A Confidence-Driven Model for Error-Resilient Computing

Chia-Hsiang Chen[1], Yejoong Kim[1], Zhengya Zhang[1], David Blaauw[1], Dennis Sylvester[1], Helia Naeimi[2], Sumeet Sandhu[2]
[1]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI
[2]Intel Corporation, Santa Clara, CA

***Abstract*—We propose an adaptive reliability enhancement structure for deeply-scaled CMOS and future devices that exhibit nondeterministic behavior. This structure forms the basis of a confidence-driven computing model that can be implemented in either a rollback recovery or an iterative dual modular redundancy method incorporating synchronous handshake schemes. The performance and cost of the computing model are estimated using a 45 nm CMOS technology and the functionality is verified by FPGA-based emulation. The confidence-driven computing model is demonstrated using a 16-bit, 12-stage CORDIC processor operating under random, transient errors. The confidence-driven computing model adapts to the fluctuating error rates at the device substrate level to guarantee the reliability of computation at the system level. This computing model costs 4.2 times smaller area and 2.7 times less energy overhead than triple modular redundancy to guarantee a system-level mean time to failure of two years.**

Keywords: reliability, transient error, confidence estimator, rollback recovery, dual modular redundancy.

I. INTRODUCTION

The scaling of semiconductor devices continues to improve the cost and performance of digital integrated circuits. CMOS device size has already reached the tens of nanometer regime. In the meantime, a variety of nano devices, such as carbon nanotube (CNT) [1], graphene [2], spin [3], nanoelectromechanical (NEM) relay [4], memristor [5], have been proposed to sustain Moore's law of scaling for years to come [6]. Although these new CMOS and post-CMOS devices boast the much anticipated high integration density and substantially lower energy consumption, they also exhibit less deterministic behavior. For example, the reduced critical charge with scaling causes the circuits to be more susceptible to accidental disruptions due to soft errors, power supply jitter, and quantum tunneling, leading to reliability concerns as scaling continues.

To overcome the reliability challenge, two common techniques can be employed. One is margining which provides sufficient slack in timing and supply voltage for the worst case; the other technique is fault detection and recovery. Margining can be wasteful as the worst case is rarely encountered. Fault detection and recovery makes use of either spatial redundancy, such as N-modular redundancy (NMR), or temporal redundancy, such as checkpointing and rollback. These methods are either expensive or not flexible enough to adapt to the fluctuating device error rates and the varying application requirements.

We propose a confidence-driven computing model that combines temporal and spatial redundancy to reduce the cost of reliability enhancement for a wide range of error rates. In particular, we use fine-grained temporal redundancy for the tunable reliability at an improved performance compared to checkpointing and rollback; and we apply partial spatial redundancy to reduce the energy and area overhead while providing the necessary protection against permanent errors. The key concept of the proposed computing model is to employ confidence estimators in each stage of computation that enforces a confidence threshold to be met before the hardened output is propagated to the following stage, as shown in Fig. 1. This computing model is emulated on FPGA incorporating real-time error injection at random locations in the circuitry. The experiments based on a 16-bit, 12-stage coordinate rotation digital computer (CORDIC) processor demonstrates several orders of magnitude reliability enhancement with a moderate throughput penalty, and the area and energy overhead are as low as 8% and 14% respectively.

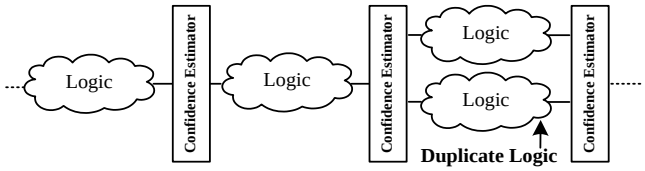

Figure 1. Proposed confidence-driven computing model incorporating confidence estimators.

II. RELATED WORK

An error-resilient computation can be achieved through either error tolerance or error correction. Error tolerance can be provided by the computation algorithm itself, an example of which is the work on algorithmic noise-tolerance (ANT) [7], where errors incurred due to supply voltage over scaling are compensated by the computation algorithm. Error tolerance can be provided at the architecture level, as what has been implemented in systems such as ERSA [8], which employs multiple cores of lower reliability to execute probabilistic applications. The scalable stochastic processor [9] enables error tolerance by transforming a datapath with balanced critical paths to one that is error-scaling friendly. The above works improve the robustness of computation, thus the supply voltage and clock frequency margins can be reduced to lower power consumption and to improve performance. However, error tolerance techniques are often limited in their applicability. ANT, ERSA and stochastic processor techniques all rely on the characteristics of a specific target computation and they need to be tailored to each individual application.

