RESEARCH ARTICLE OPEN ACCESS

Design of High Speed 128 bit Parallel Prefix Adders

T.KIRAN KUMAR[1], P.SRIKANTH[2]
[1] Dept. of ECE. MVGR college of Engineering, A.P, India
[2] Dept. of ECE., MVGR college of Engineering, A.P, India

ABSTRACT:
In this paper, we propose 128-bit Kogge-Stone, Ladner-Fischer, Spanning tree parallel prefix adders and compared with Ripple carry adder. In general N-bit adders like Ripple Carry Adders (slow adders compare to other adders), and Carry Look Ahead adders (area consuming adders) 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. While when we want to design any 128-bit operating systems and processors we can use these adders in place of regular adders. We simulate and synthesis different types of 128-bit prefix adders using Xilinx ISE 12.3 tool. By using these synthesis results, we noted the performance parameters like number of LUTs and delay. We compare these three adders in terms of LUTs (represents area) and delay values.
Keywords– prefix adder, carry operator, Kogge-Stone, Spanning tree, Ladner-Fischer.

I. INTRODUCTION

Arithmetic circuits are the ones which perform arithmetic operations like addition, subtraction, multiplication, division, parity calculation. Most of the time, designing these circuits is the same as designing multiplexers, encoders and decoders. In electronics, an adder or summer is a digital circuits [7] that performs addition of numbers. In many computers and other kind of processors, adders are other parts of the processor, many computers and other kinds of processors, where they are used to calculate addresses, table and similar. The binary adder[7,10] is the one type of element in most digital circuit designs including digital signal processors(DSP) and microprocessor data path units. 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 VLSI[10] system design. The Conventional adders discussed in section II. The details of Kogge-Stone adder, spanning tree adder and Ladner- Fischer adders are discussed, and the implementation of proposed system is described in section III. The performance and simulation results were presented and discussed in section IV.

II. CONVENTIONAL ADDERS

Ripple Carry adder

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 are the reason behind this.The 4-bit RCA figure shown below.

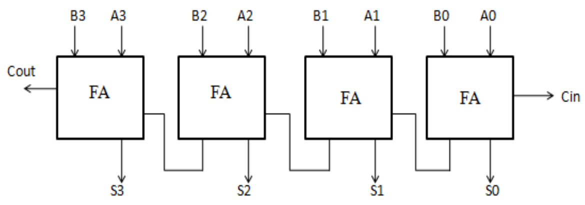

Figure1: 4-bit Ripple Carry Adder

Carry Skip adder

A carry-skip adder (also known as a carry-bypass adder) is an adderimplementation that improves on the delay of a ripple-carry adderwith little effort compared to other adders. The improvement of the worst-case delay is achieved by using several carry-skip adders to form a block-carry-skip adder. The critical path of a carry-skip-adder begins at the first full-adder, passes through all adders and ends at the sum-bit S_{n-1}. Carry-skip-adders are chained to reduce the overall critical path, since a single n-bit carry-skip-adder has no real speed benefit compared to an-bit carry-ripple-adder. The number of inputs of the AND-gate is equal to the width of the adder. For a large width, this becomes impractical and leads to additional delays, because the AND-gate has to be built as a tree. A good width is achieved, when the sum-logic has the same depth like the n-input AND-gate and the multiplexer. As an example 4-bit carry skip adder is shown.

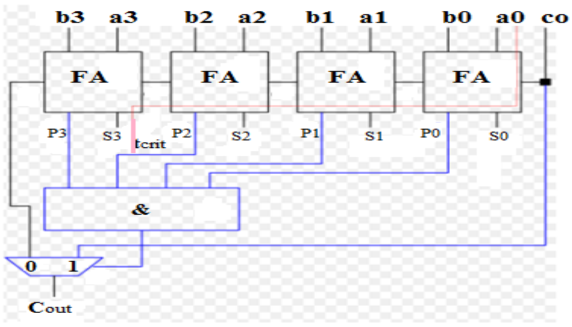

Figure2: 4-bit Carry Skip adder

III. PROPOSED ADDERS

Parallel prefix adders

The Parallel prefix adder is like a Carry Look Ahead Adder. The production of the carriers the prefix adders [1] can be designed in many different ways based on the different requirements. We use tree structure form to increase the speed [13] of arithmetic operation. Parallel prefix adders are faster adders [1] and these are faster adders [4] and used for high performance arithmetic structures in industries. The parallel prefix addition is done in 3 steps.

1. Pre-processing stage
2. Carry generation network
3. Post processing stage

Pre-processing stage

In this stage we compute, the generate and propagate signals are used to generate carry input of each adder. A and B are inputs. These signals are given by the equation 1&2.

