International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 3, Issue. 4, July-august. 2013 pp-2266-2270 ISSN: 2249-6645

Designing and Characterization of koggestone, Sparse Kogge stone, Spanning tree and Brentkung Adders

V.Krishna Kumari [(1)], Y.Sri Chakrapani[(2)]
[(1)]M.Tech, Department of Electronics & Communication Engineering
[(2)]Associate Professor Department of Electronics & Communication Engineering
[(1)(2)]Gudlavalleru Engineering college, Gudlavalleru.

ABSTRACT: *Adders are known to be the frequently used ones in VLSI designs. In digital design we have half adder and full adder, by using these adders we can implement ripple carry adder(RCA). RCA is used to perform any number of additions. In this RCA is serial adder and it has propagation delay problem. With increase in ha & fa circuits, delay also increases simultaneously. That's the reason these parallel adders (parallel prefix adders) are preferred. The parallel prefix adders are KS adder(kogge-stone),SKS adder(sparse kogge-stone),Spanning tree and Brentkung adderS. These adders are designed and compared by using area and delay constraints. Simulation and synthesis by model sim6.4b, Xilinx ise10.1i.*

I. INTRODUCTION

In Processors (DSP) and microprocessor data path units, adder is an important element. 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. This work was supported in part by NSF LSAMP and UT-System STARS awards. The FPGA ISE synthesis software was supplied by the Xilinx University program described. An efficient testing strategy for evaluating the performance of these adders is discussed. Several tree- based adder structures are implemented and characterized on a FPGA and compared with the Ripple Carry Adder (RCA) and the Carry Skip Adder (CSA). 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. These signals are variously combined using the fundamental carry operator (fco) .

$(g_L, p_L) \circ (g_R, p_R) = (g_L + p_L \bullet g_R, p_L \bullet p_R)$ (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-look ahead generator is given by:

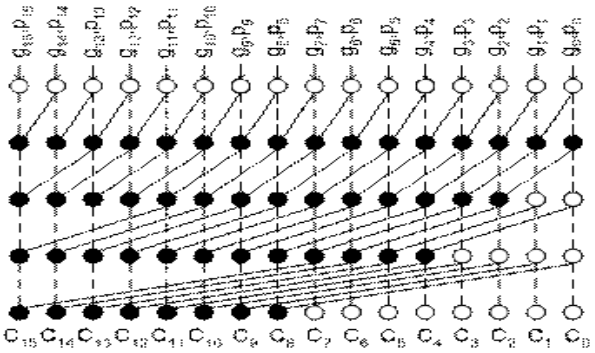
$c_4 = (g_4, p_4) \circ [(g_3, p_3) \circ [(g_2, p_2) \circ (g_1, p_1)]]$ (2)

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

$c_4 = [(g_4, p_4) \circ (g_3, p_3)] \circ [(g_2, p_2) \circ (g_1, p_1)]$ (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 $log_2 N$ 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. 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. 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. 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.

(a)



(b)

