Fault-tolerant, Real-time Reconfigurable Prefix Adder

University of Virginia Technical Report CS-2009-09
Marisabel Guevara Christopher Gregg
<mag3dn, chg5w>@virginia.edu

Abstract

In this paper, we assimilate and integrate two recent developments in prefix adder design and theory to create a fault-tolerant, real-time reconfigurable prefix adder. By exploiting the inherent redundancy of a Kogge-Stone adder we are able to extract signals to detect and correct from a single-fault. Our 16-bit design consumes less than 44% the hardware overhead of a comparable triple modular redundancy (TMR) 16-bit Kogge-Stone adder, and has a delay overhead of 33% to a standard Kogge-Stone adder. Higher bit-count adders will incur smaller delay cost. When a single fault (either a hard or soft error) is detected, the adder reconfigures itself to calculate the correct output. In the case of a non-recurring soft error, the adder will subsequently select the output of the Kogge-Stone adder, and in the event of a hard error, the adder will continue to reconfigure itself upon each successive addition to ensure proper output.

1 Introduction

Chip fabrication facilities have transitioned to $45nm$ and smaller technologies, resulting in substantial increases in both the number of hard errors, due mainly to variation, material defects, and physical failure during use, and the number of soft errors, due primarily to alpha particles from normal radiation decay, from cosmic rays striking the chip, or simply from random noise [1, 2, 3]. Soft errors are not considered to permanently break the circuit; on the other hand a hard error will permanently prevent a circuit from behaving as it was designed. It is therefore imperative that chip designers build robust fault-tolerance into computational circuits, and that these designs have the ability to detect and correct both hard and soft errors.

Traditionally, hardware based error correction has been used in critical applications such as space-based systems (i.e., satellites and deep-space vehicles). Today, however, the scaling of CMOS manufacturing technologies has raised the rate of soft and hard errors and significantly increased the number of devices per unit area, thus dually affecting the importance of fault tolerant design. Systems that must provide high-reliability are in need of fault tolerant components that present acceptable trade-offs in performance and cost. Further, the top six candidate nanoscale devices that seek to become the next building block for electronic design all have one common challenge to overcome, which is their vulnerability to noise and errors [4]. The most widely used scheme to overcome single faults is Triple-Modular-Redundancy (TMR), which by its nature incurs a three-times size overhead to any circuit [5]. While TMR provides an upper limit on size for single-fault redundancy, the large size overhead makes its use impractical for most applications. The goal, therefore, is to create circuits that can perform reasonably well while taking up less area on-chip than a TMR solution, as shown in Fig. 1.

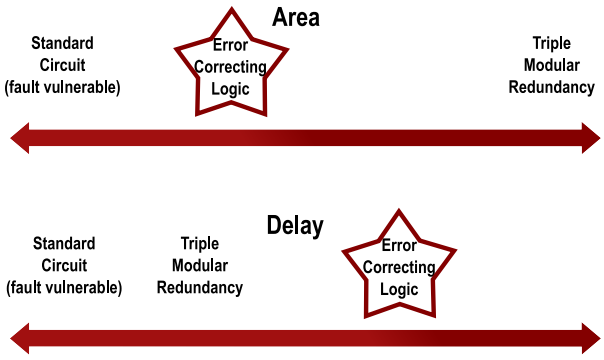

Figure 1: Area and Delay Trade-off

Some computational circuits have built-in redundancy that lies latent during normal use. Prefix-adders in general and Kogge-Stone Adders [6] (KSAs) in particular contain enough inherent redundancy in their design that can be exploited to detect faults and, in certain cases, correct them [7, 4, 8]. Our research amalgamates prefix-adder structures and the fundamentals of parity to design a novel real-time error correction scheme with a reasonable delay overhead. In section 2, we detail the redundancy found in KSAs, the traditional approach for applying parity to adders, and some of the pertinent ideas that lead to our proposed solution and circuit design. In section 3, we present our design and describe its ability to correct single faults. In particular, we explore the priority-encoder circuit that performs the fault-detection logic. In section 4, we present an analysis of the circuit, and we compare it to traditional TMR in terms of speed and overhead. Finally, in section 5 we conclude and offer ideas for future research in this domain.