In contrast, an error detection and correction approach is general-purpose in nature. Feed-forward recovery methods provide spatial redundancy such that the errors are detected and corrected with no interruption in the computation. The method is commonly known as N-modular redundancy or NMR. The fluid NMR scheme [10] demonstrates the tradeoff between power and reliability: the number of pre-allocated redundancies can be intelligently selected along with a matching voting strategy to achieve the required reliability with low power. However, cost increases exponentially when the device substrates fluctuate over a wide range of error rates [11]. The BulletProof design [12] incorporates adaptive sparing using routers in a widely-applicable defect-tolerant architecture, but the area and energy overhead are still significant.

In Fig. 2, we classify different error-recovery techniques based on their target error duration from short transient to per-

This research is supported by Intel Corporation.
978-3-9810801-7-9/DATE11/© 2011 EDAA

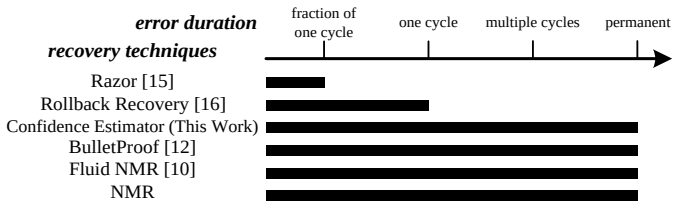

Figure 2. Error-recovery techniques and the applicable error duration.

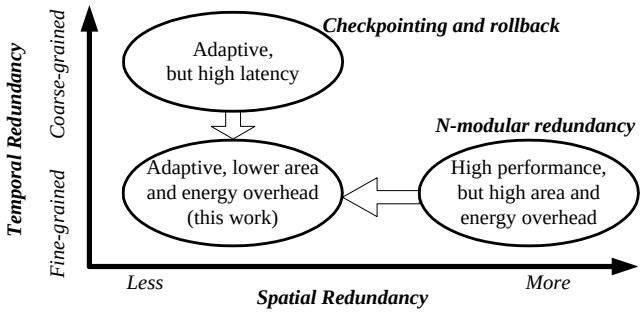

Figure 3. Combination of spatial and temporal redundancy for an efficient error-resilient computing system.

manent. NMR is capable of correcting both permanent errors and transient errors. If an error is known to last for a short duration, a more area- and energy-efficient alternative is through temporal redundancy [13], which can be implemented in a checkpointing and rollback scheme at the software level – errors are detected by repeating the computation over time, thus only one copy of the datapath is needed [14]. Nevertheless, checkpointing requires saving the context, which can be costly. Rollback incurs significant throughput and latency penalty, causing performance degradation especially when errors happen frequently. The Razor technique [15] improves on the conventional checkpointing and rollback by performing rollback at the circuit level. Razor uses a parallel shadow latch to detect and correct errors before the next clock cycle expires. In this way, Razor incurs only a small performance overhead, but the error detection window is also short, limiting its effectiveness to detecting errors that last for a fraction of a cycle. An alternative circuit-level rollback and recovery technique [16] requires a duplicate datapath for error detection and rollback buffers for correction. The rollback recovery technique has a smaller area overhead compared to NMR. Its reliability enhancement is not adjustable either.

The confidence estimator proposed in this paper can be applied with temporal redundancy to extend the error detection window to multiple clock cycles. It also allows temporal redundancy to be applied in conjunction with spatial redundancy to improve the throughput and latency. Its key feature is that it enforces a confidence threshold to be met by looking for agreements, either through repeated computation over the same datapath (temporal redundancy) or duplicate datapaths (spatial redundancy). The confidence threshold can be adjusted based on the device error rate and the application requirement. For example, when the application-required reliability is high and the device error rate is high, the confidence threshold is raised to allow more repeated computations. The resulting throughput degradation can be compensated by a fine-grained temporal redundancy to limit the number of recomputations and increase the clock frequency. The combination of spatial and temporal

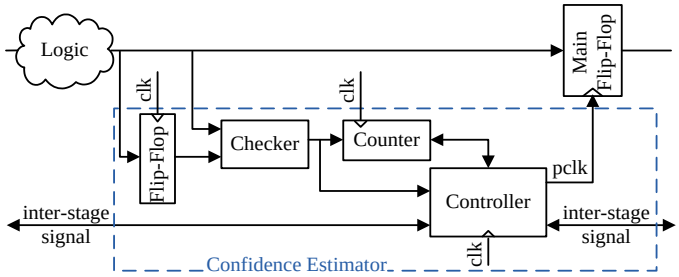

Figure 4. Block diagram of the confidence estimator.

redundancy produces an adaptive error-resilient computing as illustrated in Fig 3.

