Spring 2020

Novel approaches for reliable and efficient circuit design

Prashanthi Metku

Department: Electrical and Computer Engineering

NOVEL APPROACHES FOR RELIABLE AND EFFICIENT CIRCUIT DESIGN

by

PRASHANTHI METKU

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2020

Approved by:

Dr. Minsu Choi, Advisor
Dr. Daryl Beetner
Dr. Sahra Sedigh Sarvestani
Dr. Joe Stanley
Dr Abusayeed Saifullah

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following four articles, formatted in the style used by the Missouri University of Science and Technology:

Paper I, found on pages 6–32, Adaptive Multi-path BCH Decoder to Alleviate Hotspot-induced DRAM Bit Error Variation in 3D Heterogeneous Processor published in Journal of Semiconductor Technology and Science.

Paper II, found on pages 33–51, Novel Area-Efficient Null Convention Logic based on CMOS and Gate Diffusion Input (GDI) Hybrid Methodology accepted in Journal of Semiconductor Technology and Science.

Paper III, found on pages 52–80, A Low Power Design Technique for the Asynchronous Null-Convention Logic Circuits is intended for submission to IEEE Transactions on Very Large Scale Integration (VLSI) Systems.

Paper IV, found on pages 81–103, Energy-Performance Scalability Analysis of a Novel Quasi-Stochastic Computing Approach published in Journal of Low Power Electronics and Applications.

ABSTRACT

In this research work, a suite of approaches are presented to improve reliability of 3D heterogeneous processors (3DHP) and to reduce the area overhead of asynchronous designs. This work is primarily divided into two parts. In the first part, we present an approach for improving reliability in 3DHP. Typically, in 3DHP, thermal hotspots introduce spatial and temporal variability that results in wide bit error variation in DRAM dies. To address this issue multi- path BCH decoder is introduced. Based on the thermal gradient data generated by on-chip temperature sensors, the proposed methodology specializes in adaptively estimating the number of errors in the incoming word and also selecting the fast decoding path to correct these errors. Thus, provides DRAM error protection with minimal decoding latency. In the next part of this work, we focus on reducing the area overhead of asynchronous paradigm-driven null convention logic (NCL) design using Gate Diffusion Input (GDI). We first develop technique for realizing NCL gates. In the process, we demonstrate that there is a voltage swing at the output that may introduces errors. To address this limitation, a HYBRID approach is introduced where conventional complementary metal oxide semiconductor (CMOS) technology is integrated with GDI methodology. With this approach, we demonstrate that we can reduce the transistor count (TC) of the NCL designs while addressing the limitations due to voltage drop. To further reduce the TC of the NCL designs, GNCL is developed. This approach utilizes the regenerative buffers to overcome the performance degradation and also reduce the area overhead. Overall in this dissertation, we demonstrate reductions in area and power overheads for asynchronous designs.

ACKNOWLEDGMENTS

I want to extend my sincere gratitude to my advisor, Dr. Minsu Choi for allowing me to work and learn under his tutelage and helping me acquire invaluable insights to make this work possible. Dr. Choi’s constant support and patience has helped me develop the skills at my own pace and time. All these years working under him will go down as some of the most valuable and productive years of my student life.

I profoundly thank Dr. Daryl Beetner, Dr. Sahra Sedigh Sarvestani, Dr. Joe Stanley and Dr Abusayeed Saifullah for being my committee members and for their valuable inputs. I would also like to thank my lab partners and friends from Missouri University of Science and Technology especially Krishnan Raghavan and Vikram Chowdhury, for providing help and support during the course of my doctoral studies at Missouri S&T.

I am immensely grateful and owe my utmost gratitude towards my parents Venkateshwar Metku and Kalpana Metku for giving me the chance to be who I am today. Thank you both for believing in me and giving me strength to chase my dreams. Finally, special thanks to my brothers Virjanand Sagar, Vivekanand Sagar amd Sandeep Nallagula for constant understanding and encouragement.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION

1. INTRODUCTION

For the past few decades, scientists have been scaling devices to increasingly smaller feature sizes for enhanced performance of complementary metal-oxide semiconductor (CMOS) technology, thereby increasing the functionality of integrated circuits and systems [1, 2, 3]. However, with the exponential growth of transistor densities, power efficiency has become primary determinant of performance in todays' semiconductor industry [4]. In addition to power concerns, off-chip bandwidth trends are also expected to have a major impact on the scalability of the future designs [5]. In particular, the demand for high computing performance has increased in accordance with the requirements for smaller and more energy efficient devices. One way to obtain high computation performance is by increasing the robustness of a single processor [7]. This can be achieved by increasing its clock frequency and mounting more transistors such that more calculations could be executed. However, with the physical limits of such processors being fully exploited and an advanced version of computing strategy, heterogeneous computing i.e. using heterogeneous or hybrid platform containing more than one type of processor was introduced such that different types of tasks can be executed by processors that are specialized in them [7].

Recently, many of highly-ranked Performance computing systems include discrete Graphics Processing Unit accelerators (GPU) [8]. Systems where discrete GPUs are connected to CPUs over PCI-E bus, however, frequently suffer from a significant data copy overhead between two processors. To address this limitation, researchers in industry and academia are trying to seek a solution in a single-chip heterogeneous processors where CPU and GPU share a unified memory hierarchy [9]. However, parallelism and scalability of such heterogeneous processors are still severely constrained by limited bandwidth, high latency, and energy consumption of offchip DRAM [10, 11]. To address these bottlenecks, the processor architecture is evolving toward a 3D heterogeneous integration (commonly termed as 3DIC) [12]. In 3DIC four heterogeneous dies (i.e., CPU, GPU, analog and DRAM) are vertically interconnected by a massive number of through-Silicon Vias (TSVs). Compared with the traditional off-chip interconnects, TSVs enable a massive number of vertical channels among CPU, GPU and DRAM dies while providing much shorter distance of data travel [12, 13]. Therefore, 3DHP technology is anticipated to inherently provide much higher bandwidth, low latency and power consumption. Despite numerous unprecedented benefits, however, there is a big challenge which is a thermal reliability issue [13].

A significantly higher power density, thinned substrate, and low thermal conductivity of inter-layer material all make heat dissipation a serious problem that threatens circuit reliability and performance in 3DIC [13]. Various design-time solutions are available to tackle hotspots in 3DIC designs but the transient nature of thermal hotspots cause the design time solutions less effective. The increase in power density of 3D stacking causes an elevation in the temperature, which nominally results in an exponential rise in charge leakage of DRAM cells [13]. Therefore, requires significant increase in refresh frequency to retain data at the expense of additional power and performance overhead. Also, the spatial and temporal variability in temperature (i.e., hotspots) further complicates the DRAM reliability issues thereby requiring error detection and correction (EDAC) techniques.

The conventional 2D EDA assume a near constant bit error rate (BER) over time. Hence, EDAC engine does not need to be designed to adapt to a varying BER over time. For instance, state-of-the-art SECDED (Single Error Correction, Double Error Detection) code [14] and buses with CRC (Cyclic Redundancy Check) code [15] cannot be directly applied to the proposed 3DHP, since it is anticipated to have a varying BER and TSV failure rate over time caused by the thermally-induced reliability issues. Therefore, a novel approach to tackle this limitation is presented in the first part i.e. paper I of this dissertation.

The second and third part of this dissertation discuss about the asynchronous paradigm, null convention logic (NCL). The advantages, limitations and a methodology to address these challenges have also been part of the dissertation. Conventional synchronous logic with clocked structures have been dominating semiconductor industry over the past decades [16]. However, the continuous decreasing in the feature size and increasing operating frequency of integrated circuits (IC), clock-related issues such as clock skews, increased power at the clock edges, extra area, and layout complexity for clock distribution networks, and glitches are emerging as the dominant factor hindering increased performance [17]. These limitations have caused renewed interest asynchronous digital design. Asynchronous, clockless circuits require less power, generate less noise, and produce less electro-magnetic interference (EMI), compared to their synchronous counterparts, without degrading performance. Furthermore, delay-insensitive (DI) asynchronous paradigms have a number of additional advantages, especially when designing complex circuits, like Systems-on-a-Chip (SoCs), including substantially reduced crosstalk between analog and digital circuits, ease of integrating multi-rate circuits, and facilitation of component reuse [18, 19].

Null Convention Logic (NCL) is a delay-insensitive (DI) asynchronous. NCL was first proposed by Karl Fant and Scott Brandt in 1994 [20, 21], and further developed by Dr. Scott Smith's research group [22]. NCL initially aimed at designing Application Specific Integrated Circuit (ASIC) and Very-large-scale Integration (VLSI) circuits with lower power, lower noise, and lower electromagnetic interference (EMI). Various NCL based circuits have shown these characteristics. An NCL based Motorola STAR08 processor [23] shows the power and noise reduction up to 40% and 10dB, respectively, comparing to its synchronous counterpart. In [24], an 8-operation NCL ALUs was designed as a benchmark. The simulation result shows that the dual-rail NCL circuit consumes less power and other designs like NCL divider [25] and NCL multiply-and-accumulate unit [26] have shown the benefits of speed improvement and reduction in power consumption, noise, and EMI. However, the major drawback of NCL designs is that it requires a larger area compared with the conventional Boolean logic version. The area overhead is approximately 1.5 – 2 times as much as an equivalent synchronous design when using static CMOS gates, but less for semi-static CMOS gates [27].

This dissertation proposes and demonstrates two novel approaches to address this limitation of NCL. These approaches when compared with the conventional static CMOS methodology show a significant reduction in the transistor count which in turn helps in reducing the area overhead. However, power and delay analysis in the first approach was not fully studied. Similarly, latency analysis of the second approach will be the part of the future work. In additional to these work, additional research has conducted in the area of stochastic computing (SC) which is discussed in the fourth part of the dissertation. Traditionally, SC’s accuracy heavily depends on the stochastic bitstream length. Therefore, generating acceptable approximate results while minimizing the bitstream length is challenging, as energy consumption tends to linearly increase with bitstream length. To address this issue, a novel energy-performance scalable approach based on quasi-stochastic number generators is proposed and validated in this work.

PAPER

I. ADAPTIVE MULTI-PATH BCH DECODER TO ALLEVIATE HOTSPOT-INDUCED DRAM BIT ERROR VARIATION IN 3D HETEROGENEOUS PROCESSOR

ABSTRACT

A 3D heterogeneous processor (commonly termed as 3DHP) integrates multiple processor (such as CPU/GPU) and DRAM dies, interconnected vertically by a massive number of Through-Silicon Vias (TSVs). The 3DHP is expected to address the limited bandwidth, high latency and energy consumption of off-chip DRAM. However, spatial and temporal variability due to hotspots in on-chip thermal gradient may result in wide bit error variation in DRAM dies. This work proposes a novel adaptive multi-path BCH decoder to efficiently address this issue. Instead of having a static BCH decoder designed from the worst-case bit error probability analysis, the proposed adaptive multi-path BCH decoder offers multiple decoding paths with varying target number of error bits to correct, which is estimated from the thermal gradient data generated by on-chip temperature sensors. Thus minimizes the overall decoding latency adaptively. The proposed approach has been verified by implementing an adaptive 4-path BCH decoder in FPGA hardware. A series of decoding performance evaluation data has been generated to demonstrate the efficiency of the proposed design.

1. INTRODUCTION

Processors are evolving toward a 3D heterogeneous integration (3DIC) of CPU, GPU and DRAM dies vertically interconnected by TSVs (Through-Silicon Vias) to alleviate power, bandwidth and latency bottlenecks. Figure. 1 shows an example where four heterogeneous dies (i.e., CPU, GPU, analog and DRAM) are stacked and interconnected by TSVs. When compared with the traditional off-chip interconnects, TSVs enable a massive number of vertical channels along CPU, GPU and DRAM dies while proving a much shorter distance of data travel. Therefore, 3DHP technology is anticipated to inherently provide higher bandwidth, low latency and low power consumption. Despite the numerous unprecedented benefits, 3DHP face a big challenge which is thermal reliability issues.

The conventional 2D integration/packaging technology is mature enough to assume a near constant bit error probability (BEP) over time in DRAM. Hence, the Error Detection and Correction (EDAC) engine does not need to be designed to adapt to a varying BEP over time. However, the same EDAC strategy cannot be directly applied to 3DHP, since it is anticipated to have a varying BEP caused by hotspots (i.e., spatial/temporal variation in temperature). Various design-time solutions are available to tackle hotspots in 3DIC designs but the transient nature of thermal hotspots cause the design time solutions to be less effective. It is thus important to monitor the chip temperature during runtime using distributed temperature sensors to avoid potential temperature-induced failures. The main objective of this work is to propose and validate a novel adaptive multi-path BCH error correction decoder that provides just-enough DRAM error protection to minimize the overall decoding latency. The proposed decoder can be coupled with on-chip distributed temperature sensor network to analyze the thermal gradient to adaptively tolerate spatial/temporal bit error variance in a 3DHP.

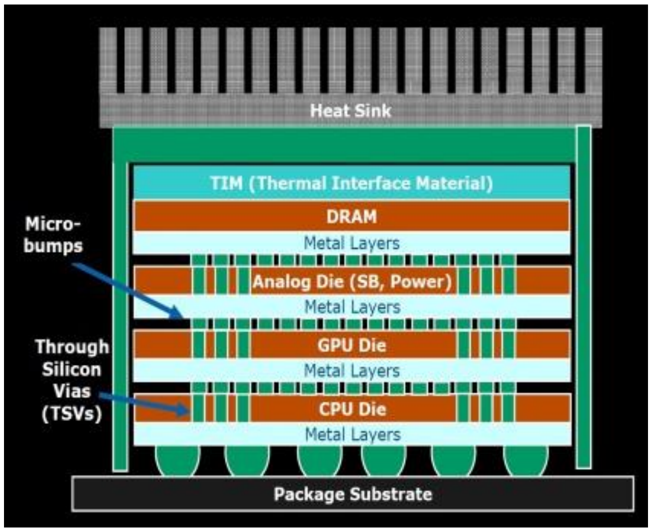


Figure 1. 3D stacking of CPU, GPU, analog and DRAM dies using TSVs [1]

This article is organized as follows. Preliminaries and review are given in Section 2. Then, the proposed adaptive multi-path BCH decoder design is extensively discussed in Section 3. Design and performance evaluation data including the area and latency are included in Section 4. Finally, concluding remarks are made in Section 5.

2. ARCHITECTURE

In 3DHP, the increase in power density of 3D stacking causes an elevation in the temperature, which nominally results in an exponential rise in charge leakage of DRAM cells. Therefore, requires significant increase in refresh frequency to retain data at the expense of additional power and performance overhead. Also, the spatial and temporal variability in temperature (i.e., hotspots) further complicates the DRAM reliability issues.

The Leakage power of DRAM cell is modeled to exponentially increase with temperature, T, as $\mathrm{P_{leakage}} = \mathrm{P^0} . \exp(-\mathrm{A} / \mathrm{A} - \mathrm{B})$ where $\mathrm{P}^{\boldsymbol{0}}$ is the room-temperature leakage power and parameters A and B are empirical constants [2], [3]. The number of discharged cells in DRAM is proportional to the dissipation of the leakage power of cells. In [4], the relation between the error rate (i.e., the number of discharged cells divided by the total number of cells) and temperature is modeled as $E_{DRAM} \propto P_{leakage}$ where $\mathrm{E_{DRAM}}$ is the error rate and $\mathrm{P_{leakage}}$ is a function of T. The retention time distribution of cells is known to be divided into two regions: 1) tail distribution, and 2) main distribution [5]. For thermally-stable operation condition, the retention time of almost all the memory cells belong to "Main Distribution". However, there are a few memory cells whose retention time does not belong to "Main Distribution." This shorter retention time distribution is defined as "Tail Distribution." The refresh characteristics of DRAM are dominated by "Tail Distribution". Further, because leakage power is exponentially increased as temperature rises, which means more and more cells fail to retain charge and become a part of the "Tail distribution". To compensate this reduction in retention time, refresh period should be shorten to refresh more frequently. Yun et al has used the data reported in [6], [7] to fit the parameters and the resulting plot is shown in Figure 2. This figure shows how DRAM error rate is related to temperature and refresh period.

In 3DHP, high temperature exponentially increases the charge leakage in DRAM memory. High refresh rates can address this issue, but reduces the performance and increases power consumption [4] [7]. The temporal and spatial change in temperature, known as hotspots further complicates the reliability issue in 3DHPs [8]. Error Correction Code (ECC) is used to address this issue in 3DIC. There are different ways the ECC can be employed depending on the number of errors detected and corrected. For example single error correction and double error detection [9] (SECDED) method can detect up to two errors but can only correct a single error.

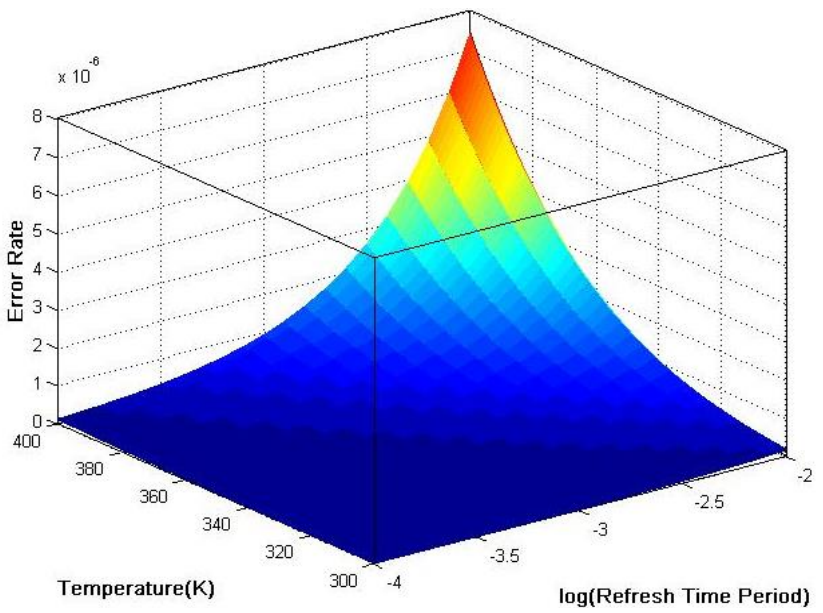


Figure 2. DRAM error rate as a function of temperature and refresh period reported in [4]

For multi-bit burst error correction, strong BCH cyclic codes can be used to provide better error correction performance [10, 11]. However, the hardware complexity of ECC circuit exponentially increases as the number of error bits to correct increases. Therefore, a novel the ECC solution with a lower area-latency product is needed to address the bit error variability caused by hotspots in 3DHP.

3. ADPATIVE MULTI-PATH DECODER DESIGN

A DRAM die consists of multiple memory cells, where each data bit is stored as a charge in the storage capacitor. The charging and discharging actions of the storage capacitor are directly related to temperature. Due to hotspots, leakage current increase which thusly discharges the charge stored by the capacitor and increases the probability of the memory errors [2].

To ensure thermal reliability and better performance of DRAM dies in 3DHP, a temperature-based adaptive ECC is proposed. Bose-Chaudhuri-Hocquenghem (BCH) codes are strong efficient error-correcting codes used to detect and correct enormous errors that have occurred in memory [11]. In 3DHP, hotspots show spatial/temporal localities as they are mainly caused by aggressive switching activities in CPU and GPU processor dies. To ensure thermal integrity among 3D-stacked dies, on-chip temperature sensors are placed to detect hotspots. When the proposed adaptive BCH decoder reads a word to decode, temperature measurement data from the distributed on-chip temperature sensor network is also read and used to calculate n_{EEB}, which is the estimated number of Error Bits for the incoming word. Then, the fastest decoding path which can be used to correct n_{EEB} number of error bits gets adaptively selected to decode the incoming codeword with the minimum decoding latency. The main advantage of the proposed adaptive multi-stage BCH decoder to single stage BCH decoder is the reduced decoding latency with area overhead.

For any integer $\mathrm{m} \geq 3$ and $\mathrm{t} < 2^{\mathrm{m}-1}$, there exists a binary t-error-correcting (n, k) BCH code, which satisfies the following conditions: (1) $\mathrm{n} = 2^{\mathrm{m}-1}$, (2) $\mathrm{n} - \mathrm{k} \leq \mathrm{mt}$, and (3) $d_{min} \geq 2t + 1$, where n is the total number of bits per codeword, k is the number of information bits, $n - k$ is the number of check bits, t is the maximum number of error bits corrected per codeword, and d_{min} is the minimum Hamming distance.

The proposed multi-path BCH decoder has multiple decoding paths with variable target t and decoding latency. As a concrete demonstration of the proposed multi-path BCH decoding approach, a 7-error correcting (511, 448) BCH decoder (i.e., $m = 9$) with $p = 4$ decoding paths with target $t = 1, 3, 5$ and 7 has been designed and verified in this work. ECC word size is normally chosen to match the size of the last level cache block, which is 64B for most current processors. Current ECC DRAMs come with $1/8_{th}$ of the capacity for storing ECC check bits, thus a 64B memory block already has 64 bits reserved for ECC. Hence, the closest n and k values (i.e., $n = 511$ and $k = 63$) are chosen for the proposed design. Figure 3 shows a block diagram of the proposed adaptive multi-path BCH decoder with $p = 4$.

The proposed adaptive multi-path BCH decoder has three main advantages over the static BCH decoder designed for a fixed t (e.g., static BCH decoders for $t = 1$ for faster decoding and $t = 7$ for higher error correction coverage). First, it can reduce the overall decoding latency, when compared to a static BCH decoder with higher t (e.g., $t = 7$). Second, it can provide better error correction coverage, when compared to a static BCH decoder with lower t (e.g., $t = 1$). Third, it can be used to decode multiple words depending on n_{EEB} for further reduction in decoding latency, as it has multiple independent decoding paths.

The proposed BCH decoder design consists of four decoding paths, each designed to correct a specified number errors. The temporal and spatial changes in the temperature are recorded by the distributed onchip temperature sensors, from which Bit Error Probability (p_{BE}) is calculated. p_{BE} is used to determine n_{EEB}.

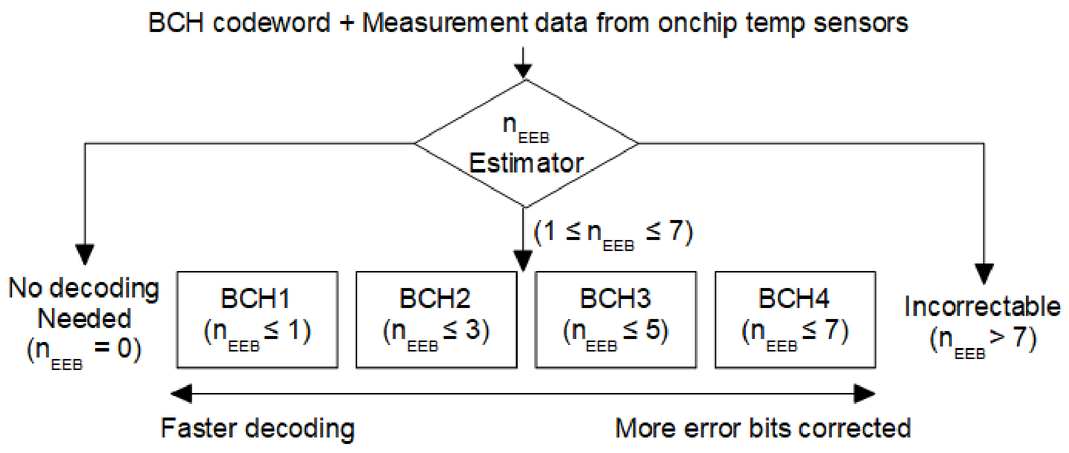

BCH codeword + Measurement data from onchip temp sensors
n_EEB Estimator
(1 ≤ n_EEB ≤ 7)
No decoding Needed (n_EEB = 0)
BCH1 (n_EEB ≤ 1)
BCH2 (n_EEB ≤ 3)
BCH3 (n_EEB ≤ 5)
BCH4 (n_EEB ≤ 7)
Incorrectable (n_EEB > 7)
Faster decoding
More error bits corrected

Figure 3. The proposed multi-path BCH Decoder. The estimated number of error bits for the incoming BCH codeword is calculated from the measurement data from onchip temperature sensors and is denoted as n_{EEB} [12]

To ensure accuracy, there is a provision for calculating n_{EEB} depending on confidence level. The confidence level is interpreted as the likehood that a particular confidence interval contains the actual n_{EEB}. For the proposed n_{EEB} estimator, a one-sided upper-bounded confidence interval is appropriate since estimation error only happens when the actual number of erroneous bits is greater than n_{EEB}. The confidence interval can be selected to calculate n_{EEB} depending on the desired accuracy. This calculation is based on p_{BE}. There exists a trade-off between n_{EEB} and the confidence level. For the higher confidence level, n_{EEB} calculation is more precise, resulting in a higher number of expected error bits. As the number of error correcting bits increases the decoding latency increases slowing down the decoding.

The first decoding algorithm for binary BCH codes was devised by Peterson in 1960 [13]. Since then, the Petersons algorithm has been refined by Berlekamp [14], Massey [15], Chien [16], Forney [17], and many others. The BCH decoder follows the sequence of decoding steps which are Syndrome Calculator [18], Error Locator Polynomial [19] and Chien Search [19]. These algorithms are interrelated, i.e. the syndrome calculator calculates the syndrome according to the received data. Error locator polynomial is generated from syndrome value, and the error location is calculated using Chien search and the transmitter can be achieved.

