Design Of High Performaance Sparse Kogge Stone Adder

[1] G. MADHUSUDHANA RAO, [2] P. JAYA BABU, [3]DUGGI RAJYA LAKSHMI

[1]H.O.D & Associate Professor, [2]Assistant Professor, [3]M.Tech Scholar
Dept. Of E.C.E,
N.V.R College of Engineering & Technology, Tenali, A.P

ABSTRACT: *A fault tolerant adder implemented by utilizing Kogge-stone configuration can correct the error due to inherent redundancy in the carry tree but no error detection is possible. This proposed design is based on fault tolerant adder that uses sparse kogge-stone adder that is capable of both fault detection and correction. Fault tolerance is achieved by utilizing two additional ripple carry adders that form the basis of triple mode redundancy adder. Triple mode redundancy is one of the most common methods utilized to create fault tolerant designs in both ASIC and FPGA implementations. The latency will be maximized because of the voter in the circuit's critical path. More advanced fault tolerant methods exist which including roving and graceful degradation approaches. Allowing fault tolerance for operating at different levels of abstraction might facilitate a more cost-effective design. Several enhancements are introduced in the design; the error recovery time is reduced by utliziing a 16-bit register, error correction is due to fault in multiple ripple carry adders is included which improves the reliability of the circuit. The power analysis and the analysis of timing for the estimation of setup time and hold time is also performed.*

KEY WORDS: Parallel Prefix technique, Efficient Sparse Kogge Stone adder, Black cell, Gray cell

I. INTRODUCTION

The binary addition is the basic arithmetic operation in any digital circuit and it becomes an essential in most of the digital Systems which including arithmetic and logic unit (ALU), microprocessors, digital signal processing (DSP) and floating point unit (FPU). With the rapid growth in portable electronic equipments and mobile

Communication devices, demand for low voltage and low power technology for VLSI applications is great increasing. In general, high speed adder which includes carry look ahead adder (CLA), carry select adder (CSA), carry bypass adder (CBA), conditional sum adder and later developed parallel prefix adder (PPA). Various prefix algorithms and tree topologies of PPAs have been implemented for solving area, delay, and power efficiencies.

Design of an appropriate tree topology is a trade-off among the fan-out, the wiring count and the logic level. The high-speed 64-bit adder is hybrid sparse radix-4 prefix tree and CSA based on energy-delay optimization methodology. In this paper, a 64-bit hybrid adder is proposed to combine both prefix tree structure (PTS) and CSA for fetching low voltage and low power features.

The three stages prefix tree of the 64-bit hybrid adder computes carries, and then the CSA with add-one circuit selects sums by these carries. Otherwise, the CBA has been added at the third stage of the PTS to diminish fan-ins, fan-outs, wiring counts and transistor counts. With respect to low voltage low power method, complementary metal oxide semiconductor

(CMOS) logic, transmission gate (TG) logic, and pass transistor logic (PTL) are applied in proposed design to fetch full swing operation at each node.

II. EXISTED SYSTEM

As the computing systems, involving high complexity arithmetic, become increasingly embedded and mobile, the concern for energy efficiency, size and speed of these systems also accrues. A large number of such applications involve media processing, such as image, video and audio based applications designed for human interface. Other such computationally intensive applications include data mining and machine learning. A common feature in these applications is that they do not require the outcome to be fully precise, rather an approximate result is adequately acceptable.

Approximate computing is an emerging trend in hardware and software design that exploits this inherent tolerance for inaccuracy for efficiency gain in terms of required hardware, speed and/or power. Several functionally approximate designs for basic arithmetic blocks including adders and multipliers have been proposed. The design of fast and efficient adders has attracted great attention in the approximate computing domain.

Unlike the low-power approximate adders, most of these these high speed, low-latency approximate adders exploit the fact that the carry propagation chain for most input combinations is shorter than that for the worst case. Some prominent examples of these high-performance adders are: error tolerant adders (ETAs) almost correct adder (ACA-I), variable latency speculative adder (VLSA) accuracy configurable adder (ACA-II), gracefully-degrading adder (GDA) and generic accuracy configurable adder (GeAr). Another type of approximate adders are low-power adders, presented.

In order to select an appropriate approximate adder for a given application, comparative performance of the available designs has to be taken into consideration. The designs mentioned above have been evaluated and compared in terms of critical path delay, required hardware and power resources and error statistics. Traditionally, the error performance evaluation and comparison presented for these adders relies on computer simulations, which can only be executed exhaustively for small-sized adders.
As the adder's size grows, time and memory resources required for the exhaustive simulations render them infeasible or impossible. For deployment in any application, systematic quantification of the computational inaccuracy in these designs is indispensable. Unlike the existing statistical performance metrics that rely on Monte- Carlo simulations, the proposed analytical model for probability of error provides an accurate measure for accuracy comparison of awide variety of approximate adders.