III. CONFIDENCE-DRIVEN COMPUTING MODEL

In the proposed confidence-driven computing model, a datapath is partitioned into short segments where the probability of error of each segment is bounded. A confidence estimator is placed at the end of each segment to harden the output for forward propagation to the next stage. A confidence estimator (CE) is composed of four components: a flip-flop, a checker, a counter, and a small controller as shown in Fig. 4. The confidence estimator samples the output of the datapath in every clock cycle and compares it with the previous sample stored in the flip-flop (through temporal redundancy), or the output of a duplicate datapath (through spatial redundancy). The checker performs the comparison and looks for either an agreement or a disagreement. The counter keeps track of the confidence level of the output: the confidence level is raised upon an agreement and reset upon a disagreement. The controller ensures that the confidence level reaches the threshold before allowing the output to be propagated through the main flip-flop by enabling the propagation clock (pclk).

An n-bit counter is capable of tracking 2^n distinct confidence levels. The lowest confidence level indicates bypass. The controller generates synchronization signals and coordinates with the neighboring stages through inter-stage signaling. We explore two types of synchronization schemes: lockstep and speculative. Their distinct effects on the output reliability, throughput, latency, as well as the area and energy overhead are described in the following sections after a brief introduction of the emulation framework and the baseline datapath.

A. FPGA-Based Emulation Framework

A CORDIC processor is used to demonstrate the functionality of the confidence-driven computing model. A CORDIC processor is constructed using multiple stages of shift and add operations. Each stage can be entirely identical, an example of which is shown in Fig. 5(a). A 16-bit, 12-stage CORDIC processor is implemented as the baseline design on a Xilinx Virtex-5 FPGA.

To emulate runtime errors, an error injection block is added to the end of each CORDIC stage, as shown in Fig. 5(b). The error injection block inverts the output bits using XOR gates based on the error rate that is selected. The random errors are generated using a set of linear feedback shift registers (LFSR) by comparing their values to some constants: if the LFSR values match the constants, bit errors are injected by inverting the bits. Since each LFSR produces 1 and 0 with nearly equal likelihood, the probability of error being generated is 2^{-x}, where x is the length of the constant. A set of maximal-length LFSRs of various lengths is constructed, each of which

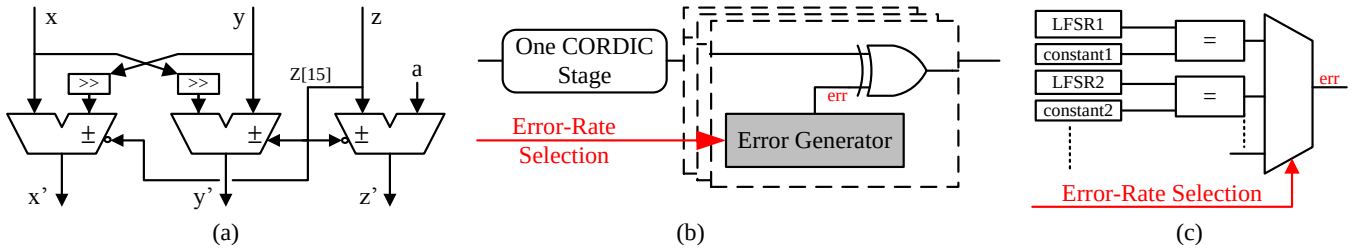

Fig. 5. (a) A single stage of a CORDIC processor composed of two shifters and three adders, (b) error injection mechanism, and (c) error generator.

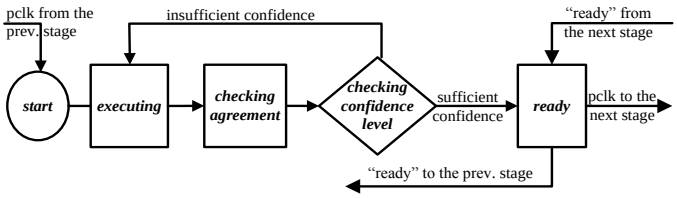

Figure 6. Lockstep synchronization flow.