2 Background

Parallel prefix adders (PPAs) speedup addition through a trade-off between three metrics: area, fan-out, and logic-depth [9]. The building blocks for PPAs are dot operators, $(g, p) \bullet (g', p') = (g + pg', pp')$, shown in Fig. 3, a propagate signal, $p_i = A_i \oplus B_i$, and a generate signal, $g_i = A_i B_i$, at each bit position as in Fig. 2. Kogge-Stone Adders (KSA) have a fan-out of 2, minimum logic-depths at

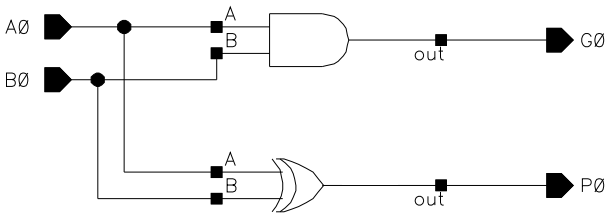

Figure 2: Kogge-Stone Input Stage

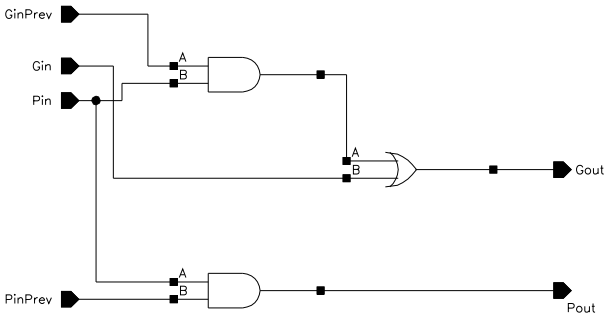

Figure 3: Kogge-Stone P/G Stage

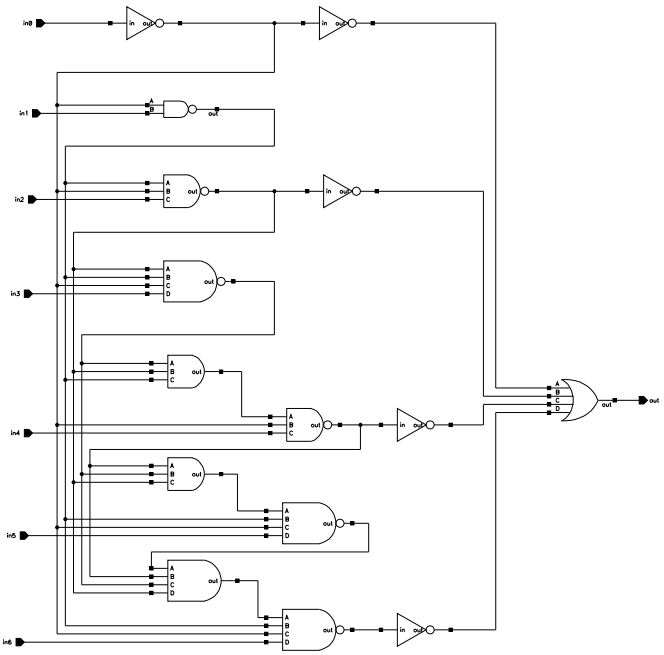

Figure 4: An 8-bit Priority Encoder, utilizing only the Least Significant Bit

$O(\log 2N)$, but a large penalty in area; Han-Carlson Adders (HCA) reduce area by increasing the logic-depth. KSAs are commonly used in Arithmetic Logic Units [10] due to their speed achieved by computing the carry-signals in parallel. The paths of the carry-logic are different for even and odd carry-bits, and this research exploits the mutual exclusion of the carry signals to achieve an adder that can produce a correct sum in the event of a hard or soft error.