$P_i = A_i \oplus B_i$..........................(1)

$G_I = A_i . B_i$.............................(2)

Carry generation network

In this stage we compute carries corresponding to each bit. Execution is done in parallel form [4].After the computation of carries in parallel they are divided into smaller pieces. carry operator contain two AND gates , one OR gate. It uses propagate and generate as intermediate signals which are given by the equations 3&4.

$P_{(i:k)} = P_{(i:j)} . P_{(j-1:k)}$.......................................(3)

$G_{(i:k)} = G_{(i:j)} + (G_{(j-1:k)} . P_{(i:j)})$.......................(4)

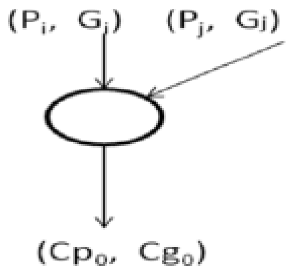

Figure3: Carry operator

The operations involved in this figure are given as.

Post processing stage

This is the final stage to compute the summation of input bits. it is same for all adders and sum bit equation given

$S_i = P_i \oplus C_i$..(5)

$C_{i+1} = (P_i . C0) + G_i$....................................(6)

Parallel Prefix Adders are classified into

1. Kogge- Stone Adder
2. Ladner-Fischer Adder
3. Spanning Tree Adder

Kogge - Stone Adder

The Kogge-Stone adder is a parallel prefix form carry look-ahead adder. It generates the carry signals in O(log n) time, and is widely considered the fastest adder design possible. It is the common design for high-performance adders in industry. This is among the type of prefix trees that use the fewest logic levels. In fact, Kogge-Stone is a member of Knowles prefix tree. The 16-bit prefix tree can be viewed as Knowles. The numbers in the brackets represent the maximum branch fan-out at each logic level. For an example 16-bit Kogge Stone adder has been shown.

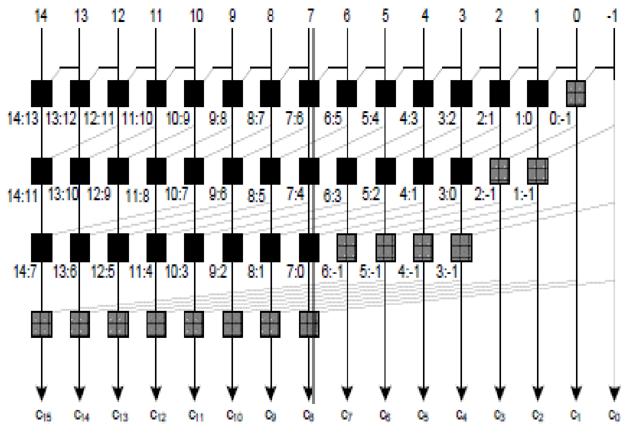

Figure 4: 16-bit Kogge-Stone Adder

The maximum fan-out is 2 in all logic levels for all width Kogge-Stone prefix trees. The key of building a prefix tree is how to implement Equation according to the specific features of that type of prefix tree and apply the rules described in the previous section.

Ladner-Fischer Adder

In 1980, Fischer and Richard Ladner presented a parallel algorithm for computing prefix sums efficiently. They show how to construct a circuit that computes the prefix sums in the circuit, each node performs an addition of two numbers. With their construction, one can choose a trade-off between the circuit depth and the number of nodes. However, the same circuit designs were already studied much earlier in Soviet mathematics. This also has *O*(log2 *n*) stages. A 16-bit Ladner-Fischer adder has been shown as an example.

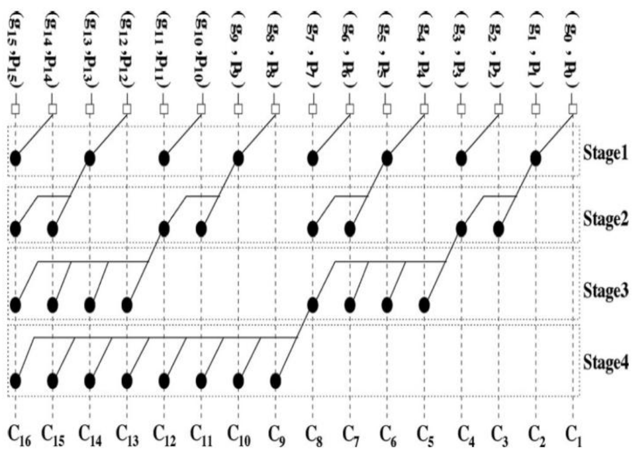
Stage1
Stage2
Stage3
Stage4
(g_{15}, p_{15}) … (g_0, p_0)
C_{16} C_{15} C_{14} C_{13} C_{12} C_{11} C_{10} C_9 C_8 C_7 C_6 C_5 C_4 C_3 C_2 C_1

Figure 5: 16- bit Ladner- Fischer adder

Spanning Tree adder

Spanning tree adders is an existing category of adders. Basically, the performances of adders are evaluated with propagation delay. This spanning tree uses minimum number of multi-input gates. It is a Hybrid adder. Which consists of generate and propagate blocks as well as the full adders. Path delay is the main concern for these prefix adders. Now an example of 16-bit spanning tree adder is shown.