3.1. SYNDROME BLOCK DESIGN FOR MULTI-PATH BCH DECODER

BCH code can be implemented in hardware and software. There have been numerous efficient decoding algorithms reported in the literature. Two recent examples are [18, 20]. For a BCH code with $\mathrm{n} = 2^{\mathrm{m}-1}$ and generator polynomial g(x), a code polynomial $c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$ is generated from the encoder and its binary representation is stored as a word in the DRAM. When this word is read from DRAM, a received polynomial (i.e., polynomial representation of the received word) is created as $r(x) = r_0 + r_1x + r_2x^2 + \cdots + r_{n-1}x^{n-1}$. Note this received word is expected to contain $\mathrm{n_{EEB}}$ number of error bits and should be decoded for error correction, if $1 \leq \mathrm{n_{EEB}} \leq 7$. In the BCH code, $r(x) = c(x) + e(x)$, where $e(x) = e_0 + e_1x + e_2x^2 + \cdots + e_{n-1}x^{n-1}$ is the error polynomial.

The initial step in BCH decoder is the syndrome calculator. It provides information to later decoding stages for error detection and correction. The received codeword is error free when the syndrome outputs are zeros. Consider a t-error-correcting BCH code of length n $= 2^{\mathrm{m}-1}$ with generator polynomial g(x), where g(x) has $\alpha, \alpha^2, \ldots., \alpha^{2t}$ roots, as g(α^i) = 0 for $1 \leq i \leq 2t$. To check whether r(x) is a code polynomial or not, one can simply test whether $r(\alpha^i) = 0$ for $1 \leq i \leq 2t$. If yes, then r(x) is a code polynomial, otherwise r(x) is not a code polynomial and error correction decoding is needed.

Syndrome calculation in BCH decoding directly deals with the errors present. For the target t, 2t syndrome components are calculated for $1 \leq j \leq 2t$,

$$S = \sum_{i=0}^{n-1} r_i \alpha^{ij} \tag{1}$$

where α is the primitive element. The polynomials which are not factorable (i.e., divisible by one and itself) are called irreducible polynomials. The root of this polynomial is called primitive polynomial [21] and it generates all non-zero field elements. These non-zero fields are used in the generation of $GF\ (2^m)$ and α is its primitive element. Thus, s_j is calculated as $s_j = ((\left(\left((r_{n-1}\alpha^j + r_{n-2})\alpha^j + r_{n-3}\right)\alpha^j \ldots. + r_1\right)\alpha^j + r_0)$.

Figure 4 shows the procedure followed for syndrome calculation. From the n bit received data the (n − 1) bit is first multiplied by the primitive element α^j then the resultant is XORed with the (n − 2) bit. The obtained result is again multiplied with α^j and the resultant is XORed with (n − 3) bit. This process is continued untill bit position 0 and the final result is s_j. The calculation of odd syndrome components from even syndrome components leads to area efficient hardware design with reduced latency [18], as the relation between them is

$$s_{2t} = s^2{}_t \tag{2}$$

For the proposed adaptive BCH decoder with 4 decoding paths (i.e., BCH1, BCH2, BCH3 and BCH4), four individual syndrome sets are needed to be calculated. Then, from

Equation 2, s_2 is easily calculated from $s_1$by squaring it (i.e. s_1^2). Similarly, for BCH2 is designed to correct upto t = 3 error bits, j ranging from one to six (i.e., 2t = 6). So, the numbers of syndrome components calculated are six (i.e. s_1 to s_6). BCH3 can correct upto five error bits, so j varies from one to ten. Hence the syndrome calculations are done from s_1 to s_{10}. For BCH4 the calculated syndromes are from s_1 to s_{14}, since the j ranges from one to fourteen. In all cases, odd syndrome components are first calculated. Then, even components are found by squaring them and the transmitter can be achieved.

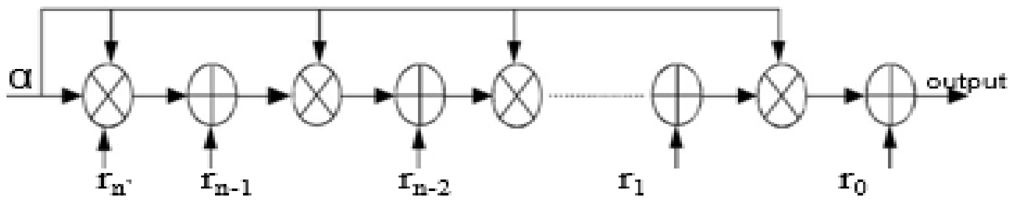


Figure 4. Syndrome block diagram [22]

3.2. ERROR LOCATOR POLYNOMIAL CALCULATOR DESIGN

The secondary stage of the BCH decoder is to determine the error locator polynomial. The syndrome calculator outputs are used to generate this polynomial [18]. The relation the between syndrome calculator and error locator is expressed as:

$$\sum_{j=0}^{t} s_{t+i-j}\Lambda_j = 0 \qquad (3)$$

where s is the syndrome and $\Lambda$is the error locator coefficient. Then, the error locator polynomial can be expressed as $\Lambda(x) = \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 + \cdots + \Lambda_t x^t$[19]. To deduce the error locator polynomial Peterson, Berlekamp and Euclidean are the most prominently used decoding algorithms in hardware the BCH decoder [23]. The Peterson algorithm is known to be the best choice, when the error correcting capability is less than or equal to three, since its computation is simple and less costly, especially in hardware [24]. Euclidean algorithm is advantageous in terms of speed and can be used in design where speed is the major objective [19]. Berlekamp algorithm has less hardware complexity in calculating the error locator polynomial and is used in the design where area is a limiting factor [24]. In designing BCH1 and BCH2 the Peterson error locator computation algorithm [19] has been used, since the number of errors to be corrected (i.e., target t) for these designs are less than or equal to three. As the Peterson error locator polynomial computation is complex for more than three errors Simplified Truncated Inverse Berlekamp Massey algorithm (SiBM) [25] is used for designing BCH3 and BCH4 as it provides better speed with less area overhead for determine error location polynomial coefficient.

According to the Peterson algorithm [19], $\Lambda(x)$ is computed directly and the degree of $\Lambda(x)$ is equal to the number of error bits occurred. For $t = 3$ as an example, Λ (i.e., coefficients of the error locator polynomial) is represented as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ s_2 & s_1 & 1 \\ s_4 & s_3 & s_2 \end{bmatrix} * \begin{bmatrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{4}$$

For BCH1, $\Lambda_0 = 1$ and $\Lambda_1 = s_1$. For BCH2, deleting the two rightmost columns and two bottom rows of matrix leads to a singular matrix, and solving the corresponding equation yields: $\Lambda 0 = 1, \Lambda 1 = s1, \Lambda_2 = (s_2 s_3 + s_5)/(s_1^3 + s_3)$, and $\Lambda_3 = (s_1^3 + s_3) + (s_1 \Lambda_2)$ [26].

The proposed decoding paths BCH3 and BCH4 have $t = 5$ and $t = 7$, respectively. Therefore, Peterson algorithm is inappropriate due to its poor hardware scalability. Therefore, the Simplified truncated inverse Berlekamp Massey algorithm (SiBM) has been used to find the coefficients of the error polynomial. SiBM [25] is known to provide higher decoding speed with less area overhead to determine the error locator polynomial coefficients. Its efficiency comes from two factors including: 1) for binary BCH codes, the Berlekamp-Massey algorithms' odd iterations can be skipped, and 2) since the error value is always binary, one error evaluator polynomial is not required. SIBM consists of an error locator polynomial and discrepancy polynomial block. Appending the error locator polynomial to the discrepancy polynomial will result in the disappearance of the error evaluator polynomial after the iteration is completed.

The SiBM processing element (PE) is capable of updating the coefficients of both error locator polynomial and discrepancy polynomial simultaneously. The odd coefficient of the extended polynomial is not capable to interact with the even coefficients. Therefore, PEs is placed as nearly two independent layers. The upper layer affects the lower layer through dependency, while the lower layer has no effect on the upper layer. The lower layer variables have little effect on the final error locator polynomial and can be simply shifted out. This further reduces the computational time and optimize SiBM algorithm.

3.3. CHIEN SEARCH BLOCK DESIGN

The final step of BCH decoder is to find the error location(s) through the Chien search method. Error location is obtained by finding the roots of the error locator polynomial [19]. The roots are searched as follows:

1. For each power of α (i.e., primitive element) for $i = 0\ to\ n - 1$, α^i is taken as the test root
2. Calculate the polynomial coefficients of the current root using coefficients of the past iteration using $\alpha^j = \alpha^{j-1} \cdot \alpha^1$ during the j_{th} iteration.
3. Calculate the sum of the polynomial coefficients.
4. When the sum is zero, an error bit is present at that location.

The factorization method [27] is used to reduce the complexity of the conventional method. It allows designing another form of the circuit of the Chien search that minimizes a large number of the used logic gates in the circuit. This block finds the error location depending on the error locator polynomial. Figure 5 shows a Chien search block diagram for t = 5 case as an example. In this figure, one is multiplied with α^t and then the resultant is multiplied with the Λ_t. The obtained result is XORed with Λ_{t-1}, again this resultant is multiplied α^t, and then XORed with Λ_{t-2} and the process repeats until Λ_0. The above said process iterates until α^0. In this way, the roots of the error location determine the error location.

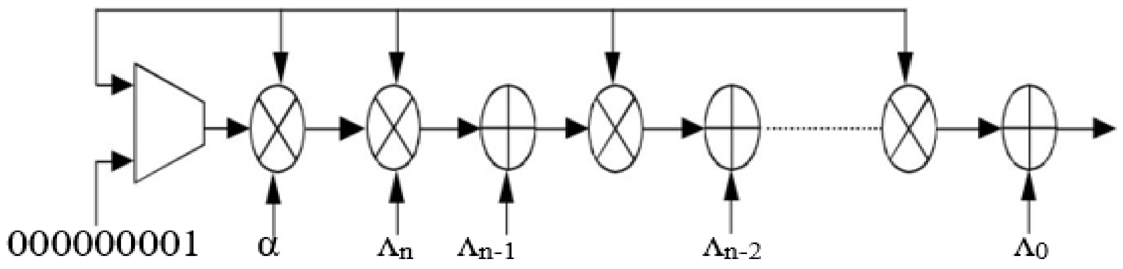


Figure 5. Chien Search block diagram for t = 5 case [19]

3.4. PARALLEL ADAPTIVE BCH DECODING FOR REDUCED DECODING

In addition to the serial decoding (i.e., decoding one codeword at a time), the proposed adaptive multi-stage BCH decoder can be utilized to correct multiple words in parallel provided incoming codewords have different n_{EEB}. Even though one or more decoding paths are occupied, if there is an unoccupied path with minimum $t \geq n_{EEB}$ of the incoming codeword, that path can start decoding it in parallel for further reduction in decoding latency. If there is no idle decoding path that can be used to decode the incoming codeword, it is temporarily stored in the storage buffer until an appropriate vacant decoding path becomes available.

Figure 6 illustrates an example of the proposed parallel decoding process. As seen in the first clock cycle ($\varphi = 1$), the n_{EEB} of the incoming codeword is 3 and the decoding path with the minimum $t \geq n_{EEB}$ is BCH2 (i.e., t = 3). So, BCH2's availability is checked and it is currently idle. Therefore, the word is given to BCH2 to be decoded. In $\varphi = 5$, another word is read from the memory, which is estimated to contain two error bits. It is given to BCH3 to be decoded, because BCH2 is currently decoding the first word and the next higher error correcting decoder path available is BCH3. Similarly, the next incoming codeword is estimated to have five error bits and is read from the memory in $\varphi = 10$.

This codeword is read from the memory in $\varphi = 10$. This codeword is given to BCH4, since BCH3 is busy decoding the second word. As seen from the figure for $\varphi = 16$ when the codeword with $n_{EEB} = 7$ is coming for decoding, it is stored in the storage buffer instead, since BCH4 is currently busy decoding another word. When the respective decoder becomes available the word stored in storage buffer is fetched to be decoded.

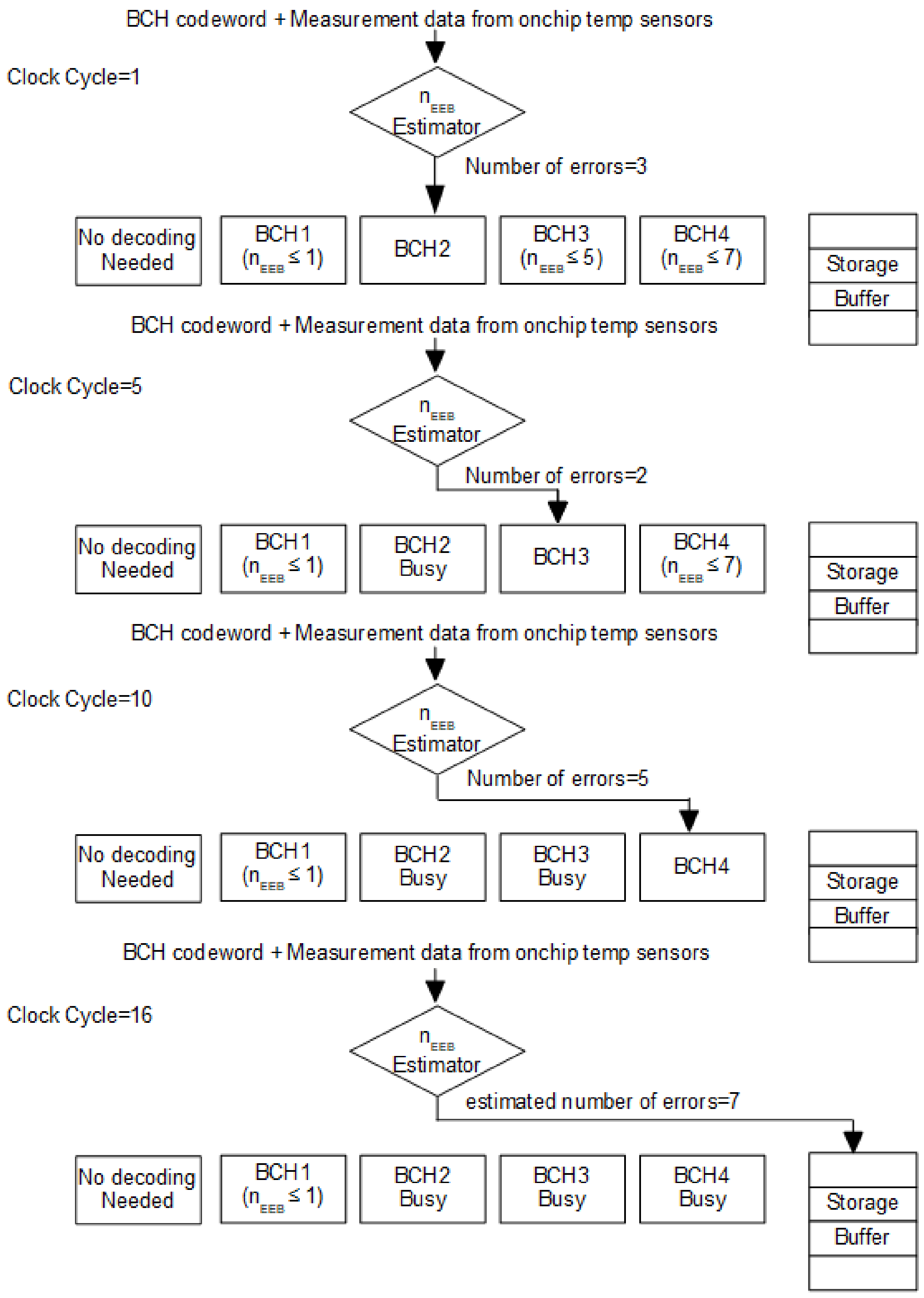


Figure 6. An example of the proposed parallel decoding

4. PERFORMANCE EVALUATION

To quantitatively demonstrate and verify the decoding performance of the proposed adaptive multi-path BCH decoder approach, the presented 7-error-correcting (511, 448) adaptive BCH decoder (i.e., $m = 9$) with $p = 4$ decoding paths with target $t = 1, 3, 5$ and 7 has been designed in Verilog HDL (Hardware Description Language) and verified. Simulation and synthesis have been carried out using Xilinx ISE tool on a Virtex5 FPGA (target device: XC5VLX30). Resource utilization and decoding latency results of the FPGA prototype are summarized in Table 1.

In FPGA, reconfigurable resources are grouped into slices which contain a set of LUTs (Look-Up Tables), flip-flops and multiplexers. These LUTs represent a group of logic gates that are hard-wired on the FPGA and stores the output depending on the input. Thus, these LUTs provide the fastest way to retrieve the output when needed. A flip-flop circuit is used for change of state and stores a single bit of data. A slice register is the group of flip-flops used to store a data word. A register has a clock, enable pin, input and output data ports. For every clock cycle depending on the input, the output is updated and stored. It should be noticed that as the error correcting capability of the decoder increases the hardware resources utilized also increases leading to a large area overhead. As in this Table 1, hardware implementations of different BCH decoder paths show poor scalability as t increased.

Therefore, a decoding path with minimum $t \geq n_{EEB}$ should be used to decode. This would adaptively minimize the overall decoding time. The area overhead of the proposed adaptive 4-path BCH decoder compared to the static BCH decoder designed to tolerate maximum t = 7 is calculated using the number of occupied slices used in both designs. The static decoder which has only one decoding path of BCH4 utilizes 1, 776 FPGA slices and the proposed adaptive 4-path BCH decoder design utilizes 2, 628 slices. Therefore, the area overhead of the proposed adaptive 4-path BCH decoder is *only 47.97%*, even though it has 4 physically separate decoding paths. This area overhead is relatively small, since decoding paths with smaller t (i.e., BCH1, BCH2 and BCH3) have considerably lower hardware complexities and require only 852 (i.e., 47.97% of BCH4's 1,776) additional slices to be realized.

Table 1. FPGA resource utilization and decoding latency of four decoding paths in the proposed adaptive multi-path BCH decoder

	BCH1 (t = 1)	**BCH2 (t = 3)**	**BCH3 (t = 5)**	**BCH4 (t = 7)**
# of Slice Registers	49	93	220	444
# of Slices LUT	107	586	1817	2536
# of LUT FF used	44	98	203	397
# of Occupied Slices	44	220	558	1776
Decoding Latency	7.21 ns	12.73 ns	28.59s	39.33s

4.1. SERIAL DECODING PERFORMANCE EVALUTION

In this section, serial decoding performance of the prototype adaptive error-correcting (511, 448) BCH decoder (i.e., m = 9) with p = 4 decoding paths with target t = 1, 3, 5 and 7 (adaptive 4-path BCH decoder, in short) will be evaluated. In the proposed adaptive multi-path BCH decoding approach, distributed onchip temperature sensors in

3DHP provide temerature measurement data for DRAM die to the n_{EEB} estimator as shown in Figure 3. Then, the n_{EEB} estimator calculates the p_{BE} of the codeword that is being read from DRAM with a certain user-provided confidence level as previously discussed in Section 3.

For variable p_{BE} , the probability of having k number of error bits occurred in n-bit BCH codeword can be calculated using binomial equation, $p(k) = n_k\, k_{BE}\, (1 - p_{BE})^{nk}$. For the prototype adaptive 4-path BCH decoder, the distribution of six different error probabilities can be calculated as follows:

1) P(0), where no decoding is necessary, since n_{EEB}= 0;
2) P(1), where BCH1 decoding path is used to correct single bit error;
3) P(2 ∧ 3), where BCH2 decoding path is used to correct 2 to 3 error bits;
4) P (4 ∧ 5), where BCH3 decoding path is used to correct 4 to 5 error bits;
5) P (6 ∧ 7), where BCH4 decoding path is used to correct 6 to 7 error bits.
6) P (> 7) = 1 − $^{\mathbf{P}}{}_{i=\mathbf{1}}$ P (i), which means uncorrectable.

Figure 7 depicts the distribution of error probabilities for various p_{BE} values, which are arbitrarily chosen as 0.04, 0.004, 0.0004 and 0.0004. This graph illustrates implications of p_{BE} on the utilization of 4 decoding paths. For example, when p_{BE} = 0.04, the probability of codeword having more than seven error bits per codeword is dominant; thereby, almost all codewords are uncorrectable. On the other hand, p_{BE} = 0.00004 yields codewords with no error bits (i.e., codewords with no decoding is necessary) almost 98% of the time. Therefore, no useful error correction decoding happens in both of these extreme cases and the utilization of decoding paths is extremely low. The p_{BE} = 0.004 case gives higher utilization of 4 decoding paths as P (0) ≈ 13% (i.e., no error - no decoding needed) and P

(> 7) ≈ 0% (i.e., almost no uncorrectable codewords). In this case, BCH2 with t = 3 shows the highest utilization of 45% among 4 decoding paths.

4.1.1. Average Decoding Latency for Various Bit Error Probabilities. Table 2 shows the average decoding latency of the proposed adaptive 4-path BCH decoder for variable p_{BE}. From Figure 7, for $p_{BE} = 0.04$ case, the excepted number of error bits is mostly more than seven. Therefore, codewords containing more than seven errors are not correctable by the proposed design. Hence, its average decoding latency is near zero as almost no decoding occurs. The other extreme case of $p_{BE} = 0.00004$ is similar, as the decoding paths are under-utilized as the most of codewords are error-free requiring no decoding. Notably, the other two intermediate cases (i.e., $p_{BE} = 0.004$ and 0.0004) can be used to represent the error distribution of 3DHP with hotspot induced p_{BE} variation. The p_{BE}= 0.004 case has an order of magnitude higher p_{BE} when compared with $p_{BE} = 0.0004$ case, so it can be used as an exemplary p_{BE} for hotspots in 3DHP. Also, the other has 10 times lower p_{BE}, so it can be used an exemplary p_{BE} for the other area not affected by the hotspots.

Table 2. Average decoding latency (ADL) for different p_{BE} values. For $p_{BE} = 0.04$, P (>7) ≈ 100%, which means almost all codewords are uncorrectable and not decoded. For the other extreme, $p_{BE} = 0.00004$, 98% of codewords are error-free and do not need decoding

p_{BE}	**0.04**	**0.004**	**0.0004**	**0.00004**
nEEB	Highest	Mid-high	Mid-low	Lowest
max P (t)	P (> 7) ≈ 100%	P (2 ∧ 3) = 46%	P (0) = 81.5%	P (0) = 98%
ADL	≈ 0	11.50 ns	1.44 ns	0.15 ns

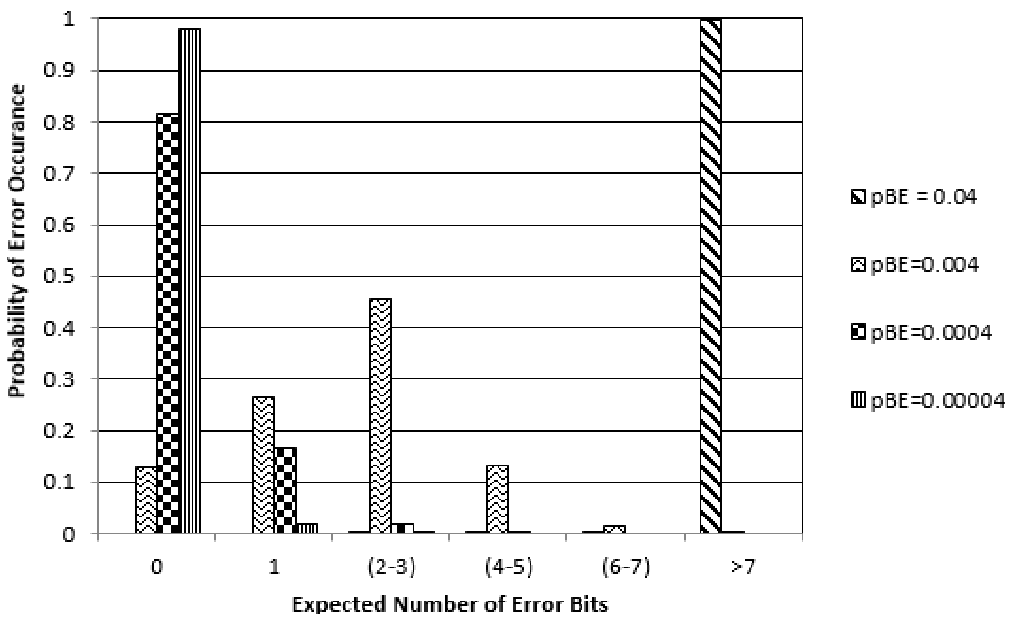


Figure 7. Distribution of error probabilities for variable p_{BE} values

Table 3. Cumulative error coverage for various p_{BE} values

p_{BE}	P (0)	P (1)	P (2 ∧ 3)	P (4 ∧ 5)	P (6 ∧ 7)	P (> 7)
0.04	8.7e-10	1.9e-08	1.6e-06	4.01e-05	0.00047	1
0.004	0.1289	0.3936	0.84945	0.9820	0.99878	1
0.0004	0.8151	0.9817	0.999938	0.99999	0.99999	1
0.00004	0.9797	0.9997	0.999997	0.999998	0.99999	1

As clearly shown in Figure 7, the average decoding latency for codewords read from the area not affected by hotspots (i.e., $p_{BE} = 0.0004$) is 1.14 ns, which is considerably lower than 11.50 ns decoding latency of the hotspot-affected codewords. Notably, even this slowest 11.50 ns decoding latency is significantly lower than the worst-case (i.e., t = 7) static BCH decoding latency of 39.33 ns reported in Table 1.

4.1.2. Cumulative Error Coverage. The proposed adaptive 4-path BCH decoder can correct up to 7 error bits per word. Table 3 shows the cumulative error coverage for variable p_{BE}. As per the results shown in the table maximum error coverage is found near more than seven errors for high p_{BE} value. As the p_{BE} value decreases error coverage is maximum for small number of errors.

There is tradeoff between error coverage and latency. The staged BCH decoder for any input irrespective of varying p_{BE} offer maximum error coverage with maximum latency overhead. Whereas the adaptive multi-stage BCH decoder relies on p_{BE} and offers required error coverage with the reduced latency. Therefore the proposed model can provide efficient error coverage with less delay for varying temporal and spatial changes.