Our adder is primarily based on an idea presented by Ndai, et. al whereby an *n*-bit KSA is split into two distinct *n*-bit HCAs [11] by utilizing the redundancy inherent in the KSA architecture[7]. This configuration will, in fact, correct for up to 50% errors on the adder if all the errors happen to occur in either the even- or odd-half of the adder. Their method for error-correction, however, is not real-time. They rely on a one-time, post-testing stage physical reconfiguration that corrects for hard errors. While this is certainly useful, the downside is that it cannot correct soft errors. We utilize their idea to build an additional HCA stage onto the end of a KS adder and then use multiplexers to select the even or odd carry-bits of the HCA stage in the event of an error in the opposite path. This research explores error detection and correction schemes as an extension to the reconfigurable adder design and hence provides the necessary reconfiguration in real-time.

Parity checking counts the number of set bits in a signal to detect an error [12]. The use of parity checking in memory or communication is common, but in the realm of logic circuitry the time and/or area overhead is often deemed too expensive. The following parity-checking comparison guarantees correctness in the event of a single-error in an adder circuitry [13]:

$$P_{sum} \stackrel{?}{=} P_{input\,A} \oplus P_{input\,B} \oplus P_{carry}$$

where $P_{input\,A} \rightarrow$ parity of input A, $P_{input\,B} \rightarrow$ parity of input B, $P_{input\,C} \rightarrow$ parity of input C, and $P_{sum} \rightarrow$ parity of the sum bits

The above equality will be broken by a single-fault, hence it may be used for single soft error detection. The problem with this technique is that an error in the carry circuitry will not be detected due to fault aliasing, where k errors in the carry bits and k errors in the sum bits make a total of $2k$ errors (always even and hence undetected)[13]. For this technique to guarantee single-fault detection, the adder must produce a set of correct carry bits.

To guarantee a correct set of carry bits, independent hardware to calculate a second set of carry signals is necessary. [14] includes a theoretical evaluation of the complexity of such an implementation. A standard KSA has an approximate complexity of $5n+2n\log_2 n$ ($3n$ from the initial stage, $2n$ from the final sum, and $2n\log_2 n$ for the dot operators, where n is the number of bits). The redundant carry can be generated either in between dot operators, with an additional two XOR gates, or inside each dot operator, with the addition of a single XOR gate. Two additional XOR gates are necessary for the bit-by-bit parity comparison and the input bits. The resulting complexity of the redundant carry is $2n + 6n + 2n\log_2 n$, resulting in more than 100% additional complexity than the original KS Adder. This analysis motivates an alternative solution for fault detection as this approach is clearly expensive; by comparison, our design has only $6n + \log_2 n\ + 1$ complexity.

3 A Fault Tolerant Adder Circuit

The complete design of a 16-bit reconfigurable KSA is shown in Fig. 5. There are three separate components to the design: a Kogge-Stone component, a Han-Carlson component, and the priority encoder and NOR component which feed the *Normal* and *Even* signals. The characteristic of a KSA that this research exploits is the fact that dot op-

erators on the same bit-path do not receive any data signals from adjacent bits. This forms a redundant path for calculating carries, as shown in [7].

Our design exploits the property of KSAs that adjacent bits are calculated using independent hardware, although in fact the first row of dot operators receives its input signals from adjacent propagate/generate stages. Thus, the proposed design can detect and recover from a single fault that occurs on any of the dot operators, but assumes that the outputs of the propagate/generate input stage are correct. Techniques to overcome this assumption would rely on additional hardware, and this paper does not explore this possibility.

The fault detection we implemented builds off the redundancy discussed above. The outputs of the original KSA provide a complete set of carry bits, and the outputs of the additional HC Stage provide a set of carry bits that are generated from hardware not shared by immediate neighbors. A single fault will thus corrupt only one set of carry bits, and the other is guaranteed to be correct. In the absence of a single fault, XORing the two sets of carry bits one by one will yield a vector of logic zeroes. In the case of a fault, the XOR will generate one or more logic ones, the one located in the least significant bit position signaling which bit position was the victim of the error. Hence, both the Normal and Even/Odd selection signals can be generated from the above analysis.

