# Towards fault tolerant parallel prefix adders in nanoelectronic systems

Wenjing Rao
UC San Diego
CSE Department
wrao@cs.ucsd.edu

Alex Orailoglu
UC San Diego
CSE Department
alex@cs.ucsd.edu

## ABSTRACT

Future nanoelectronics based arithmetic components will enjoy abundant hardware, yet at the same time confront severe unreliability challenges. We focus on the fault tolerance of high performance parallel prefix adders (PPA), and exploit the inherent redundancy in PPAs to develop efficient fault tolerance approaches. We show that the internal invariant inherent in the parallel prefix adders provides support for online fault detection and fault masking. Furthermore, based on the particular regular structure of PPAs, an online diagnosis scheme can be developed, thus enabling the application of reconfigurability of nanoelectronics for the highly flexible online repair approaches. In contrast to traditional fault tolerance techniques that rely solely on significant external overhead, the proposed approach opens up a new genre of efficient fault tolerance techniques for arithmetic components in the nanoelectronic environment.

## 1. INTRODUCTION

As CMOS is reaching its physical limits due to quantum physical effects and fabrication limitations [1], a number of nanoelectronic devices have been proposed as promising candidates, including SET [2], RTD [3], CNT [4], QCA [5] and molecular electronics [6]. Even though each device candidate operates on its specific physical basis, a number of characteristics are shared among the nanoelectronic devices due to their commonality of nanometer level scales.

These fundamental changes at the device level provide significant benefits due to the density boost, but at the same time introduce new challenges as well [1, 7]. The most severe challenge confronting in common all the nanoelectronic device candidates is unreliability, in particular, the extremely small scale of the nanoelectronic devices leading to low immunity to noise and errors. Since ultra low power is used with the device operation based on quantum effects, the influence of stray charge, crosstalk, temperature fluctuation and cosmic particle caused single event upset results in massive online fault occurrences in nano scale transistors [1, 7, 8]. In addition, multiple faulty behaviors, including transient, semi-permanent and permanent faults are all projected to occur in the nanoelecronic environment. As a result, online fault tolerance is of significant importance in ensuring the fundamental requirement of computational correctness for nanoelectronics based systems.

Adders constitute the most basic and fundamental building blocks for arithmetic components. In the nanoelectronic environment, high performance parallel adders exhibit significant advantages over serial adders, because the hardware abundance and massive parallelism supported by nano devices diminishes the relative importance of the area constraint. Fault tolerance techniques for the most basic form of parallel adders, Carry Lookahead Adders, have been developed in [9] for nanoelectronic systems. In this paper, we focus on the general genre of parallel adders, namely the parallel prefix adders (PPA). The delivery of high performance and the regularity in the prefix network both make PPAs highly promising for the future nanoelectronic systems. Specifically, the design of a Kogge-Stone parallel prefix adder on defective nanoelectronic fabrics has been illustrated in [10].

Since PPAs utilize a parallel prefix network to calculate carry bits in advance for performance reasons, the extra hardware used in PPAs opens up opportunities for developing efficient fault tolerance schemes. To answer the questions of whether or how the inherent hardware redundancy can be exploited for fault tolerance purposes, one needs to examine both the redundancy in parallel carry generation, and the redundancy necessitated in various fault tolerance approaches. In this paper, we exploit the inherent relationship among the output bits of a PPA's parallel prefix network for fault tolerance purposes. Furthermore, we show that such inherent redundancy can be used to support multiple fault tolerance approaches, thus facilitating efficient implementation of reliable PPAs in the nanoelectronic environment.

The paper is organized as follows. In section 2, we provide the preliminaries for PPAs and the general genres of fault tolerance schemes. The proposed schemes of exploiting extant redundancy to achieve fault tolerant PPAs are presented in section 3. In section 4, we discuss the capability of the proposed fault tolerance approaches, and section 5 concludes the paper.

## 2. PRELIMINARIES

In this section, we first present a brief introduction to PPAs, so as to set up the basis for the proposed approach. We then briefly summarize the various fault tolerance approaches applicable in the nanoelectronic environment. In the later sections we will show that these fault tolerance schemes can all be supported by the extant redundancy in PPAs.

### 2.1 Introduction to PPA

In order to avoid the delay of the rippling carries, a PPA computes the signals of "generate" ($g$) and "propagate" ($p$) and calculates carry bits in parallel [11]. Based on $(G[0, i-1], P[0, i-1])$, the block *g, p* signals, each bit of carry signal $c[i]$ can be calculated directly:

$$c[i] = G[0, i-1] + P[0, i-1] \cdot c[0]$$

. For an $n$-bit addition, a PPA calculates first all the $n$ pairs of block $(G[0, i], P[0, i])$ signals, where $i = [0, n-1]$, and then evaluates the carry signals and sum signals correspondingly.