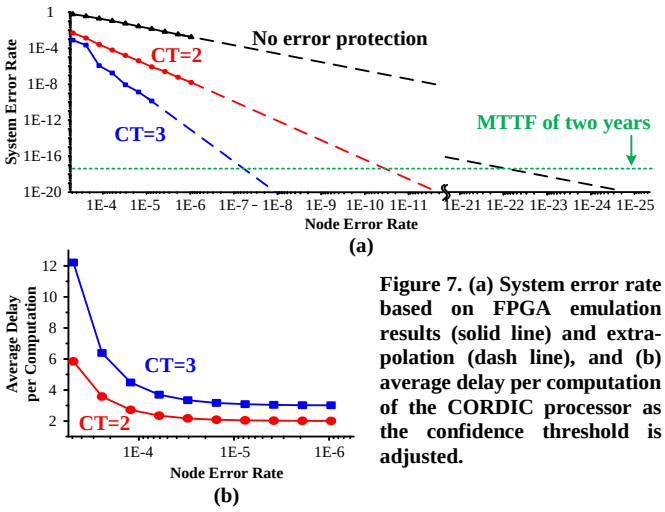

Figure 7. (a) System error rate based on FPGA emulation results (solid line) and extrapolation (dash line), and (b) average delay per computation of the CORDIC processor as the confidence threshold is adjusted.

produces a different error probability. An error-rate selector, shown in Fig. 5(c), sets the error rate by choosing one of the LFSRs.

Each bit of the datapath assumes a different error rate due to the unequal propagation paths [17]. We capture the unequal error rates by mapping the gate level description of the CORDIC processor in software and apply independent and identical circuit node level error injection across the entire datapath. We find that the least significant bit of each CORDIC stage, or bit 0, has the lowest error rate due to the shortest path and fewest number internal nodes along the path. Bit 1 error rate is 2.8 times higher; bit 2 error rate is 3.9 times higher; and bits 3 to 15 error rates are the highest at 4.3 times due to the long paths. This experiment used a 16-bit ripple carry adder topology. A different datapath topology could have equally well been used and would have yielded a different error rate profile. The error rate profile is accounted for in the error injection block by setting the bit 0 at the base error rate and tuning up the error rates of the remaining bits accordingly.

Fault simulation can be accelerated by hardware emulation by up to six orders of magnitude for complex designs [18] compared to software-based simulations. A hardware emulation setup was also used in [8], which relied on pre-stored error vectors supplied through scan chains. In contrast, our FPGA based emulation uses real-time, on-FPGA error generation for the maximum performance.

The 12-stage CORDIC processor is partitioned into two, three or four segments and each segment terminates at a confidence estimator. Properties of the confidence estimator are described in Subsections B and C below by assuming a roll-back recovery (RR) method incorporating only temporal redundancy. The incorporation of spatial redundancy is described in Subsection D in an iterative dual modular redundancy (IDMR) method.

B. Lockstep Synchronization

In the lockstep synchronization scheme, the output of one stage is guaranteed to be hardened before it is propagated to the next stage. A slow stage, due to errors, holds back the computation and leaves the neighboring stages waiting. The flow chart of the lockstep synchronization is shown in Fig. 6. The pclk signal from the previous stage triggers the start of a new computation in the current stage. The output is sampled and it is checked for agreement. The counter accumulates the confidence level (CL) until it reaches the confidence threshold (CT), at which point the current stage enters the ready state. The controller waits until the following stage signals "ready" and then enables the pclk to allow the output to be propagated to the following stage. After the output exits the current stage, a "ready" signal is passed to the previous stage.

The confidence threshold is the primary knob to tune the output reliability level. A confidence threshold of 2 requires 2 agreements and a threshold 3 requires 3 agreements. Note that the confidence estimator looks for consecutive agreements in time or concurrent agreements through redundant spatial copies, which is different from the majority voting scheme used in NMR. Any disagreement resets the confidence level and restarts the confidence accumulation process.

Fig. 7 shows the emulation results of the CORDIC processor on FPGA. Confidence estimators are placed at the end of 6th stage and 12th stage of the CORDIC processor to enhance the system reliability. Fig. 7(a) shows the system error rate (the probability of an incorrect system output) as a function of the circuit node error rate (the probability of error of any circuit node). Without any reliability enhancement mechanism, the error rate of the CORDIC processor is three orders of magnitude higher than the node error rate due to the large number of circuit nodes that are subject to errors. With the confidence estimators and as we increase the confidence threshold, the system error rate decreases by at least four orders of magnitude when the node error rate is at 10^{-5} or lower.