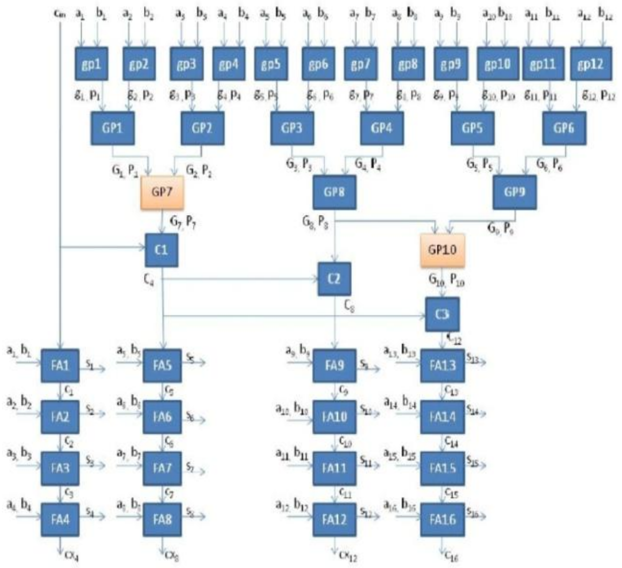
gp1 gp2 gp3 gp4 gp5 gp6 gp7 gp8 gp9 gp10 gp11 gp12
GP1 GP2 GP3 GP4 GP5 GP6
GP7 GP8 GP9 GP10
C1 C2 C3
FA1 FA2 FA3 FA4 FA5 FA6 FA7 FA8 FA9 FA10 FA11 FA12 FA13 FA14 FA15 FA16

Figure 6: 16-bit Spanning Tree Adder

IV. SIMULATION RESULTS AND COMPARISIONS

Various adders were designed using Verilog language in Xilinx ISE Navigator 12.3 and all the simulations are performed using Modelsim 6.5b simulator. The performance [12] of proposed adders are analyzed and compared. In this proposed architecture, the implementation code for 128-bit Kogge-Stone, Spanning Tree adder, Ladner-Fischer adders were developed and compared with 128-bitRipple carry adder and corresponding values of delay and area were observed. Table 1 shows the comparisons of 128-bit adders. The simulated outputs of 128-bit proposed adders are shown in Figure7, 8, 9 & 10.

Table 1: Comparison of 128-bit adders

Adder	Delay (ns)	Memory (KB)	No of LUTs	No of Slices
Ripple carry	161.256	191788	9312	146
Kogge Stone	21.959	211308	1272	727
Ladner-Fischer	26.245	199980	483	276
Spanning Tree	19.496	194860	410	231