The generation of the *g, p* block signals falls into the prefix computation framework: for two adjacent or overlapping blocks $B_L$ and $B_R$, let their associated block $g, p$ signal pairs be $(g_L, p_L)$ and $(g_R, p_R)$, respectively; then the $g, p$ signals for the merged block $B = B_L B_R$ can be obtained by $g = g_L + g_R \cdot p_L$ and $p = p_L \cdot p_R$. Intuitively, the "generate" signal ($g$) of carry in the large block takes place if 1) the left block generates a carry, or 2) the right block generates a carry and

978-3-9810801-3-1/DATE08 © 2008 EDAA

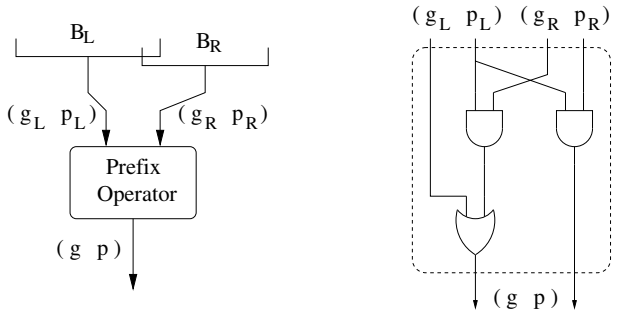



**Figure 1: Functionality and internal structure of a prefix operator**

the left block propagates it. The large block "propagates" a carry if both the left block and the right block propagate it. A *prefix operator* can be defined for the generation of *g, p* pairs. Figure 1 illustrates the composition of *g, p* signals from two blocks $B_L$ and $B_R$ by a prefix operator with its gate-level implementation.

In a PPA, a network is constructed to calculate all the $(G[0,i], P[0,i])$ pairs. Figure 2 illustrates the parallel prefix networks of three example PPAs, where each node in the network represents a prefix operator. The three examples illustrate multiple tradeoff points of hardware, performance, and interconnect overhead in PPA designs. For each of the PPAs shown in figure 2, the inputs are the 16 individual bit-level *g, p* signals. Through the well-constructed prefix network, *g, p* signals of larger blocks are constructed by applying prefix operators over consecutive smaller blocks. In the end, all the block signals $(G[0,i], P[0,i])$ are generated at the 16 output positions, facilitating the fast evaluation of carry and sum signals.

## 2.2 Fault tolerance in nanoelectronic systems

In the nanoelectronic environment, the massive occurrence of online faults makes aggressive fault tolerance approaches a fundamental requirement for the implementation of any functional system [1]. Essentially, any fault tolerance approach relies on a certain amount of redundancy. Consequently, exploiting the extant redundancy within the system reduces the extra hardware expense needed for fault tolerance purposes, thus opening up opportunities for the development of highly efficient fault tolerance schemes for nanoelectronic systems. In this section, we provide a brief introduction to a set of fault tolerance schemes that are generally applicable to nanoelectronic systems. We show in subsequent sections that the multiple fault tolerance approaches, including fault detection, masking and diagnosis, can all benefit from inherent redundancy exploitation.

Traditional CMOS based systems are highly reliable. Therefore, the main concern on online faults in adders has been addressed by fault detection only [12, 13, 14, 15, 16, 17]. However, fault detection by itself cannot guarantee the basic reliability requirement in a nanoelectronic environment; fault tolerance approaches, such as fault masking and online repair, need to be applied to deal with the high occurrence of online faults. These approaches typically come with the high cost of tremendous hardware redundancy. It is shown that general fault masking schemes such as NMR and N-MUX [18] result in immense hardware cost when used to handle the fault rates in a nanoelectronic environment.

Nanoelectronic systems naturally support online repair schemes by their inherent regular structure and reconfigurability [19, 10, 20]. Consequently, online repair based schemes can provide the advantages of flexibility in dealing with the high occurrence of faults. In contrast to fault masking schemes, online repair schemes require less hardware overhead [18], especially for regular systems where backup units can be shared. Related research in online repair for adders includes [9, 21, 22].

However, the cost of online repair based schemes, i.e., the spare hardware needed to replace the faulty units, hinges heavily on the efficiency of an online fault diagnosis procedure. If a fault can be precisely localized within a small component, then the repair procedure can replace the faulty component with minimum hardware cost. A coarse-grain fault diagnosis, on the other hand, results in the replacement of a big chunk of hardware.

