*IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*
*e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 6, Issue 5 (Jul. - Aug. 2013), PP 41-48*
*www.iosrjournals.org*

# Design and implementation of Parallel Prefix Adders using FPGAs

M.Venkata Durga rama raju[1], A.Deepthi[2]
[(1)]*M.E, Department of Electronics & Communication Engineering*
[(2)]*Asst.Professor, Department of Electronics & Communication Engineering*
[(1)(2)]*MVSREC, Nadargul, Hyderabad.*

***Abstract:*** *Adders are known to have the frequently used in VLSI designs. In digital design we have half adder and full adder, main adders by using these adders we can implement ripple carry adders. RCA use to perform any number of addition. In this RCA is serial adder and it has commutation delay problem. If increase the ha&fa simultaneously delay also increase. That's why we go for parallel adders(parallel prefix adders). IN the parallel prefix adder are ks adder(kogge-stone),sks adder(sparse kogge-stone),spaning tree and brentkung adder. These adders are designd and implemented on FPGA sparton3E kit. Simulated and synthesis by model sim6.4b, Xilinx ise10.1.*

**Submitted Date 26 June 2013** **Accepted Date: 01 July 2013**

## I. Introduction

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor datapath 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

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 [1]. These signals are variously combined using the *fundamental carry operator* (fco) [2].

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

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:

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

A simple rearrangement of the order of operations allows parallel operation, resulting in a more efficient tree

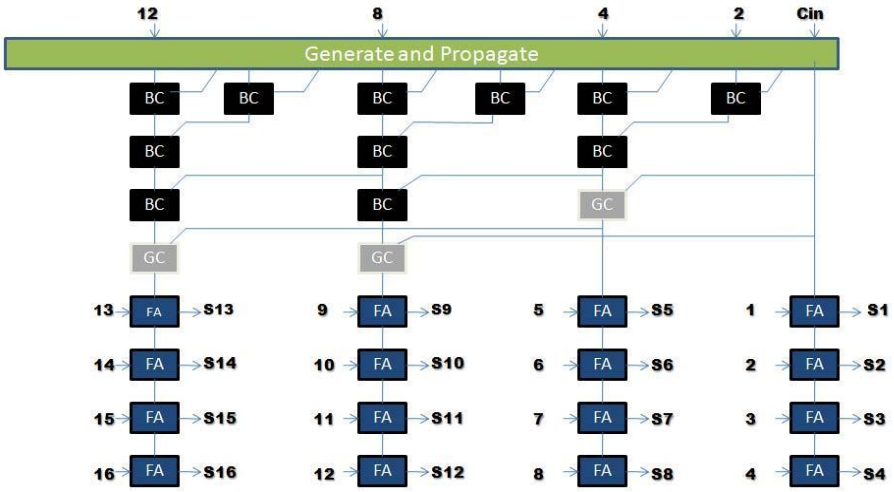
structure for this four bit example:

$$c_4 = [(g_4, p_4) \circ (g_3, p_3)] \circ [(g_2, p_2) \circ (g_1, p_1)] \quad (3)$$

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, 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.

978-1-4244-9593-1/11/$26.00 ©2011 IEEE 168

(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 carry-lookahead (CLA) adder is also examined [6]. 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 [7].

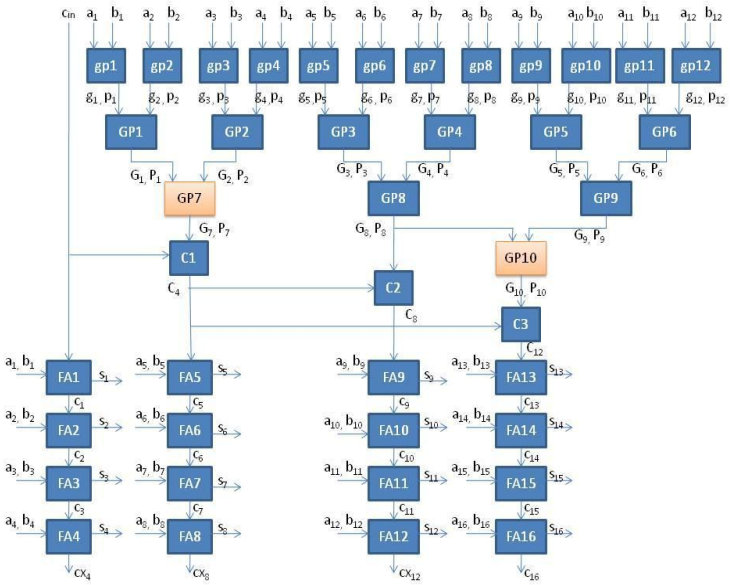



**Fig. 2**. Spanning Tree Carry Lookahead Adder (16 bit)

## 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 [9]. 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 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 as those presented in [8-10].

