA. This adder uses limited number of propagate and generate cells than the other 3 adders. It takes less area to implement than the KSA and has less wiring congestion. The operation of the 16 bit brent kung adder is given below [3].This adder uses less BC's and GC's than kogge stone adder and has the better delay performance which is observed in Agilent 1692A logic analyzer.

III. SIMULATION RESULTS

Table.3 contains the results obtained. The adder abbreviations used in the table and the following discussions are: BK for the Brent-Kung adder, KS for the Kogge-Stone adder, SK for Sparse Kogge Stone Adder, RC for Ripple Carry Adder. In the table, area is measured in Slice Look-Up Tables (LUT) units which represent configurable logic units within the FPGA. Interestingly the synthesis tool synthesized a simple ripple carry adder regardless of the optimization strategy. The adder was implemented by configuring the slices within the FPGA as full adder components. Hence the number of lookup tables matched the operand bit-size in every case. Table.3 give the area results with the software set for area. It is apparent from these tables that the area optimization strategy produces adders which are significantly smaller compared to those produced with complexity optimization.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

In Table we can observe that generally the adder areas compare with the characteristics of their type. The BK exhibits the smallest size while the KS is the largest adder. This shows that in certain cases the tool optimized circuits reverse the algorithmic superiority of a design.

Table.3 Area Results Obtained With Area Optimization

	RC	KS	SK	BK
I/P Arrival Time	18.43	12.72	12.65	10.82
Path delay	21.65	20.26	15.87	5.96
Slices	18	21	29	26
4 i/p LUT's	32	37	51	45
Bonded IOB's	51	51	66	56

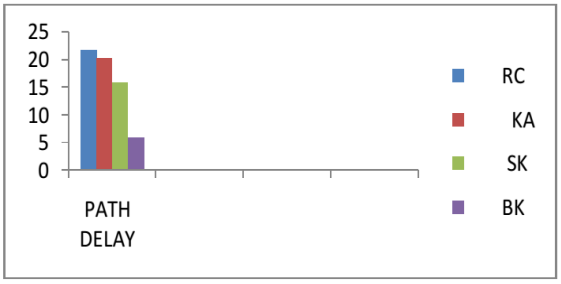

Figure.6 Comparison of path delays for Adders

These adders are implemented in verilog HDL in Xilinx 13.2 ISE design suite and then verified using Xilinx virtex 5 FPGA through chip scope analyzer [7], [8] and [9]. And these were tested using Agilent 1692A logic analyzer.