4.2. PARALLEL DECODING PERFORMANCE

As discussed in Section 3.4, multiple independent decoding paths given in the proposed adaptive multi-path BCH decoder can be utilized to decode multiple codewords from DRAM in parallel. The performance of the proposed parallel decoding technique will be extensively evaluated using the adaptive 4-path BCH decoder design as an example in this section. A cycle-accurate simulator has been implemented in Matlab to generate simulation results for the proposed parallel decoding technique. Bit error variations caused by hotspots are simulated by introducing the following user-provided simulation parameters:

1. p_{BEH} : the increased bit error probability due to hotspots.
2. p_{BEC} : the baseline bit error probability unaffected by hotspots.
3. f_{hot}: the relative frequency of codewords subject to p_{BEH} .

4. f_{cold}: the relative frequency of codewords subject to p_{BEC} , where $f_{hot} + f_{cold} = 100\%$.

   $size_{buf}$: the storage buffer size

Accordingly, each respective decoder's status is checked for its availability. If the decoder is not available then the next available higher error correcting decoder (i.e., the decoder which can correct more number of errors when compared to the required no of error correction) is selected for decoding. When there is no available decoder with $t \geq m_{EEB}$, then the respective word is stored in the storage buffer. If the storage buffer is completely occupied then the word is not read from the memory. Table 4 shows the average decoding latency for various p_{BEH} /p_{BEC} , f_{hot}/f_{cold} and $size_{buf}$ values chosen arbitrarily. It can be noticed that decoding latencies differ with $size_{buf}$ and f_{hot}/f_{cold} ratio. Decoding latency increases with the increases in the number of words read from hotspot region.

Table 4. Parallel decoding simulation results showing the average decoding latency by varying pBE_H /pBE_C , f_{hot}/f_{cold} and $size_{buf}$

pBE_H /pBE_C	$size_{BUF}$	f_{HOT}/f_{COLD} 40/60	f_{HOT}/f_{COLD} 60/40	f_{HOT}/f_{COLD} 80/20
0.003/0.002	4	15.7 ns	15.8 ns	16.8 ns
	8	15.6 ns	15.69 ns	16.4 ns
	16	15.8 ns	15.8 ns	16.2 ns
0.009/0.002	4	17.1 ns	20 ns	23.3 ns
	8	16.8 ns	19.7 ns	23.1 ns
	16	16.6 ns	19.63 ns	22.8 ns
0.011/0.005	4	23.2 ns	26.9 ns	30.2 ns
	8	22.9 ns	26.7 ns	29.8 ns
	16	22.5 ns	26.5 ns	29.6 ns

The obtained results also indicate that the average decoding latency for the proposed adaptive multi-path BCH decoder leveraging the parallel decoding technique has less decoding latency when compared to the static BCH decoder with fixed t = 7, which has a constant 39.33 ns decoding latency. Thus, it can be concluded that the proposed adaptive 4-path BCH decoder can achieve significantly lower decoding latency ranging from 15.7 ns to 29.6 ns for p_{BEH} /p_{BEC} , f_{hot}/f_{cold} and $size_{buf}$ values chosen with are a overhead of 47.97%.

5. CONCLUSION

In this paper, a novel adaptive multi-path BCH decoder design approach is proposed and validated to address the bit error variation issue caused by hotspots in 3DHP. The proposed design has multiple decoding paths with variable decoding latency and area trade-off. For each word read from DRAM, thermal gradient data from the on-chip temperature sensors is utilized to estimate the expected number of error bits. Then, the fastest possible decoding path which is able to correct the expected number of error bits is adaptively selected to reduce the overall decoding time. Also, a parallel decoding approach leveraging the multiple independent decoding paths of the proposed decoder design is also proposed and validated in this work. To clearly evaluate the latency reduction performance and area overhead of the proposed approach, an adaptive 4-path BCH decoder has been implemented in FPGA hardware. Then, its serial and parallel decoding performances along with area overhead have been extensively evaluated. The proposed adaptive 4-path BCH decoder can achieve significantly reduced average decoding latency ranging from 15.7 ns to 29.6 ns for variable p_{BEH} /p_{BEC} , f_{hot}/f_{cold} and $size_{buf}$ value sets chosen with area overhead of 47.97%.

BIBLOGRAPHY

[1] S. Naffziger, “Technology impacts from the new wave of archi-tectures for media-rich workloads,” in VLSI Technology (VLSIT), 2011 Symposium on, 2011, pp. 6–10.

[2] W. Liao, F. Li, and L. He, “Microarchitecture level power and thermal simulation considering temperature dependent leakage model,” in Low Power Electronics and Design, 2003. ISLPED ’03. Proceedings of the 2003 International Symposium on, 2003, pp. 211–216.

[3] J.-H. Ahn, B.-H. Jeong, S.-H. Kim, S.-H. Chu, S.-K. Cho, H.-J. Lee, M.-H. Kim, S.-I. Park, S.-W. Shin, J.-H. Lee et al., “Adaptive self-refresh scheme for battery operated high-density mobile DRAM applications,” in Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian. IEEE, 2006, pp. 319–322.

[4] W. Yun, K. Kang, and C.-M. Kyung, “Thermal-aware energy minimization of 3D-stacked L3 cache with error rate limitation,” in Circuits and Systems (ISCAS), 2011 IEEE International Sym-posium on, 2011, pp. 1672–1675.

[5] T. Hamamoto, S. Sugiura, and S. Sawada, “On the retention time distribution of dynamic random access memory (DRAM),” Electron Devices, IEEE Transactions on, vol. 45, no. 6, pp. 1300– 1309, 1998.

[6] M. Cho, Y. Kim, D. Woo, S. Kim, M. Shim, Y. Park, W. Lee, and B.-I. Ryu, “Analysis of Thermal Variation of DRAM Retention Time,” in Reliability Physics Symposium Proceedings, 2006. 44th Annual, IEEE International. IEEE, 2006, pp. 433–436.

[7] M. Hashimoto and R. Baumann, “Investigation of cell leakage and data retention in eDRAM,” in Solid-State Circuits Conference, 2000. ESSCIRC’00Proceedings of the 26rd European, 2000, pp. 356–359.

[8] M. Guan and L. Wang, “Temperature aware refresh for DRAM performance improvement in 3D ICs,” in Quality Electronic Design (ISQED), 2015 16th International Symposium on, 2015, pp. 207–211.

[9] M. Richter, K. Oberlaender, and M. Goessel, “New Linear SEC-DED Codes with Reduced Triple Bit Error Miscorrection Probability,” in On-Line Testing Symposium, 2008. IOLTS’08. 14th IEEE International, 2008, pp. 37–42.

[10] P.-Y. Chen, C.-L. Su, C.-H. Chen, and C.-W. Wu, “Generalization of an Enhanced ECC Methodology for Low Power PSRAM,” IEEE Trans. Comput., vol. 62, no. 7, pp. 1318–1331, 2013.

[11] Z. Chishti, A. Alameldeen, C. Wilkerson, W. Wu, and S.-L. Lu, “Improving cache lifetime reliability at ultra-low voltages,” in Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on, 2009, pp. 89–99.

[12] K. K. Prashanthi Metku, Ramu Seva and M. Choi, “Multi-Stage BCH Decoder to Mitigate Hotspot-Induced Bit Error Variation,” in 2015 International SoC Design Conference, pp. 89–90.

[13] W. Peterson, “Encoding and error-correction procedures for the Bose-Chaudhuri codes,” IRE Transactions on Information Theory, vol. 6, no. 4, pp. 459–470, 1960.

[14] E. Berlekamp, “On decoding binary Bose-Chadhuri Hocquenghem codes,” IEEE Trans. Inf. Theory, vol. 11, no. 4, pp. 577–579, 1965.

[15] J. Massey, “Step-by-step decoding of the Bose-Chaudhuri- Hocquenghemcodes,” IEEE Trans. Inf. Theory, vol. 11, no. 4, pp.580–585, 1965.

[16] R. Chien, “Cyclic decoding procedures for Bose- Chaudhuri-Hocquenghem codes,” IEEE Trans. Inf. Theory, vol. 10, no. 4, pp. 357–363, 1964.

[17] G. Forney, “On decoding BCH codes,” IEEE Trans. Inf. Theory, vol. 11, no. 4, pp. 549–557, 1965.

[18] H. Kristian, H. Wahyono, K. Rizki, and T. Adiono, “Ultrafast scalable BCH decoder with efficient-Extended Fast Chien Search,” in Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 4, 2010, pp. 338–343.

[19] X. Zhang and Z. Wang, “A Low-Complexity Three-Error Correcting BCH Decoder for Optical Transport Network,” IEEE Trans. Circuits Syst. II, vol. 59, no. 10, pp. 663–667, 2012.

[20] Y.-M. Lin, H.-C. Chang, and C.-Y. Lee, “Improved High Code Rate Soft BCH Decoder Architectures With One Extra Error Compensation,” IEEE Trans. VLSI Syst., vol. 21, no. 11, pp. 2160–2164, 2013.

[21] Lempel, “Analysis and synthesis of polynomials and sequences over,” IEEE Trans. Inf. Theory, vol. 17, no. 3, pp. 297–03, 1971.

[22] N. Ahmadi, M. Sirojuddiin, A. Nandaviri, and T. Adiono, “An optimal architecture of BCH decoder,” in Application of Information and Communication Technologies (AICT), 2010 4th International Conference on, 2010, pp. 1–5.

[23] Y. Liu, “Channel Coding,” Henan Science and Technology Press,1992.

[24] X. Qi, X. Ma, D. Li, and Y. Zhao, “Implementation of accelerated BCH decoders on GPU,” in Wireless Communications & Signal Processing (WCSP), 2013 International Conference on, 2013, pp. 1–6.

[25] M. Yin, M. Xie, and B. Yi, “Optimized algorithms for binary BCH codes,” in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on, 2013, pp. 1552–1555.

[26] X. Zhi-yuan, L. Na, and L. Le-le, “New decoder for triple error correcting binary BCH codes,” in Industrial Electronics and Applications, 2008. ICIEA 2008. 3rd IEEE Conference on, 2008, pp. 1426–1429.

[27] E.-H. El Idrissi Anas, E. Rachid, and H. Lamari, “A low power error detection in the Chien Search Block for Reed Solomon code,” in Complex Systems (ICCS), 2012 International Conference on, 2012, pp. 1–3.

II. NOVEL AREA-EFFICIENT NULL CONVENTION LOGIC ON CMOS AND GATE DIFFUSION INPUT (GDI) HYBRID METHODOLOGY

ABSTRACT

A Null convention logic (NCL) is a promising delay insensitive paradigm for constructing asynchronous circuits. Traditionally, NCL circuits are implemented utilizing complementary metal oxide semiconductor (CMOS) technology that has large area overhead. To address this issue, a HYBRID methodology is introduced for realizing NCL circuits in this paper. The proposed approach utilizes both CMOS and gate diffusion input (GDI) techniques to significantly reduce the area. Compared with the conventional static CMOS NCL counterpart, the HYBRID implementation of an NCL up counter demonstrate an average of 10% reduction in the transistor count.

1. INTRODUCTION

The clocked synchronous paradigm currently dominates the semiconductor design industry [1]. However, there are major drawbacks of this synchronous approach, including critical timing analysis and clock skew issues [2]. Typically, a precise clock distribution network is used to address these limitations, which is a tedious and complex task. Moreover, with the decreasing feature size, power consumption of clock distribution network is found to be rapidly increasing, which is a major limiting factor for emerging low power semiconductor industry [3]. Since asynchronous designs consumes less power, produce less noise and electromagnetic interference (EMI) than their synchronous counterparts, there is renewed interest in this field [4].

Asynchronous circuits are characterized into two classifications: bounded-delay and delay-insensitive (DI) models [3]. Bounded-delay models consider that the both gate and wire delays are bounded and therefore, require extensive timing analysis to determine the delay [1]. On the other hand, DI circuits assume both interconnects and logic elements delay are unbounded and wire forks within the components are isochronic [5]. However, wires connecting the components do not adhere to this isochronic fork assumption, ensuring the correct operation regardless on the input availability. Hence, DI circuits require little timing analysis and yield average case performance rather than the worst-case performance of bounded-delay and traditional synchronous paradigms [6].

Literature provides several DI paradigms such as Seitz's, DIMS, Anantharaman's, Singh's, and David's Phased Logic and NULL Convention Logic [1]. Most of these DI methods (Seitz's, DIMS, Anantharaman's, Singh's, David's, Phased Logic) either depends on C-element or synchronous design to achieve DI. More elaborate description of these DI methodologies can be found in [7]. Conversely, NCL methodology uses a library of hysteresis state holding functionality gates to attain DI. These gates enable transistor level optimization, which help in reducing the overall circuit area [8]. Hence, NCL is the best alternative for integrating asynchronous digital design into the predominantly synchronous semiconductor design industry.

The NCL paradigm consists of 27 hysteresis state holding logic gates [2]. These gates are traditionally implemented using one of the CMOS techniques: static, semi-static, differential or dynamic methods. Detailed information regarding these approaches can be found in [8]. Among these methods, the static CMOS method is most commonly used as it results in less leakage and noise compared to other CMOS techniques [9]. However, the main drawback of static CMOS implementation is the area overhead. The results indicate that the area occupied by static CMOS NCL implementation is approximately 1.5-2 times the equivalent synchronous design [7]. To address this drawback, this paper proposes a HYBRID technique for designing NCL circuits. The proposed approach integrates both CMOS and gate diffusion input techniques to realize NCL designs.

Gate diffusion input methodology is a low power design technique that utilizes only two transistors to implement different functions [1]. The input configuration required to implement various function can be found in [9-12]. Hence, by applying GDI methodology for realizing NCL gates, total transistor count is reduced, which in turn reduces switching power. However, the biggest drawback of GDI methodology is the voltage drop at the output which results in performance degradation [12-15]. Hence, a new HYBRID methodology that can address both of these limitations, voltage drop of GDI and area overhead of CMOS is proposed in this work.

The aim of this work is to utilize both CMOS and GDI techniques to design NCL circuits. This approach where in both CMOS and GDI based NCL gates are used to design NCL circuits helps in reducing the transistor count when compared to the conventional static CMOS approach. To validate the performance of the proposed approach, a variety of NCL up-counter increment (NUI) circuits were realized and compared with the static CMOS methodology. The proposed approach shows a minimum of 4% reduction in the transistor count when compared with the static CMOS approach.

The rest of the paper is organized as follows: Section 2 discusses the preliminaries and review of NCL and GDI. Section 3 presents the design description of the HYBRID methodology. Section 4 illustrates the simulation results, followed by conclusion in Section 5.

2. PRELIMINARIES AND REVIEW

2.1. NULL CONVENTION LOGIC

NCL is a clockless DI model that works correctly regardless of input accessibility. It is a self timed logic model where both data and control are integrated to a single signal and communication is accomplished through local handshaking [1]. To provide synchronization DATA and NULL states are used which are obtained using dual or quad-rail logic. A dual-rail signal, D utilizes two wires D0 and D1 to represent values DATA0, DATA1 and NULL [4] as shown in Figure 1. The NULL state (D0 = 0, D1 = 0) symbolizes that D is not available. The DATA0 state (D0 = 1, D1 = 0) and DATA1 state (D0 = 0, D1 =1) represents Boolean logic 0 and 1 [8]. These two rails are mutually exclusive and cannot be asserted at the same time. This means that if both the rails are high, the state is known as an invalid/illegal state.

The framework of NCL system is shown in Figure 2. As observed from the figure, combinational logic (CL) is always sandwiched between two DI registers and these adjacent DI registers communicate through request and acknowledge signals ki and ko. NCL utilizes a special set of logic element known as threshold gates for realizing the combinational logic, DI registers and competition detection circuits [3].

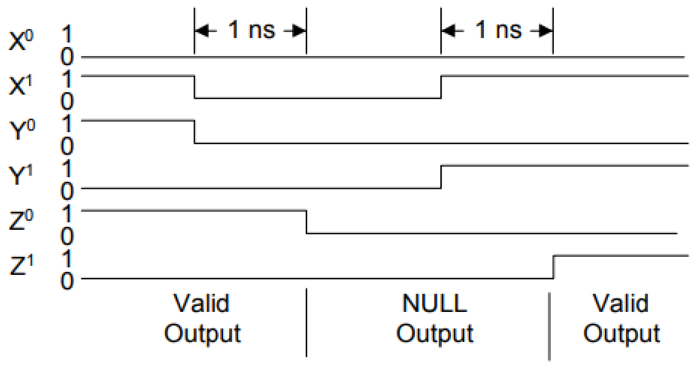


Figure 1. Dual-rail representation of NCL AND function: Z = X • Y: initially X=DATA1 and Y=DATA0, so Z=DATA0; next X and Y both transition to NULL, so Z transitions to NULL; then X and Y both transition to DATA1, so Z transitions to DATA1 [7]

There are 27 threshold gates and the primary type of threshold gate depicted in Figure 3(a), is known as THmn, where $1 \leq m \leq n$ [2]. Here, n represents the number of inputs and m denotes the number of inputs that need to be asserted for the output to be asserted. The secondary type of threshold gate illustrated in Figure 3(b) is refered as a weighted threshold gate, denoted as THmn Ww1w2…wR. The constant equation is w1, w2…wR > 1, where w1, w2…wR are the integer weights of input1, input2 … inputR, respectively [5]. These threshold gates have built-in hysteresis behavior to ensure DI. Hysteresis in NCL ensures that two DATA wavefront are not overwritten and are always separated by a NULL wavefront.

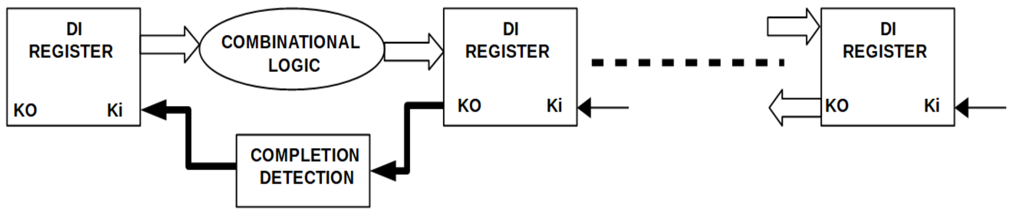


Figure 2. NCL system framework [3]

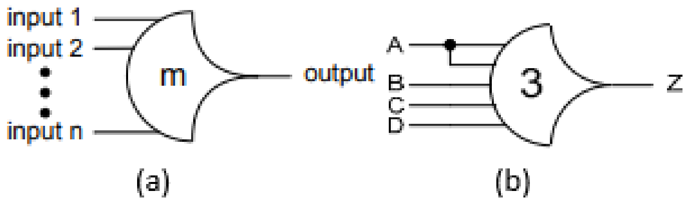


Figure 3. (a) THmn threshold gate (b) TH34w2 threshold gate [3]

The general algebraic expression of an NCL gate is the combination of set and hold equations. The set equation defines the functionality of the gate and the hold equation determines till when the gate should be asserted once it is asserted. The set equation i.e. the functionality of each NCL gate is presented in [1] whereas; the hold equation remains the same for every gate, which is simply OR-ing, all the inputs. Therefore, the general equation for an NCL gate is given by Z = set + (Z- • hold), where Z- is the previous output value and Z is the current value. Prevailing methodologies utilized for realizing NCL circuits are static and semi-static CMOS technology. Figure 4 and Figure 5 depicts the static and semi-static CMOS implementation of TH23 gates.

As depicted in Figure 3(b), the semi-static implementation only requires set and set' expressions are utilized to realize TH23 gate. To achieve hysteresis, the semi-static implementation uses weak feedback inverters, which slows down the gate operation leading to large latency overhead. This limitation is addressed by using static CMOS implementation that utilizes pull-up (set) and pull-down networks (reset) as shown in the Figure 3(a). As observed from the figure, the additional circuitry is required to maintain the built-in hysteresis property of NCL gates. This leads to an area overhead where NCL designs are approximately 1.5 - 2 times larger than the equivalent synchronous designs [7].

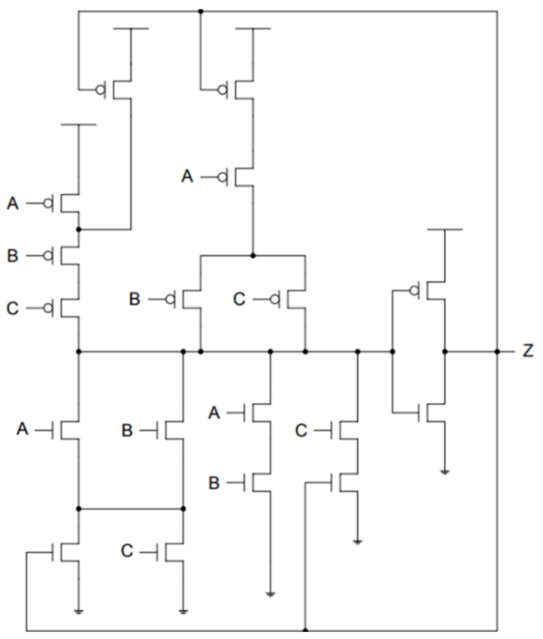


Figure 4. Transistor level realization of TH23 gate using Static CMOS methodology [7]

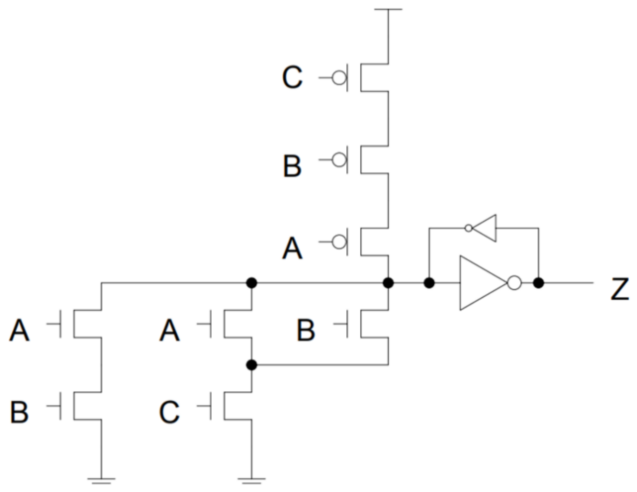
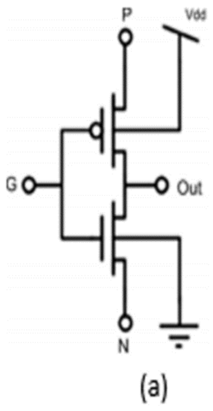


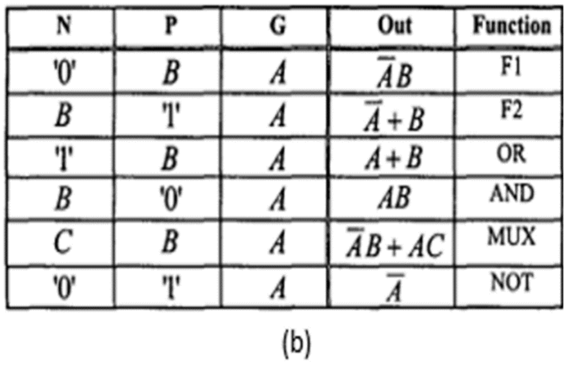
Figure 5. Transistor level implementation of TH23 gate using semi-static methodology [7]

Therefore, it is crucial to address this limitation so that NCL designs become viable alternative for synchronous design. This drawback is be addressed by utilizing a low power design methodology called GDI.

2.2. GATE DIFFUSION INPUT

GDI is a low power design technique commonly used in synchronous design to reduce area and dynamic power consumption [1]. The structure of basic GDI cell is depicted in Figure 6. It has three inputs G (common gate input of both the nMOS and the pMOS), P (input to the source/drain of the pMOS), N (input to the source/drain of the nMOS). The bulks of nMOS and pMOS transistors are constantly connected to GND and V_{DD}, respectively [6].





N	P	G	Out	Function
'0'	B	A	$\overline{A}B$	F1
B	'1'	A	$\overline{A}+B$	F2
'1'	B	A	$A+B$	OR
B	'0'	A	AB	AND
C	B	A	$\overline{A}B+AC$	MUX
'0'	'1'	A	$\overline{A}$	NOT

(b)

Figure 6. (a) Basic GDI cell structure (b) Different functions input configurations [10]

Various logic functions of GDI cell for different input configurations are illustrated in Figure 6(b). Since, the pull-up and pull-down networks of these functions are not always connected to power supply (V_{DD}) and ground (GND), a voltage drop at the output is observed. This drawback is the biggest limitation of GDI methodology based implementation [9-15]. Similarly, by realizing NCL gates using GDI technique, voltage swings prevail in the circuit. Therefore, this work mainly focuses on addressing this issue such that GDI technique can be used for realizing NCL circuit.

The next section presents the mechanism for realizing the NCL gates using GDI methodology. The efficacy of the proposed approach is verified by realizing the several NUI circuits and comparing with the static CMOS methodology.

3. THE PROPOSED HYBRID METHODOLOGY

First, the mechanism to realize NCL gates using GDI methodology, also known as FNCL approach is discussed. Since this approach utilizes both F1 and F2 functions unique to GDI as shown in Figure 3, it is named as FNCL. The limitation of voltage drop of FNCL approach is also presented in this subsection. Finally, the HYBRID methodology, which utilizes both CMOS and GDI techniques to address the area overhead limitation of NCL designs, is described in detail.

3.1. REALIZATION OF NCL USING FNCL APPROACH (BASED ON F1 AND F2 FUNCTIONS OF GDI GATE)

To realize NCL threshold gates using FNCL approach, first the Boolean expression of THmn gate is factorized. Then, based on the factorized expression GDI function F1, F2 and MUX are utilized to implement the gate. As an example, steps for realization of TH22 gate is shown below:

Step 1: Factorized expression of TH22 gate is: Z = AB + Z (A+B)

Where, A, B are the inputs and Z is the output.

Step 2: The GDI functions are utilized to realize AND (AB) and OR (A+B) expressions. Among all the GDI functions, only F1 and F2 are utilized, as they demonstrate voltage drop for only one input combination compared to the others (AND, OR) functions. Hence, F1 and F2 are used to implement AB and A+B as shown in the Figure 7.

