DESIGN AND IMPLEMENTATION OF HIGH SPEED ADDERS

[1]V Saranya Devi [2]K.C.Kullayappa Naik
[1]M.Tech, QIS Institute of Technology, Vengamukkalapalem, AP-India,
E-mail: saranya.vsd@gmail.com
[2]HOD of ECE Department, Associate Professor, E-mail: kcknaik@gmail.com

ABSTRACT— *Parallel-prefix adders (also known as carry tree adders) are known to have the best performance in VLSI designs. However, this performance advantage does not translate directly into FPGA implementations due to constraints on logic block configurations and routing overhead. This paper investigates two types of carry-tree adders (the Kogge-Stone, sparse Kogge-Stone) and compares them to the simple Ripple Carry Adder (RCA). These designs of varied bit-widths were implemented on a Xilinx Spartan 3E FPGA and delay measurements were made with a high-performance logic analyzer. Due to the presence of a fast carry-chain, the RCA designs exhibit better delay performance up to 128 bits. The carry-tree adders are expected to have a speed advantage over the RCA as bit widths approach 128.*

1. INTRODUCTION

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor data path units. Such, extensive research continues to be focused on improving the power delay performance of the adder. In the 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 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 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 adders 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.

2. ARCHITECTURE OF PARALLEL PREFIX ADDERS

This section describes the basic adders used in this thesis. The adders implemented on FPGAs are the Kogge-Stone adder, ripple carry adder and sparse Kogge-Stone adder. The ripple carry adder is one of the simplest adder designs. The Kogge-Stone adder is an example of a parallel prefix adder. The internal blocks used in the adder designs are described in detail in this section.

Ripple Carry Adder: The ripple carry adder is one of the simplest adders. It consists of a cascaded series of full adders. For example 4-bit adder can be constructed by cascading four full adders together as shown in Figure. The ripple carry adder is relatively slow as each full adder must wait for the carry bit to be calculated from the previous full adder. The worst case delay of a ripple carry adder occurs when cin propagates from the first stage to the most significant bit position. The delay for an N-bit adder is given by,

$$t_{adder} = (N-1)t_{carry} + t_{sum}$$

Kogge-Stone Adder:

The Kogge-Stone adder is classified as a parallel prefix adder since the generate and the propagate signals are pre-computed. In a tree-based adder, carries are generated in tree and fast computation is obtained at the expense of increased area and power. The main advantage of this design is that the carry tree reduces the logic depth of the adder by essentially generating the carries in parallel. The parallel-prefix adder more favorable in terms of speed due to the O(log2*n*) delay through the carry path compared to O(*n*) for the RCA. The Kogge-Stone adder is widely used in high-performance 32-bit, 64-bit, and 128-bit adders as it reduces the critical path to a great extent compared to the ripple carry adder.

The operation of the tree-based adder can be understood using the concept of the fundamental carry operation (*fco*). This operator works on the generate and propagate pairs as defined by,

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

Where gL, pL are the left input generate and propagate pairs and gR, pR are the right input generate and propagate pairs to the cell. For example, in a 4-bit carry look ahead adder, the carry combination equation can be expressed as,

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

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

:

:

$$= g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2 + p_4 \cdot p_3 \cdot p_2 \cdot g_1$$

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

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

:

:

$$= g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2 + p_4 \cdot p_3 \cdot p_2 \cdot g_1$$

Since the *fco* obeys the associativity property, the expression can be reordered to yield parallel computations in tree based structure,

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

The complete schematic for the 8-bit Kogge-Stone adder is shown in Figure. An 8-bit Kogge-Stone adder is built from eight generate and propagate (GP) blocks, twelve black cell (BC) blocks, eight gray cell (GC) blocks, and eight sum blocks. To be aesthetic, an extra column has been added in our design to show the computation of *c8*. In practice, c8 is generated by just adding an extra gray cell in the last column.

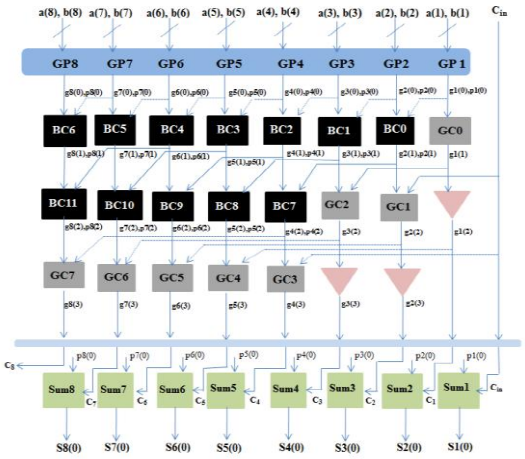

Figure 2.1: 8-bit Kogge-Stone adder

Sparse Kogge-Stone Adder:

The sparse Kogge-Stone adder consists of several smaller ripple carry adders (RCAs) on its lower half and a carry tree on its upper half. Thus, the sparse Kogge-Stone adder terminates with RCAs. The number of carries generates is less in a sparse Kogge-Stone adder compared to the regular Kogge-Stone adder. The functionalities of the GP block, gray cell and black cell remains exactly the same as the regular Kogge-Stone adder. The schematic for a 16-bit sparse Kogge-Stone adder is shown in Figure 2.2. Sparse and regular Kogge-Stone adders have essentially the same delay when implemented on an FPGA although the former utilizes much less resources.