Because a single error can propagate through higher-order bits in a prefix-adder [15], we needed a circuit to determine whether the least significant bit that resulted in a logic one is at an odd or even bit position. In other words, we needed to know if the first, third, fifth, etc., carry bit was the first to show the error, or whether it was the second, fourth, sixth, etc bit. One circuit that accomplishes this is a priority encoder, although only the last output bit of the encoder is necessary to determine if the first set bit is even or odd thus reducing the overhead that would be necessary for a complete priority encoder. See Fig. 4 for a circuit diagram of a modified eight-bit priority encoder circuit (*note: only seven bits are necessary for the circuit*).

The HC component of the adder produces carries that are XORed with those produced by the KS component, but the HC carries are independent of the KS carries adjacent to them. The subsequent XOR for each carry pair is utilized by the *Normal* OR gate and the priority encoder to set the *Normal* and *Even* signals correctly. If there are no errors, the NOR gate sets the *Normal* signal to a logic one, so that the outputs of the multiplexers come from the KS component of the adder. If, on the other hand, there is an error, *Normal* is set to a logic zero, and the priority encoder sets the *Even* signal to produce output from the non-faulty Han-Carlson adder.

4 Results

Using the *Spectre* SPICE simulator and $6\mu m$ technology, we compared our reconfigurable design to that of a standard KSA in both 4-bit and 16-bit implementations. In both cases, we measured the delay overhead of the additional dot operator and logic to generate the normal and even/odd signals on the same additions. Fig. 6 shows

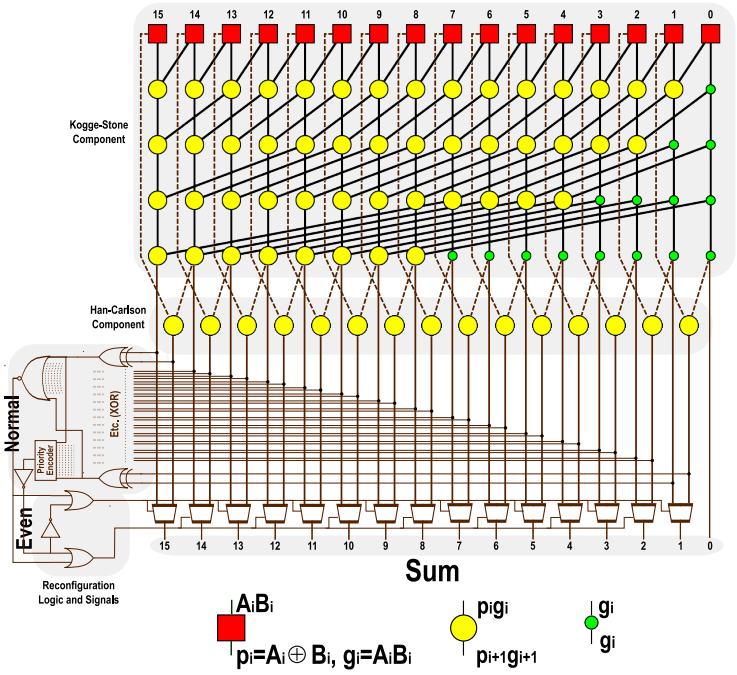

Figure 5: 16-bit fault-tolerant, real-time reconfigurable adder
The inputs $A15$ $B15$, ..., $A0$ $B0$ feed into the propagate/generate input stage at the top, where each square is implemented as shown in Fig. 2. The large circles represent dot operators, and the circuit for this logic is shown in Fig. 3. The small circles represent buffers to bring the propagate/generate signals to the next stage.

the delay for a 4-bit addition during one transition from high to low and from low to high during the add. Fig. 7 shows the delay for a 16-bit addition during a transition from high to low and from low to high during the add.