To summarize, the unique reliability challenge encountered by nanoelectronic systems results from the combination of the following two perspectives. On the one hand, the high fault rates demand powerful fault tolerance approaches that are not necessitated in traditional CMOS based systems. On the other hand, the price of delivering such fault tolerance capability is excessively high, and unaffordable even with the hardware abundance of the nanoelectronic environment. Specifically, although various fault tolerance approaches exist, including fault detection, fault masking, and reconfiguration based online repair, all these approaches rely on redundancy and demand tremendous amount of extra hardware.

# 3. FAULT TOLERANCE FOR PPA

## 3.1 Redundancy in parallel prefix network

In a PPA, carry bits are computed in parallel by generating the block *g, p* signals for every bit position. As is shown in figure 2, for every output bit at position $i$, the signal pair $(G[0,i], P[0,i])$ is calculated through a tree of prefix operator nodes covering all the inputs from bit 0 to bit $i$. Overall, for a 16 bit adder, the prefix network consists of 16 trees, with possible sharing of intermediate results, in generating the outputs in parallel.

In a PPA, for performance purposes, all the $(G[0,i], P[0,i])$ signals are calculated in parallel, with hardware redundancy used in a prefix network. However, based on the inherent correlations among the neighboring $g, p$ signals, a copy of the same output can be generated through an alternative path. For instance, between any two adjacent output positions $i$ and $i-1$, we have:

$$G[0,i] = g[i] + G[0,i-1] \cdot P[0,i-1]$$

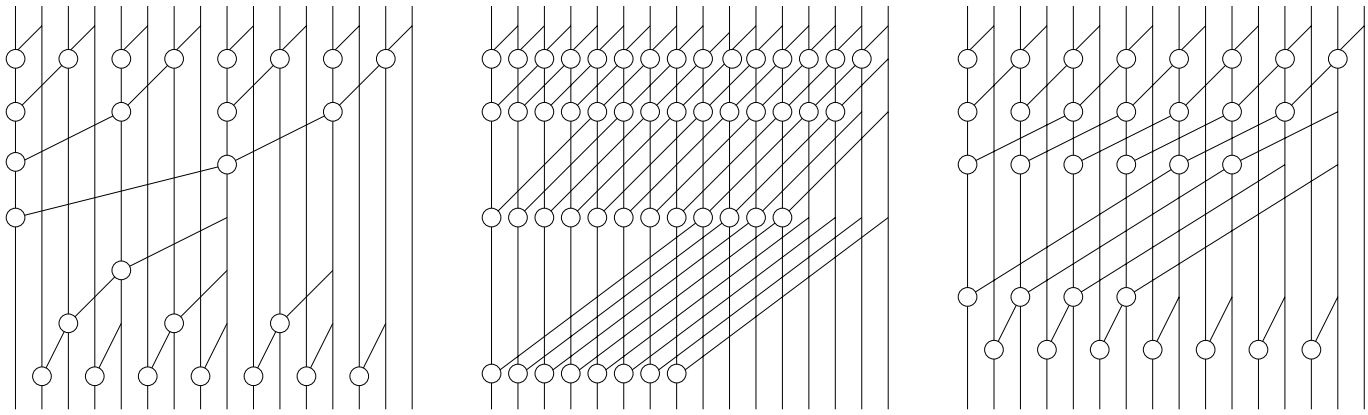
$$P[0,i] = p[i] \cdot P[0,i-1]$$

where $g[i]$ and $p[i]$ are simply bit level $g, p$ signals. In general, we can define the prefix operator as $\circ$ on a signal pair $(g,p)$, so that $(g,p) = (g_L, p_L) \circ (g_R, p_R)$. Therefore, the correlation between adjacent output positions can be represented as:

$$\begin{aligned} & (G[0,i], P[0,i]) \\ = \; & (g[i], p[i]) \circ (G[0,i-1], P[0,i-1]) \\ = \; & (g[i], p[i]) \circ (g[i-1], p[i-1]) \circ (G[0,i-2], P[0,i-2]) \\ = \; & (g[i], p[i]) \circ ... \circ (g[j], p[j]) \circ (G[0,j-1], P[0,j-1]) \\ & (0 < j < i) \end{aligned}$$

Basically, an alternative path of generating $(G[0,i], P[0,i])$ can be formed in the prefix network by using the signals from an adjacent output position $(i-1)$. Generating a redundant $(g,p)$ pair through the alternative path costs thus only one extra prefix operator for each output position $i$. The efficient construction of such redundant copies provides the opportunity to implement cost-effective fault tolerance approaches.

In summary:



Brent−Kung adder | Kogge−Stone adder | A hybrid Brent−Kung / Kogge−Stone adder

**Figure 2: Parallel prefix network examples**

- Through the exploitation of the prefix correlation, for every output position, a redundant copy of the same signal can be generated from the adjacent output position by adding an additional prefix operator.