## IV. Method Of Study

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. The Xilinx ISE 12.2 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 CoreGenerator 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).

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 [8]. They compared the design of

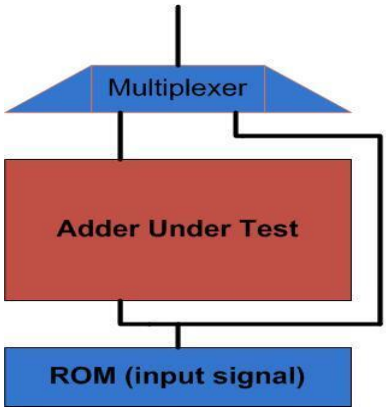



**Fig. 3**. Circuit used to test the adders adder

etermine 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 I**: **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. 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.

**Fig. 4**. Test setup showing the Logic Analyzer and Spartan 3E development board

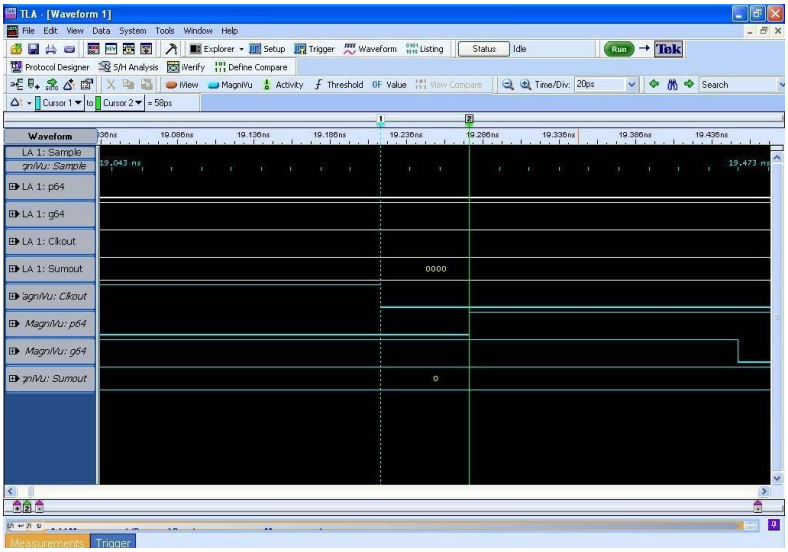



**Fig. 5**. Screen shot of a delay measurement for a 64 bit adder using MagniVu timing (blue traces) on the TLA 7012.

## V. Discussion of Results

The simulated adder delays obtained from the Xilinx ISE synthesis reports are shown in Fig. 6. The simulation results for the carry skip adders are not included because the ISE software is not able to correctly identify the critical path through the adder and hence does not report accurate estimates of the adder delay.

Observe that a semi-log plot is employed, so as expected the tree-adder delay plots as a straight line on this graph. Somewhat surprising is the fact that the sparse Kogge-Stone adder has about the same delay as the regular Kogge -Stone adder. Because the sparse Kogge Stone completes the summation process with a 4 bit RCA, which are optimized via the fast carry chain, its performance is expected to be intermediate between the regular Kogge-Stone adder and the RCA. The impact of the routing overhead would seem to be a likely cause. However, according to the synthesis reports, the delay with the logic only makes the regular Kogge -Stone slightly faster. This will need to be a topic of further investigation.

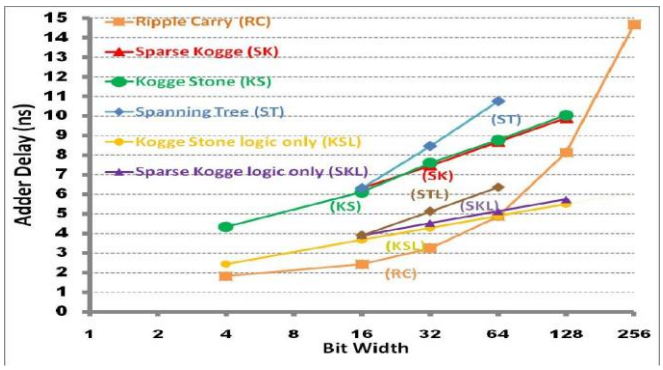



**Fig. 6**. Simulation results for the adder designs

Overall, when the delay due to routing overhead is removed, the tree adders are now closer to the simple RCA design. The RCA adder exhibits the best delay with widths up to 64 bits when the routing delay is excluded and out to 128 bits with the routing delay included.