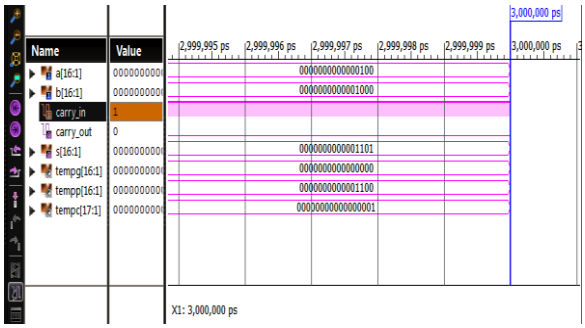

Fig7: Look head adder result

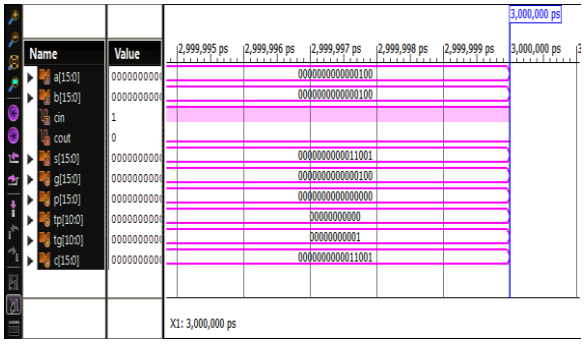

Fig8: Brentkung adder Result

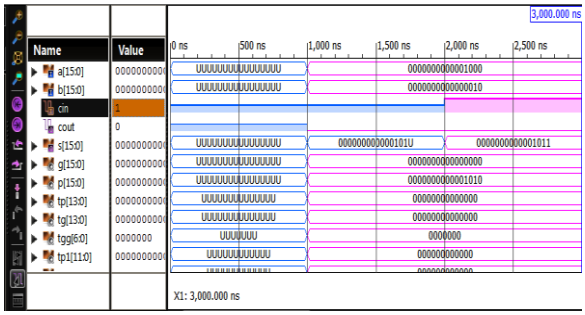

Fig9: koggestone Adder result

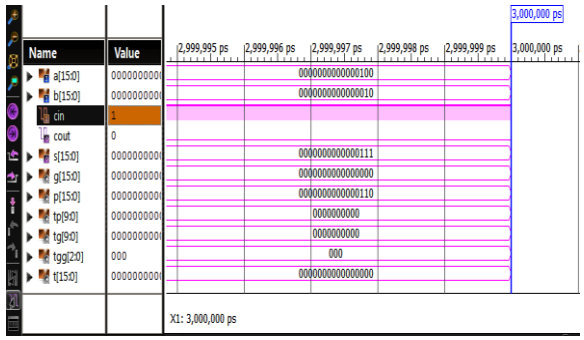

Figure10: spinning tree adder result

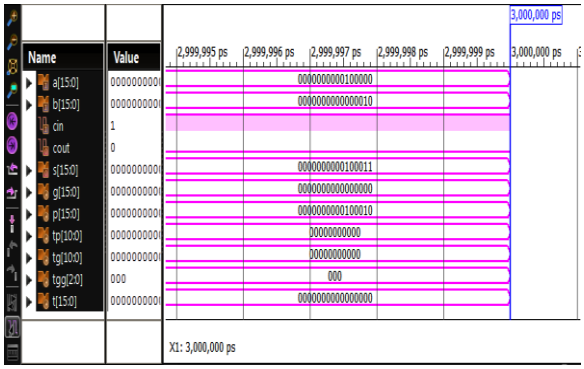

Figure11: sparse koggestone adder result

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

IV CONCLUSION

Parallel-prefix adders are not as effective as the simple ripple-carry adder at low to moderate bit widths. We have indications that the carry-tree adders eventually surpass the performance of the linear adder designs at high bit-widths, expected to be in the 128 to 256 bit range. This is important for large adders used in precision arithmetic and cryptographic applications where the addition of numbers on the order of a thousand bits is not uncommon. Because the adder is often the critical element which determines to a large part the cycle time and power dissipation for many digital signal processing and cryptographically implementations, it would be worthwhile for future FPGA designs to include an optimized carry path to enable tree based adder designs to be optimized for place and routing. The testability and possible fault tolerant features of the Brent kung adder are topics for future research.

REFERENCES

[1] David H.K.Hoe, Chris Martinez and Sri Jyothsna Vundavalli", Design and Characterization of Parallel Prefix Adders using FPGAs", 2011 IEEE 43rd Southeastern Symposium in pp. 168-172, 2011.

[2] N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson–Addison-Wesley, 2011.

[3] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp. 260-264, 1982.

[4]D.Harris,"A Taxonomy of Parallel Prefix Netw orks," in Proc. 37th Asilomar Conf Signals Systems and Computers,pp.2213–7,2003.

[5] P. M. Kogge and H. S. Stone, "A Parallel Algo rithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. On Computers, Vol.C-22,No8,August1973.

[6] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, "Easily Testable Cellular Carry Lookahead Adders," Journal of Electronic Testing: Theory and Applications19,285-298,2003.

[7] T. Lynch and E. E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," IEEE Trans. on Computers, vol. 41, no. 8, pp. 931-939, Aug. 1992.

[8] Beaumont-Smith, A, Cheng-Chew Lim ,"Parallel prefix adder design", Computer Arithmetic, 2001. Proceedings. 15th IEEE Symposium ,pp. 218 – 225,2001.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989

[9] K. Vitoroulis and A. J. Al-Khalili, "Performance of Parallel Prefix Adders Implemented with FPGA technology," IEEE Northeast Workshop on Circuits and Systems, pp. 498-501, Aug. 2007. 172.

[10] S. Xing and W. W. H. Yu, "FPGA Adders: Performance Evaluation and Optimal Design," IEEE Design & Test of Computers, vol. 15, no. 1, pp. 24-29, Jan. 1998.