High Speed and Low Power Design of Parallel Prefix Adder

[1]B. Krishna, [2]P. Siva Durga Rao, [3]N. V. G. Prasad

[1, 2, 3]Dept. of ECE, Sasi Institute of Technology & Engineering, Tadepalligudem, AP, India

Abstract

Parallel-prefix adders (also known as carry-tree adders) are known to have the best performance in VLSI designs compared to that of conventional Ripple Carry Adder (RCA). However, each type of parallel prefix adder has its own pros and cons and are chosen according to the design requirement of the application. This paper investigates mainly two types of carry-tree adders, the brent kungg adder and the Kogge-Stone adder and compares them. These designs were implemented on a Xilinx Virtex 5 FPGA and found that brent kungg adder occupies less area compared to that of the koggestone adder because of its minimal number of usage of nodes and lesser wiring interconnects. And 128 and 144 bit widths of the Koggestone adder were also designed and compared to the corresponding bitwidths of the ripplecarry adder and found ripple carry performs faster up to 128 bit and as bitwidths increase beyond 128 bits parallel pefix adders performs faster.

Keywords

Rca, FPGA, ASIC, VlSI

I. Introduction

The binary adder is the critical element in most digital and microprocessor data path units. As such, extensive research continues to be focused on improving the power-delay performance of the adder. In VLSI implementations, parallel-prefix adders are known to have the best performance. Reconfigurable logic such as Field Programmable Gate Arrays (FPGAs) has been gaining in popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for many practical designs involving mobile DSP and telecommunications applications and a significant reduction in development time and cost over Application Specific Integrated Circuit (ASIC) designs. The power advantage is especially important with the growing popularity of mobile and portable electronics, which make extensive use of DSP functions. However, because of the structure of the configurable logic and routing resources in FPGAs, parallel-prefix adders will have a different performance than VLSI implementations. In particular, most modern FPGAs employ a fast-carry chain which optimizes the carry path for the simple Ripple Carry Adder (RCA).

In this paper, the practical issues involved in designing and implementing tree-based adders on FPGAs are described. Several tree-based adder structures are implemented and characterized on a FPGA and compared with the Ripple Carry Adder (RCA). Finally, some conclusions and suggestions for improving FPGA designs to enable better tree-based adder performance are given.

II. Carry-Tree Adder Designs

Parallel-prefix adders, also known as carry-tree adders, pre-compute the propagate and generate signals [1]. These signals are variously combined using the fundamental carry operator (fco) [2].

(g_L, p_L) o $(g_R, p_R) = (g_L + p_L \bullet g_R, p_L \bullet p_R)$ (1)

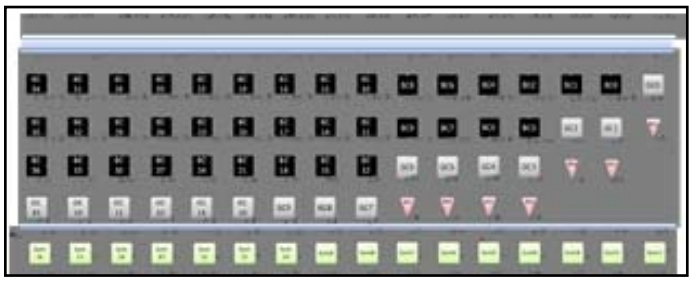
Due to associative property of the fco, these operators can be combined in different ways to form various adder structures. For, example the four-bit carry-lookahead generator is given by:

$c_4 = (g_4, p_4)$ o $[(g_3, p_3)$ o $[(g_2, p_2)$ o $(g_1, p_1)]\ ]$ (2)

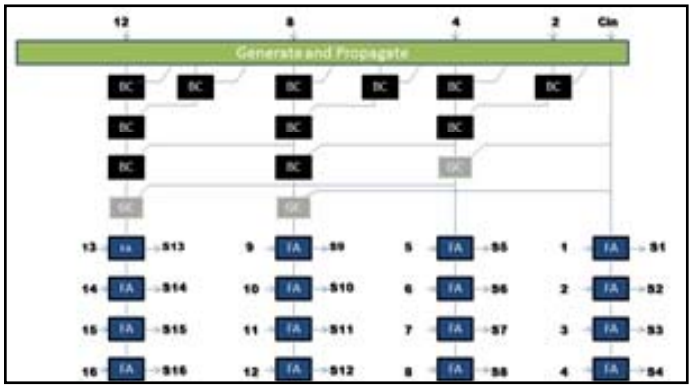
A simple rearrangement of the order of operations allows parallel operation, resulting in a more efficient tree structure for this four bit example:

$c_4 = [(g_4, p_4)$ o $(g_3, p_3)]$ o $[(g_2, p_2)$ o $(g_1, p^1)]$ (3)

It is readily apparent that a key advantage of the tree-structured adder is that the critical path due to the carry delay is on the order of log2N for an N-bit wide adder. The arrangement of the prefix network gives rise to various families of adders. For a discussion of the various carry-tree structures, see [1, 3]. For this study, the focus is on the Kogge-Stone adder [4], known for having minimal logic depth and fanout (see Fig. 1(a)). Here we designate BC as the black cell which generates the ordered pair in equation (1); the Gray Cell (GC) generates the left signal only, following [1]. The interconnect area is known to be high, but for an FPGA with large routing overhead to begin with, this is not as important as in a VLSI implementation. The regularity of the Kogge-Stone prefix network has built in redundancy which has implications for fault-tolerant designs [5]. The sparse Kogge-Stone adder, shown in Fig 1(b), is also studied. This hybrid design completes the summation process with a 4 bit RCA allowing the carry prefix network to be simplified. Another kind of carry tree adder, brent kungg adder is also implemented as shown in fig. 2.



(a)



(b)

Fig. 1(a): 16 Bit Kogge-Stone Adder and (b). Sparse 16-Bit Kogge-Stone Adder

The general form of the koggestone adder and Sparse koggestone adder is shown in fig. 2(a), 2(b). Another kind of carry tree for general consideration we have is Spanning tree Carry-Lookahead (CLA) Adder. Like the sparse Kogge-Stone adder, this design terminates with a 4-bit RCA.

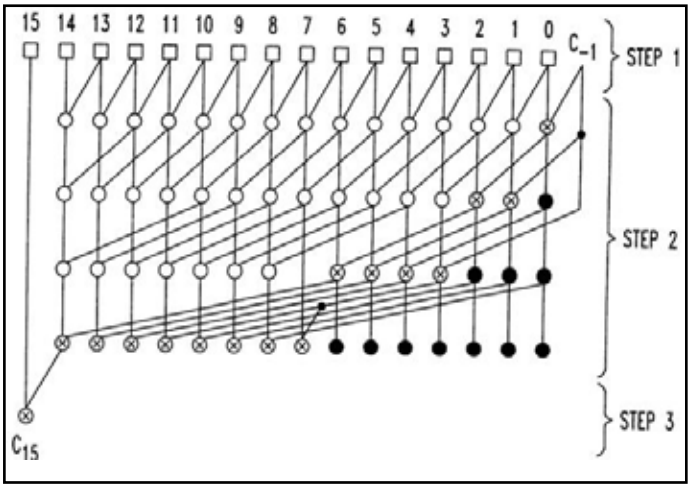


Fig. 2(a): General Implementation form of Koggestone Adder (16 bit)

The implementation structure of sparse koggestone adder is shown in fig. 2(b). Sparse kogge stone has RCA's and redundancy is reduced comapared to its previous adder.

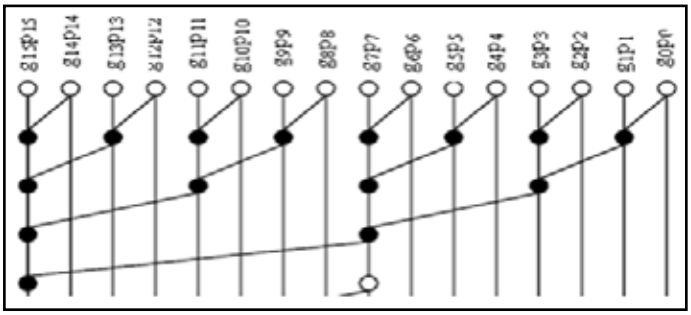


Fig. 2(b): General Implementation form of Koggestone Adder (16 bit)