- Multiple redundant copies of a signal can be generated by extending the prefix correlation to output positions further apart. However, when the redundant copy is built based on a more remote output position, the cost in terms of delay, hardware and interconnect increases.

## 3.2 Online fault detection

Since a redundant copy of each output signal can be generated with an insignificant amount of hardware and a small delay, checking for the conformity of the two copies of the same signal delivers online fault detection capability. Specifically, the redundant copy of output signal at position $i$ can be implemented from the output signal at position $i-1$, as the redundant copy generated through the immediate adjacent one costs the least in terms of hardware and delay, and has the best interconnection locality. These two copies of the same signal are directly compared using an XOR gate.

We can see from figure 2 that, for the Kogge-Stone PPA, no adjacent output positions share a common prefix operator. Such a property ensures a fine-grained fault detection capability, indicating at a bit level whether each output signal is faulty or not. In general, for PPAs without such a property, fault detection is achieved at the adder component level. Nonetheless, the *fault-secure* property of the PPA is maintained [23, 24]. In other words, any single fault occurrence can be guaranteed detected. For a fault to evade the redundant copy comparison based detection, the faulty computational unit has to either fan out to *all* the output positions, or to *none* of the output positions. Apparently, neither condition is satisfied for any node in the PPA network. Therefore, a faulty output is always distinguishable from the fault-free one.

The fault detection capability is essentially associated with the way each basic computational unit (in this case, a prefix operator) fans out to multiple output positions. Whether the approach can provide fine-grained fault detection at the bit level depends on the overlapping of computational units between the redundant copies to be compared. If a common prefix operator fans out to the two adjacent output positions $i$ and $i-1$, when the original copy at position $i$ is compared with the redundant copy generated at position $i-1$, both might be infected by the same faulty prefix operator, thus possibly conforming to a faulty bit.

## 3.3 Fault masking

A single fault masking capability achievable by TMR requires at least three copies of each signal. At a first sight, it seems fault masking can be supported easily by extending the fault detection approach to generate multiple copies of each output signal through inherent redundancy. However, the effectiveness of such fault masking schemes hinges on the disjointness of the hardware to generate the three copies, as a fault in one shared unit might cause the duplicated faulty result to outweigh in the majority vote. Consequently, one needs to examine carefully component overlaps in a PPA network, in order to make efficient use of the extant redundancy for fault masking purposes.

The Kogge-Stone PPA poses a unique characteristic in its structure. Notably in the Kogge-Stone PPA, not a single common node is ever shared between adjacent output positions. Although the set of odd numbered output positions has a large set of overlapped prefix operator nodes, as do the even ones, there is no overlap whatsoever between these two sets of components. In other words, the hardware is divided into two disjoint sets with a "clear cut". The internal redundancy structured into such two disjoint sets naturally supports the generation of two copies of the same signal with no overlapping in hardware. Consequently, the inherent correlation between adjacent output positions can be exploited directly for fault masking approaches.

Traditional TMR approaches utilize triple the amount of original hardware to form three copies of a computation with complete disjointness. For a Kogge-Stone PPA, since the second copy of an output signal can be generated with the internal redundancy disjointly, only the third copy of the computation in a TMR approach needs to be added. In fact, to obtain a disjoint third copy of the output signals for TMR purposes, only half the amount of the original hardware in the Kogge-Stone PPA needs to be added. By duplicating either the hardware involved in the set of odd number output positions, or the even ones, the third copy of each output signal can be constructed. Consequently, a TMR based fault masking can be achieved with 1.5 times the original hardware for a Kogge-Stone PPA, instead of the triplication in a directly implemented TMR approach.

In the Brent-Kung or hybrid PPAs, the overall overlapping of hardware across output positions is so pervasive that the contribution of extant redundancy is hardly of any help in forming redundant copies with fully disjoint hardware. Essentially, a Kogge-Stone PPA utilizes more internal hardware redundancy to achieve higher performance, in comparison to the other implementations of a PPA. In addition, the redundancy in a Kogge-Stone PPA is organized in a highly regular manner. This regularity, together with the abundance of internal hardware redundancy makes it feasible to exploit the existing redundancy

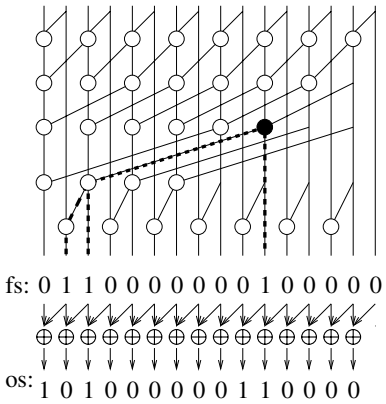



**Figure 3: An example of fanout signature (fs) and observable signature (os) for the prefix operator node in black**

for fault masking purposes without sacrificing the high performance.