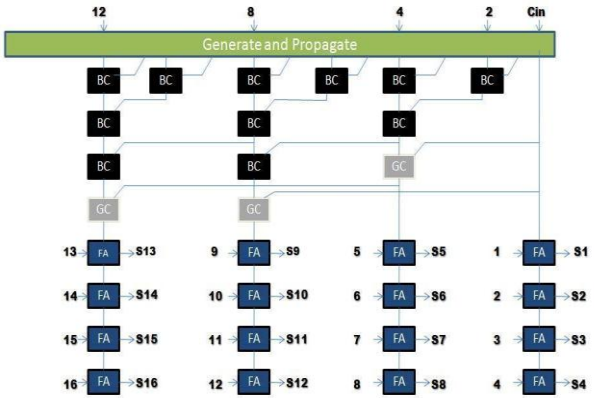
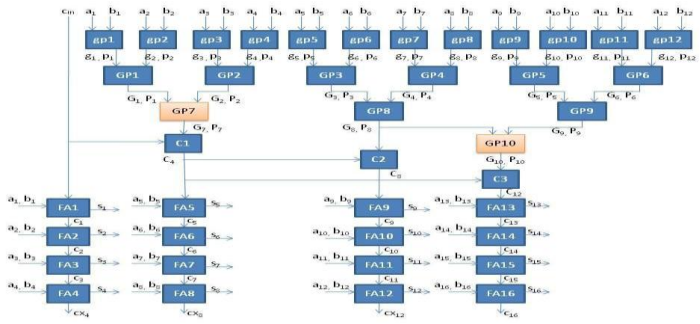
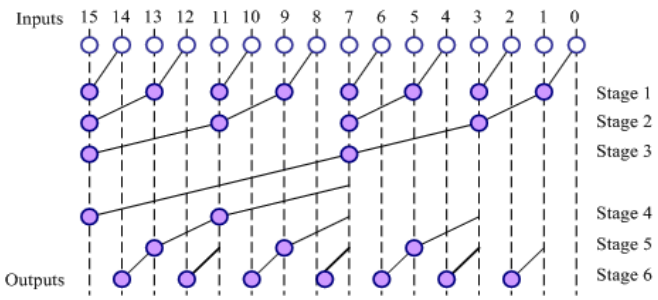


Fig. 1. (a) 16 bit Kogge-Stone adder and (b) sparse 16-bit Kogge-Stone adder

Another carry-tree adder known as the spanning tree and Brent kung carry-look ahead (CLA) adders are examined . Like 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. Also of interest for the spanning-tree CLA is its testability features .

(a)





(b)

Fig. 2. (a)Spanning Tree Carry Look ahead Adder (16 bit) (b) 16 bit Brent kung adder.

III. RELATED WORK

The ripple carry adder with the carry-lookahead, 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 Virtex II yielded similar results. In 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 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 spanning tree adder are 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. In addition to being able to compare the simulation data with measured data using a high-speed logic analyzer, our results present a different perspective in terms of both results and types of adders .

The adders to be studied were designed with varied bit widths up to 128 bits and coded in VHDL. The functionality of the designs were verified via simulation with ModelSim 6.4b. The Xilinx ISE 10.1 software was used to synthesize the designs onto the Spartan 3E FPGA. In order to effectively test for the critical delay, two steps were taken. First, a memory block (labeled as ROM in the figure below) was instantiated on the FPGA using the Core Generator to allow arbitrary patterns of inputs to be applied to the adder design. A multiplexer at each adder output selects whether or not to include the adder in the measured results, as shown in Fig. 3. A switch on the FPGA board was wired to the select pin of the multiplexers. This allows measurements to be made to subtract out the delay due to the memory, the multiplexers, and interconnect (both external cabling and internal routing.

IV. IMPLEMENTATION

Xing and Yu noted that delay models and cost analysis for designs developed for VLSI technology do not map Second, the parallel prefix network was analyzed to directly to FPGA designs. They compared the design of 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.

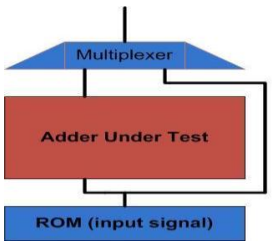


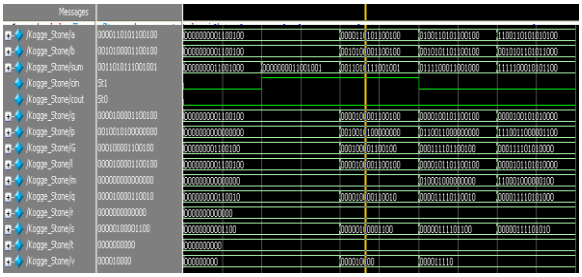
Fig. 3. Circuit used to test the adders adder

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. The designs were also synthesized for speed rather than area optimization.

The adders were tested with a Tektronix TLA7012 Logic Analyzer. The logic analyzer is equipped with the 7BB4 module that provides a timing resolution of 20 ps under the MagniVu setting. This allows direct measurement of the adder delays. The Spartan 3E development board is equipped with a soft touch-landing pad which allows low capacitance connection directly to the logic analyzer. The test setup is depicted in the figure below.