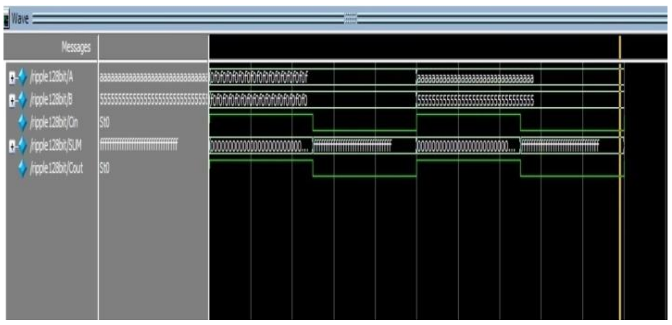

Figure 7: Simulated Output of 128-bit Ripple carry Adder

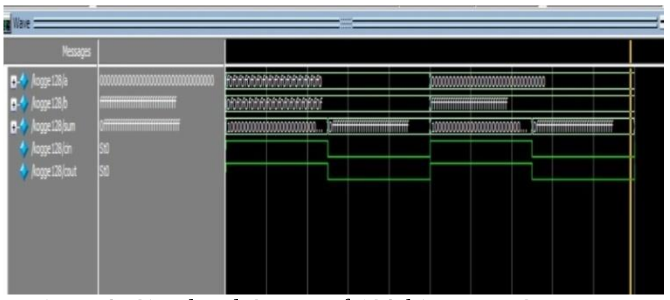

Figure 8: Simulated Output of 128-bit Kogge Stone Adder

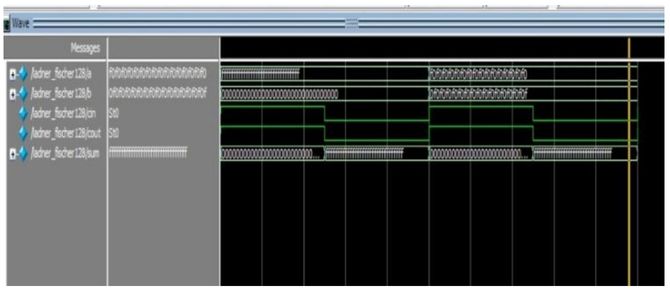

Figure 9: Simulated Output of 128-bit Ladner-Fischer Adder

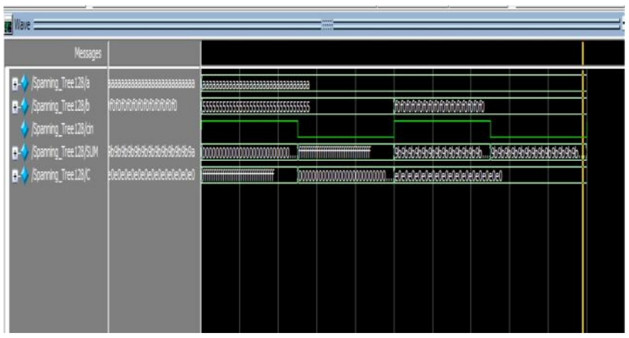
Figure 10: Simulated Output of 128-bit Spanning Tree Adder

V. CONCLUSION

The proposed adders are faster because of less delay and area efficient compared to other basic adders. Among these three prefix adders Spanning Tree adder has better performance among the other prefix adders from low order bits to high order bits. The performance comparisons between these adders are measured in terms of area and delay. It would be interesting to investigate the design of the 256 bit adders. These adders are popularly used in VLSI implementations.

ACKNOWLEDGEMENT

T.Kiran Kumar would like to thank Mr.P.Srikanth, Assistant Professor, ECE Department who had been guiding throughout the project and supporting me in giving technical ideas about the paper and motivating me to complete the work efficiently and successfully.

REFERENCES

[1] Y. Choi, “Parallel Prefix Adder Design”, Proc. 17th IEEE Symposium on Computer Arithmetic, pp 90-98, 27th June 2005.
[2] R. P. Brent and H. T. Kung, “A regular layout for parallel adders”, IEEE trans, computers, Vol.C-31,pp. 260-264,.March 1982.
[3] Kogge P, Stone H, “A parallel algorithm for the efficient solution of a general class Recurrence relations,” IEEE Trans. Computers, Vol.C-22, pp 786-793,Aug. 1973.
[4] R. Zimmermann, “Non-heuristic operation and synthesis of parallel-prefix adders,” in International workshop on logic and architecture synthesis, December 1996,pp. 123-132.
[5] C.Nagendra, M. J. Irwin, and R. M. Owens, “Area -Time-Power tradeoffs in parallel adders”, Trans. Circuits Syst. II, vol.43, pp. 689– 702 Oct.1996.
[6] R. Ladner and M. Fischer, “Parallel prefix computation, Journal of ACM.La.Jolla CA,Vol.27,pp.831-838,October 1980.
[7] Reto Zimmermann. Binary Adder Architectures for Cell-Based VLSI an their Synthesis. Hartung-Gorre, 1998.
[8] D. Harris, “A taxonomy of parallel prefix networks,” in Signals, Systems and Computers,2003. Conference Record of Thirty Seventh Asilomar Conference on, vol. 2, the Nov. 2003,pp.2217.
[9] N. H. E. Weste and D. Harris, CMOS VLSI Design, 4th edition, Pearson Addison-Wesley, 2011.
[10] H. Ling, High-speed binary adder," IBM Journal of Research and Development, vol. 25,no. 3, pp. 156 March 1981.
[11] 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.
[12] D. H. K. Hoe, C. Martinez, and J. Vundavalli, “Design and Characterization of Parallel Prefix Adders using FPGAs, ”IEEE 43rd Southeastern Symposium on System Theory, pp. 170-174, March 2011.
[13] T. Matsunaga, S. Kimura, and Y. Matsunaga.“Power-conscious syntheses of parallel prefix adders under bitwise timing constraints,” Proc. the Workshop on Synthesis And System Integration of Mixed Information technologies(SASIMI), Sapporo, Japan, October 2007,pp. 7–14.