## 3.4 Fine grained diagnosis for online repair

Online repair based fault tolerance approaches take advantage of the reconfigurability in nanoelectronic systems, thus providing high flexibility in dealing with high and variable fault rates. Reconfiguration based online repair is invoked after a fault has been detected and then diagnosed to be within a certain location. The amount of hardware necessitated in the repair phase hinges on the precision of the online diagnosis; therefore, high resolution online diagnosis is crucial to the efficiency of online repair based fault tolerance approaches.

The main challenge of online diagnosis lies in the lack of controllability in an online environment. Furthermore, in a traditional offline diagnosis, all the fault signatures can be stored in a dictionary, which can be referred to when diagnosis is performed. In an online environment, the approach of using a dictionary to store the signature of every fault is prohibitively expensive, as one would have to store all the outputs under every fault, for every possible input combination.

To achieve online diagnostic capability, one has to rely on the inherent regular structure of the components, so as to achieve full diagnostic resolution. In other words, each faulty component needs to have a unique syndrome to be distinguished from another. Otherwise, an *aliasing fault* would result in significant hardware overhead in the repair stage, since all the components in the ambiguity set would need to be replaced by spare ones.

As in the fault detection approach, by generating a redundant copy of each output signal through its adjacent signal, the conformity of the two copies essentially serves as a syndrome of the faulty component. Basically, under a fault-free scenario, every pair should agree, and the XOR results of each output position between the two redundant copies should form a zero vector. We will refer to the vector of XOR results as *observable signature* henceforth. When a fault occurs, the observable signature is a non-zero vector. Basically, if every fault results in a unique *observable signature*, then fine-grained diagnosis can be achieved with full resolution.

Essentially, the observable positions of a faulty unit are only the output positions that the unit can fan out to. We define the *fanout signature* of a unit by its associated combination of output positions it fans out to. The relationship between a *fanout signature (fs)*, and the correspondent *observable signature (os)*, can be expressed as $os[i] = fs[i] \oplus fs[i-1]$. This shows that they have the same distinction capability, since at output position 0 there is no unit involved and invariably $fs[0] = 0$. Figure 3 shows the fanout signature $fs$, as well as the observable signature $os$ for the faulty black unit in a hybrid PPA.

Since *fanout signature* and *observable signature* are essentially identical in representing the diagnostic resolution, we can examine the diagnostic capability by focusing on the *fanout signature* of each faulty unit, since it is more relevant to the prefix network structure. Apparently, two units with the same *fanout signature* cannot be distinguished apart. In such a case, the units with the same signature fall into an *ambiguity group*. Figure 4 illustrates all the *ambiguity groups* of the three PPA designs. For each PPA, the prefix operator nodes that are marked by the same number belong to one ambiguity group, and the nodes that are not marked by any number can be diagnosed with full resolution. For instance, the Brent-Kung adder has four ambiguity groups, with group 1 consisting of 4 elements, groups 2 and 4 consisting of 2 elements each, and group 3 consisting of 3 elements. Since the associated diagnosis cannot distinguish between the members in an ambiguity group, if the observable signature indicates a fault occurring in an ambiguity group, the subsequent reconfiguration procedure in effecting a repair needs to replace all the members.

| 16-bit adder | Capability | extra hw / extant hw |
|---|---|---|
| Kogge-Stone PPA | bit-level detection | 15 / 49 |
| Hybrid PPA | adder-level detection | 15 / 32 |
| Brent-Kung PPA | adder-level detection | 15 / 26 |

**Table 1: Fault detection analysis**

# 4. FAULT TOLERANCE ANALYSIS

In this section, based on the proposed technique of exploiting the existing redundancy in parallel adders, we provide an analysis of the fault tolerance capability from the following perspectives:

- Fault detection and masking capability and overhead.
- Online fault diagnostic resolution and the relevant repair costs.
- Incomplete fault manifestation.