In [10], the authors considered several parallel prefix adders implemented on a Xilinx Virtex 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 Virtex 5 FPGA. Here 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 spanning tree adder are representative of the latter category. However for designing 128 bit width koggestone adder a series of 16 bit koggestone adders of eight number are placed in parallel.The operation between these stages are maintained by connecting the carry out of the present stage to the next stage and so on for each stage until the final eighth stage of adder.The output carry of the final stage is considered as the total carry out of the 128 bit adder and hence intermediate carry's are not vital in this scenario. Similarly the same is the case for designing 144 bit adder where the difference is it has an extra 16 bit stage compared to the 128 bit adder. Similarly the designing of other carry tree adders like sparse kogge stone adder and spanning tree adder for various bit widths can also be studied.
Another kind of parallel prefix adder proposed in this paper is brent kungg adder. The 16 bit architecture of the brent kungg adder is shown in fig. 3.This adder is a combination of forward carry tree and an inverse carry tree .Forward carry tree is same as sparsekogge stone adder. Compared to the koggestone adder it has the advantage of occupying lesser area which has become possible because of the reduced number of black cells and grey cells and interconnects as well.The logic levels are also reduced.Power consumption is also comparatively low in brent kungg adder. However as the bitwidth increases Brentkungg adder would not be a preferred choice in the perspective of speed(delay is increased) because of its routing complexity which increases proportional to the bitwidth.Sparsekogge stone adder and spanning tree adder has reduced redundancy compared to koggestone . But our focus in this paper is with brent kungg and koggestone adder.

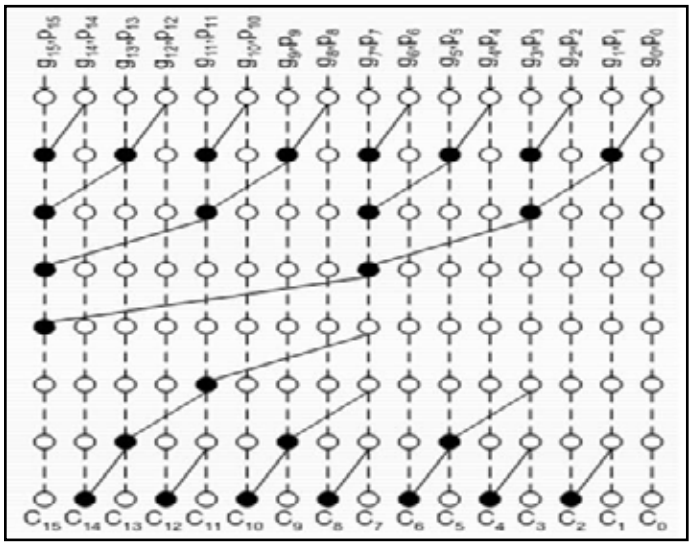


Fig. 3: 16 bit BrentKungg Adder Architecture

III. Method of Study

The adders to be studied were designed with varied bit widths of 16 bits, 128 bits and 144 bits coded in Verilog. The functionality of the designs were verified via simulation with ModelSim. The Xilinx ISE 10.1 software was used to synthesize the designs onto the Virtex 5 FPGA and delay measurements are made using quartus software.
The parallel prefix network was analyzed to determine if a specific pattern could be used to extract the worst case delay. Considering the structure of the Generate- Propagate (GP) blocks (i.e., the BC and GC cells), we were able to develop the following scheme, by considering the following subset of input values to the GP blocks.
Table 1: Subset of (g, p) Relations Used for Testing

(g_L, p_L)	(g_R, p_R)	$(g_L + p_L g_R, p_L p_R)$
(0,1)	(0,1)	(0,1)
(0,1)	(1,0)	(1,0)
(1,0)	(0,1)	(1,0)
(1,0)	(1,0)	(1,0)

If we arbitrarily assign the (g, p) ordered pairs the values (1,0) = True and (0, 1) = False, then the table is self-contained and forms an OR truth table. Furthermore, if both inputs to the GP block are False, then the output is False; conversely, if both inputs are True, then the output is True. Hence, an input pattern that alternates between generating the (g, p) pairs of (1, 0) and (0, 1) will force its GP pair block to alternate states. Likewise, it is easily seen that the GP blocks being fed by its predecessors will also alternate states. Therefore, this scheme will ensure that a worse case delay will be generated in the parallel prefix network since every block will be

active. In order to ensure this scheme works, the parallel prefix adders were synthesized with the "Keep Hierarchy" design setting turned on (otherwise, the FPGA compiler attempts to reorganize the logic assigned to each LUT). With this option turned on, it ensures that each GP block is mapped to one LUT, preserving the basic parallel prefix structure, and ensuring that this test strategy is effective for determining the critical delay.