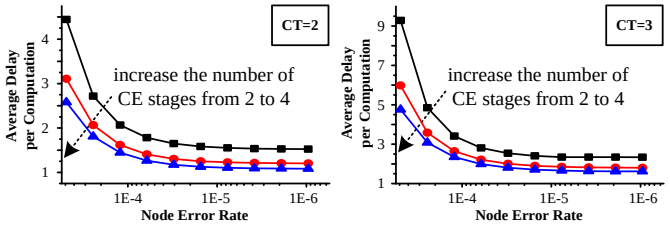

Figure 8. Average delay per computation improves with more fine-grained placement of confidence estimators: (a) at confidence threshold = 2, and (b) at confidence threshold = 3.

To translate the error rates in practical terms, if we were to guarantee a mean time to failure (MTTF) of two years at a 1 GHz clock frequency, the required node error rate is 10^{-22} if no reliability enhancement technique is used. This extremely low node error rate is possible with current mainstream CMOS technology but will likely become difficult with continued device scaling and the new generation of nano devices. By inserting confidence estimators, the required node error rate is relaxed to 10^{-11} with a confidence threshold of 2 and 10^{-8} with a threshold of 3. The threshold can be adjusted at runtime based on the underlying circuit error rate and the application requirement. A high confidence threshold provides better protection but it also decreases the throughput. Fig. 7(b) shows the average delay per computation (the inverse of throughput): when the node error rate is moderate to low, a confidence threshold of 2 requires at least 2 clock cycles per computation and a threshold of 3 requires at least 3 cycles per computation. When the circuit node error rate is high, the delay becomes significant using a higher threshold. Frequent node errors slow down the process of gathering agreements. It is therefore more advantageous to operate the confidence estimators at the lowest confidence threshold that provides the necessary reliability. The runtime configurable confidence threshold accommodates device fluctuations and different reliability requirements among applications. Note that increasing the confidence threshold has only a minor impact on the energy consumption because the datapath under protection usually does not incur additional activities during checking.

To reduce the throughput penalty, we now show how confidence estimators can be placed in finer-grained intervals. A fine-grained placement of confidence estimators shortens the path and bounds the probability of error, contributing to a faster convergence towards the required reliability. The sampling clock frequency can also be increased due to the shortened delay per stage. Fig. 8 shows the improved average delay per computation as more stages of confidence estimators are inserted to the same CORDIC processor. The improvement becomes more significant at high circuit node error rates, thanks to the faster convergence towards the confidence threshold. The delay improvement is also attributed to the increased clock frequency by up to 2.4 times as more confidence estimator stages are inserted, as shown in Table I.

The fine-grained placement of confidence estimators increases the area and energy penalties. Table I presents the comparison among different placement choices by listing the normalized clock frequency, area and energy, which are estimated based on the synthesis results using a 45 nm CMOS technology. Adding one stage of confidence estimator reduces the clock frequency by 30% and introduces an 8% area overhead. Additional stages of confidence estimators improve the clock frequency and the area overhead increases. However, there is a limit to how fine-grained the confidence estimators can be efficiently placed due to the diminishing improvement in throughput, and the escalating cost of area and energy. Design time decisions need to be made based on the expected range of circuit error rates along with the area and energy constraints imposed by the design.

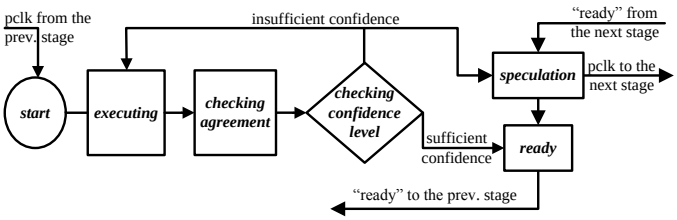

Figure 9. Speculative synchronization flow.

TABLE I. AREA AND ENERGY OF THE ROLLBACK RECOVERY METHOD (LOCKSTEP SYNCHRONIZATION)

	1-stage CE	2-stage CE	3-stage CE	4-stage CE
Normalized clock period	1.30	0.76	0.60	0.54
Normalized area	1.082	1.210	1.337	1.464
Normalized energy (CT=2)	1.14	1.20	1.36	1.53
Normalized energy (CT=3)	1.15	1.23	1.46	1.64
Normalized energy (CT=4)	1.16	1.26	1.54	1.76

C. Speculative Synchronization

The speculative synchronization scheme allows a computation to proceed to the next stage even if the confidence level has not reached the confidence threshold. Compared to the lockstep synchronization, the speculative execution shortens the latency and permits a higher throughput. The flow chart of the speculative synchronization is shown in Fig. 9. Under this scheme, the confidence estimators become transparent gate keepers: tentative output is passed to the next stage when the next stage is ready to accept a new computation, while the tentative output is still being hardened by the current stage. When the current stage finally reaches the confidence threshold, its controller signals "ready" to the previous stage, indicating that it is ready to accept a new computation. The speculative synchronization cuts the idle cycles when one stage is complete and waiting for the previous stage to finish accumulating confidence. The scheme assigns tentative work to otherwise idle stages. The lockstep synchronization can be applied in the final stage to ensure that the final output is hardened to meet the confidence threshold.