## 4.1 Fault detection and fault masking discussion

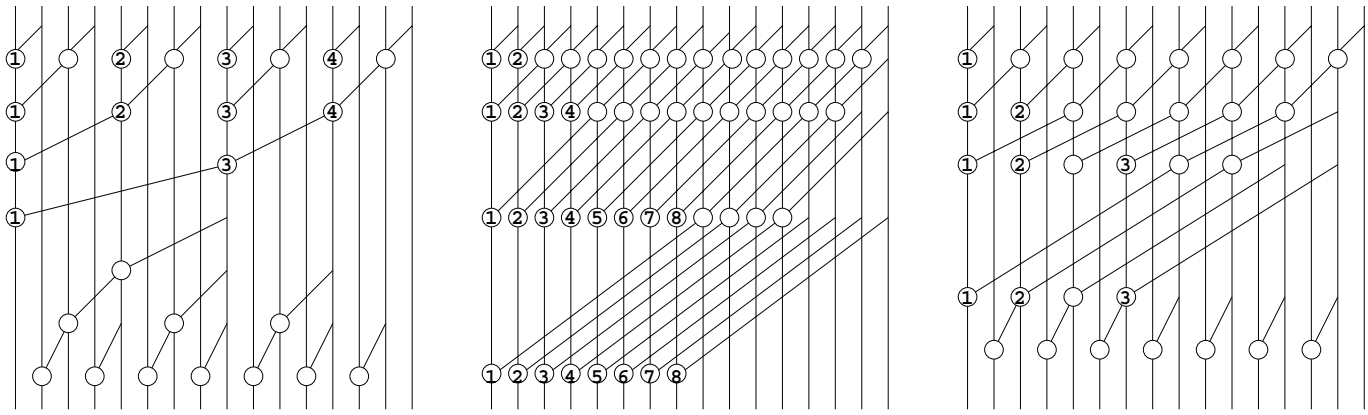
Table 1 summarizes the fault detection capability and hardware overhead for the parallel adders under discussion. The third column compares the extra hardware needed to generate the redundant copies of signals as a fraction of the existing amount of hardware in the adder. For each of the PPA cases, a total of 15 prefix operator nodes need to be added at the 15 output positions to generate the redundant signal copies, while the existing hardware consists of the prefix operator nodes within the network.

Comparing to a conventional duplication based online fault detection scheme, where the extra hardware equals the existing amount of hardware, the proposed approach utilizes significantly lower hardware overhead. Performance-wise, the proposed approach costs an insignificant two-gate delay to generate the redundant signal copy.

For the Kogge-Stone PPA, the amount of extra hardware consists of three parts: 15 prefix operator nodes at the output end, a redundant copy of the network for the odd output positions with 24 internal nodes, and 8 extra nodes at the output of the extra network. Overall, fault masking can be achieved by twice the amount of the original hardware (47 extra nodes). This is significantly less than double the amount of extra hardware (98 nodes) required in a TMR approach. For the proposed approach, the performance overhead for fault masking is a constant 4 gate delay in the CLA, and 2 gate delay in the Kogge-Stone PPA.

| Adder | Brent-Kung | Kogge-Stone | hybrid |
|---|---|---|---|
| Repair cost for one faulty node | 1.85 | 1.90 | 1.63 |

**Table 2: Expected repair cost considering the ambiguity groups**



Brent−Kung adder | Kogge−Stone adder | A hybrid Brent−Kung / Kogge−Stone adder

**Figure 4: Ambiguity groups of PPAs**

## 4.2 Diagnostic resolution analysis

As is shown in figure 4, for a certain number of nodes in each of the parallel prefix networks, aliasing exists for the proposed online diagnosis approach. The overhead introduced by the loss of diagnostic resolution lies in the extra hardware needed to replace the entire ambiguity group, if one of the group members is faulty. Overall, according to the ambiguity group information illustrated in figure 4, the expected number of nodes needed in the repair phase to replace one faulty node can be computed. Table 2 lists the expected number of nodes to replace a faulty one for each PPA designs.

Observing that the loss of diagnostic resolution is mainly due to the fact that all the members in the same ambiguity group have the same fanout signature, one can improve the diagnostic resolution using additional hardware to make each member's fanout signature distinct. Note that the extra hardware needs to be added within the same framework of the diagnosis scheme. In other words, new output positions need to be added to distinguish each member of an ambiguity group, while the only way to observe any fault at the output positions is through the comparison of redundant copies of the same signal. Such redundancy required to enhance diagnostic resolution can also benefit from exploiting the internal correlation in the prefix computation.

Figure 5 illustrates an example of adding three nodes to distinguish the four nodes, $a, b, c, d$ in the same ambiguity group of a Brent-Kung PPA. Based on the property of prefix operation, the three additional output positions form the same signal as the highest (15th) output position in the original PPA. For example, the fan out line from node $c$ consists of $(G[8,15], P[8,15])$ while the 14th output position consists of $(G[0,14], P[0,14])$. With one of the additional prefix operator nodes, these two pairs are merged. According to the prefix operation:

$$(G[8,15], P[8,15]) \circ (G[0,14], P[0,14]) = (G[0,15], P[0,15])$$

Therefore, the additional node fans out a redundant copy of the signal from the 15th position. Note that full fault diagnostic resolution can be achieved for all the three additional nodes as well.