The overhead from the additional dot operator and logic in terms of time is 0.8 ns, as shown in Fig. 6 and Fig. 7. This can be deemed a maximum since for KSAs with higher bit counts this overhead is a smaller percentage of the total addition time. For our 4-bit implementation, the overhead represented an additional 39% in time. In the 16-bit implementation, the percentage of this same overhead is reduced to 33%. The delay of an addition in a standard KSA can be approximated by $t_p = t_{PG} + t_\bullet \times \log N + t_{XOR}$ where $t_{PG}$is the time for a single propagate/generate cell to compute, $t_\bullet$is the propagation delay of a dot operator stage, and t_{XOR} is the delay of one XOR gate. The additional HC stage for the reconfigurable implementation adds $t_\bullet$delay, the normal signal generation adds a propagation delay of $t_{XOR} + t_{OR}$, and the priority encoder to produce the even/odd signal adds a delay of t_{PE}, for a new propagation delay from inputs to sum of $t_P = t_{PG} + t_\bullet \times (\log N + 1) + 2t_{XOR} + t_{OR} + t_{PE}$. This overhead is dependent on N only in the t_{PE} term, the propagation delay of an N-bit priority encoder, which according to [16] is bound by $O(log_2 N)$.

Extending this analysis of hardware overhead, a KSA has 49 dot operator nodes, as seen in Fig. 5. The proposed reconfigurable adder requires an additional 15 dot operator nodes, represented by the additional HC Stage in Fig. 5, whereas a TMR single-fault tolerant

Fault-tolerant 16-bit adder	extra num nodes/standard num nodes
Reconfigurable KSA	15/49
TMR KSA	98/49

Table 1: Hardware Overhead Analysis

KSA requires an additional two fully independent KSAs, or 98 dot operators as summarized in Table 1. This high-level view of the hardware overhead provides a rough insight into the cost in area of the proposed reconfigurable KSA versus a TMR implementation of a fault tolerant KSA. Our proposed design requires less than 44% more hardware than a standard KSA, as compared to 66% for a TMR implementation.

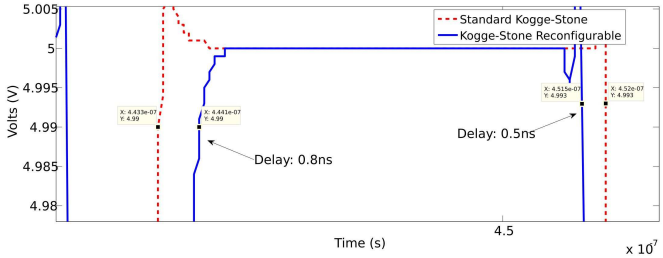

Figure 6: Delay of 4-bit fault-tolerant adder compared to a Kogge-Stone adder (sum bit 5)

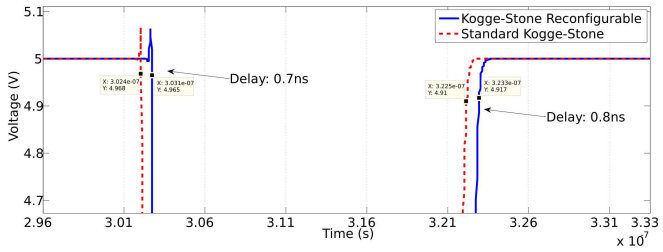

Figure 7: Delay of 16-bit fault-tolerant adder compared to a Kogge-Stone adder

5 Conclusion and Future Work

We have designed a real-time reconfigurable prefix adder with minimal delay overhead, and reduced area cost compared to a Triple-Modular-Redundant circuit. The circuit will correct one hard or soft error per calculation, and it will correct up to 50% errors if they all occur on either the even- or odd-half of the adder. This research has contributed to the advancement of fault tolerant adders, providing computer architects and system designers another safeguard against errors at the circuit level. Further, applications where reliability is of highest importance also can benefit from an alternative single-fault tolerant adder with different delay and area overhead costs than existing techniques. Finally, the field of nanoscale electronics also benefits from a design that exploits the inherent redundancy of a commonly used logic structure to guarantee fault tolerance, especially at a scale where additional devices are less expensive in terms of area and monetary cost than in CMOS technology.