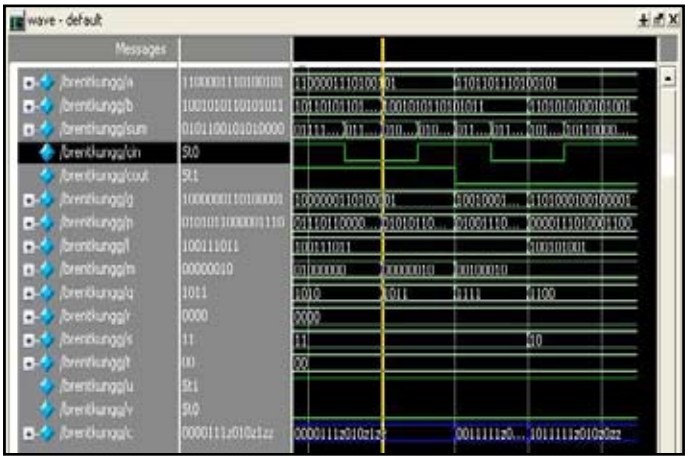


Fig. 4: Screen Shot of Simulation Result of a 16 Bit Brentkungg Adder

IV. Discussion of Results

The design summary of brentkungg adder obtained from the Xilinx ISE synthesis reports are shown in Fig. 5.The number of slice Look Up Tables(LUT's) utilized is 29 where as the koogestone adder utilizes 68 slice LUT's.Hence there is a drastic reduction in number of slices that a brentkungg adder utilize for implementation.

Project File:	adders.ise
Module Name:	brentkungg
Target Device:	xc5vfx200t-2ff1738
Product Version:	ISE 10.1 - Foundation Simulator
Design Goal:	Balanced
Design Strategy:	Xilinx Default (unlocked)
adders Partition	
No partition information was found.	
Device Utilization	
Slice Logic Utilization	**Used**
Number of Slice LUTs	29
Number used as logic	29
Number using O6 output only	29

Fig. 5: Design Summary of Brentkungg Adder

The other carry tree adders also requires more area compared to brent kungg adder. The table 2 shows the percentage reduction in the area in the perspective of slice LUT's occupancy of various parallel prefix adders.

Table 2: % of LUT's Occupied By Each Adder

Parallel Prefix Adder Type	Number of LUT's required	% of LUT's reduced in brent kungg adder
KoggeStone Adder	68	57.35%
Sparse KoggeStone Adder	44	34.09%
Spanning Tree Adder	32	9.36%

The dealy measurement for ripple carry adder and koggestone adder are made with quartus software for varied bit widths of 16 bits, 128 bits and 144 bits.It is observed that ripple carry adder performs faster up to 128 bits and as the bitwidth goes beyond koggestone adder operates faster.The figure 6 depicts the plot of bitwidth vs delay.