Step 3: The GDI MUX cell is used to determine final output i.e. whether to pass set data or hold data based on the previous results. The MUX is configured such that the output F1 cell (AB) and the O2 output of F2 cell (A+B) are fed to the sources of pMOS and nMOS as shown in Figure 7. This will allow to select set equation (AB) when Z=0 and hold data (A+B) when the Z is 1.

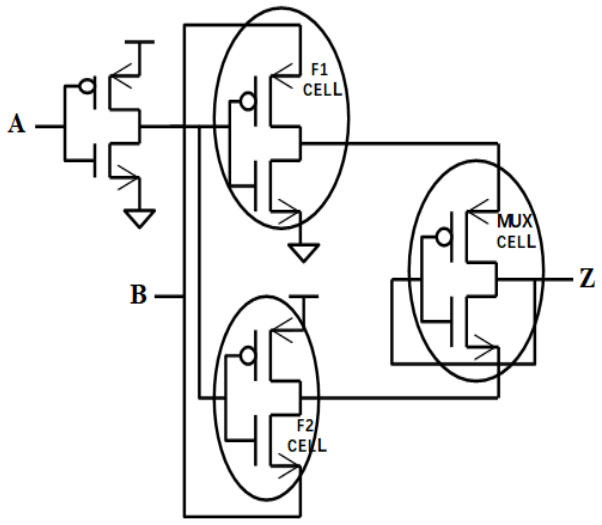


Figure 7. FNCL implementation of TH22 gate

Therefore, the proposed FNCL approach requires only eight transistors to implement TH22 gate. Compared to the static CMOS approach, a 20% reduction in the transistor count is observed. However, the main drawback of this approach is the performance degradation due to the substantial voltage drop at the final output. The mechanism to address this limitation is discussed in next subsection.

3.2. PERFORMANCE DEGRADATION OF FNCL APPROACH

The performance degradation is due to the different input configuration of the nMOS and pMOS transistors. A voltage drop of Vtp (threshold voltage of pMOS) and V_{DD} - Vtn (threshold voltage of nMOS) for pMOS and nMOS transistors are observed when their sources are not tied to V_{DD} and GND respectively [9].

To demonstrate this phenomenon, the simulation results of the proposed TH22 gate is illustrated in Figure 8. It is observed in Figure 8, when either of the inputs or any one of the inputs are low (A=0, B=0, Z=0), the outcome is Vtp rather than strong low '0'. This can be explained as follows: whenever A = 0, the voltage at the pMOS source of MUX cell is 0. Since, pMOS passes weak '0', the result would be Vtp. Conversely, when A and B are high, the output is V_{DD} without any voltage drop since pMOS passes strong '1'. Hence, three out of four input combinations result in voltage drop.

This voltage swing issue further escalates when two FNCL gates are interconnected. To validate this conclusion an NCL Full adder (FA) circuit is implemented using the FNCL approach. The structure of FA and its simulation results are depicted in Figure 9 and Figure 10. From Figure 10 it is observed that TH23 gates generates carryout and TH34w2 gates utilizes these results to generate the sum (S0, S1). When the FA circuit is simulated, voltage swings (for logic low) at carryout was ~ 0.1V, whereas for sum it was ~ 0.38V.

This increased voltage swing at the sum output is due to the voltage drop at TH23 gate being carried on to TH34w2 gate. Therefore, realizing the whole circuit using FNCL gates is not viable option. To address this limitation, novel HYBRIB approach is also proposed in this paper.

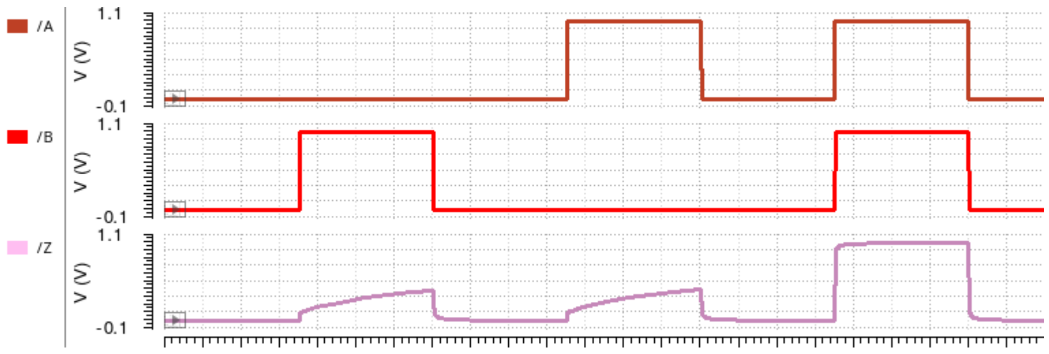


Figure 8. Simulation results demonstrating voltage drop of FNC TH22 gate

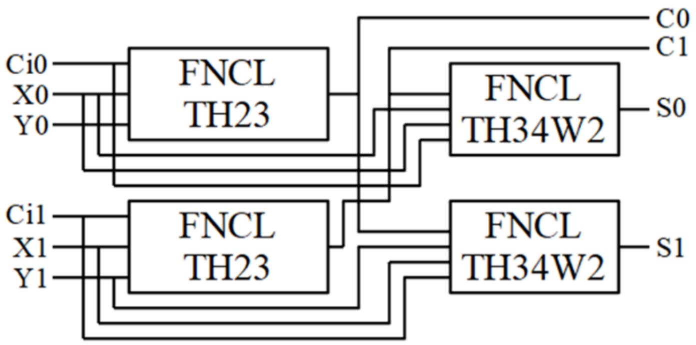


Figure 9. Structure of FNCL FA

This voltage swing issue further escalates when two FNCL gates are interconnected. To validate this conclusion an NCL Full adder (FA) circuit is implemented using the FNCL approach. The structure of FA and its simulation results are depicted in Figure 6 and Figure 9. From Figure 10 it is observed that TH23 gates generates carryout and TH34w2 gates utilizes these results to generate the sum (S0, S1). When the FA circuit is simulated, voltage swings (for logic low) at carryout was ~ 0.1V, whereas for sum it was ~ 0.38V. This increased voltage swing at the sum output is due to the voltage drop at TH23 gate being carried on to TH34w2 gate. Therefore, realizing the whole circuit using

FNCL gates is not viable option. To address this limitation, novel HYBRIB approach is also proposed in this paper.

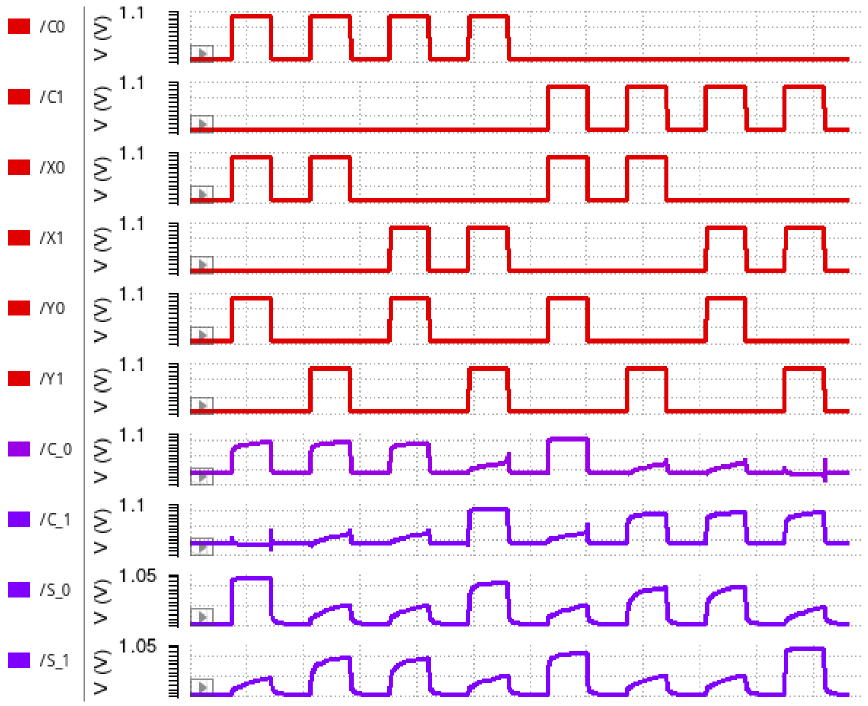


Figure 10. Simulation results of FA validating the volatge drop at sum is greater than carryout

3.3. CMOS-GDI HYBRID APPROACH

The design of the HYBRID model is inspired by the observation that in NCL system framework the DI combinational logic (CL) is always enclosed between DI registers. In other words, inputs or outputs of CL always pass through a DI register to ensure synchronization (two DATA wavefronts are not overwriting). The idea of the HYBRID methodology is to redesign this NCL structure using both static CMOS and FNCL techniques. Figure 11 depicts the framework of NCL system using HYBRID methodology. The difference between the original and the new (HYBRID) structure is the method by which NCL blocks CL, DI register and completion detection (CD) are realized. The FNCL approach is utilized to realize CL and CD blocks, while static CMOS method is used to implement the DI registers (CMOS_DI_reg).

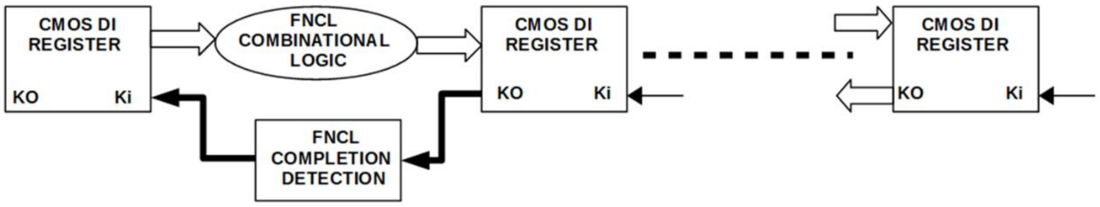


Figure 11. The proposed HYBRID framework

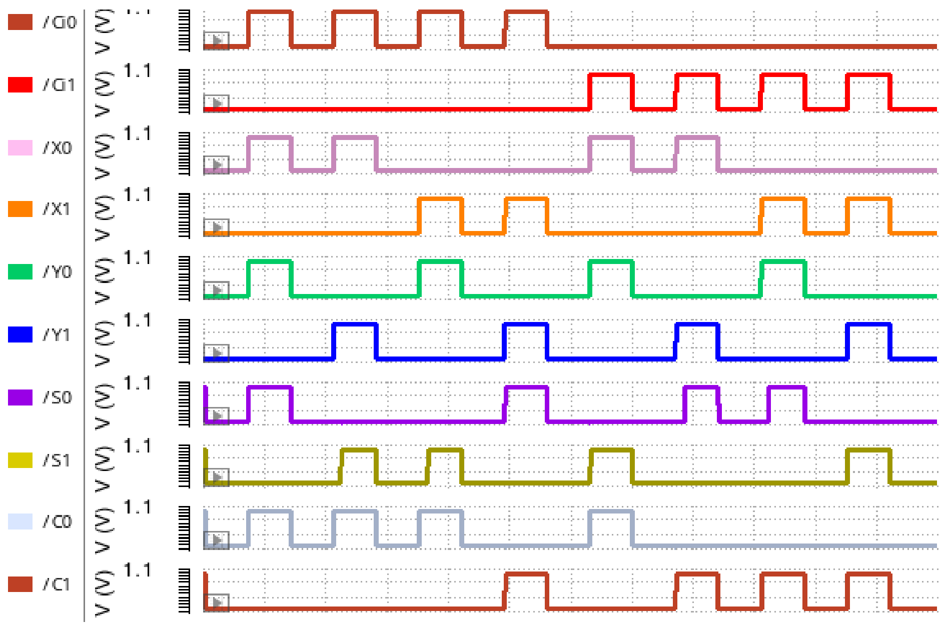


Figure 12. Simulation results of a 1-bit full adder using HYBRID approach

As discussed, the FNCL blocks (CL, CD) yield a voltage drop at their output. This limitation can be addressed by transferring these outputs through the CMOS_DI_reg. The

CMOS_DI_reg have strong pull-up and pull-down network which helps to restore signal strength and generate an output of either V_{DD} or ground. Thus, the HYBRID approach prevents the voltage drop of the GNCL blocks from progressing to the next stage. Figure 12 shows the application of this idea to a one-bit full adder circuit and the simulation results depicts that HYDRIB approach has no performance degradation.

In summary, the FNCL approach is proposed to address the area overhead limitation of static CMOS methodology. However, the voltage drop at the output hinders this approach for designing NCL system. Therefore, a HYDRID methodology, which utilizes both the FNCL and static CMOS techniques to address the drawbacks of both the approaches are introduced. To validate the effectiveness of the HYBRID methodology, the proposed approach is applied to a case study of different NCL up-counter increment (NUI) designs. A comparative study of the NUI circuits when implemented using static CMOS and HYBRID methodologies are presented in the next section.

4. PERFORMANCE EVALUATION

This section presents the comparative results of NUI designs when implemented using static CMOS and HYBRID methodologies. All the designs are realized in 45nm technology using Cadence general-purpose design kit (PDK) which provides the standard cell library and associated technology files for circuit realization. The schematics are simulated using Specter simulator with V_{DD} = 1V and temperature = 27^{o}c.

Serval alternative designs for NCL up-counter increment circuits are realized to verify the viability of the proposed approach. The proposed HYBRID methodology achieves a significant reduction in transistor count compared to the conventional static CMOS NCL designs.

4.1. NCL GATES UTILIZED FOR REALIZING NUI CIRCUITS

The NCL gates used for implementing various NUI designs along with their transistor count for CMOS and HYBRID methodologies are depicted in Figure 13. As illustrated in Figure 10, an average of 6% decrement in the number of transistors utilized for implementing these NCL gates using FNCL methodology is observed.

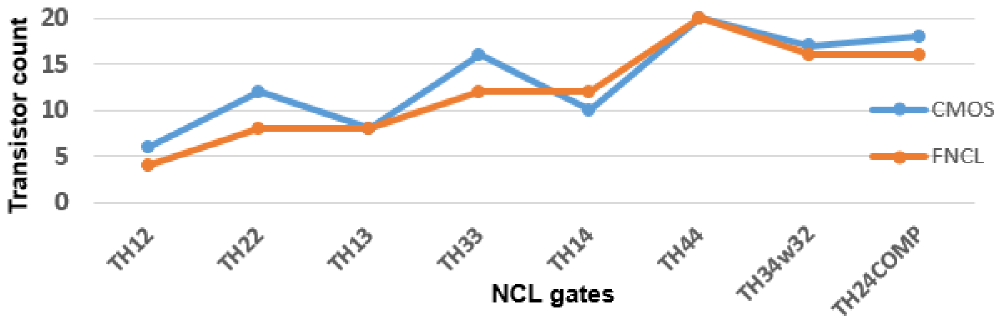


Figure 13. Number of transistors utilized by CMOS and FNCL

4.2. TRANSISTOR COUNT FOR VARIOUS NUI IMPLEMENTATIONS

Table 1, presents the transistor count (TC) for various NUI designs implemented via static CMOS and HYBRID methodologies. As observed from the Table 1, HYBRID methodology utilizes a smaller number of transistors compared to the CMOS implementation. The percentage reduction in transistor count for each model is illustrated in the Figure 14. Compared to the conventional static CMOS methodology, an incomplete AND NUI shows a 7% reduction in TC when implemented using the proposed approach.

Similarly, the Reduced Dual-Rail, Factored Dual-Rail, Complex Dual-Rail NUI circuits show a 19.3 %, 12.3% and 4% reduction in TC when realized using HYBRID approach.

Table 1. Comparison of static CMOS and HYBRID methodologies

Model Type	**STATIC CMOS**		**HYBRID**	
	TC of only CL	**Total TC including DI registers**	**TC of Only CL**	**Total TC including DI registers**
Incomplete AND	216	492	180	456
Reduced Dual-Rail	460	764	340	616
Factored Dual-Rail	308	584	236	512
Complex Dual-Rail	212	488	192	468

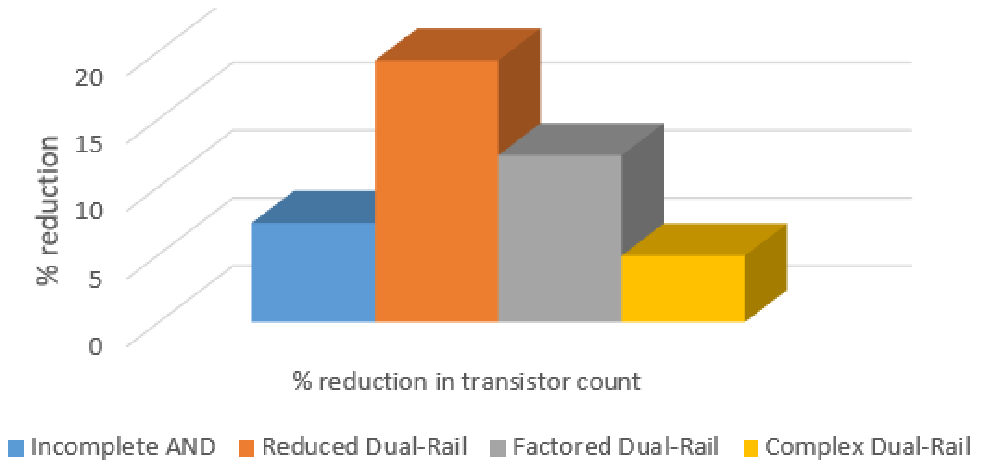


Figure 14. Percentage reduction in the transistor count

5. CONCLUSION

In this paper, a novel CMOS-GDI HYBRID methodology is proposed and validated to address the area overhead in conventional NCL based on static CMOS implementation. It utilizes two types of design techniques, static CMOS and GDI to realize NCL designs. The proposed approach demonstrated an average of 10% reduction in the transistor count when several NUI are realized using the proposed approach. This enhancement provides the scope for NCL to be an alternative for synchronous designs. The impact of HYBRID method on power consumption and latency will be the part of the future work.

BIBLOGRAPHY

[1] Bandapati, Satish K., and Scott C. Smith. "Design and characterization of NULL convention arithmetic logic units." Microelectronic engineering 84, no. 2 (2007): 280-287.

[2] F. A. Parsan and S. C. Smith, “CMOS implementation of static threshold gates with hysteresis: A new approach,” in Proc. IEEE/IFIP 20th Int VLSI and System-on-Chip (VLSI-SoC) Conf, Oct. 2012, pp. 41–45.

[3] Smith, Scott C., Ronald F. DeMara, Jiann S. Yuan, D. Ferguson, and D. Lamb. "Optimization of NULL convention self-time circuits." INTEGRATION, the VLSI journal 37, no. 3 (2004): 135-165.

[4] Bandapati, Satish K., Scott C. Smith, and Minsu Choi. "Design and characterization of NULL convention self-timed multipliers." IEEE design & test of computers 20, no. 6 (2003): 26-36.

[5] R. Bonam, S. Chaudhary, Y. Yellambalase, and M. Choi, “Clock-free nanowire crossbar architecture based on null convention logic (ncl),” in Proc. 7th IEEE Conf. Nanotechnology (IEEE NANO), Aug. 2007, pp. 85-89.

[6] Bailey, Andrew D., Jia Di, Scott C. Smith, and H. Alan Mantooth. "Ultra-low power delay-insensitive circuit design." In 2008 51st Midwest Symposium on Circuits and Systems, pp. 503-506. IEEE, 2008.

[7] Smith, Scott Christopher, and Ronald F. Demara. "Gate and throughput optimizations for null convention self-timed digital circuits." Doctor of Philosophy, Dissertation (2001).

[8] F. A. Parsan and S. C. Smith, “CMOS implementation comparison of ncl gates,” in Proc. IEEE 55th Int. Midwest Symp. Circuits and Systems (MWSCAS), Aug. 2012, pp. 394–397.

[9] Mader, Roy, Eby G. Friedman, Ami Litman, and Ivan S. Kourtev. "Large scale clock skew scheduling techniques for improved reliability of digital synchronous VLSI circuits." In 2002 IEEE International Symposium on Circuits and Systems. Proceedings, vol. 1, pp. I-I. IEEE, 2002.

[10] Morgenshtein, Arkadiy, Michael Moreinis, and Ran Ginosar. "Asynchronous gate-diffusion-input (GDI) circuits." IEEE transactions on very large scale integration (vlsi) systems 12, no. 8 (2004): 847-856.

[11] Morgenshtein, Arkadiy, Alexander Fish, and Israel A. Wagner. "Gate-diffusion input (GDI): a power-efficient method for digital combinatorial circuits." IEEE transactions on very large scale integration (VLSI) systems 10, no. 5 (2002): 566-581.

[12] Morgenshtein, A. Fish, and A. Wagner, “Gate-diffusion input (gdi)-a novel power efficient method for digital circuits: a design methodology,” in Proc. 14th Annual IEEE Int. ASIC/SOC Conf, 2001, pp. 39–43.

[13] Morgenshtein, A. Fish, and I. A. Wagner, “Gate-diffusion input (gdi) - a technique for low power design of digital circuits: analysis and characterization,” in Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2002, vol. 1, 2002, pp. I–477–I–480 vol.1.

[14] Morgenshtein, I. Shwartz, and A. Fish, “Gate diffusion input (gdi) logic in standard CMOS nanoscale process,” in Proc. IEEE 26-th Convention of Electrical and Electronics Engineers in Israel, Nov. 2010, pp. 000 776–000 780.

[15] Morgenshtein, V. Yuzhaninov, A. Kovshilovsky, and A. Fish, “Full-swing gate diffusion input logiccase study of low-power cla adder design,” INTEGRATION, the VLSI journal, vol. 47, no. 1, pp. 62–70, 2014.

III. A LOW POWER DESIGN TECHNIQUE FOR THE ASYNCHRONOUS NULL-CONVENTION LOGIC CIRCUITS

ABSTRACT

Null Convention Logic (NCL) is a robust clock-less technique for designing asynchronous delay-insensitive circuits. The traditional complementary metal oxide semiconductor (CMOS) approach is often used for designing NCL circuits, which tends to occupy a large area. To address this issue, a low power design technique Gate Diffusion Input (GDI) is introduced for designing the NCL circuits. This GDI design methodology is the promising alternative for the static CMOS designs, which allows the reduction in area and power consumption while maintaining the low complexity of the logic design. In this paper, a novel GDI based NCL designs are proposed and designed. However, the voltage swings in the GDI approach leads to the considerable amount of voltage drop at the output. This limitation is addressed by using low threshold transistors where a voltage drop is expected, and high threshold transistors are used for the regenerative inverters at the output. The proposed approach has been verified by designing the NCL Ripple Carry Adder (RCA), uunpipelined multiplier, pipelined multiplier and unpipelined ALU by using the GDI technique. These models are designed and simulated using Cadence Virtuoso and an average of 13.5% reduction in the transistor count is observed for these GDI based NCL models when compared to the CMOS models.

1. INTRODUCTION

With the increasing clock rate and decreasing IC feature size, meeting timing closure requirement in the presence of clock skew is a major problem [1–3]. To address this issue, a high-performance device allocates a large part of its area to clock drivers, which increases power dissipation, significantly at clock edges, where switching occurs [4–6]. Power dissipation deteriorates the operation of high-performance devices, which is a major concern for the emerging low power industry [7, 8]. Since, asynchronous digital designs are inherently robust in the sense of power dissipation, exhibits low noise and electro-magnetic interference, there is a renewed interest in this area [9–11].

Asynchronous circuits are classified into two types: bounded-delay and delay insensitive (DI) models. Bounded- delay models such as Micropipelines [12] and Huffman [13] circuits consider both gate delays and wire delays to be bounded. The delays are added according to the worst-case scenarios, therefore require extensive timings to ensure the correctness of the circuits [14]. On the other hand, delay-insensitive models assume that both the gate and wire delays are unbounded, and wire forks within basic components are isochronic. However, the wire connecting the components doesn't abide with the isochronic assumption. Therefore, DI models can operate correctly regardless of input availability [15, 16].

In the literature, several DI paradigms are available for designing asynchronous circuits [17]. These paradigms include phasedlogic, null convention logic (NCL), Seitz's [18], Singh's [19], David's [20], Anantharaman's [21] and DIMS [22] approaches. Among these approaches, NCL is the most commonly used DI paradigms and it exhibits great optimization potential over other DI methods. NCL designs use special types of gates called threshold gates which has built in hysteresis state holding property to achieve DI and to ensure that the gate is input complete [23].

Recent literature reports several CMOS methodologies that are typically utilized to implement these gates [24]. These methodologies include dynamic, semi- static, differential and static implementation. Detailed information about the dynamic, semi-static and differential implementation of NCL gates can be found in [25–27]. These CMOS methodologies either rely on parasitic capacitance (dynamic) or feedback mechanism (semi-static and differentia) to hold state information. Parasitic capacitance based NCL gates are susceptible to noise, leakage and charging problems, while a feedback inverter slows down the gate's operation due to the intrinsic switching contention.

Typically, these constraints are addressed with the use of static NCL gate implementation providing faster and reliable operation. The static NCL gates comprises of set and hold blocks which determines whether to perform the gate functionality or hold data [28]. However, due to the increase in area overhead observed with NCL implementation, their use is limited in the semiconductor industry [29]. To alleviate this issue, a novel GNCL methodology is introduced in this paper to reduces the area overhead of NCL gates. Furthermore, additional components are introduced within the methodology that improves the power robustness of NCL gates.

To address the issue of area overhead, gate diffusion input technique [30, 31], comprising of a basic GDI cell that can implement various complex Boolean circuits is utilized. Since, a GDI–based implementation utilizes just two transistors to implement such circuits, a reduction in transistor count is observed which in turn reduces switching power

[32, 33]. However, due to the application of different inputs to the pMOS and nMOS sources, a voltage drop is observed at the output. To address the issue of voltage drop, the cascaded inverters methodology is adopted. It is then demonstrated that, voltage swing at the output can be observed in GDI-based NCL designs. To overcome this limitation regenerative buffer are introduced where multi-threshold transistors are utilized for effectiveness.