Figures 7 and 8 depict the measured results using the TLA. A comparison between the tree adders and the RCA is given in Figure 7. The basic trends are the same: the tree adders exhibit logarithmic delay dependence on bit widths and the RCA has linear performance. An RCA as large as 160 bits wide was synthesizable on the FPGA, while a Kogge-Stone adder up to 128 bits wide was implemented. The carry-skip

adders are compared with the Kogge-Stone adders and the RCA in Figure 8. Carry skip adders with a skip of four and eight were implemented. The poor performance of the carry skip adders is attributable to the significant routing overhead incurred by this structure.

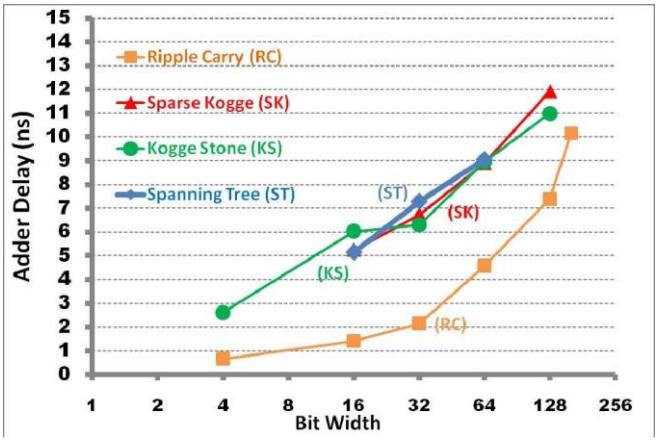



**Fig. 7**. Measured results for the parallel-prefix adder designs compared with the RCA

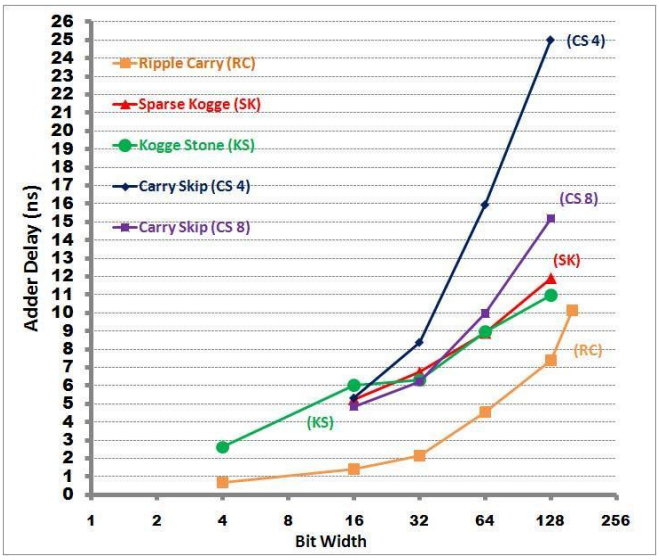



**Fig. 8**. Measured results for the carry-skip adders compared to the RCA and Kogge-Stone adders

The actual measured data appears to be a bit smaller than what is predicted by the Xilinx ISE synthesis reports. An analysis of these reports, which give a breakdown of delay due to logic and routing, would seem to indicate that at adder widths approaching 256 bits and beyond, the Kogge-Stone adder will have superior performance compared to the RCA. Based on the synthesis reports, the delay of the Kogge-Stone adder can be predicted by the following equation:

$$t_{KS} = (n+2)\ _{\text{LUT}} + \rho_{KS}(n) \qquad (4)$$

where $N = 2^n$, the adder bit width, $_{\text{LUT}}$ is the delay through a lookup table (LUT), and $\rho_{KS}(n)$ is the routing delay of the

Kogge -Stone adder as a function of *n*. The delay of the RCA can be predicted as:

$$t_{RCA} = (N-2)\ _{\text{MUX}} + \tau_{\text{RCA}} \qquad (5)$$

where $_{\text{MUX}}$ is the mux delay associated with the fast-carry chain and $\tau_{\text{RCA}}$ is a fixed logic delay. There is no routing delay assumed for the RCA due to the use of the fast-carry chain. For the Spartan 3E FPGA, the synthesis reports give

the following values: $_{LUT}$ = 0.612 ns, $_{MUX}$ = 0.051 ns, and $\tau_{RCA}$ = 1.715 ns. Even though $_{MUX}$ << $_{LUT}$, it is expected

That the Kogge-Stone adder will eventually be faster than the RCA because $N = 2^n$, provided that $\rho_{KS}(n)$ grows relatively slower than $(N - 2)_{MUX}$. Indeed, Table II predicts that the Kogge-Stone adder will have superior performance at $N = 256$.