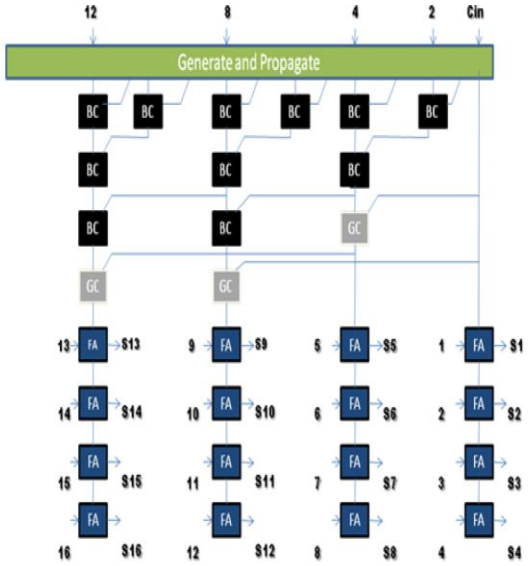

Figure 2.2: 16-bit sparse kogge-stone adder

3. ALLIED WORK

The Basys board is a circuit design and implementation platform that anyone can use to gain experience building real digital circuits. Built around Xilinx Spartan-3E Field Programmable Gate Array and Cypress EZUSB controller, then the Basys board provides complete, ready-to-use a hardware suitable for hosting circuits ranging from basic logic devices to complex controller. A large collection of on-board I/O devices and all required FPGA support circuits are included, so the countless designs can be created without the need for any other components.

Four standard expansion connectors allow designs to grow beyond the Basys board using breadboards, user-designed circuit boards, or Pmods (Pmods are inexpensive analog and digital I/O modules that offer A/D & D/A conversions, motor drivers, sensor inputs and many other features). Signals on the 6-pins connectors are protected against ESD damage and short-circuits ensuring a long operating life in any environment.

The Basys board work seamlessly with all versions of the Xilinx ISE tools, including the free Web Pack. It ships with a USB cable that provides power and a programming interfaces. So no other power supplies or programming cables are required.

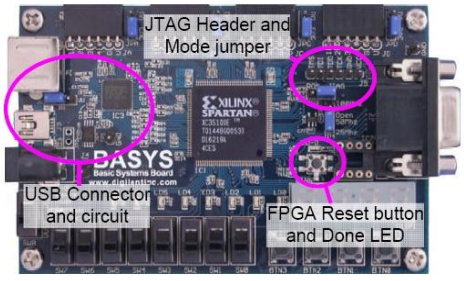

Figure 3.1. Basys programming circuit location

4. RESULTS

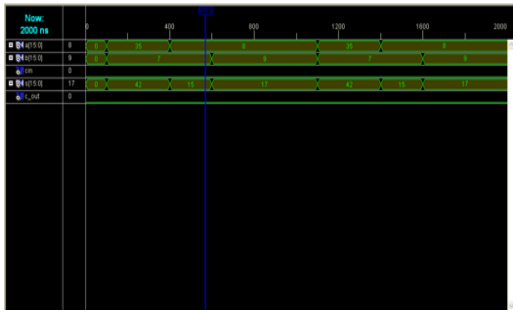

Fig 4.1 Simulation Result of 16bit RC

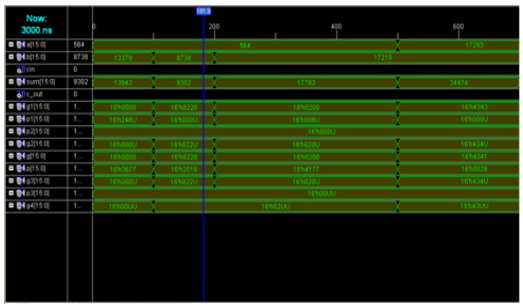

Fig 4.2 Simulation Results of KSA -16 bit

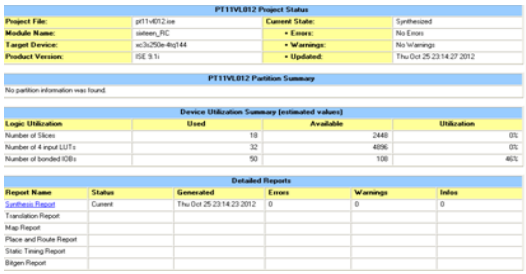

Fig 4.3 Design summary Report of 16bit RC

Experimental Result

	Delay	No.of slice	Memory
4bit RC	8.696ns	4	152556 KB
4bit KSA	9.838ns	4	151532 KB
16bit RC	24.686	18	152556 KB
16bit KSA	21.700ns	21	154604 KB
16it SKSA	20.042ns	22	153580 KB
128bit RC	162.446ns	147	159724 KB
128bit KSA	26.877ns	646	173612 KB
128bit SKSA	75.400ns	272	164844 KB

5. CONCLUSION

The Above Experimental Results proved that parallel prefix adders are very high speed than normal Ripple carry Adders when it will increase the width of the adders. All adders will successfully synthesized using Xilinx9.1 synthesis tool and simulation will done using ISE simulator. We are generating a synthesis reports for Spartan 3E Fpga.

6. REFERENCES

[1]Design and Characterization of Parallel Prefix Adders using FPGAs David H. K. Hoe, Chris Martinez and Sri Jyothsna Vundavalli Department of Electrical Engineering The University of Texas, Tyler dhoe@uttyler.edu 978-1-4244-9593-1/11/$26.00 ©2011 IEEE

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

[3] D. Harris "A Taxonomy of Parallel Prefix Networks" in Proc 37th Asilomar Conf. Signal Systems and Computers, pp. 2213–7, 2003.

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

[5] P. Ndai, S. Lu, D. Somasekhar, and K. Roy "Fine- Grained Redundancy in Adders" Int. Symp. on Quality Electronic Design, pp. 317-321, March 2007.

[6] M. Becvar and P. Stukjunger "Fixed-Point Arithmetic in FPGA" Ac-ta Polytechnic, vol. 45, no. 2, pp. 67-72, 2005.

[7] 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.

[8] 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.

[9] 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.

[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.

[11] 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.