The aim of this work is to propose a low power design approach to reduce area overhead of NCL circuits. With a comprehensive simulation study, the reduction in transistor count and power robustness are demonstrated in this paper. Overall, a 13% - 14% reduction in the number of transistors and a 14% - 30% decrement in the dynamic power is shown. The contributions of this paper include: 1. Implementation of NCL gates using GDI technique (GDI-NCL). 2. Application of regenerative buffers at the output of GDI-NCL gates to overcome the voltage swing. 3. A generalized (GNCL) approach that uses multi-threshold transistor technique to reduce area (transistor count), voltage swing and power is presented.

This article is organized as follows. Section 2 presents the preliminaries and review of NCL and GDI. Section 3 describes the limitation of NCL and the technique for overcoming it. An extensive discussion of the proposed design is carried out in Section 4. Performance evaluation data for various NCL circuits are included in Section 5. Finally, the summary and concluding remarks are made in Section 6.

2. PRELIMINARIES AND REVIEW

NCL is a popular delay-insensitive (DI) methodology for designing asynchronous circuits where accurate results regardless of input availability are observed. Therefore, NCL circuits are clockless, self-timed logic paradigms that can integrate data and control into a single signal. To achieve the delay-insensitive behavior, NCL must exhibit two primary characteristics; (1) symbolic completeness and (2) input completeness, achieved through dual- rail or quad- rail logic [14].

Dual rail logic consists of two wires D0 and D1, representing the states DATA0, DATA1 and NULL. The DATA 0 (D0 = 1, D1= 0) is equivalent to Boolean logic 0, DATA1 state (D0= 0, D1= 1) constitutes Boolean logic 1 whereas D0 = 0, D1= 0 represents NULL state. NULL state refers to the scenario when no DATA is available at the input. Furthermore, the state of D0 = 1 and D1 = 1, refers to the invalid stage. Similarly, quad-rail has four wires Q0, Q1, Q2 and Q3, each representing different stages from the set DATA0, DATA1, DATA2, DATA3, and NULL. Both rails are mutually exclusive, so that no two rails can be asserted simultaneously [21].

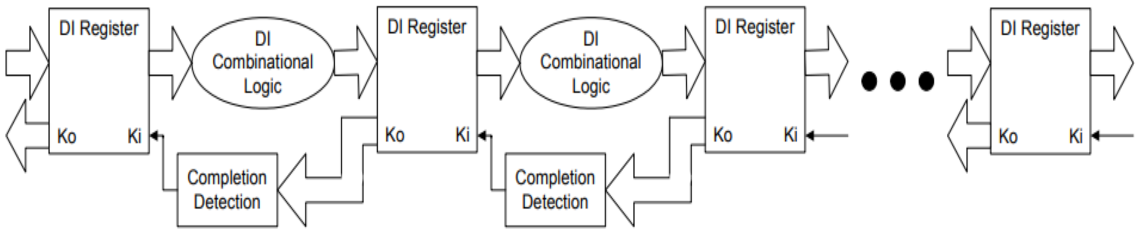


Figure 1. NCL framework [17]

Moreover, NCL systems are composed of DI combinational logic blocks that are enclosed between DI registers as shown in Figure 1. The structure of NCL is similar to synchronous designs, except for the case of completion detection (CD) where data synchronization between the logic blocks is enabled. Communication between the adjacent DI registers are carried out by local handshaking signals i.e. request signal (Ki) and acknowledge signal (Ko). These signals avoid overwriting two DATA wavefront by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. These acknowledge signals are then combined in the CD circuitry to produce the request signal(s) to the previous register stage [23].

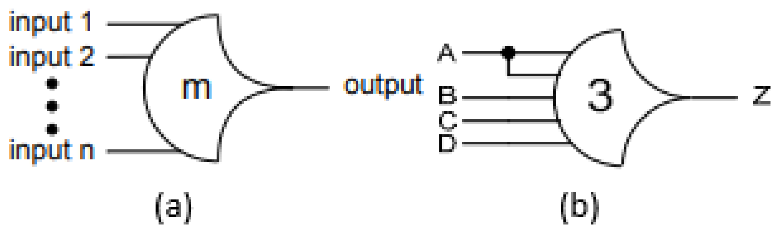
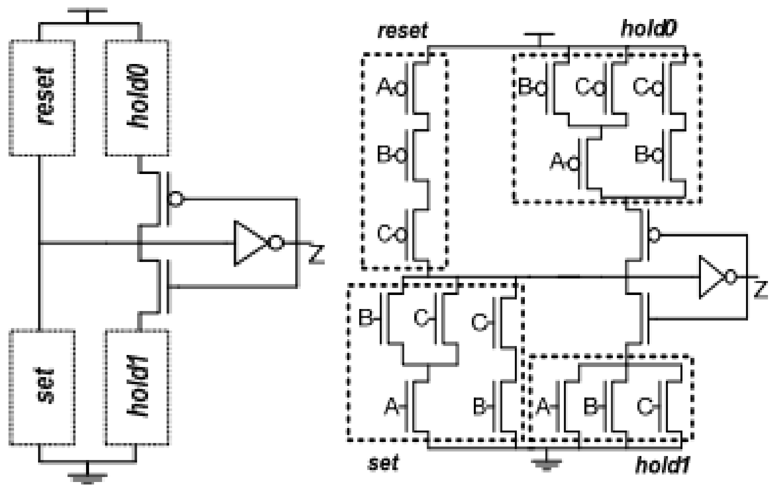


Figure 2. (a) THmn threshold gate (b) TH34w2 threshold gate [34]

The fundamental building block for the designing any NCL circuits are the threshold gates, which are of two types: threshold gate and weighted threshold gate [28]. Figure 2(a) depicts the primary type of threshold gate THmn, where $1 \leq m \leq n$. Here, n represents the total number of inputs and m indicates the minimum number of inputs to be asserted for the output to be asserted. The weighted threshold gates are represented as THmnWw1w2wR, where w1, w2, ...wR, each > 1 signifies the integer weights of input1, input2,..... input R, respectively. Figure 2 (b) illustrate the weighted threshold gate TH34W2 gate.

There are 27 basic NCL gates constituting from two to four variable functions. These gates have built- in hysteresis state holding capacity that ensures that the gate is input complete, which means that the output remains constant until all inputs are deserted. The functionality (set equation) of 27 NCL gates can be found in [34] and the structure of static NCL gate implementation is depicted in Figure 3(a).

Typically, a static NCL gate consists of two pull down (set, hold1) and two pull up (reset, hold0) networks as depicted in Figure 3(a). The set block defines the functionality of the gate and determines when the output should be asserted. Similarly, hold1 network is used to retain data until all inputs are deasserted which is obtained by OR-ing all the inputs. Therefore, hold1 is equivalent to n-input OR gate, where n is the total number of inputs [24].



(a) (b)

Figure 3. (a) Structure of static CMOS implementation of NCL gates (b) Static CMOS transistor level implementation of TH23 NCL gate [29]

The general equation for implementing static NCL gates is given as Z = set + (Z' • hold1), where z is the new output value and z- denotes past output value. Similarly, the general form of Z complement is represented as Z' = reset + (Z' • hold0), where reset and hold0 is the complement of set and hold1 [29]. These equations can be used for implementing any NCL gates and Figure 3(b) depicts the transistor level implementation of TH23 gate with set and hold1 equations as AB+BC+AC and A+B+C.

As seen from the Figure 3(a), extra logic is required to implement hold1 and hold0 increases the area overhead. This drawback is one of the factors limiting the use of NCL designs in the semiconductor industry, and the following section discusses this drawback in detail.

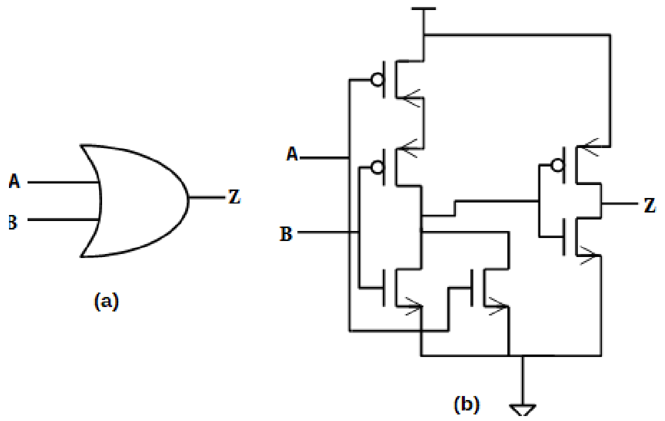


Figure 4. Conventional Boolean OR gate (a) symbol (b) Static CMOS implementation

3. LIMITATION OF NCL

It is estimated that, for a functionally identical design, NCL will have a substantial increase in area relative to the conventional synchronous designs. Moreover, the area consumption in static NCL designs approximately 1.5 – 2 times that of an equivalent synchronous design [2]. To clearly demonstrate this idea, a realization of AND function is compared with conventional Boolean and NCL design next. A gate and transistor level implementation of an AND function through conventional Boolean and NCL design is shown in Figure 4 and Figure 5 [28].

In Figure 4(b), it can be observed that conventional two input OR function requires only six transistors. The NCL AND comprises of TH12 and TH22 threshold gates as illustrated in Figure 5(a). Furthermore, as seen in Figure 5(b), TH22 gate require twelve transistors and TH12 gate which is equivalent to 2-bit conventional OR gate require six transistors for their implementation through static CMOS approach. Overall, NCL logic requires twenty transistors to implement NCL OR gate, which is approximately three times higher than the conventional synchronous design. Thus, proving that NCL design has a higher transistor count and with the International Technology Roadmap for Semiconductors (ITRS) anticipates that asynchronous circuits will account for 47% of chip area by 2020 [34], it is very important to address this limitation. Furthermore, as asynchronous circuits are incredibly useful in low area and power applications such as microcontrollers, embedded medical products, encryption engines for smart card applications, fault-attack-resistant cryptographic circuits, ternary logic, sensor networks, and IOT devices, it is very important to address this limitation [10].

The two main causes for the increased area are the use of dual rail logic and the requirement of additional circuitry for state holding capacity and this is observed in Figure 3(a). Since, the dual rail is required to obtain symbolic and input completeness, the internal gate logic must be altered to achieve any reduction in area. To manipulate the internal logic for NCL gates, this work introduces a GDI-based approach. The implementation of GDI-based NCL is discussed in detail in the next section.

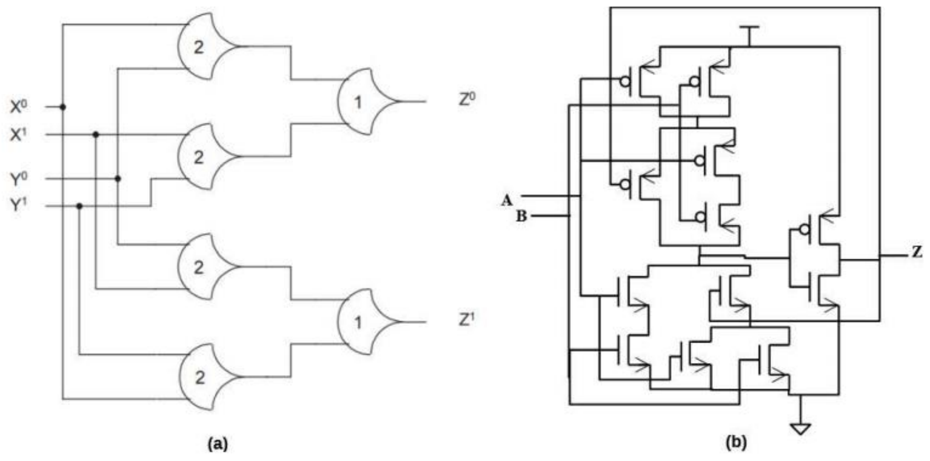


Figure 5. (a) NCL input-complete OR function: Z = X + Y [28] (b) Static CMOS implementation of TH22 NCL gate

4. PROPOSED APPROACH FOR DESIGNING NCL CIRCUITS

We begin this section by describing a basic implementation of area efficient gate diffusion input-based NCL (GDI-NCL) design paradigm. Next it is demonstrated that GDI-based designs suffer from drop in output voltage. Next, regenerative buffers are introduced where multi-threshold gates are utilized to eliminate voltage drops.

4.1. BASIC IMPLEMENTATION OF NCL GATES USING GDI TECHNIQUE (GDI-NCL TECHNIQUE)

To utilize GDI methodology, the Boolean expression of a THmn or weighted NCL gates must be factorized into series of AND and OR functionality. Next, based on the resulting expression, GDI-AND, GDI-OR and GDI-MUX configuration as shown in Figure 6 can be used for the design of NCL gates [30]. The algebraic representation for TH23 gate, which is shown in Figure 7, is given as

$$Z= AB + BC + AC + Z'(A+B+C) \tag{1}$$

where A, B, C, D are the inputs, Z' = previous result and Z = current output. Factorizing the above equation gives

$$Z = A (B + C) + BC + Z'(A+B+C) \tag{2}$$

This expression implementation using GDI-NCL technique is as explained as follows.

Step 1: First the second term in the equation 2 is implemented using GDI-AND. As shown in Figure 6(a), GDI-AND cell has two inputs G and P, and the output is measured at drain D. So, to implement BC, B and C are the given as input to G and N and the resultant output O1 is taken at D.

Step 2: Similarly, algebraic expression B +C is implemented using GDI-OR, where B, C are applied to nodes G and N and the output O2 is measured at D.

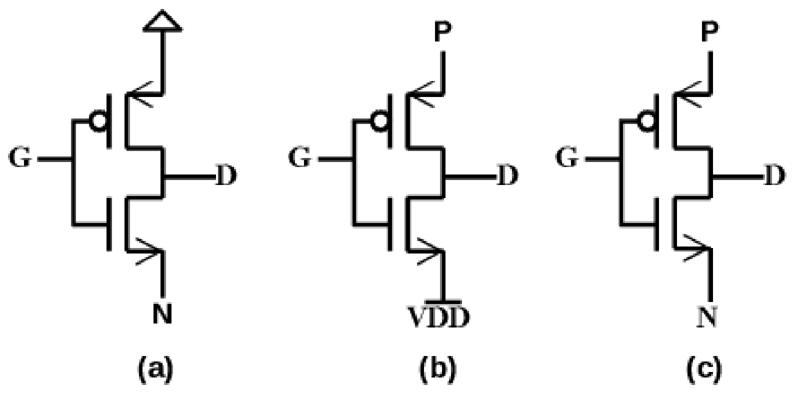


Figure 6. Structures of different GDI functionality cells: (a) GDI-AND (b) GDI-AND (c) GDI-MUX

Step 3: GDI-MUX is used to realize A(B+C) + BC. To implement this expression, input A is provided to node G and the output of step1 and step 2 i.e. O1 and O2 are given to input P and N of GDI-MUX. The output O3 is taken from node D.

Step 4: Another GDI OR are used to implement A+B+C, where A and O2 are applied to nodes G and P and the output O4 is generated at node D.

Step 5: Finally, GDI MUX is used to realize the final expression given in equation 2. To this MUX, Z' the previous result is given as the input to node G, O3 is passed to node P and O4 is provided to node N. Thus, the result i.e. the current output (Z) is measured at node D of this GDI MUX. Note: If Z'= 0 the output would be [A(B + C) + BC] i.e. the gate operators on current data, else, value at node N is selected which is nothing but A+B+C i.e. holds the previous results.

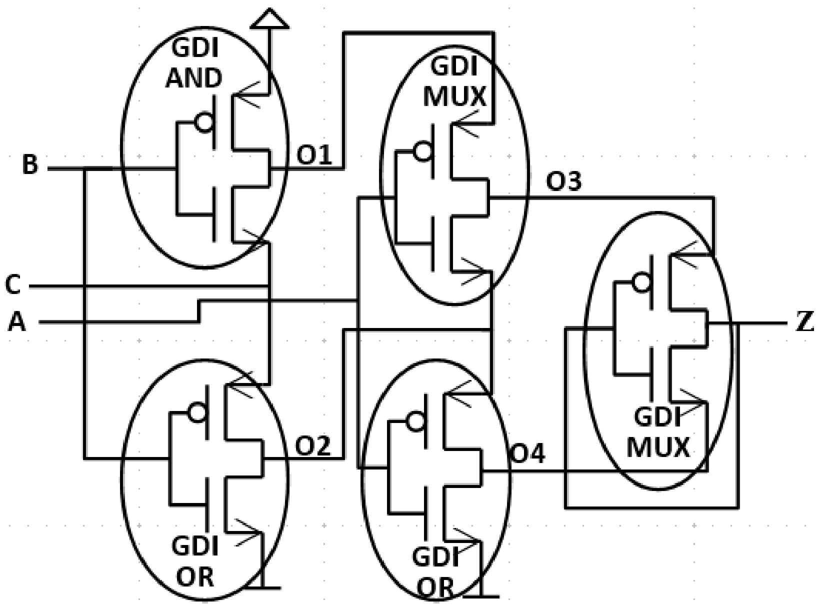


Figure 7. GDI implementation of TH23 NCL gate

The correctness of this implementation can be evaluated by verifying the output for different input combinations and the results are described in Table 1. As observed in Table 1, the result (Z) of the proposed approach is identical to the actual algebraic expression results. As seen from the Figure 7, the total number of transistors (#t) required to implement this GDI based TH23 gate is 10, while the CMOS implementation require 18 transistors.

Thus, a 44.4% reduction in the transistor count can be observed for TH23 gate. Similarly, all the NCL gates can be realized using GDI-NCL technique and compared to static CMOS implementation, GDI-NCL technique approximately provides 30% - 50% reduction in the transistor count as depicted in Figure 8. The algebraic factorization, the method of realization and #t used for implementation few NCL gates using proposed GDI-NCL technique is presented in Figure 9.

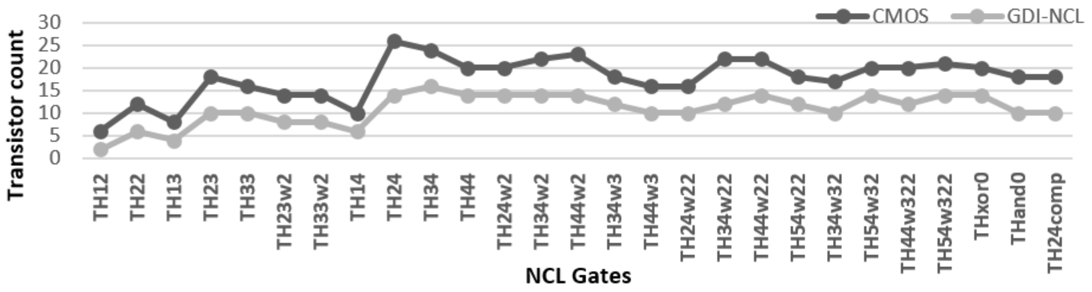
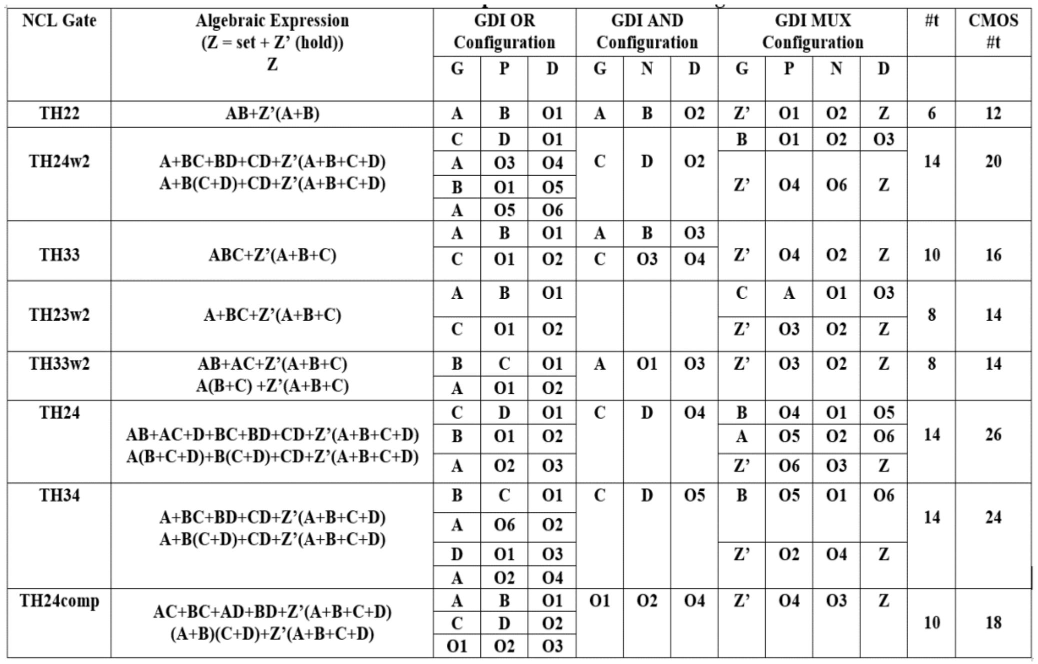


Figure 8. Number of transistors required for implementing NCL gates using CMOS and GDI-NCL methodology

The main disadvantage of GDI-NCL implementation is that the full output voltage swing cannot be achieved for all input combinations as shown in the Figure 10. This limitation is due to the structure of the inputs applied to the pMOS and nMOS transistors of a GDI-OR, GDI-AND, GDI-MUX cells. Since, the pMOS and nMOS transistors are a strong pull- up and pull- down networks, the application of any voltage other than VDD and gnd to their sources results in a voltage drop of Vtp (pMOS) and (V DD − Vtn) (nMOS), where, Vtp and Vtn represents the threshold voltage for pMOS and nMOS transistor.

Table. 1 GDI-NCL TH23 gate results for different input combinations

Inputs	**Node Voltages**					**Algebraic Expression Results** Z = A(B+C)+BC+Z'(A+B+C)
	O1	**O2**	**O3**	**O4**	**Z**	
A= 1; B = C = Z' = 0	0	0	0	1	0	0
A = C = 1; B = Z' = 0	0	1	1	1	1	1
A= B = Z' = 0; C = 1	0	1	0	1	0	0



NCL Gate	Algebraic Expression (Z = set + Z' (hold)) Z	GDI OR Configuration			GDI AND Configuration			GDI MUX Configuration				#t	CMOS #t
		G	P	D	G	N	D	G	P	N	D		
TH22	AB+Z'(A+B)	A	B	O1	A	B	O2	Z'	O1	O2	Z	6	12
TH24w2	A+BC+BD+CD+Z'(A+B+C+D)	C	D	O1	C	D	O2	B	O1	O2	O3	14	20
	A+B(C+D)+CD+Z'(A+B+C+D)	A	O3	O4				Z'	O4	O6	Z		
		B	O1	O5									
		A	O5	O6									
TH33	ABC+Z'(A+B+C)	A	B	O1	A	B	O3	Z'	O4	O2	Z	10	16
		C	O1	O2	C	O3	O4						
TH23w2	A+BC+Z'(A+B+C)	A	B	O1				C	A	O1	O3	8	14
		C	O1	O2				Z'	O3	O2	Z		
TH33w2	AB+AC+Z'(A+B+C)	B	C	O1	A	O1	O3	Z'	O3	O2	Z	8	14
	A(B+C) +Z'(A+B+C)	A	O1	O2									
TH24	AB+AC+D+BC+BD+CD+Z'(A+B+C+D)	C	D	O1	C	D	O4	B	O4	O1	O5	14	26
	A(B+C+D)+B(C+D)+CD+Z'(A+B+C+D)	B	O1	O2				A	O5	O2	O6		
		A	O2	O3				Z'	O6	O3	Z		
TH34	A+BC+BD+CD+Z'(A+B+C+D)	B	C	O1	C	D	O5	B	O5	O1	O6	14	24
	A+B(C+D)+CD+Z'(A+B+C+D)	A	O6	O2									
		D	O1	O3				Z'	O2	O4	Z		
		A	O2	O4									
TH24comp	AC+BC+AD+BD+Z'(A+B+C+D)	A	B	O1	O1	O2	O4	Z'	O4	O3	Z	10	18
	(A+B)(C+D)+Z'(A+B+C+D)	C	D	O2									
		O1	O2	O3									

Figure 9. GDI-NCL implementation of few NCL gates

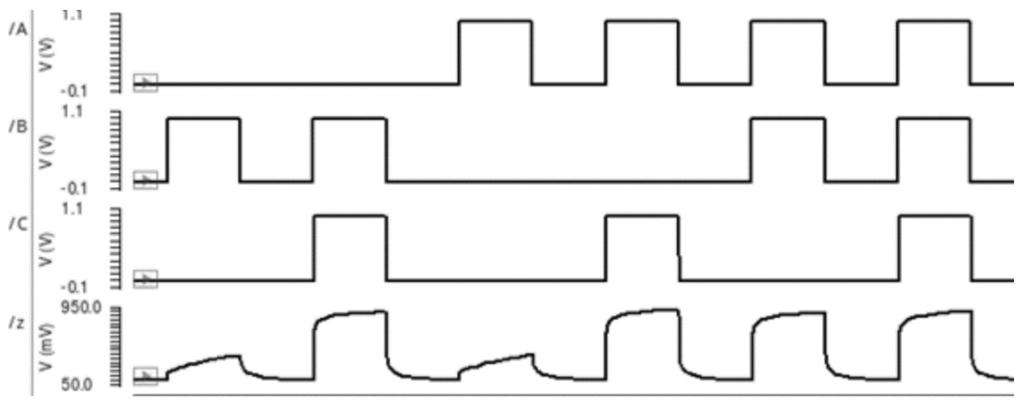


Figure 10. Voltage drop at the output of GDI-NCL TH23 gate

Inputs			Node Voltages				Final Voltage (Z)	
A	B	C	N1 (O1)	N2 (O2)	N3 (O3)	N4 (O4)	N5 = 0	N5 = 1
0	0	0	Vtp	Vtp	Vtp	Vtp	>Vtp	>Vtp
0	1	0	Vtp	VDD - Vtn	Vtp	VDD - Vtn	>Vtp	< VDD - Vtn
1	0	1	Vtp	VDD - Vtn	VDD - Vtn	VDD - Vtn	< VDD - Vtn	< VDD - Vtn
1	1	1	VDD-Vtn	VDD - Vtn	VDD - Vtn	VDD - Vtn	< VDD - Vtn	< VDD - Vtn

Figure 11. Voltage drop at the output of GDI-NCL TH23 gate for different input combinations

To overcome the performance degradation and achieve a full swing output voltage, a regenerative buffer can be utilized. Regenerative buffer consists of two back- to- back inverters that can restore the signal strength. Therefore, to avoid the voltage drop in GDI-NCL gates, a regenerative buffer is used before the output of every gate as shown in Figure 12. The difference in voltage output (Z) levels before and after the use of regenerative buffers are shown in Figure 12 and Figure 13 respectively. As depicted in Figure 13, regenerative buffer increases the signal strength and allow the final output voltage to achieve a strong zero (gnd) or a strong one (VDD). Note that regenerative buffer approach for output voltage swing restoration is quite straightforward and different buffering techniques can be found in literature [32]. However, the usage of the regenerative buffer increases the transistor count, delay and power. Compared to the basic GDI-NCL implementation, regenerative buffers will increase the transistor count by four for each gate. The static power consumption is increased due to the direct leakage path.