The speculative synchronization provides almost identical reliability enhancement compared to the lockstep scheme as shown in Fig. 10(a), and the speculative scheme demonstrates an appreciable improvement in throughput, or average delay per computation, compared to the lockstep scheme when the

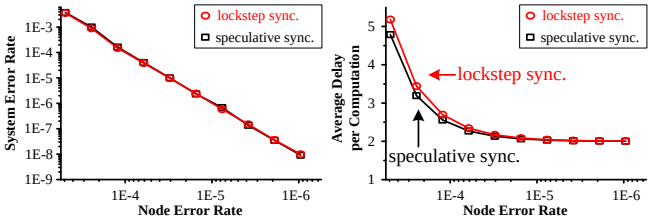

Figure 10. (a) System error rate and (b) average delay of the CORDIC processor using different synchronization schemes (assume a rollback recovery method using a confidence threshold of 2).

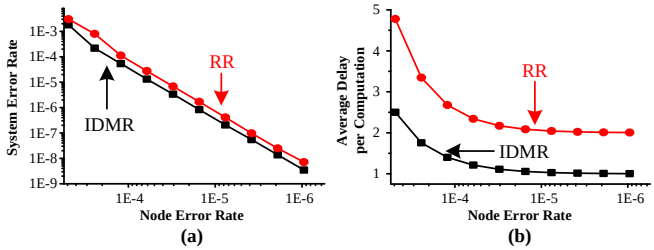

Figure 11. (a) System error rate and (b) average delay of the CORDIC processor using rollback recovery (RR) and iterative DMR (IDMR) method (assume a confidence threshold of 2).

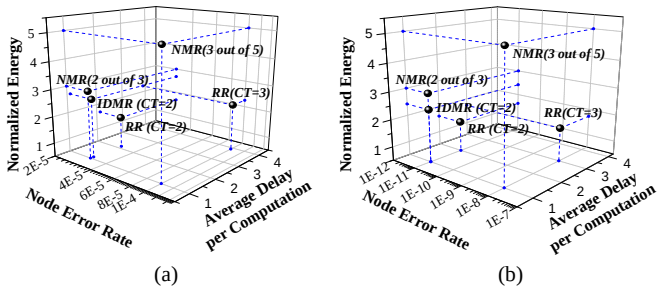

Figure 12. Normalized energy and latency overhead of two cases: (a) a probabilistic application that requires a 10^{-5} system error rate, and (b) a highly reliable general-purpose computing application that requires a MTTF of 2 years.

TABLE II. AREA AND ENERGY OF THE ITERATIVE DMR METHOD (LOCKSTEP SYNCHRONIZATION AND CT = 2)

	1-stage CE	2-stage CE	3-stage CE	4-stage CE
Normalized area	2.03	2.15	2.28	2.40
Normalized energy	2.12	2.17	2.27	2.42

circuit node error rate is high. Moreover, the speculative synchronization can substantially improve the latency of computation. When the circuit node error rate is moderate to low, the latency of computation is almost independent of the confidence threshold because a tentative output is passed along without waiting, followed by parallel checking performed at all the stages. Therefore, the speculative synchronization is especially attractive for latency-sensitive applications.

The area overhead of the speculative scheme is higher than the lockstep scheme by a negligible proportion, and its energy overhead is up to 6 % higher. The higher energy cost is due to the higher switching activity in speculative execution.

D. Spatial Redundancy

The above discussions assume a rollback recovery (RR) method that utilizes only the temporal redundancy. In fact, spatial redundancy can be incorporated in conjunction with the temporal redundancy to achieve additional gains in performance. The simplest way to include spatial redundancy is by providing a duplicate datapath in the form of dual modular redundancy (DMR). A duplicate datapath allows the confidence to be accumulated quickly, thus increasing throughput and minimizing latency. DMR is less expensive than TMR. Errors detected in DMR trigger recomputations for error correction in an iterative DMR (IDMR) method. The inter-stage synchronization is performed using either the lockstep or the speculative scheme described above.