**Table II**: **Delay Results for the Kogge-Stone Adders**

| **N** | **Synth. Predict** | **Route Delay** | **Route Fitted** | **Delay *tKS*** | **Delay *Trca*** |
|---|---|---|---|---|---|
| 4 | 4.343 | 1.895 | 1.852 | 4.300 | 1.817 |
| 16 | 6.113 | 2.441 | 2.614 | 6.286 | 2.429 |
| 32 | 7.607 | 3.323 | 3.154 | 7.438 | 3.245 |
| 64 | 8.771 | 3.875 | 3.800 | 8.696 | 4.877 |
| 128 | 10.038 | 4.530 | 4.552 | 10.060 | 8.141 |
| 256 | – | – | 5.410 | 11.530 | 14.669 |

(all delays given in ns)

The second and third columns represent the total predicted delay and the delay due to routing only for the Kogge-Stone adder from the synthesis reports of the Xilinx ISE software. The fitted routing delay in column four represents the predicted routing delay using a quadratic polynomial in $n$ based on the $N = 4$ to 128 data. This allows the $N = 256$ routing delay to be predicted with some degree of confidence as an actual Kogge-Stone adder at this bit width was not synthesized. The final two columns give the predicted adder delays for the Kogge-Stone and RCA using equations (4) and (5), respectively. The good match between the measured and simulated data for the implemented Kogge-Stone adders and RCAs gives confidence that the predicted superiority of the Kogge-Stone adder at the 256 bit

**Width is accurate.**

This differs from the results in [10], where the parallel-prefix adders, including the Kogge-Stone adder, always exhibited inferior performance compared with the RCA (simulation results out to 256 bits were reported). The work in [10] did use a different FPGA (Xilinx Virtex 5), which may account for some of the differences.

The poor performance of some of the other implemented adders also deserves some comment. The spanning tree adder is comparable in performance to the Kogge-Stone adder at 16 bits. However, the spanning tree adder is significantly slower at higher bit widths, according to the simulation results, and slightly slower, according to the measured data. The structure of the spanning tree adder results in an extra stage of logic for some adder outputs compared to the Kogge-Stone. This fact coupled with the way the FPGA place and route software arranges the adder is likely the reason for this significant increase in delay for higher order bit widths. Similarly, the inferior performance of the carry-skip adders is due to the LUT delay and routing overhead associated with each carry-skip logic structure. Even if the carry-skip logic could be implemented with the fast-carry chain, this would just make it equivalent in speed to the RCA. Hence, the RCA delay represents the theoretical lower limit for a carry-skip architecture on an FPGA.

## VI. Summary And Future Work

Both measured and simulation results from this study have shown that parallel-prefix adders are not as effective as the simple ripple-carry adder at low to moderate bit widths. This is not unexpected as the Xilinx FPGA has a fast carry chain which optimizes the performance of the ripple carry adder. However, contrary to other studies, 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 cryptographical 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. This would improve their performance similar to what is found for the RCA. We plan to explore possible FPGA architectures that could implement a "fast-tree chain" and investigate the possible trade-offs involved. The built -in redundancy of the Kogge-Stone carry-tree structure and its implications for fault tolerance in FPGA designs is being studied. The testability and possible fault tolerant features of the spanning tree adder are also topics for future research.

## References


[1] N. H. E. Weste and D. Harris, *CMOS VLSI Design*, 4th edition, Pearson–Addison-Wesley, 2011.
[2] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Comput.*, vol. C-31, pp. 260-264, 1982.
[3] D. Harris, "A Taxonomy of Parallel Prefix Networks," in *Proc. 37th Asilomar Conf. Signals Systems and Computers*, pp. 2213–7, 2003.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] M. Bečvář and P. Štukjunger, "Fixed-Point Arithmetic in FPGA," *Acta Polytechnica*, vol. 45, no. 2, pp. 67-72, 2005.
[10] 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.