Without loss of generality, this approach can be applied to make each member of any ambiguity group distinguishable, thus elevating the diagnostic capability to full resolution for all the PPAs. For an ambiguity group of $n$ members, $n-1$ new nodes need to be added. Table 3 lists the number of extra nodes needed to achieve full resolution in each of the PPA designs. It also shows the average hardware overhead of repair for one node based on the original PPA. According to tables 2 and 3, the hardware overhead for repair is significantly reduced for the online diagnosis approach. Moreover, the full diagnostic resolution approach cuts down significantly the hardware repair cost.

## 4.3 Inconsistent fault manifestation

The erroneous output in an online environment depends on the 1) the faulty component, and 2) the input combination. Basically, a fault manifests only when the input combination stimulates the fault and sensitizes its propagation path towards the output. Similarly, if an internal faulty signal is not directly connected to the output, it might lose its observability when it reaches the output positions, after going through intermediate components.

In the example of figure 3, a faulty output of the black block fans out to three output positions $\{5, 13, 14\}$. Of the three positions, the fault will definitely manifest at the 5th position, since the faulty signal goes directly to the output there. Whether the fault will manifest at the 13th output position, however, depends on the input to the node as well as the internal logic of the node which the faulty signal goes through before reaching the output. Certainly, an annihilation of the faulty signal at the 13th output position makes it impossible for any fault effect to propagate to the 14th one. Otherwise, whether the fault manifests at the 14th output position depends on the nodes it propagates through at the 14th position.

Such an *inconsistent fault manifestation* makes online diagnosis challenging, since each fault might have a number of possible fanout signatures. This increases fault aliasing possibility, thus directly impacting diagnostic resolution. In general, any online diagnosis approach relies on certain fault manifestation signatures to identify the faulty components, thus suffering from loss of certainty in fault signatures, due to inconsistent fault manifestation. The following observations regarding inconsistent fault manifestation can help improve diagnostic resolution:

- Inconsistent fault manifestation occurs when a faulty signal goes through certain computation units before finally reaching the output. Therefore, for faulty signals that can directly reach an output position, the manifestation is consistent. For example, in figure 3, the 5th output position has consistent fault manifestation for the faulty unit marked as black.
- Inconsistent fault manifestation has unidirectional impact; for any faulty unit, the output bits reachable from it might be incor-

| 16-bit adder | Brent-Kung | Kogge-Stone | Hybrid |
|---|---|---|---|
| Number of extra nodes | 7 | 14 | 6 |
| Avg repair cost for one faulty node | 1.27 | 1.29 | 1.19 |

**Table 3: HW overhead / repair cost for full diagnostic resolution**

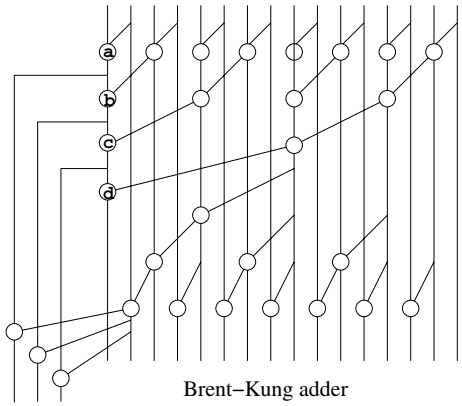



**Figure 5: An example of achieving full diagnostic resolution**

rect or correct, yet the output bits that are not reachable from the faulty block are always correct.

- Whether a faulty signal propagates through a unit and manifests at the output depends on two issues: the internal logic of the unit, and the other inputs of the unit. For a PPA, the internal logic of each prefix operator is known. Under the assumption of uniformly distributed input signals, the inconsistent fault manifestation at the output side can be quantified and evaluated in a probabilistic sense.

Although inconsistent fault manifestation introduces challenges to online diagnostic resolution, it has no impact on fault detection or fault masking. Since inconsistent fault manifestation unidirectionally changes faulty output positions to fault-free ones, neither fault detection nor fault masking capability will be compromised, since the only thing that matters in these two schemes is the correctness of the results, i.e., the signals on the output side.

# 5. CONCLUSION

One of the most severe challenges for nanoelectronic systems is the reliability challenge, which has been known to require tremendous hardware overhead to overcome. The fundamental question of whether reliability can be achieved efficiently determines the success of the future nanoelectronic systems. In this paper, we propose a novel way to enhance the reliability of parallel prefix adders for nanoelectronic systems in a highly efficient approach.