The iterative DMR method demonstrates slightly higher reliability enhancement than the rollback recovery method as shown in Fig. 11(a). The improvement is due to the difference in confidence accumulation policies: an error in DMR invalidates one pair of computations from both the primary and the duplicate datapath, thus the number of agreements needed on average to reach the confidence threshold is slightly higher than the rollback recovery method. Iterative DMR incurs less throughput penalty compared to the rollback recovery method, as shown in Fig. 11(b), but the energy and area are 58% and 64% higher respectively by comparing the results in Table II and Table I (when using four stages of confidence estimators).

The iterative DMR method detects permanent errors when a confidence estimator consistently reports disagreements and fails to meet the confidence threshold over a large number of clock cycles. An adaptive sparing method can be used to dynamically allocate additional spatial redundancy to alleviate permanent errors. This enhancement follows the existing work on sparing [12], [19], and it will be part of our future work.

IV. COMPARISON AND DISCUSSION

In a practical operating environment, the circuit node error rate can fluctuate over time due to supply jitter, temperature variation, noise, and other environmental effects. The effects on different parts of a chip can also vary, e.g., some parts are subject to a higher temperature or a higher noise level due to run-time activities. The confidence-driven computing model based on confidence estimators allows the dynamic adjustment of the confidence thresholds to accommodate variations in time and differences among parts of a chip.

Two case studies are shown in Fig. 12 for two applications: a probabilistic application that tolerates a high system error rate and a general-purpose computing application that requires a very low system error rate. In the first case study targeting a probabilistic application, the confidence-driven computing model guarantees a given system error rate of 10^{-5}, when the underlying circuit node error rate varies between 10^{-5} and 10^{-4}. The protection can be accomplished in several ways: rollback recovery or iterative DMR with either lockstep or speculative synchronization. For example, using rollback recovery and speculative synchronization, the confidence threshold can be set to either 3 or 2 to accommodate node error rate of 10^{-4} and 10^{-5}, respectively. In comparison, TMR guarantees the 10^{-5} system error rate when the node error rate is at

TABLE III. OVERHEAD COMPARISON FOR A PROBABILISTIC APPLICATION

	Technique				
	Confidence-Driven Model			***NMR***	
Applied strategy	RR* (CT=2)	RR* (CT=3)	IDMR** (CT=2)	2 out of 3 (TMR)	3 out of 5
Max node error rate	$3x10^{-5}$	10^{-4}	$3x10^{-5}$	$3x10^{-5}$	10^{-4}
Area overhead	47%	47%	140%	200%	400%
Energy overhead	66%	125%	168%	200%	400%
Delay in cycles	2.13	3.9	1.11	1	1

Assume a required system error rate of 10^{-5}.

*RR: rollback recovery, **IDMR: iterative dual modular redundancy

TABLE IV. OVERHEAD COMPARISON FOR A RELIABLE COMPUTING APPLICATION

	Technique				
	Confidence-Driven Model			***NMR***	
Applied strategy	RR* (CT=2)	RR* (CT=3)	IDMR** (CT=2)	2 out of 3 (TMR)	3 out of 5
Max node error rate	10^{-11}	10^{-8}	10^{-11}	10^{-11}	10^{-8}
Area overhead	47%	47%	140%	200%	400%
Energy overhead	62%	73%	142%	200%	400%
Delay in cycles	2	3	1	1	1

Assume a required system error rate of 1.6×10^{-17}(MTTF of 2 years at a 1 GHz clock frequency).

*RR: rollback recovery, **IDMR: iterative dual modular redundancy

10^{-5}, but it fails to adapt when the circuit error rate deteriorates.

The confidence-driven model has low area and energy overhead, which are 4.2 times and 1.6 times lower compared to TMR as shown in Table III based on the rollback recovery method. The iterative DMR method is more costly, but still 30% more area-efficient and 16% more energy-efficient than TMR. Compared to the fluid NMR that is able to adapt to the range of node error rates, the rollback recovery method costs 8.5 times smaller area and 3.2 times lower energy.

The second case study targets a highly reliable general-purpose computing application that requires two years of MTTF. We estimate that the rollback and recovery method costs 4.2 times smaller area and 2.7 times lower energy than TMR as shown in Table IV.

V. CONCLUSION

We propose a confidence-driven computing model that enhances the system reliability even when the underlying circuits operate non-deterministically. The reliability enhancement is enabled by the confidence estimators using both spatial and temporal redundancy. Fine-grained temporal redundancy lowers the performance penalty, while the reduced spatial redundancy cuts the area and energy overhead. As a result, the confidence estimator based methods can be realized more efficiently compared to the competing alternatives, such as TMR.

We explore two choices in implementing the confidence-driven computing: rollback recovery and iterative DMR. The rollback recovery method is more area and energy efficient, and the iterative DMR method features a higher throughput. The two methods can adopt one of the two synchronization schemes: lockstep or speculative. The speculative scheme permits a lower latency and average delay per computation with a negligible increase in energy.