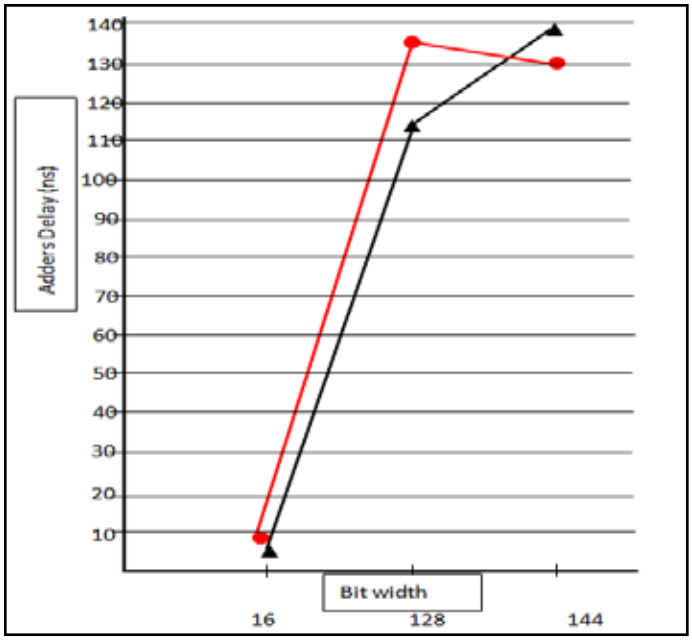


Fig. 4: Bitwidth vs Delay Measurements of RCA and Koggestone Adders

However, the spanning tree adder is significantly slower at higher bit widths. The structure of the spanning tree adder results in an extra stage of logic for some adder outputs compared to the Kogge-StoneThe delay of ripple carry adder for 16 bit is obtained as 9.166ns which is comparatively less than a delay of Carry tree adder(Koggestone adder) which is 9.563 ns. And similarly same is the case with 128 bit adders whose delays are registered as 113.224ns and 133.701ns for RCA and KSA adders respectively. But for 144 bitwidth they are obtained as 138.243ns for RCAand 130.389ns for KSA.

Hence for 144 bit adders Carry tree adders(Kogge stone adder) has lesser delay than RCA and therefore performs faster.The table 3 shows the delays of all adders with varied btwidths.

Table 3: Comaparision of Delay's in All Adders of Various Bit Widths

Bit width	Delay in ns	
	Ripple Carry Adder	Kogge Stone Adder
16 - bit	9.166	9.563
128 - bit	113.224	133.701
144 - bit	138.243	130.839

V. Summary and Future Work

The koggestone adder and brentkungg adder have been designed and synthesized.The adder of latter type is found to occupy lessslice of LUT's compared to former type.The delay however brent kunng adder is not a preferred choice in case of higher bitwidths . Throughout, there were several adder topologies with different advantages and disadvantages and I finally came to the decision to choose Brentkung adder architecture. Its implementation was simpler than Radix-4 Kogge-Stonetree.As a result the layout of the circuit would be less complicated. Even though it is almost equals koggestone adder in terms of dealy it can be used for lesser routing complexity applications. 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. To make the brentkungg adder to operate as fast as possible, we can look into some ideas to minimize the propagation delay such as modifying the Brent-Kung tree, added 3 more dot products and cut the number of stages for the critical path from 7 to 5. For the Brent-Kung tree in order to get the carryout C14, the adder has to wait for the product of C7&C11, then C11&13 and finally C13&P14. On the other hand for the idea of making Brent-Kung tree C7 directly producted with C13 at the fourth stage. This helps the C14 to be available at the fifth stage. This can be a topic of future research and should understand how good this can be implemented retaining its original properties

References

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

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

[3] S. Knowles,"A Family of Adders", Proc. 15th IEEE Symposium onComputer Arithmetic, pp 277-281, 2001.

[4] M. M. Ziegler, M. R. Stan,"A Unified Design Space for Regular Parallel Prefix Adders", IEEE Journal of Design, Automation and Test in Europe Conference and Exhibition, Vol. 2, pp 1386 - 1387, 2004.

[5] T. Hans, D. A Carlson,"Fast Area-Efficient VLSI Adders", Proc. 8thIEEE Symposium on Computer Arithmetic, pp 49-56, 1987.

[6] Y. Choi,"Parallel Prefix Adder Design", Proc. 17th IEEE Symposium onComputer Arithmetic, pp 90-98, 27th June 2005.

[7] V. Ionescu, I. Bostan, L. Ionescu,"Systematic Design for IntegratedDigital Circuit Structures", IEEE Journal of Semiconductor Conference,2004, Vol. 2, pp 467 – 470, 2004.

[8] R. E. Ladner, M. J. Fischer, "Parallel Prefix Computation", JACM, Vol. 27-4, pp 831-838, 1980.

[9] P. M. Kogge, H. S. Stone,"A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations", IEEE Trans. on Computers, Vol. C-22, No. 8, August 1973.

[10] K. Vitoroulis, 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.

[11] S. Xing, 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.

[12] D. Harris, "A Taxonomy of Parallel Prefix Networks", In Proc. 37th Asilomar Conf. Signals Systems and Computers, pp. 2213–2217, 2003.