The N-bit approximate adder, given in Fig. 2, is constructed using L sub-adders. The ith sub-adder, for i ¼ 1; 2; . . . ; L, is a Ri þ Si-bit precise adder. The output of Sub-adder 1 is always correct as there is no loss of accuracy due to carry chain truncation. However, the outputs of Sub-adder 2 to Sub-adder L can be erroneous since addition with the carry generated by the previous less significant bits has been eliminated.

Since the output sum is obtained by gathering the outputs of all the sub-adders, error in any sub-adder's output can contribute to error in the final output. Error in the ith sub-adder occurs when the Ri prediction bits of the sub-adder's input are all propagating carry-in and the previous less significant bits of adder's input, that are not input to this sub-adder, are generating carry-out. The error occurs because the generated carry-out is not propagated due to the broken carry chain between sub-adders.

Let G0i and Pi represent the carry-out generation (with carry-in equal to zero) and carryin propagation events that lead to error in the output of this ith sub-adder, respectively. Since for Sub-adder Pi and G0i represent conditions on disjoint groups of bits of the adder's inputs, they can be modeled as independent events. Thus, the probability of intersection of independent events is equal to product of probabilities of individual events. It should be noted here that the input bits are perfectly independent if they are uniformly distributed and the probabilities of events Pi and G0i can be parameterized in terms of number of bits being added. In case of non-uniformly distributed inputs, these probabilities depend on both the number and position of bits and the events G0i and Pi may not be independent.

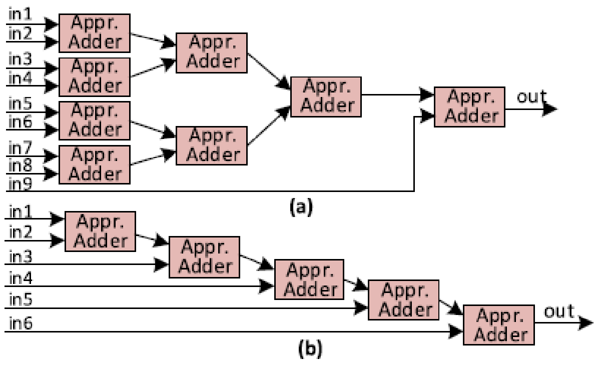


FIG.1: EXISTED SYSTEM

We found the overall probability of error in an output of approximate adder. Now, we find the PMF of error value, aiming at understanding the distribution of this overall probability among all the possible error values. This is done by individually considering all possible error cases. In every case, error value is found by identifying the sub-adders with erroneous outputs and the weights of carry bits that are discarded due to the carry chain truncation. The corresponding probabilities are computed using the same concepts that we developed. As explained, the error in an approximate adder is due to the missed addition of carry bit(s) due to the broken carry chain. Consequently, the approximate sum is deficient by an amount equal to the weight of this bit location.

III. PROPOSED SYSTEM

The binary adder is the critical element in most digital circuit designs including digital signal processors (DSP) and microprocessor units. And such as, extensive research continues to be focused on improving the power-delay performance of the adders. VLSI implementations, parallel prefix adders are known to have the best performance. Reconfigurable logic like Field Programmable Gate Arrays (FPGAs) has been gaining the popularity in recent years because it offers improved performance in terms of speed and power over DSP-based and microprocessor-based solutions for several 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 significant with the growing popularity of mobile and portable electronics, which make extensive utilize 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). Parallel-prefix structures are found to be common in adders which have high performance because of the delay is logarithmically proportional to the width of an adder. Such structures can usually be divided into three stages.

1. Pre-processing stage
2. Carry generating network
3. Post Processing Stage

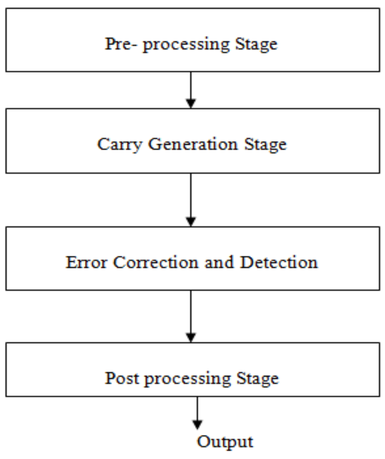


FIG.2: PROPOSED SYSTEM

Fig.2. shows the architecture of Parallel prefix adder. Kogge Stone Adder (KSA) is a parallel Prefix Adder. It is considered as fastest and is widely used in industry for high performance arithmetic circuits. KSA employs the 3-stage structure of the CLA adder. In KSA carries are computed fast by computing the carries in parallel. The carry computation method which is leads to speed up the overall operation significantly. This reduces the area and maximizing the speed. As shown in Fig. 2, the construction of KSA involves three processing stage.