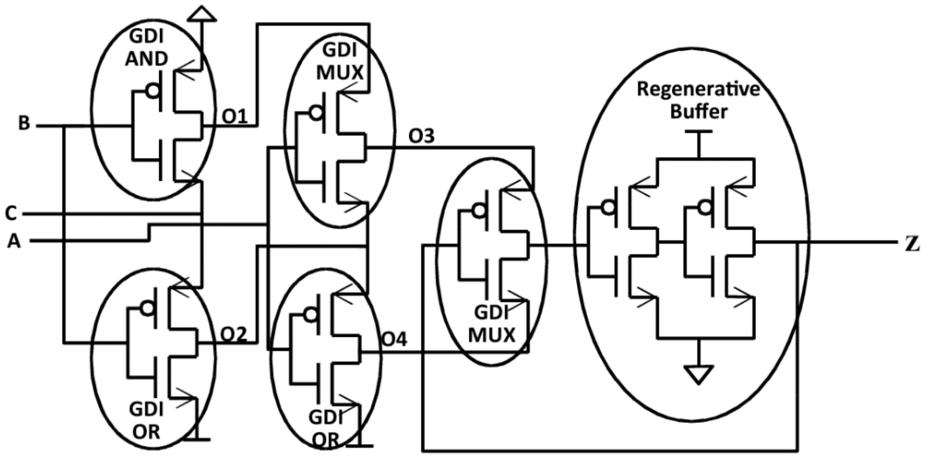


Figure 12. Addition of regnerative buffer at the output of GDI-NCL TH23

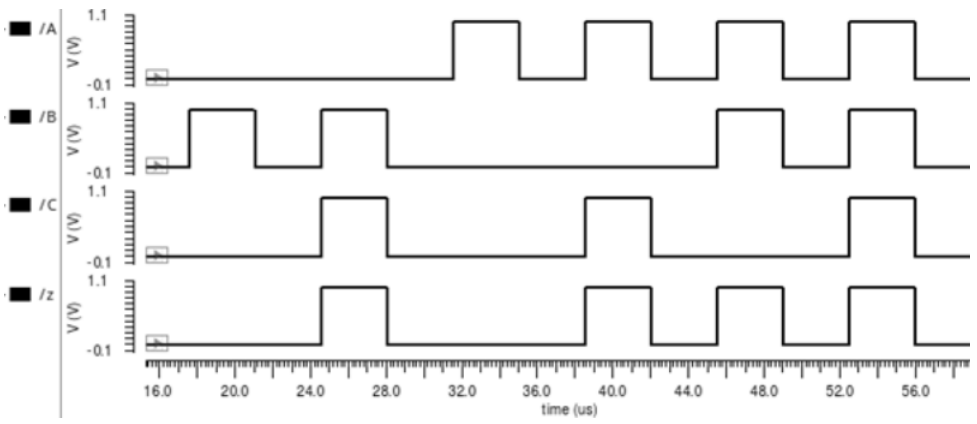


Figure 13. Output wavefprm of GDI-NCL TH23 gate after the use of renerative buffers

This limitation can be addressed by designing the circuit using multi-threshold transistors technique. As seen in [35], leakage current, the main contributor towards static power exponentially decreases with the increasing threshold voltage. Therefore, high threshold transistors can be used to decrease static power, but with overhead latency. So, to achieve low power and latency, both low threshold and high threshold transistor are utilized for implementing NCL gates. Low threshold transistors are used in the critical paths of the circuit, while the rest of the circuit is designed with high threshold transistors. The integration of high threshold transistors into non- critical paths is usually practiced to maintain high performance (i.e. to reduce leakage power), while low threshold transistors are used in critical paths to maintain speed. Thus, the multi-threshold transistor technique can be applied to GDI-NCL technique to reduce the static power even when regenerative buffers are used.

4.2. GENERALIZED APPROACH (GNCL) FOR REALIZING NCL GATES

To achieve a low power design, low threshold transistors are used specifically in path where a voltage drop is to be expected, and the rest of the circuit is designed using high threshold transistors. Therefore, GDI-AND, GDI-OR and GDI-MUX are realized using low threshold transistors and regenerative buffers are implemented using high threshold transistors.

For example, consider the scenarios when TH23 gate are realized using the GNCL approach. As described in Figure 14, GDI based AND, OR and MUX gates, realized using low threshold transistors, are utilized for realizing BC, B + C, A A(B + C) + BC and (B +

C) + BC + Z’(A+B+C). On the other hand, regenerative buffers are designed using high threshold transistor to produce a full swing final output.

Even the final design that is described in this section exhibits a considerable impact on latency which will be discussed as part of the future effort. In the next section, the proposed methodology, several combination circuits are implemented and comparative results are presented.

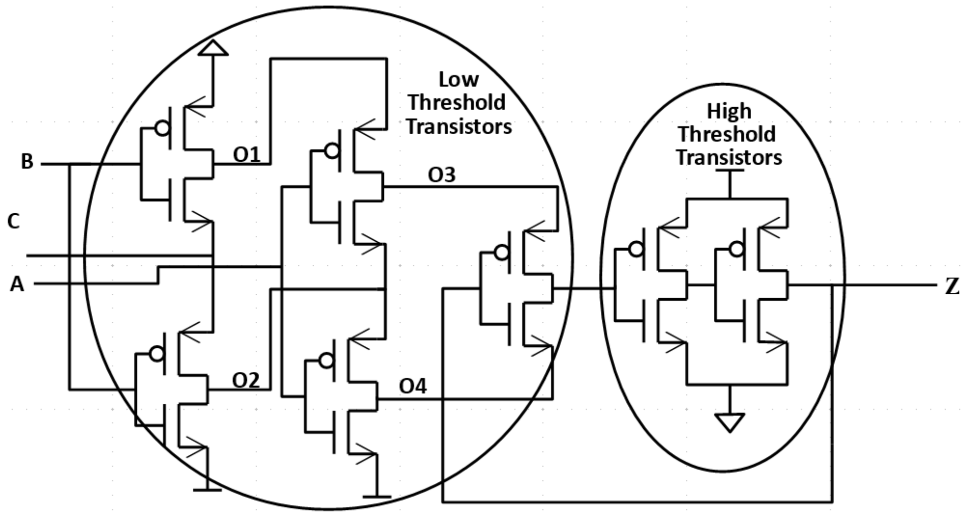


Figure 14. GNCL implementation of TH23 gate

5. SIMULATION RESULTS

This section compares the results for various NCL circuits implemented using static CMOS and GNCL approaches. These designs include 4-bit NCL ripple carry adder (RCA), 4-bit unpipelined NCL multiplier (UMUL), pipelined NCL multiplier (MUL) and 4-bit unpipelined NCL arithmetic logic unit (UALU). The internal logic and structures of these designs can be found in [23, 28]. To study the impact of proposed approach on dynamic power consumption, three CMOS models are realized using different threshold voltage transistors namely; high threshold model (High Vth), low threshold model (Low Vth) and nominal threshold model (Std Vth). The High Vth model consists of NCL gates implemented through high threshold voltage transistors where low power and high latency is observed. Similarly, Low Vth and Std Vth models respectively use low threshold and nominal threshold voltage transistors for implementation. These low threshold transistors tend to have low latency, but high-power consumption and standard threshold transistors provide medium delay and medium power dissipation. The GNCL design performance is compared individually with all of these CMOS designs.

All the designs are realized in 45nm technology using Cadence proprietary general process design kit (gpdk45). A process design kit contains the process technology and needed information to do device-level design in the Cadence environment. The schematics are implemented in Cadence Virtuoso tool with VDD = 1V and temperature = 27°C. The designs are simulated with the Spectre simulator in the Cadence Virtuoso using gpdk45 high, low and nominal threshold MOSFET transistors with W/L ratio of 2.6. All these transistors are minimum size and simulations were carried on all the possible input patterns. The performance comparison is based on number of transistors (#t), static power consumption (SP) and dynamic power consumption (DP). The average of all patterns is computed to determine the dynamic power consumption. To quantitatively demonstrate and verify the performance of the proposed approach, a detailed analysis of NCL gates are when implemented using GNCL technology is first presented.

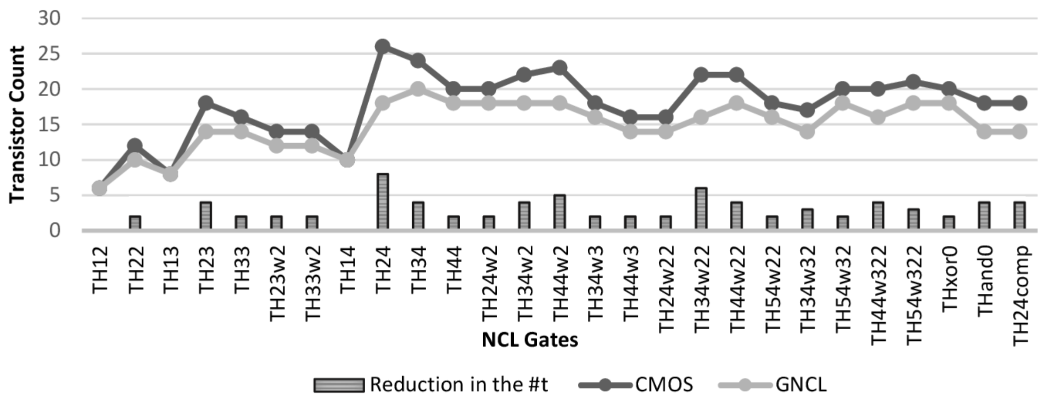


Figure 15. Transistor counts for 27 NCL gates using CMOS and GNCL techniques

5.1. 27 FUNDAMENTAL NCL GATES

As discussed in Section 2, NCL gates are the basic building of any NCL circuits. Therefore, comparison between CMOS and GNCL implementation of NCL gates in terms of transistor count and power is analyzed first.

5.1.1. Transistor Count. Figure 15 illustrate the #t required by the CMOS [28] and GNCL methodologies to implement 27 NCL gates. As shown in the graph, the GNCL approach shows a reduction in #t for all gates except for TH1n gates. TH1n gates are equivalent to n-input OR gates and require 2(n + 1) #t for their static CMOS implementation. Whereas, conventional GDI technique will only require (n−1)∗2 number of transistors. But, the conventional GDI approach has a drawback of voltage drop that affects the performance of the gate. To overcome this limitation, the proposed approach introduces regenerative buffers, hat increases the total #t by four. Therefore in total, the GNCL technique require ((n−1)∗2+ 4) #t, which is equal to the CMOS approach. Thus, the GNCL based TH1n gates have the same number of transistors as CMOS approach. However, GNCL approach reduces the number of transistors in the range of 2-8 for the other NCL gates. Therefore, an average 2.9% reduction in the transistor count for 27 NCL gates is observed.

5.1.2. Power Consumption. Power analysis of 27 NCL gates when implemented using different models is analyzed here. The difference in the static power and dynamic power consumption of the different implementation is shown below.

5.1.2.1. Static power. Static power quantifies the power consumed when the circuit is not in operation. Static power is largely dependent on the leakage current. The leakage mechanisms that dominates the total power dissipation in a transistor are subthreshold leakage and gate leakage that increases with reduction in threshold. As a result, higher the vth, smaller the static power. Thus, high vth transistors will result in lower static power dissipation compared to nominal vth and low vth transistors. Similarly, in comparison with low vth transistor, nominal vth transistor results in low static power dissipation. Therefore, as depicted in Figure 16, High Vth, and Std Vth models results in low static power than the GNCL model. But, the GNCL model provides better results than Low vth model as it contains both high and low vth transistors.

5.1.2.2. Dynamic power. Dynamic power, also known as switching power, depends on transient power and load capacitance power consumption. Transient power is the amount of power consumed when a device transitions from one state to another. Therefore, it is directly tied to the number of transistors in a device that change states. As a result, the decrease in the number of transistors can reduce dynamic power. Since, the GNCL model requires less #t than CMOS model, it therefore consumes less dynamic ower as shown in Figure 17. However, for some of the NCL hates such as T H1n gates, GNCL model has higher dynamic power than High Vth and Std Vth models, because generative buffers require higher power to increase the signal strength.

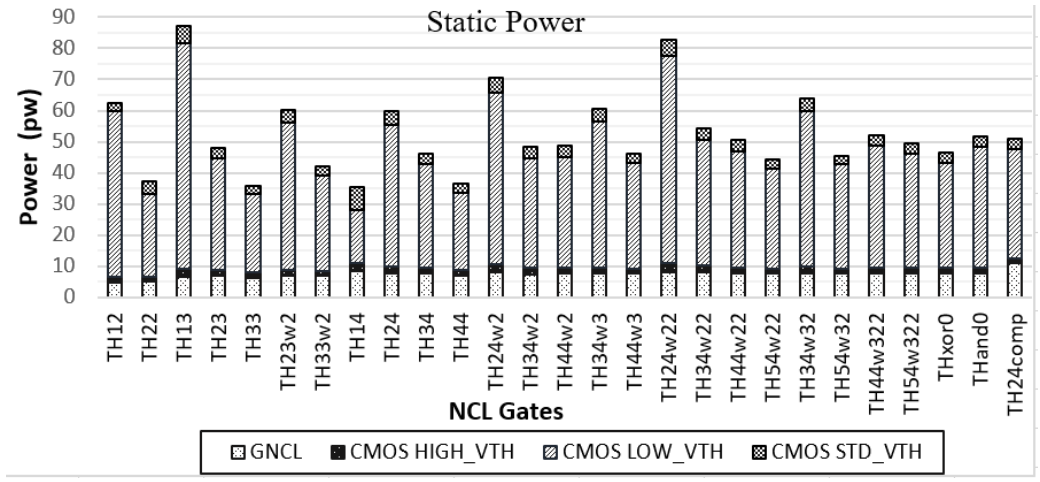


Figure. 16. Static power consumption of 27 NCL gates

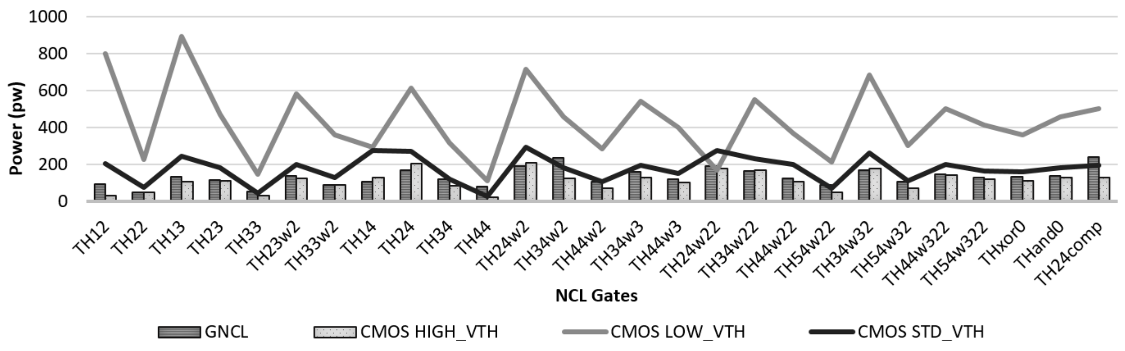


Figure 17. Dynamic power consumption of 27 NCL gates

5.2. COMBINATIONAL CIRCUITS

To further analyze the impact of the proposed approach on larger designs, several combinatorial circuits such as RCA, UMUL, MUL and UALU were realized using these different models (High Vth, Std Vth, Low Vth and GNCL) of NCL gates. The following are the comparative results of these NCL circuits.

5.2.1. Transistor Count. Table 2 summarize the total number of transistors required to implement RCA, UMUL, MUL and UALU circuits using CMOS and GNCL approaches. As shown in the table, all CMOS models require the same number of transistors, as they differ only in the type of transistor used. Therefore CMOS circuits consume more #t relative to GNCL designs for implementing any of these circuits. For an example, the CMOS models require 1128 transistors to realize a 4-bit RCA, while the GNCL requires only 960 transistors. Therefore, the proposed approach reduces the transistor count by 14%. Similarly, for the other circuits, an average reduction of 13.4% in the transistor count is observed with the use of GNCL approach.

Table 2. Total number of transistor used for implemententing various NCL circuit

MODEL	**RCA**	**UMUL**	**MUL**	**UALU**
High_Vth/Std_Vth/ Low_Vth	1128	2040	2574	4084
GNCL	960	1760	2238	3520
Reduction in the transistor count	14%	13.7%	13%	13.4%

5.2.2. Power Consumption. Power analysis of RCA, UMUL, MUL and UALU designs when implemented using different models is analyzed here. The difference in the static power and dynamic power consumption of the different implementation is shown below.

Table 3. Static power consumption for different NCL circuits

MODEL	**RCA (nW)**	**UMUL (nW)**	**MUL (nW)**	**UALU (nW)**
High_Vth	0.05	0.3	2.17	0.63
Low_Vth	3.72	5.9	8.5	11.4
Std_Vth	0.36	0.5	1.23	1.09
GNCL	1.50	2.3	1.28	3.20

Table 4. Dynamic power consumption for different NCL circuits

MODEL	**RCA (nW)**	**UMUL (nW)**	**MUL (nW)**	**UALU (nW)**
High_Vth	13.42	21.06	28.34	17.16
Std_Vth	16.45	25.93	33.9	24.55
Low_Vth	22.44	45.83	68.40	30.53
GNCL	13.79	21.91	29.84	18.42

5.2.2.1. Static power. As discussed earlier, static power consumption of NCL gates, high Vth transistors consumes less static power than nominal Vth and Low vth transistors. So, as seen from Table 3, for any circuit (RCA, UMUL, MUL and UALU), High Vth model always shows low static power consumption compared to other models. Similarly, Std Vth models designed with nominal Vth transistors consume less power than the proposed method. However, compared to Low Vth models, the GNCL based circuit's shows an average 80% reduction in static power.

5.2.2.2. Dynamic power. In Table 4, dynamic power consumption for different NCL circuits is summarized using CMOS and GNCL approaches. As seen in Table 6, GNCL designs reduce dynamic power by an average of 4% for High Vth, 21.7% for High Vth and 41% for Low Vth models.

In summary, among the presented CMOS models, GNCL shows best performance in terms of transistor count and dynamic power. Although, the static power of High_Vth designs are small, high Vth transistors are rarely used to design a complete device because they increase latency. Hence, a complete device is not entirely designed with high Vth transistors. It is therefore best to compare the performance of the GNCL approach to the Std_Vth and Low_Vth models. Compared to these two models, GNCL shows 13% -14% reduction in the transistor count and 30% - 40% reduction in dynamic power. The static power consumption of GNCL model can further reduced by replacing low Vth transistors by nominal Vth transistors.

6. CONCLUSION

This paper has proposed a novel GNCL methodology, which achieves low area and power consumption by using GDI and multi-threshold technique. The proposed approach shows a 13%-14% reduction in the transistor count and a 14%-30% decrement in the power consumption when compared to the conventional CMOS NCL counterpart. This considerable enhancement in terms of area and power will further increase the use of NCL in asynchronous digital designs, competing with conventional synchronous designs. The impact of the GNCL methodology on latency is not discussed and will be part of future effort.

BIBLOGRAPHY

[1] M. T. Moreira, M. Arendt, F. G. Moraes, and N. L. V. Calazans, “Static differential ncl gates: Toward low power,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 62, no. 6, pp. 563–567, Jun. 2015.

[2] P. A. Beerel, R. O. Ozdag, and M. Ferretti, A designer’s guide to asynchronous VLSI. Cambridge University Press, 2010.

[3] L. D. Tran, G. I. Matthews, P. Beckett, and A. Stojcevski, “Null convention logic (ncl) based asynchronous design—fundamentals and recent advances,” in 2017 International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom). IEEE, 2017, pp. 158–163.

[4] N. Nemati, P. Beckett, M. C. Reed, and K. Fant, “Clock-less DFT-less test strategy for null convention logic,” IEEE Transactions on Emerging Topics in Computing, vol. 6, no. 4, pp. 460–473, Oct. 2018.

[5] V. M. Wijayasekara, A. T. Rollie, R. G. Hodges, S. K. Srinivasan, and S. C. Smith, “Abstraction techniques to improve scalability of equivalence verification for ncl circuits,” Electronics Letters, vol. 52, no. 19, pp. 1594–1596, 2016.

[6] M. Chang and W. Chang, “Asynchronous fine-grain power-gated logic,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 6, pp. 1143–1153, Jun. 2013.

[7] J. Sudhakar, Y. Alekhya, and K. S. Syamala, “A dual-rail delay-insensitive ieee-754 single-precision null convention floating point multiplier for low-power applications,” in Innovations in Electronics and Communication Engineering. Springer, Jan. 2018.

[8] M. T. Moreira, P. A. Beerel, M. L. L. Sartori, and N. L. V. Calazans, “Ncl synthesis with conventional eda tools: Technology mapping and optimization,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 6, pp. 1981–1993, Jun. 2018.

[9] Y. Bai, R. F. DeMara, J. Di, and M. Lin, “Clockless spintronic logic: A robust and ultra-low power computing paradigm,” IEEE Transactions on Computers, vol. 67, no. 5, pp. 631–645, May 2018.

[10] M. Chang, P. Yang, and Z. Pan, “Register-less null convention logic,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 64, no. 3, pp. 314–318, Mar. 2017.

[11] S. Le, S. K. Srinivasan, and S. C. Smith, “Automated verification of input completeness for ncl circuits,” Electronics Letters, vol. 54, no. 20, pp. 1158–1160, 2018.

[12] I. E. Sutherland, “Micropipelines,” Commun. ACM, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[13] S. H. Unger and S. Y. H. Su, “Asynchronous sequential switching circuits,” and Cybernetics IEEE Transactions on Systems, Man, vol. SMC-3, no. 3, p. 302, May 1973.

[14] A. S. C. Smith, B. R. F. Demara, B. J. S. Yuan, C. D. Ferguson, and C. D. Lamb, “Optimization of null convention self-timed circuits,” INTEGRATION, the VLSI journal, vol. 36, no. 31, pp. 135–165, 2004.

[15] K. Van Berkel, “Beware the isochronic fork,” Integration, the VLSI journal, vol. 13, no. 2, pp. 103–128, 1992.

[16] C. A. R. Hoare, Developments in concurrency and communication. Addison-Wesley Longman Publishing Co., Inc., 1991.

[17] S. C. Smith, R. F. Demara, J. S. Yuan, M. Hagedorn, and D. Ferguson, “Delay-insensitive gate-level pipelining,” Integration, the VLSI journa, vol. 30, no. 2, pp. 103–131, 2001.

[18] C. L. Seitz, System Timing, in Introduction to VLSI Systems. Addison-Wesley, 1980.

[19] N. P. Singh, “A design methodology for self-time systems,” Master’s Thesis, MIT/LCS/TR-258, Laboratory for Computer Science, 1981.

[20] I. David, R. Ginosar, and M. Yoeli, “An efficient implementation of boolean functions as self-timed circuits,” IEEE Transactions on Computers, vol. 41, no. 1, pp. 2–11, Jan. 1992.

[21] Smith, S. C, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, “Null convention multiply and accumulate unit with conditional rounding, scaling and saturation,” Journal of Systems Architecture, vol. 47, no. 12, pp. 977–998, 2002.

[22] J. Sparso and J. Staunstrup, “Design and performance analysis of delay insensitive multi-ring structures,” in Proc. Twentysixth Hawaii Int. Conf. System Sciences [1993], vol. i, Jan. 1993, pp. 349–358 vol.1.

[23] S. K. Bandapati and S. C. Smith, “Design and characterization of null convention arithmetic logic unit,” Microelectronic engineering, vol. 84, no. 6, pp. 280–287, 2007.

[24] F. A. Parsan and S. C. Smith, “CMOS implementation comparison of ncl gates,” in Proc. IEEE 55th Int. Midwest Symp. Circuits and Systems (MWSCAS), Aug. 2012, pp. 394–397.

[25] G. E. Sobelman and K. Fant, “CMOS circuit design of threshold gates with hysteresis,” in Proc. IEEE Int. Symp. Circuits and Systems (ISCAS), vol. 2, May 1998, pp. 61–64 vol.2.

[26] M. Shams, J. C. Ebergen, and M. I. Elmasry, “Modeling and comparing CMOS implementations of the c-element,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 6, no. 4, pp. 563–567, Dec. 1998.

[27] S. Yancey and S. C. Smith, “A differential design for c-elements and ncl gates,” in Proc. 53rd IEEE Int. Midwest Symp. Circuits and Systems, Aug. 2010, pp. 632–635.