V. SIMULATION AND SYNTHESIS REPORT

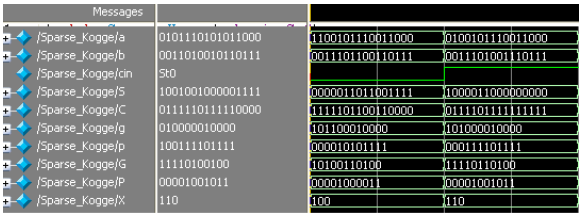


(a)

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	37	9,312	1%
Logic Distribution			
Number of occupied Slices	24	4,656	1%
Number of Slices containing only related logic	24	24	100%
Number of Slices containing unrelated logic	0	24	0%
Total Number of 4 input LUTs	37	9,312	1%
Number of bonded IOBs	50	232	21%

(b)

Fig.4:(a)Kogge stone simulated wave form (b) KS device utilization.

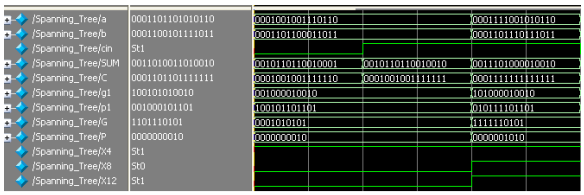


(a)

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	51	9,312	1%
Logic Distribution			
Number of occupied Slices	30	4,656	1%
Number of Slices containing only related logic	30	30	100%
Number of Slices containing unrelated logic	0	30	0%
Total Number of 4 input LUTs	51	9,312	1%
Number of bonded IOBs	65	232	28%

(b)

Fig.5:(a)Sparse kogge stone simulated wave form (b) SKS device utilization.

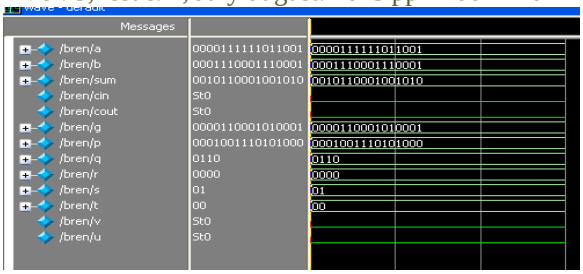


(a)

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	32	9,312	1%
Logic Distribution			
Number of occupied Slices	24	4,656	1%
Number of Slices containing only related logic	24	24	100%
Number of Slices containing unrelated logic	0	24	0%
Total Number of 4 input LUTs	32	9,312	1%
Number of bonded IOBs	65	232	28%

(b)

Fig.6:(a)Spanning tree simulated wave form (b) Spanning tree device utilization.



(a)

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	24	4656	0%
Number of 4 input LUTs	43	9312	0%
Number of bonded IOBs	49	232	21%

(b)

Fig.7:(a) Brent kung simulated wave form (b) Brent kung device utilization

REFERENCE

[1]. 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.
[2]. D. Gizopoulos, M. Psarakis, A. Paschalis, and Y. Zorian, "Easily Testable Cellular Carry Lookahead Adders," *Journal of Electronic Testing: Theory and Applications 19*, 285-298, 2003.
[3]. 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.
[4]. M. Bečvář and P. Štukjunger, "Fixed-Point Arithmetic in FPGA," *Acta Polytechnica*, vol. 45, no. 2, pp. 67-72, 2005.
[5]. P. M. Kogge and 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.
[6]. P. Ndai, S. Lu, D. Somesekhar, and K. Roy, "Fine-Grained Redundancy in Adders," *Int. Symp. on Quality Electronic Design*, pp. 317-321, March 2007.
[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]. N. H. E. Weste and D. Harris, *CMOS VLSI Design*, 4th edition, Pearson–Addison-Wesley, 2011.
[9]. R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.*, vol. C-31, pp. 260-264, 1982.
[10]. D. Harris, "A Taxonomy of Parallel Prefix Networks," in *Proc. 37th Asilomar Conf. Signals Systems and Computers*, pp. 2213–7, 2003.