Because scaling of electronic components has resulted in an increased number of hard and soft errors, more research is needed to create hardware based error correction schemes for different types of logic. Adders are circuits with much regularity, yet work still remains to be done to extend fault tolerance to irregular logic blocks. There has already been significant work on memory circuits (see [17], for instance), but research on arithmetic logic has lagged behind, or focused on critical systems that can make the sacrifices (monetary, speed, size, etc.) necessary to ensure fault tolerant operation [18]. Virtually all next-generation systems will exhibit high soft and hard errors and hence more research needs to focus on making all logic circuits fault-tolerant.

References

[1] S. Mukherjee, J. Emer, and S. Reinhardt, "The soft error problem: an architectural perspective," *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 243–247, Feb. 2005.

[2] A. Johnston, G. Swift, and D. Shaw, "Impact of cmos scaling on single-event hard errors in space systems," *Low Power Electronics, 1995., IEEE Symposium on*, pp. 88–89, Oct 1995.

[3] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, pp. 43–52, Feb. 2005.

[4] W. Rao and A. Orailoglu, "Towards fault tolerant parallel prefix adders in nanoelectronic systems," *Design, Automation and Test in Europe, 2008. DATE '08*, pp. 360–365, March 2008.

[5] R. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM Journal of Research and Development*, vol. 6 (2), pp. 200–209, April 1962.

[6] P. Kogge and H. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Transactions on Computers*, vol. C-22, pp. 783–791, August 1973.

[7] P. Ndai, S.-L. Lu, D. Somesekhar, and K. Roy, "Fine-grained redundancy in adders," *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, pp. 317–321, March 2007.

[8] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, "A 6.5ghz 54mw 64-bit parity-checking adder for 65nm fault-tolerant microprocessor execution cores," *VLSI Circuits, 2007 IEEE Symposium on*, pp. 46–47, June 2007.

[9] M. Ziegler and M. Stan, "A unified design space for regular parallel prefix adders," *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2, pp. 1386–1387 Vol.2, Feb. 2004.

[10] S. Ghosh, P. Ndai, and K. Roy, "A novel low overhead fault tolerant kogge-stone adder using adaptive clocking," *Design, Automation and Test in Europe, 2008. DATE '08*, pp. 366–371, March 2008.

[11] T. Han and D. Carlson, "Fast area-efficient vlsi adders," *Proc. 8th Symp. Comput. Arithmetic*, pp. 49–56, 1997.

[12] P. K. Lala, ed., *Self-checking and fault-tolerant digital design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[13] F. Sellers, L. Hsiao, and L. Bearnson, *Error Detecting Logic for Digital Computers*. McGraw-Hill, 1968.

[14] J. Biernat, "The complexity of fault-tolerant adder structures," *Dependability of Computer Systems, 2008. DepCos-RELCOMEX '08. Third International Conference on*, pp. 316–323, June 2008.

[15] "Priority encoder (8:3 bits)." Avaliable: http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/10-gates/45-priority/priority83.html, Nov 2008.

[16] C. Kun, S. Quan, and A. Mason, "A power-optimized 64-bit priority encoder utilizing parallel priority look-ahead," *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 2, pp. II–753–6 Vol.2, May 2004.

[17] M. Myjak, D. Blum, and J. Delgado-Frias, "Enhanced fault-tolerant cmos memory elements," *Circuits and Systems, 2004. MWSCAS '04. The 2004 47th Midwest Symposium on*, vol. 1, pp. I–453–6 vol.1, July 2004.

[18] V. Nelson, "Fault-tolerant computing: fundamental concepts," *Computer*, vol. 23, pp. 19–25, Jul 1990.