A. Pre-processing stage:

This step involves computation of generate and propagate signals which are corresponding to each pair of bits in A and B. These signals are given by the logic equations below:

$$pi = Ai \ XOR \ Bi \qquad (1)$$
$$gi = Ai \ AND \ Bi \qquad (2)$$

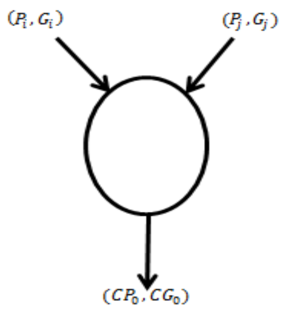


FIG.3: CARRY GENERATION STAGE

B. Carry generating network: Fig.3. Show the block differentiates KSA from other adders and is the main force behind its high performance. This step involves computation of carries which corresponding to each bit. It uses group propagate and generate as intermediate signals which are given by the logic equations below:

$$CP0 = Pi \ AND \ Pj \qquad (3)$$
$$CG0 = Gi \ OR \ Pi \ AND \ Gj \qquad (4)$$

C. Post Processing Stage: This step is final step and is common to all adders of this family. It involves computations of sum bits. Sum bits are computed by the logic given below:

$$Si = Pi \ XOR \ Ci{-}1 \qquad (5)$$

SPARSE KOGGE-STONE ADDER GENERATOR: This generates Verilog code for adders with large numbers of bits. While a complete adder would produce the output of all bits, this just outputs a series of carry bits at fixed intervals. These can be utilized as the carry-in bits for a series of smaller adders. This is useful in particular for FPGAs, where small ripple-carry adders can be much faster than the general-purpose logic thanks to fast connections among neighboring slices. This allows a large adder to be composed of many smaller adders by generating the intermediate carries quickly.

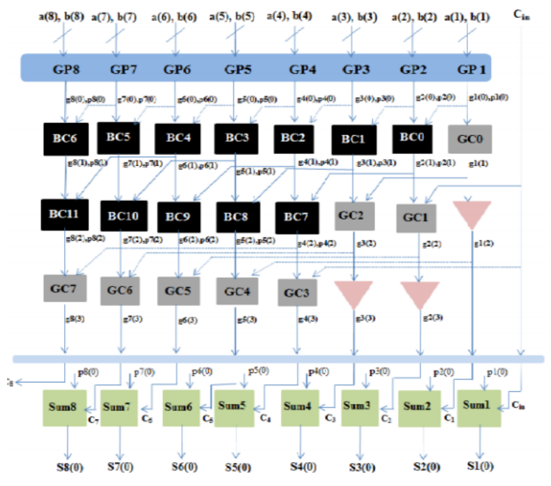


FIG.4: SPARSE KOGGE STONE ADDER

The Kogge-Stone adder (KSA) 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 mximized 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 becomes more favorable in terms of speed due to the O(log2n) delay through the carry path compared to O(n) for the Ripple Carry Adder(RCA). The differences in terms of logic depth and number of logic blocks can be seen by comparing which illustrates a 8 bit Kogge-stone Adder, with which depicts a 8 bit RCA.