[28] Smith, S. Christopher, and R. F. Demara, Gate and throughput optimizations for null convention self-timed digital circuits. Doctor of Philosophy, Dissertation, 2001.

[29] F. A. Parsan and S. C. Smith, “CMOS implementation of static threshold gates with hysteresis: A new approach,” in Proc. IEEE/IFIP 20th Int. Conf. VLSI and System-on-Chip (VLSI-SoC), Oct. 2012, pp. 41–45.

[30] A. Morgenshtein, M. Moreinis, and R. Ginosar, “Asynchronous gate-diffusion-input (gdi) circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 12, no. 8, pp. 847–856, Aug. 2004.

[31] A. Morgenshtein, I. Shwartz, and A. Fish, “Gate diffusion input (gdi) logic in standard CMOS nanoscale process,” in Proc. IEEE 26-th Convention of Electrical and Electronics Engineers in Israel, Nov. 2010, pp. 000 776–000 780.

[32] A. Morgenshtein, A. Fish, and I. A. Wagner, “Gate-diffusion input (gdi): a power-efficient method for digital combinatorial circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 5, pp. 566–581, Oct. 2002.

[33] Morgenshtein, Arkadiy, Alexander Fish, and Israel A. Wagner. "Gate-diffusion input (GDI)-a technique for low power design of digital circuits: analysis and characterization." In 2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353), vol. 1, pp. I-I. IEEE, 2002.

[34] S. C. Smith, “Design of an FPGA logic element for implementing asynchronous null convention logic circuits,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 15, no. 6, pp. 672–683, Jun. 2007.

[35] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, “Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits,” Proceedings of the IEEE, vol. 91, no. 2, pp. 305–327, Feb. 2003.

IV. ENERGY-PERFORMANCE SCALABILITY ANALYSIS OF A NOVEL QUASI-STOCHASTIC COMPUTING APPROACH

ABSTRACT

Stochastic computing (SC) is an emerging low-cost computation paradigm for efficient approximation. It processes data in forms of probabilities and offers excellent progressive accuracy. Since SC's accuracy heavily depends on the stochastic bitstream length, generating acceptable approximate results while minimizing the bitstream length is one of the major challenges in SC, as energy consumption tends to linearly increase with bitstream length. To address this issue, a novel energy-performance scalable approach based on quasi-stochastic number generators is proposed and validated in this work. Compared to conventional approaches, the proposed methodology utilizes a novel algorithm to estimate the computation time based on the accuracy. The proposed methodology is tested and verified on a stochastic edge detection circuit to showcase its viability. Results prove that the proposed approach offers a 12% – 60% reduction in execution time and a 12% – 78% decrease in the energy consumption relative to the conventional counterpart. This excellent scalability between energy and performance could be potentially beneficial to certain application domains such as image processing and machine learning, where power and time-efficient approximation is desired.

Index Terms— stochastic computing; energy-performance scalability; low discrepancy sequence

1. INTRODUCTION

With rapidly advancing technology, energy efficiency has become one of the major design challenges in digital circuits and systems. Studies demonstrate that energy efficiency can be improved by reducing both the computational time and power consumption [1]. However, reducing these factors affects the performance of the system. In other words, reducing the power consumption affects the overall performance of the system. This challenge intensifies the current demand for low-power high-performance systems, and therefore a novel methodology to handle this challenge is required. One such promising technique that exploits probability theory "stochastic computation" can address these limitations [1]. Stochastic computing (SC), which was invented in the 1960s by Gaines [2, 3], recently regained significant attention mainly due to its approximate computation method. This computation method offers progressive accuracy scalability [4] that can be well exploited in the applications where approximated accuracy is accepted. This includes media processing, neural networks, factor graphs, LDPC codes, fault-tree analysis, image processing, and filters [5–10]. However, mainstream adoption of SC is limited due to the long run-time and inaccuracy [1]. As explained in [11], a random number generator (RNG), also known as a stochastic number generator (SNG), plays a significant role in determining the area and energy consumption. The commonly used SNG is the linear feedback shift register (LFSR), and several optimization techniques to improve the output accuracy of the LFSR-based SNGs are presented in the literature [12–18]. As presented in [19], increasing the length of stochastic sequences (SS) increases operating time and power consumption.

To address this issue, [11] introduced a quasi-stochastic bit sequence generation (QSNG) that utilizes the distributed memory elements of a field-programmable gate array (FPGA) for designing the SNGs. However, no comment on energy reduction has been reported in [11]. Therefore, in this work a detailed analysis and methodology for energy reduction is presented to improve the overall performance. In this paper, a novel energy-performance scalable methodology based on quasi-stochastic number generators is proposed and validated. Compared to the conventional approaches, the proposed methodology utilizes a novel algorithm to estimate the computation time based on the accuracy. Finally, a comprehensive simulation-based study is presented in this paper to demonstrate the reductions in operating time and energy consumption. Overall, a 12% – 60% reduction in the operating time and a 12% –78% saving in terms of the energy consumption relative to the conventional LFSR counterpart are observed. This paper is organized as follows. In Section 2, background of Stochastic computing and quasi-stochastic bit sequence generation are discussed. Section 3 provides a novel energy-efficient quasi-stochastic computing algorithm to calculate the number of clock cycles based on the peak signal-to-noise ratio. The simulation results to validate the proposed approach are presented in Section 4. Finally, Section 5 asserts the conclusion.

2. BACKGROUND

2.1. STOCHASTIC COMPUTING

SC is a computation technique that uses finite length binary bitstreams to encode stochastic numbers [19]. The length of the bitstream and the number of 1s and 0s in the binary bitstream determine the encoded probability value [1]. The basic circuits used in stochastic computation are shown in Figure 1. The operation of these circuits rely on the type of number interpretations, namely unipolar (UP), bipolar (BP) or inverted bipolar (IBP) formats as presented in [19]. The unipolar format represents the real number x in the range of [0, 1], using bipolar x is represented in between [−1, 1] and IBP ranges from [−1, 1], where the Boolean values 0 and 1 are represented as 1 and −1 in the stochastic number (SN) [11]. Detailed explanations of various SN formats are clearly discussed in [14].

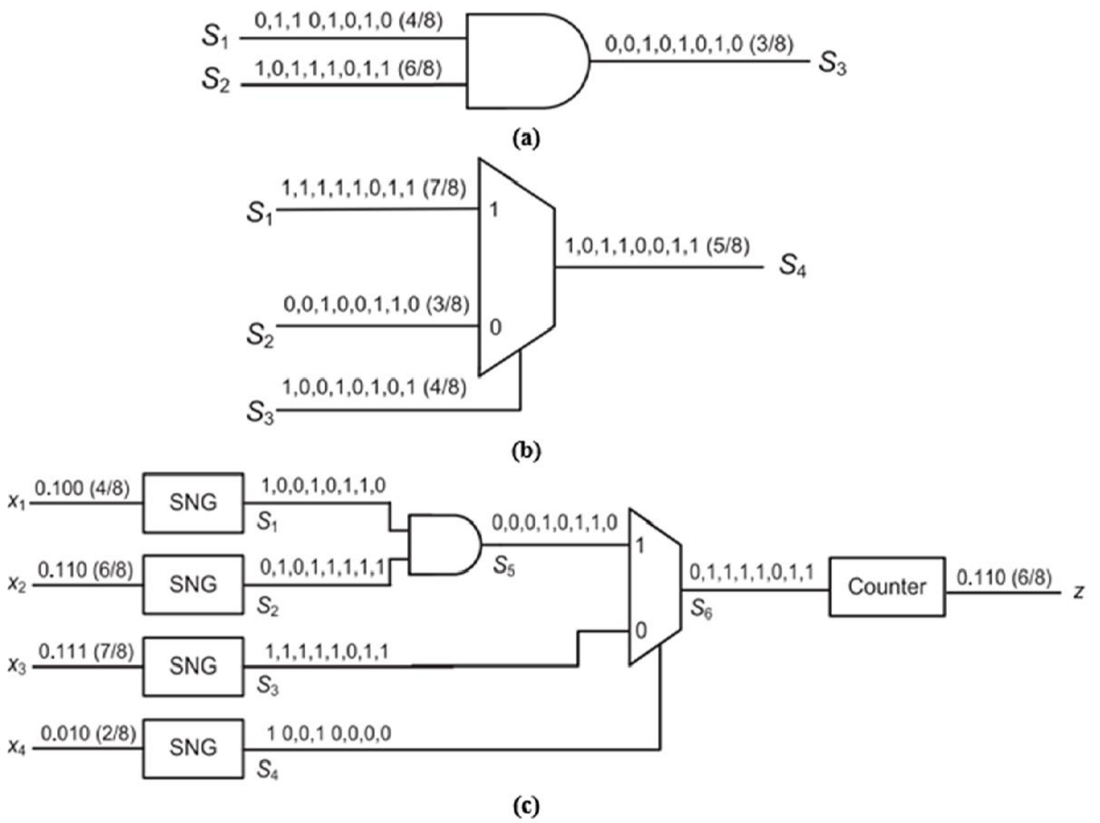


Figure 1. Basic circuits used in stochastic computation: (a) AND gate used as a stochastic multiplier. (b) Multiplexer used as a scaled stochastic adder. (c) Stochastic circuit for realizing the arithmetic function z = x1x2x4 + x3(1 − x4) [19]

The probability value in SC is represented by a binary bitstream of 0s and 1s with specific length L [19]. For the binary representation of 0.5, in the bitstream of length L, half of the bits are represented by 1s and the other half with 0s [11]. For example, one way of representing 0.5 with a bitstream of 8 bits is 01010101. Dependency or correlation between the inputs also plays an important role in representing a stochastic number [19].This inherent feature of SC limits its performance over certain applications compared to conventional binary implementations [20]. For example, an AND gate is used as a multiplier in SC. If two input SS (x and y, namely) are identical (e.g., x = y = 01012 = 0.5), output z will also be 01012 = 0.5, which is not an accurate result because the accurate output stochastic bitstream should have three 0's and one 1 (i.e., 0.25). Another extreme case can happen when $x = \bar{y}$, where output will be 0000 = 0.0.

As shown in these two examples, stochastic bitstream length should be large enough to have the output stream to converge to an accurate value. Therefore, SC is considered to be viable for applications such as image processing and machine learning where fast and efficient approximate computation is desired. To achieve acceptable accuracy, bitstream length L should be large enough to have the final result converged to a value with acceptable approximation error. To address this limitation, a new approach quasi-stochastic bit sequence generation, leveraging FPGA implementation of low-discrepancy (LD) bitstreams for faster convergence has been proposed in [11].

2.2. QUASI-STOCHASTIC BIT SEQUENCE GENERATION

In this approach, the LD sequence and distributed memory elements of the FPGAs (i.e., the LUTs are used for designing the SNGs) [11]. Compared to conventional hardware pseudo random number generation scheme such as LFSR methodology, LD sequences prevent the occurrence of random fluctuation by uniformly spicing the 0s and 1s in the stochastic bit streams [21]. They allow a fraction of the points inside any subset of [0, 1) to be as close as possible, such that uniformity is maintained between the low-discrepancy points [11]. This helps to reduce gaps and clustering points as illustrated in Figure 2.

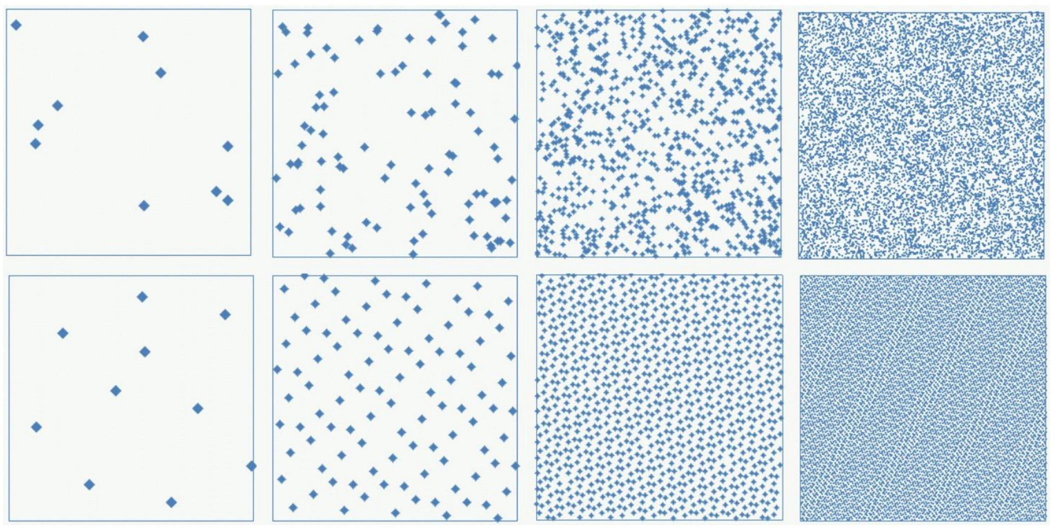


Figure 2. Distribution of pseudo-random points (top) and LD points (bottom) in the unit square [22]

In this QSNG methodology, the stochastic sequence is obtained by multiplying the pre-computed fixed direction vectors with binary numbers [11]. The general structure for generating the binary base two LD sequence consists of bit-wise XOR gates, a multiplication circuit, and RAM to store the directional vectors. In the multiplication circuit, each bit from the counter output is multiplied by each n-bit direction vector to produce n-bit intermediate direction vectors [11].

The bit-wise XOR-ing of these n-bit intermediate direction vectors will result in n-bit LD sequence. At the comparator, these LD sequences are compared with the input binary numbers to generate stochastic number [11]. For example, to generate an SS of bit length of 256 (2^8), eight-bit length direction vectors, which can generate an eight-bit length LD sequence every clock cycle, are required. In summary, SNG plays an important role in determining the SC properties such as size and computation time. In a LFSR-based SNG, L clock cycles are required to fully generate an SS of length L bits [19]. On the other hand, the length of the SS in QSNG methodology determines the size of the binary counter, which in turn determines the computation time [11]. With energy becoming the predominant factor in the current computing systems, novel techniques to address this limitation is required.

Therefore, the primary focus of this work is to present an energy-efficient SC approach for image processing application based on the proposed QSNG methodology. A systematic approach called EQSNG (energy-efficient quasi-stochastic number generation) is proposed that minimizes the energy consumption by detecting the lowest number of clock cycles for a specified accuracy. This methodology is used to assess SC's accuracy in various test images. To the best of our knowledge, this is the only SC design that outperforms its conventional LFSR-based SC in terms of energy-performance scalability. The energy-performance scalability of SC based on QSNG is discussed in detail in Section 3.

3. ENERGY PERFORMANCE SCALABILITY OF NOVEL QUASI-STOCHASTIC COMPUTING APPROACH

We begin this section by discussing major factors affecting the accuracy of a processed image. Next, the effect of computation time on accuracy and energy consumption is demonstrated. Lastly, the proposed energy efficient algorithm that introduces energy-performance scalability in SC is discussed in detail. In most of the image processing techniques, the quality of the processed image is determined by its accuracy. Accuracy can be quantified using several error metrics, such as maximum error, mean square error (MSE), and so on [23]. In this work, PSNR is used to quantify the acceptability of noisy image. It is measured in the unit of dB and determines the similarities between two images (e.g., input image and processed output image). PSNR value can be calculated by Equation 1 [23]:

$$PSNR = 10\ log_{10}\frac{MAX_I^2}{MSE} \tag{1}$$

where $MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|\ \mathrm{I(i,j)} - \mathrm{K(i,j)}\ |^{\ 2}$ is the mean square error between the error-free and the erroneous image, $\mathrm{MAX_I}$ is the maximum image pixel value (e.g., 255 in 8-bit grayscale image), m and n represent the width and height of the target image in terms of the number of pixels, and I(i, j) and K(i, j) represent the pixel values of the error-free image and the erroneous/noisy image, respectively. For the gray scale images, MSE is determined based on their brightness values.

As seen from Equation 1, $\mathrm{MAX_I}$ plays an important role in determining the accuracy of the image and the length of the SS. According to [11, 19], high precision (in terms of accuracy) output can be achieved when an SC circuit operates on a large number of stochastic bit streams. Since each bit of an SS takes a clock cycle to be processed, computation time linearly increases with the increase in the size of the stochastic bit stream. Therefore, with increasing accuracy, computation time tends to increase. Note the computation time refers to the total number of clock cycles required to generate output SS. In physics, power is how fast energy is used or transmitted and power is calculated as the amount of energy divided by the time it took to use the energy. Its unit is the watt, which is one joule per second of energy used. Likewise, power is the amount of energy used per each unit time (i.e., 126 clock cycles) in a clocked digital circuits. Then, energy can be calculated by multiplying power by the total number of clock cycles used. Therefore, the number of clock cycles and energy consumption are proportional. In a conventional digital circuit designed to process data given in binary radix encoding, energy-performance scalable computing is quite limited, as the total number of clock cycles needed to process inputs to generate output is solely determined by how the circuit is designed and optimized. Also, power consumed per clock cycle is purely dependent upon the complexity of the circuit. Besides, stochastic computing has much higher inherent potential for efficient utilization of energy-performance scalability. The term energy-performance scalability in this paper refers to the fact that when accuracy is high, energy consumption will be high. However, for many image processing applications, a desirable accuracy is more than enough. Therefore, savings in energy can be achieved for acceptable accuracy. If more clock cycles are used, more energy will be needed, but higher quality output will result and vice versa. Such an inherent tradeoff can be beneficial in certain application domains such as image processing and artificial neural networks where quick low-power approximation is desired. The proposed quasi-stochastic computing approach is to address the slow convergence problem of conventional Stochastic computing while offering excellent energy-performance scalability.

To prove that the proposed approach is viable, an edge detection scheme is performed on the gray scale image “clock.” The impact of computation time on accuracy and energy is depicted in Figure 3. As seen from the graph, the accuracy in terms of PSNR of the image and the energy consumption tends to increase linearly with the number of clock cycles. Hence, it is practical to choose the minimum number of clock cycles that can satisfy the minimum required accuracy for the best possible energy-performance balance. To address this energy-accuracy trade-off, we propose an energy-accuracy scalable EQSNG design that can determine the number of iterations based on the acceptable PSNR threshold for an image.

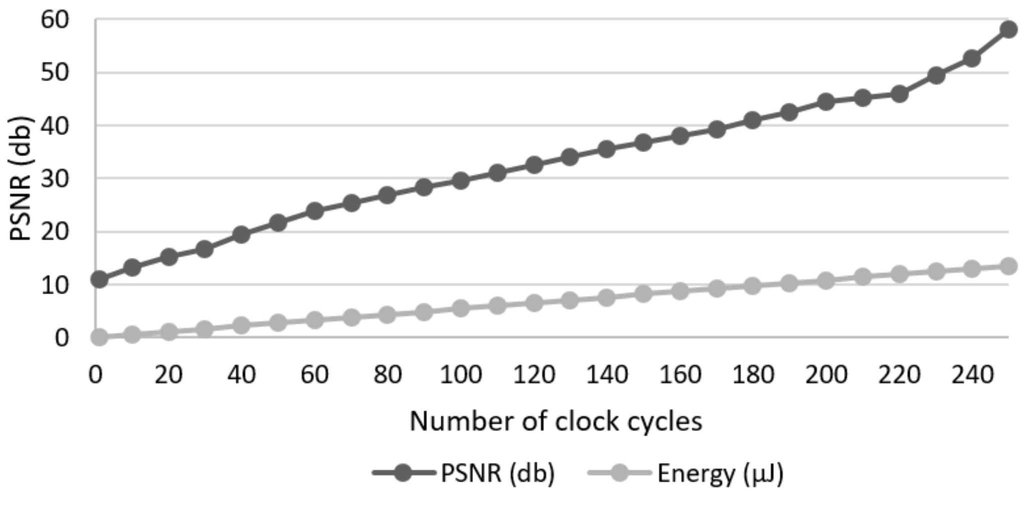


Figure 3. Accuracy and energy consumption during edge detection of clock test image

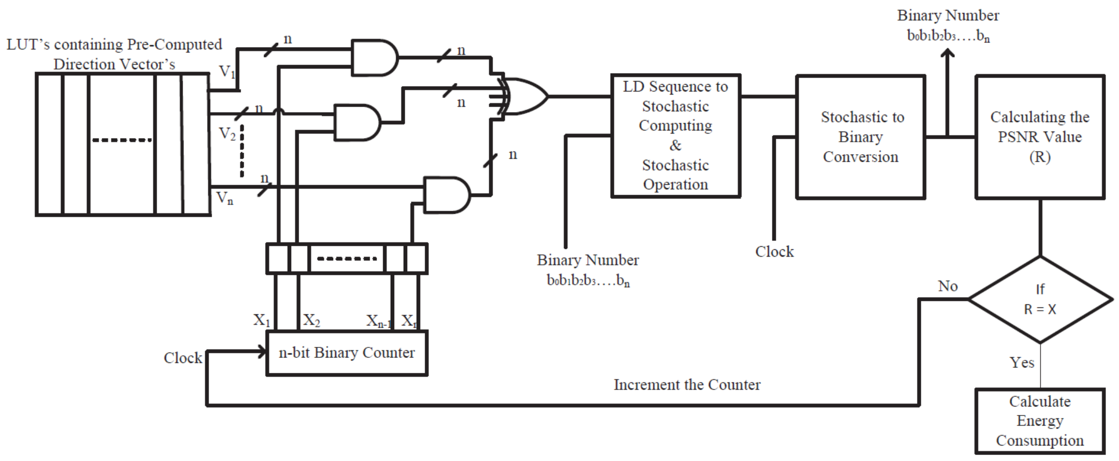

LUT's containing Pre-Computed Direction Vector's
V1
V2
Vn
n
n-bit Binary Counter
Clock
X1
X2
Xn-1
Xn
LD Sequence to Stochastic Computing & Stochastic Operation
Binary Number b0b1b2b3….bn
Stochastic to Binary Conversion
Clock
Binary Number b0b1b2b3….bn
Calculating the PSNR Value (R)
If R = X
No
Yes
Calculate Energy Consumption
Increment the Counter

Figure 4. Structure of EQSNG

The acceptability of the target image can be achieved by just comparing the equivalent error rate with the corresponding acceptable error rate threshold. This acceptable error rate threshold is assumed to be a user-defined value in this work. The general design of the energy efficient QSNG model (EQSNG) is depicted in Figure 4. The optimal number of iterations is calculated based on the user-defined peak signal-to-noise ratio (PSNR). The process to estimate optimal number of iterations is shown in Algorithm 1. The first step is to store the pre-computed direction vectors in the random-access memory (e.g., look up tables of the FPGA). Then, each bit of n-bit directional vectors is multiplied with the n-bit binary counter output using an AND gate. The resulting binary numbers are XORed up to obtain the final LD sequence. This LD sequence is compared to the binary input value to generate an SS on which stochastic operations are performed. The resultant stochastic output is again converted to binary number at the stochastic binary conversion block. This post-processed binary output is processed in MATLAB to determine the image quality (i.e., accuracy).

To determine the image accuracy, the mean square error (MSE) that accurately measures the error in the reference image is calculated first. The resultant MSE value is used for calculating PSNR ($PSNR_{Current}$). If the calculated PSNRCurrent is less than the user defined target PSNR value ($PSNR_{Target}$), the counter is incremented and the whole process is carried out till the desired PSNR is achieved. Since the counter is incremented by increasing the clock cycles, the total energy consumption is calculated by multiplying the power by the number of clock cycles. As the proposed approach can converge at a much faster rate, they require few clock cycles to achieve the desired PSNR value, which in turn further reduces the energy consumption.

Algorithm 1: EQSNG Algorithm

```
Data: $Image_{Input}$, $PSNR_{Target}$
Result: Energy, ClockCycles
Initialization;
ClockCycles = 0;
PSNRCurrent = 0;
LOOP: if $PSNR_{Current} < PSNR_{Target}$ then
    if ClockCycles == 0 then
        Calculate Power;
    end
    ClockCycles += 1;
    Generate $Image_{Output}$;
    Calculate $PSNR_{Current}$;
    Go to Loop
else
    $Energy = Power \times ClockCycles$;
end
```

Hence, the proposed approach provides acceptable image quality with fewer clock cycles and less energy consumption. Compared to the conventional SC approach based on LFSR, the EQSNG methodology can generate an acceptable quality edge detection image with excellent energy efficiency. To demonstrate and verify the energy-performance scalability of the EQSNG approach, the proposed methodology is implemented on a stochastic edge detection circuit for 8-bit grayscale image processing. In the next section, the proposed methodology is applied to several test images and comparative results are presented and analyzed.

4. SIMULATION-BASED ENERGY-PERFORMANCE SCALABILITY ANALYSIS

This section compares the results for various test images implemented using conventional LFSR and EQSNG approaches. These test images on which edge detection is performed are shown in Figure 5, which are called clock, crowd, and aerial. The edge detection circuit based on Robert’s cross algorithm [5] was used for the proposed energy-performance scalability analysis. To study the impact of the proposed approach on energy consumption, target PSNR values are arbitrarily selected. Next, the computation time (i.e., number of clock cycles) required to achieve the specified accuracy is determined and corresponding energy consumption is calculated.

The circuits have been realized on a Xilinx Virtex 4 SF FPGA (XC4VLX15) device and synthesized using Xilinx ISE 12.1 design suite. The QSNG uses the LD sequence and distributed memory elements (LUTS) of the FPGAs for designing the SNGs. Therefore, an FPGA is used. The performance of the proposed technique has been extensively evaluated using a 8-bit grayscale images (i.e., each pixel value is represented using a stochastic bit-length of $2^8 = 256$ bits) as an example in this section. A cycle-accurate simulator has been implemented in MATLAB to generate simulation results for the proposed technique. The pixel values of the images were extracted using MATLAB and were given as the 8-bit binary input to the stochastic edge detection circuit. Then, the output extracted from the post-synthesis simulation results was processed in MATLAB to determine the accuracy.