We identify the extant correlation in parallel prefix computation and exploit it for fault tolerance purposes. We develop a number of techniques to construct fault detection, fault masking and online fault diagnosis approaches utilizing the extant hardware redundancy. The proposed approach for parallel prefix adders provides an efficient means of enhancing the reliability of the basic arithmetic building blocks in the nanoelectronic environment. Moreover, the proposed techniques open up a new way to develop powerful fault tolerance schemes with very low overhead and overcome the reliability challenge in nanoelectronic systems efficiently.

# 6. REFERENCES

[1] ITRS, *International Technology Roadmap for Semiconductors Emerging Research Devices*, 2006.

[2] M. A. Kastner, "The Single-Electron Transistor", *Review of Modern Physics*, vol. 64, pp. 849–858, 1992.

[3] P. Mazumder, S. Kulkarni, M. Bhattacharya, J. P. Sun and G. I. Haddad, "Digital Circuit Applications of Resonant Tunneling Devices", *Proceedings of the IEEE*, vol. 86, n. 4, pp. 664–686, April 1998.

[4] P. Avouris, J. Appenzeller, R. Martel and S. Wind, "Carbon Nanotube Electronics", *Proceedings of the IEEE*, vol. 91, n. 11, pp. 1772–1784, 2003.

[5] C. S. Lent, P. D. Tougaw, W. Porod and G. H. Bernstein, "Quantum Cellular Automata", *Nanotechnology*, vol. 4, pp. 49–57, 1993.

[6] Y. G. Krieger, "Molecular Electronics: Current State and Future Trends", *J. Structural Chem*, vol. 34, pp. 896–904, 1993.

[7] P. Beckett and A. Jennings, "Towards Nanocomputer Architecture", in *Asia-Pacific Computer System Architecture Conference*, pp. 141–150, 2002.

[8] M. Forshaw, R. Stadler, D. Crawley and K. Nikolic, "A Short Review of Nanoelectronic Architectures", *Nanotechnology*, vol. 15, pp. 220–223, 2004.

[9] W. Rao, A. Orailoglu and R. Karri, "Fault Identification in Reconfigurable Carry Lookahead Adders Targeting Nanoelectronic Fabrics", in *ETS*, pp. 63–68, 2006.

[10] D. B. Strukov and K. K. Likharev, "CMOL FPGA: A Reconfigurable Architecture for Hybrid Digital Circuits with Two-terminal Nanodevices", *Nanotechnology*, vol. 16, pp. 888–900, Apr 2005.

[11] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford University Press, 2000.

[12] R. J. Sellers, M. Hsiao and L. W. Bearnson, *Error Detecting Logic for Digital Computer*, McGraw-Hill, 1968.

[13] B. W. Johnson, J. H. Aylor and H. H. Hana, "Efficient Use of Time and Hardware Redundancy for Concurrent Error Detection in a 32-bit VLSI Adder", *IEEE Journal of Solid-State Circuits*, pp. 208–215, February 1988.

[14] J. G. G. Langdon and C. K. Tang, "Concurrent Error Detection for Group Look-ahead Binary Adders", *IBM J. Res. Develop*, pp. 563–573, Sep 1970.

[15] M. Nicolaidis, "Carry Checking/Parity Prediction Adders and ALUs", *IEEE Transactions on VLSI Systems*, vol. 11, pp. 121–128, Jan 2003.

[16] D. P. Vasudevan and P. K. Lala, "A Technique for Modular Design of Self-Checking Carry-Select Adder", in *DFT*, pp. 325–333, 2005.

[17] B. K. Kumar and P. K. Lala, "On-line Detection of Faults in Carry-Select Adders", in *ITC*, pp. 912–918, 2003.

[18] K. Nikolic, A. Sadek and M. Forshaw, "Architectures for Reliable Computing with Unreliable Nanodevices", in *IEEE-NANO*, pp. 254–259, 2001.

[19] A. DeHon and M. J. Wilson, "Nanowire-based Sublithographic Programmable Logic Arrays", in *FPGA*, pp. 123–132, 2004.

[20] S. C. Goldstein, M. Budiu, M. Mishra and G. Venkataramani, "Reconfigurable Computing and Electronic Nanotechnology", in *ASAP*, pp. 132–143, 2003.

[21] S. Peng and R. Manohar, "Fault Tolerant Asynchronous Adder through Dynamic Self-reconfiguration", in *ICCD*, pp. 171–179, 2005.

[22] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re and A. Salsano, "Fault Localization, Error Correction, and Graceful Degradation in Radix 2 Signed Digit-Based Adders", *IEEE Transactions on Computers*, vol. 55, pp. 534–540, May 2006.

[23] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[24] P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design*, Morgan Kaufmann, 2000.