The confidence-driven computing model represents a promising solution that bridges the gap between the varying application-required reliability and the fluctuating device behavior in future technologies. The placement of confidence estimators, along with the tuning of the confidence threshold and the synchronization scheme will allow us to construct a reliability-diverse computer architecture with computing elements that provide a range of reliability levels at appropriate energy cost to deliver the required performance.

ACKLOWEDGEMENT

The authors would like to thank Farhana Sheikh, Keith Bowman, Feng Xue, Tanay Karnik, Chris Ramming and Shekhar Borkar for guidance and suggestions.

REFERENCES

[1] H. Wei, *et al.,* "Efficient Metallic Carbon Nanotube Removal Readily Scalable to Wafer-Level VLSI CNFET Circuits," in *Symp.VLSI Circuits*, pp. 237-238, 2010.

[2] S. J. Han, *et al.,* "Study of Channel Length Scaling in Large-Scale Graphene," in *Symp.VLSI Circuits*, pp. 26-27, 2010.

[3] T. Inokuchi, *et al.,* "Reconfigurable Characteristics of Spintronics-based MOSFETs for Nonvolatile Integrated Circuits," in *Symp.VLSI Circuits*, pp. 119-120, 2010.

[4] F. Chen, *et al.,* "Demonstration of Integrated Micro-Electro-Mechanical Switch Circuits for VLSI Applications," in *IEEE Int. Solid-State Circuit Conf.*, pp. 26-27, 2010.

[5] S. H. Jo, K. H. Kim, and W. Lu, "Programmable Resistance Switching in Nanoscale Two-Terminal Devices," *Nano Lett.*, vol. 9, no. 1, pp. 496-500, 2009.

[6] International Technology Roadmap for Semiconductors 2009 [Online] Available: http://www.itrs.net/Links/2009ITRS/Home2009.htm

[7] R. Hegde and N. R. Shanbhag, "Energy-Efficient Signal Processing via Algorithmic Noise-Tolerance", in *Proc.Int. Symp. Low Power Electron. and Design*, pp. 30–35, 1999.

[8] L. Leem, *et al.,* "ERSA: Error Resilient System Architecture for Probabilistic Applications," in *Design, Automation, and Test in Europe Conf.,* pp. 1560-1565, 2010.

[9] S. Naravana, *et al.,* "Scalable Stochastic Processors," in *Design, Automation, and Test in Europe Conf.*, pp. 335-338, 2010.

[10] J. Sartori, J. Sloan, and R. Kumar, "Fluid NMR – Performing Power/Reliability Tradeoffs for Applications with Error Tolerance," in *Workshop on Power Aware Computing and Systems*, 2009.

[11] K. Nikolic, A. Sadek, and M. Forshaw, "Fault-Tolerent Techniques for Nanocomputers," *Nanotechnology*, vol. 13, no. 3, pp. 357-362, May, 2002.

[12] K. Constantinides, *et al.,* "BulletProof: A Defect-Tolerant CMP Switch Architecture," in *Int. Symp. High-Performance Comput. Architecture*, pp. 5-16, 2006.

[13] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," in *VLSI Test Symp.*, pp 86-94, 1999.

[14] K. G. Shin, and H. Kim, "A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods," in *IEEE Trans. Comput.*, vol. 43, no. 10, pp. 1151-1162, 1994.

[15] D. Ernst, *et al.,* "Razor: A Lowpower Pipeline Based on Circuit-Level Timing Speculation," in *Proc. IEEE/ACM Int. Symp.Microarchitecture*, pp. 7–18, 2003.

[16] H. Naeimi and A. DeHon, "Fault-Tolerant Sub-Lithographic Design with Rollback Recovery," *Nanotechnology*, vol. 19, no. 11, Feb., 2008.

[17] B. E. S. Akgul, *et al.,* "Probabilistic CMOS Technology: A Survey and Future Direction," in *VLSI IFIP Int. Conf.* , pp. 1-6, 2006.

[18] A. Pellegrini, *et al.,* "Crash Test: A Fast High-Fidelity FPGA-Based Resiliency Analysis Framework," in *IEEE Int. Conf. Comput. Design*, pp. 363-370, 2008.

[19] S. Gupta, S. Feng, A. Ansari, J. Blome, and S. Mahlke, "The StageNet Fabric for Constructing Resilient Multicore Systems," in *Proc. IEEE/ACM Intl. Symp. on Microarchitecture*, pp.141–151, 2008.