IV. RESULTS

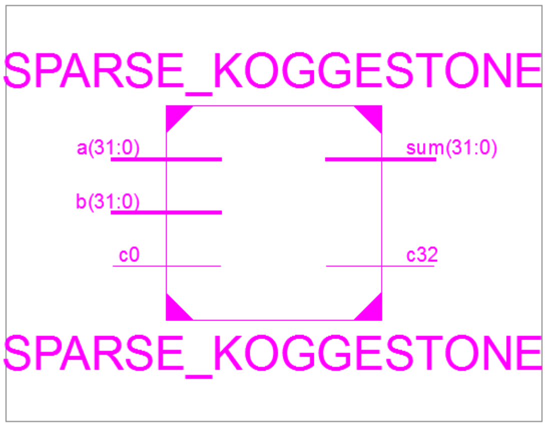


FIG.5: RTL SCHEMATIC

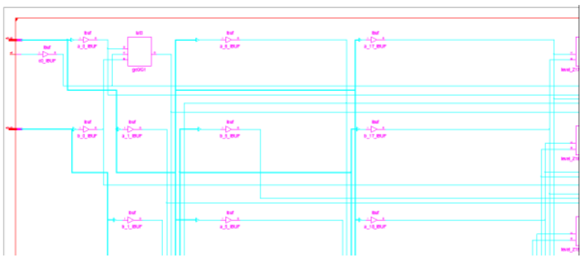


FIG.6: TECHNOLOGY SCHEMATIC

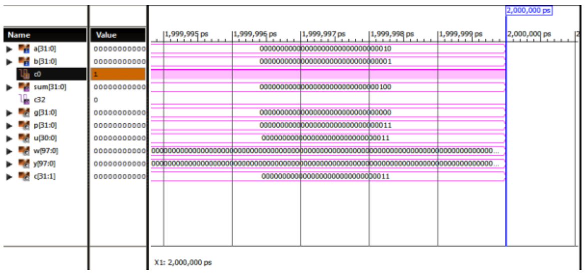
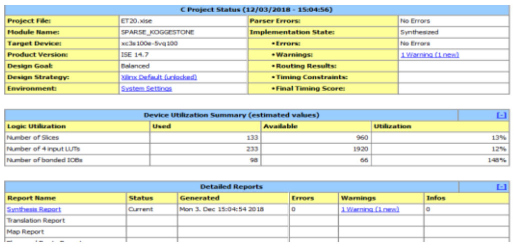


FIG.7:
OUTPUT



C Project Status (12/03/2018 - 15:04:56)			
Project File:	ET20.xise	**Parser Errors:**	No Errors
Module Name:	SPARSE_KOGGESTONE	**Implementation State:**	Synthesized
Target Device:	xc3s100e-5vq100	**• Errors:**	No Errors
Product Version:	ISE 14.7	**• Warnings:**	1 Warning (1 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	133	960	13%
Number of 4 input LUTs	233	1920	12%
Number of bonded IOBs	98	66	148%

Detailed Reports					[-]
Report Name	**Status**	**Generated**	**Errors**	**Warnings**	**Infos**
Synthesis Report	Current	Mon 3. Dec 15:04:54 2018	0	1 Warning (1 new)	0
Translation Report					
Map Report					

FIG.8: REPORT

V. CONCLUSION

The Parallel Prefix Adder is a type of process that increase the speed of arithmetic operations of the system. A Sparse Kogge-Stone adder which is fully fault tolerant in its lower half (i.e., in the ripple carry adders) was proposed. Simulation results demonstrate that this design is able to detect and correct error in its chains. The simulation and synthesis of the Sparse Kogge-Stone adder have been performed on ISim (VHDL/Verilog) simulator. So, the Modified Sparse Kogge-Stone adder can be utilized for various high speed applications. In future scope, the parallel prefix adders must be tested for other adders also to optimize the area and timing both.

VI. REFERENCES

[1] I. Koren, Computer Arithmetic Algorithms. Natick, MA, USA: A K Peters, 2002.

[2] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. thesis, Swiss Federal Institute of Technology, (ETH) Zurich, Zurich, Switzerland, 1998, Hartung-Gorre Verlag.

[3] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, no. 3, pp. 260–264, Mar. 1982.

[4] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," IEEE Trans. Comput., vol. C-22, no. 8, pp. 786–793, Aug. 1973.

[5] J. Sklansky, "Conditional-sum addition logic," IRE Trans. Electron. Comput., vol. EC-9, pp. 226–231, Jun. 1960.

[6] T. Han and D. A. Carlson, "Fast area-efficient VLSI adders," in Proc. IEEE 8th Symp. Comput. Arith. (ARITH), May 18–21, 1987, pp. 49–56.

[7] R. E. Ladner and M. J. Fischer, "Parallel prefix computation," J. ACM, vol. 27, no. 4, pp. 831–838, Oct. 1980.

[8] Chris D. Martinez, L. P. Deepthi Bollepalli, and David H. K. Hoe, "A Fault Tolerant Parallel-Prefix Adder for VLSI and FPGA Design" , 44th IEEE Southeastern Symposium on System Theory, 2012.

[9] R. Iris, D. Hammerstrom, J. Harlow, W. H. Joyner Jr., C. Lau, D. Marculescu, A. Orailoglu, M. Pedram, "Architectures for Silicon Nanoelectronics and Beyond," Computer, vol. 40, no. 1, pp. 25-33, Jan. 2007.

[10] P. M. Stone 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, pp. 786-793, Aug. 1973

[11] S. Ghosh, P. Ndai, and K. Roy, "A Novel Low Overhead Fault Tolerant KoggeStone Adder using Adaptive Clocking," Design, Automation and Test in Europe, pp. 366- 371, 2008.

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

[13] D. H. K. Hoe, C. Martinez, and J. Vundavalli, "Design andCharacterization of Parallel Prefix Adders using FPGAs," IEEE 43rd Southeastern Symposium on System Theory, pp. 170-174, March 2011.

[14] S. Ghosh, P. Ndai and K. Roy, "A Novel Low Overhead Fault Tolerant Kogge-Stone Adder using Adaptive