Figure 5. Open source test images used for edge detection: (a) clock (b) crowd (c) aerial

The circuits have been realized on a Xilinx Virtex 4 SF FPGA (XC4VLX15) device and synthesized using Xilinx ISE 12.1 design suite. The QSNG uses the LD sequence and distributed memory elements (LUTS) of the FPGAs for designing the SNGs. Therefore, an FPGA is used. The performance of the proposed technique has been extensively evaluated using a 8-bit grayscale images (i.e., each pixel value is represented using a stochastic bit-length of $2^8 = 256$ bits) as an example in this section. A cycle-accurate simulator has been implemented in MATLAB to generate simulation results for the proposed technique. The pixel values of the images were extracted using MATLAB and were given as the 8-bit

binary input to the stochastic edge detection circuit. Then, the output extracted from the post-synthesis simulation results was processed in MATLAB to determine the accuracy.

To quantitatively demonstrate and verify the performance of the proposed approach, energy consumption is determined by using the following simulation parameters: 8-bit grayscale images and its desired PSNR value. Table 1 shows the number of clock cycles and energy consumed for achieving the desired quality of image. As per the results shown in the table, energy consumption for the proposed EQSNG methodology is significantly lower than the traditional approach (LFSR) for the same target PSNR. As seen from the Table 1, the number of clock cycles for EQSNG to achieve the desired quality of the image is considerably less than LFSRs. The proposed EQSNG implementation of the edge detection circuit reduces the computation time by a factor of 3.5 times on average when compared to LFSR based approach. For instance, to achieve a PSNR of 31.53 dB for the Aerial test image, the energy consumed by the EQSNG and LFSR approach are 0.14 µJ and 0.63 µJ, which is a substantial saving.

To quantitatively demonstrate and verify the performance of the proposed approach, energy consumption is determined by using the following simulation parameters: 8-bit grayscale images and their desired PSNR value. Table 1 shows the number of clock cycles and amount of energy consumed for achieving the desired quality of image. As per the results shown in the table, energy consumption for the proposed EQSNG methodology is significantly lower than the traditional approach (LFSR) for the same target PSNR. As seen from Table 1, the number of clock cycles for EQSNG to achieve the desired quality of the image is considerably less than LFSRs. The values in

Table 1, are obtained by designing both the LFSR and EQSNG models and verified via simulation studies.

The proposed EQSNG implementation of the edge detection circuit reduces the computation time by a factor of 3.5 times on average when compared to the LFSR based approach. For instance, to achieve a PSNR of 31.53 dB for the aerial test image, the energy consumed by the EQSNG and LFSR approach are 0.14 µJ and 0.63 µJ, which is a substantial saving. Therefore, the energy consumption reduces by 77.7%. Similarly, the energy consumed by LFSR and EQSNG methodologies to achieve a PSNR of 28 dB for the clock test image is 0.054 µJ and 0.05 µJ energy. Thus, the proposed approach reduces energy consumption by 12.2% as presented. Compared to the LFSR approach, for the Crowd test image with a PSNR of 40.30 dB, the EQSNG approach saves about 18.6% of energy. The, reduction in energy consumption for various PSNR values by using the proposed approach is depicted in Figure 6.

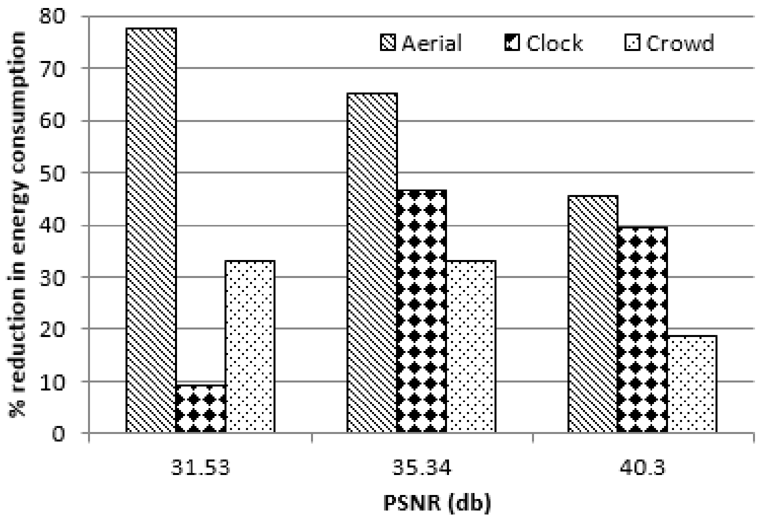


Figure 6. Reduction in energy consumption for various PSNR values using EQSNG methodology compared to LFSR approach

Table 1. Table showing the no of clock cycles and energy consumption for various PNSR

Test Image	Approach		Target PSNR (dB)					
			22.6	**25.13**	**28.13**	**31.53**	**35.34**	**40.30**
Aerial	EQSNG	# of clk cycles	7	10	14	26	47	77
		Energy (µJ)	0.038	0.054	0.0076	0.14	0.25	0.419
	LFSR	# of clk cycles	17	23	100	198	225	240
		Energy (µJ)	0.054	0.073	0.32	0.63	0.72	0.768
Clock	EQSNG	# of clk cycles	4	7	10	19	30	53
		Energy (µJ)	0.022	0.038	0.05	0.1	0.16	0.29
	LFSR	# of clk cycles	4	10	18	37	95	151
		Energy (µJ)	0.013	0.032	0.057	0.11	0.3	0.48
Crowd	EQSNG	# of clk cycles	8	13	18	28	45	80
		Energy (µJ)	0.043	0.07	0.098	0.15	0.24	0.43
	LFSR	# of clk cycles	14	22	50	70	112	165
		Energy (µJ)	0.044	0.07	0.16	0.224	0.36	0.53

The proposed EQSNG implementation of the edge detection circuit reduces the computation time by a factor of 3.5 times on average when compared to the LFSR based approach. For instance, to achieve a PSNR of 31.53 dB for the aerial test image, the energy consumed by the EQSNG and LFSR approach are 0.14 µJ and 0.63 µJ, which is a substantial saving. Therefore, the energy consumption reduces by 77.7%. Similarly, the energy consumed by LFSR and EQSNG methodologies to achieve a PSNR of 28 dB for the clock test image is 0.054 µJ and 0.05 µJ energy. Thus, the proposed approach reduces energy consumption by 12.2% as presented. Compared to the LFSR approach, for the Crowd test image with a PSNR of 40.30 dB, the EQSNG approach saves about 18.6% of energy. The, reduction in energy consumption for various PSNR values by using the proposed approach is depicted in Figure 6.

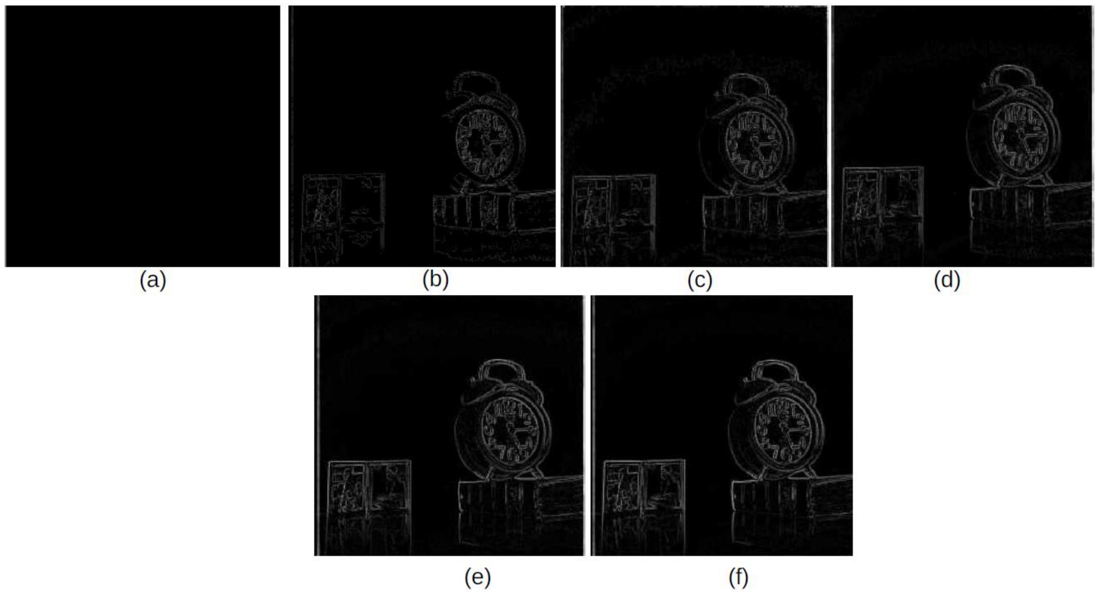


Figure 7. Edge detection on the clock test image using the proposed EQSNG SC apporach: (a) PSNR = 22.2 dB; 4 clock cycles. (b) PSNR = 25.13 dB; 7 clock cycles. (c) PSNR = 28.12 dB; 10 clock cycles. (d) PSNR = 31.53 dB; 18 clock cycles. (e) PSNR = 35.34 dB; 35 clock cycles. (f) PSNR = 40.30 dB; 55 clock cycles

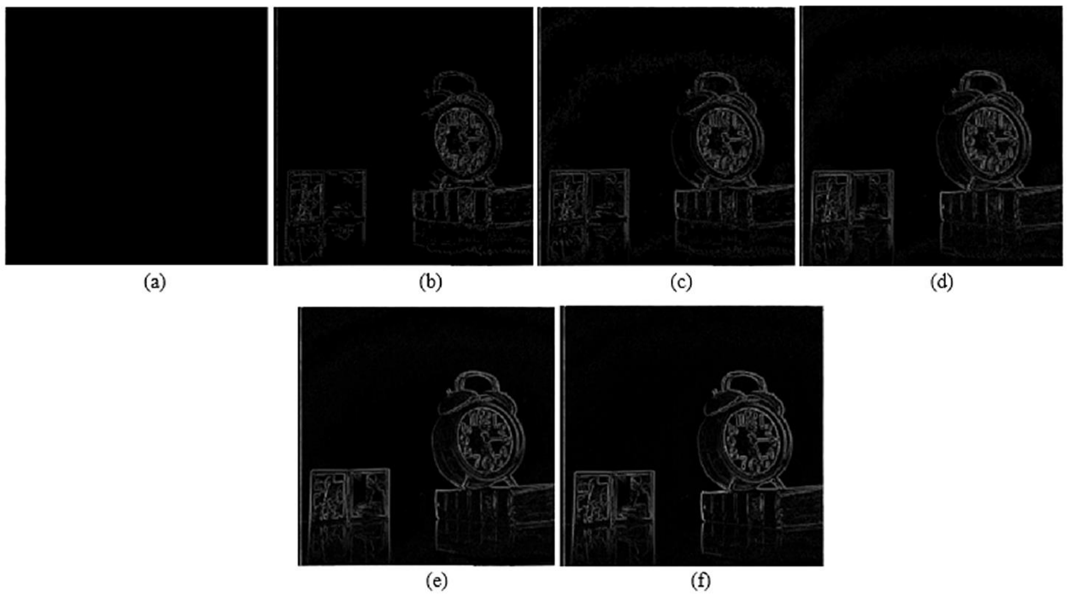


Figure 8. Edge detection on the clock test image using conventional LFSR-based SC apporach: (a) PSNR = 22.2 dB; 4 clock cycles. (b) PSNR = 25.13 dB; 10 clock cycles. (c) PSNR = 28.12 dB; 18 clock cycles. (d) PSNR = 31.53 dB; 37 clock cycles. (e) PSNR = 35.34 dB; 95 clock cycles. (f) PSNR = 40.30 dB; 151 clock cycles

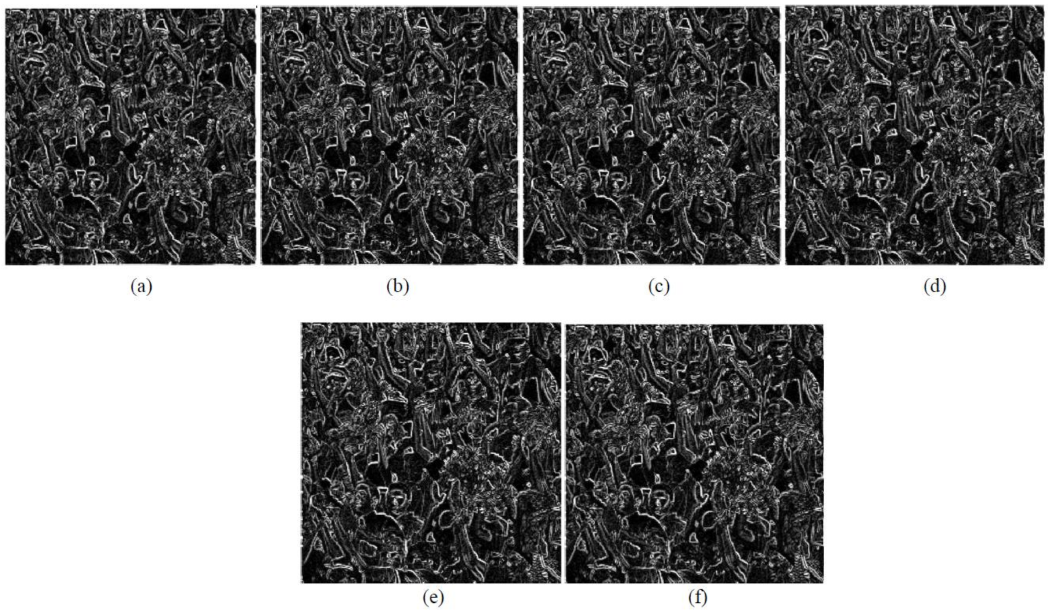


Figure 9. Edge detection on the crowd test image using the proposed EQSNG SC apporach: (a) PSNR = 22.2 dB; 8 clock cycle. (b) PSNR = 25.13 dB; 13 clock cycles. (c) PSNR = 28.12 dB; 18 clock cycles. (d) PSNR = 31.53 dB; 28 clock cycles. (e) PSNR = 35.34 dB; 45 clock cycles. (f) PSNR = 40.30 dB; 80 clock cycles

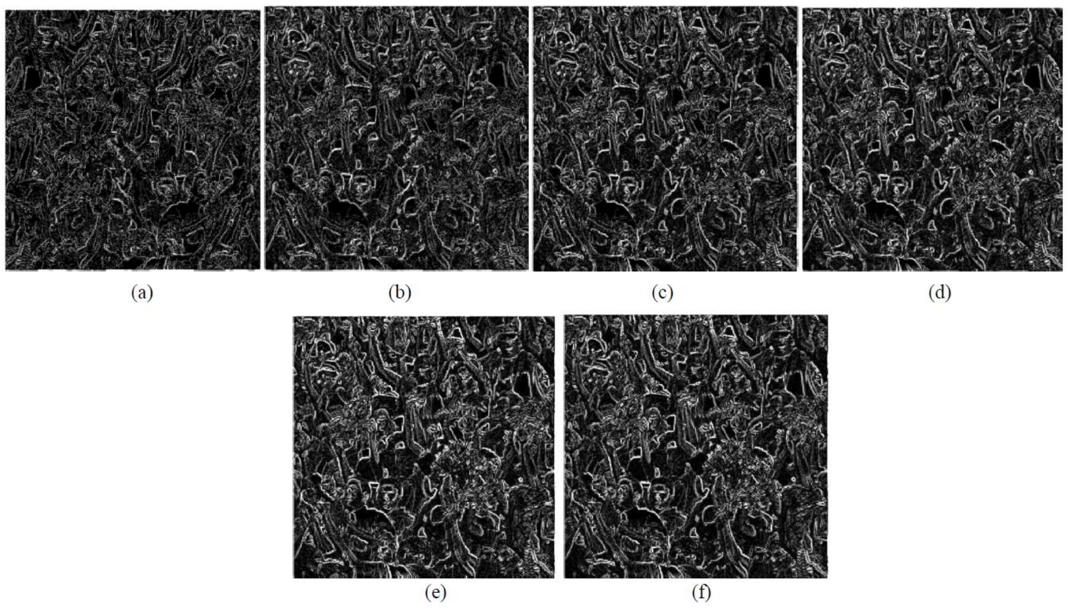


Figure 10. Edge detection on the crowd test image using conventional LSFR-based SC apporach: (a) PSNR = 22.2 dB; 14 clock cycles. (b) PSNR = 25.13 dB; 22 clock cycles. (c) PSNR = 28.12 dB; 50 clock cycles. (d) PSNR = 31.53 dB; 70 clock cycles. (e) PSNR = 35.34 dB; 112 clock cycles. (f) PSNR = 40.30 dB; 165 clock cycles

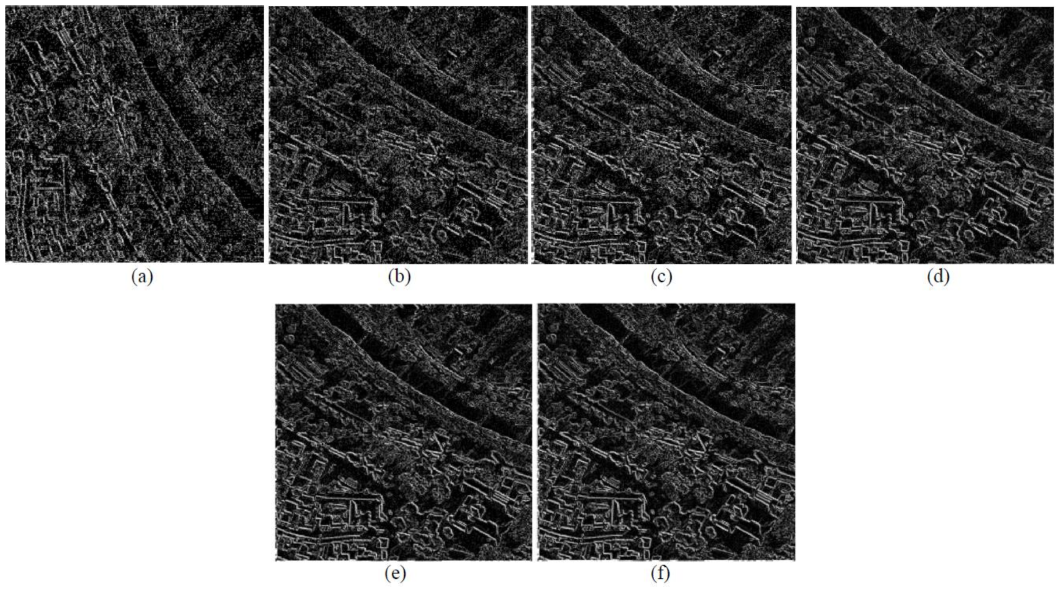


Figure 11. Edge detection on the aerial test image using the proposed EQSNG SC apporach: (a) PSNR = 22.2 dB; 7 clock cycles. (b) PSNR = 25.13 dB; 10 clock cycles. (c) PSNR = 28.12 dB; 15 clock cycles. (d) PSNR = 31.53 dB; 26 clock cycles. (e) PSNR = 35.34 dB; 47 clock cycles. (f) PSNR = 40.30 dB; 77 clock cycles

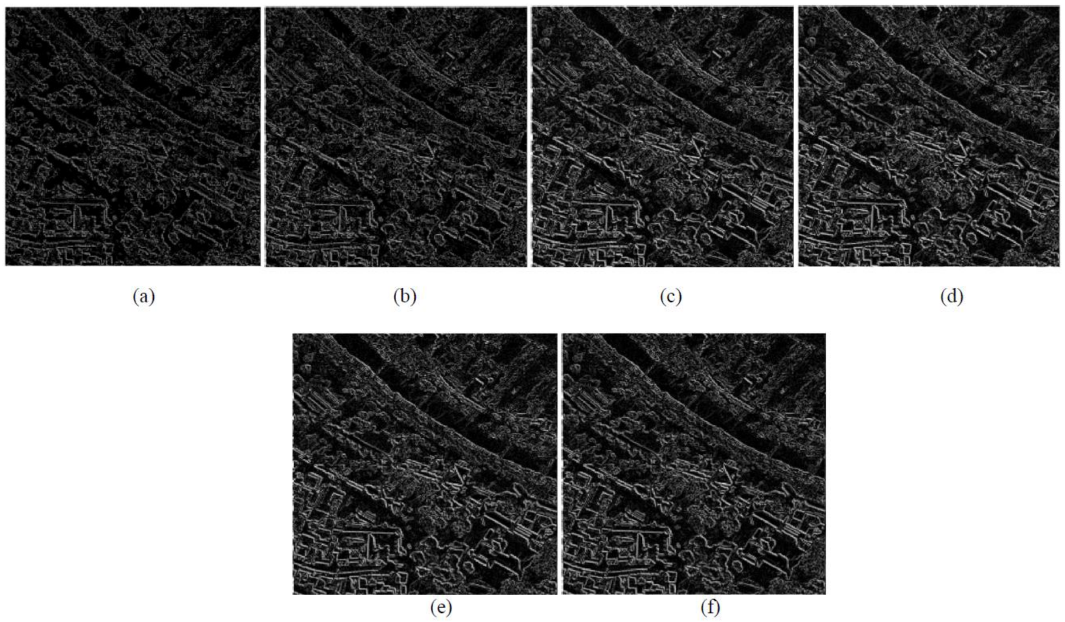


Figure 12. Edge detection on the aerial test image using conventional LSFR-based SC apporach: (a) PSNR = 22.2 dB; 17 clock cycles. (b) PSNR = 25.13 dB; 23 clock cycles. (c) PSNR = 28.12 dB; 100 clock cycles. (d) PSNR = 31.53 dB; 198 clock cycles. (e) PSNR = 35.34 dB; 225 clock cycles. (f) PSNR = 40.30 dB; 248 clock cycles

From Table 1, it should be noticed that as the PSNR (i.e., accuracy) increases the number (#) of clock cycles utilized also increases. Therefore, the higher the computation time, the better the quality of the image as illustrated in Figures 7–12. These figures show that the proposed approach utilizes a smaller number of clock cycles to achieve the same accuracy as the LFSR approach due to faster stochastic value convergence. Therefore, using the proposed EQSNG methodology, execution time and energy consumed can be reduced while achieving an acceptable level of accuracy.

In summary, 12%–78% reduction in the energy consumption is observed. Moreover, compared to LFSR based approach, the proposed EQSNG implementation on average reduces the computation time by a factor of 2.5 times. This excellent energy-quality scalability of the proposed approach may also be beneficial to the other application domains (e.g., signal processing, machine vision, and deep learning) where efficient reduced-precision computation is desired.

5. CONCLUSION

In this paper, a novel EQSNG is introduced and verified via extensive simulation-based analysis where low computation time and energy consumption are achieved. The proposed approach is efficient enough to offer 12–60% reduction in execution time and a 12–78% decrease in energy consumption relative to the conventional LFSR counterpart. This considerable enhancement in terms of time and energy will further promote the viability of SC over conventional approaches in application domains such as image processing and machine learning where low-power approximation fast is desired.

BIBLOGRAPHY

[1] Alaghi, A.; Qian, W.; Hayes, J.P. The promise and challenge of stochastic computing. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 2017, 37, 1515–1531.

[2] Gaines, B.R. Stochastic computing. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NY, USA, 18–20 April 1967; ACM: New York, NY, USA, 1967; pp. 149–156.

[3] Gaines, B. Stochastic computing systems. In Advances in Information Systems Science; Springer: Berlin, Germany, 1969; pp. 37–172.

[4] Moons, B.; Verhelst, M. Energy-Efficiency and Accuracy of Stochastic Computing Circuits in Emerging Technologies. Emerg. Sel. Top. Circuits Syst. IEEE J. 2014, 4, 475–486.

[5] Alaghi, A.; Li, C.; Hayes, J.P. Stochastic circuits for real-time image-processing applications. In Proceedings of the 50th Annual Design Automation Conference, Austin, TX, USA, 29 May–7 June 2013; ACM: New York, NY, USA, 2013, p. 136.

[6] Naderi, A.; Mannor, S.; Sawan, M.; Gross, W.J. Delayed stochastic decoding of LDPC codes. IEEE Trans. Signal Process. 2011, 59, 5617–5626.

[7] Aliee, H.; Zarandi, H.R. Fault tree analysis using stochastic logic: A reliable and high speed computing. In Proceedings of the IEEE 2011 Proceedings-Annual Reliability and Maintainability Symposium (RAMS), Lake Buena Vista, FL, USA, 24–27 January 2011; pp. 1–6.

[8] Li, P.; Lilja, D.J. Using stochastic computing to implement digital image processing algorithms. In Proceedings of the 2011 IEEE 29th International Conference on Computer Design (ICCD), Amherst, MA, USA, 9–12 October 2011; pp. 154–161.

[9] Chang, Y.N.; Parhi, K. Architectures for digital filters using stochastic computing. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada, 26–31 May 2013; pp. 2697–2701.

[10] Saraf, N.; Bazargan, K.; Lilja, D.J.; Riedel, M.D. IIR filters using stochastic arithmetic. In Proceedings of the 2014 IEEE Design, Automation and Test in Europe Conference and Exhibition (DATE), Dresden, Germany, 24–28 March 2014; pp. 1–6.

[11] Seva, R.; Metku, P.; Choi, M. Energy-Efficient FPGA-Based Parallel Quasi-Stochastic Computing. J. Low Power Electron. Appl. 2017, 7, 29.

[12] Li, P.; Lilja, D.J. Accelerating the performance of stochastic encoding-based computations by sharing bits in consecutive bit streams. In Proceedings of the 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), Washington, DC, USA, 5–7 June 2013; pp. 257–260.

[13] Ichihara, H.; Ishii, S.; Sunamori, D.; Iwagaki, T.; Inoue, T. Compact and accurate stochastic circuits with shared random number sources. In Proceedings of the 2014 32nd IEEE International Conference on Computer Design (ICCD), Seoul, Korea, 19–22 October 2014; pp. 361–366.

[14] Alaghi, A.; Hayes, J.P. A spectral transform approach to stochastic circuits. In Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, 30 September–3 October 2012; pp. 315–321.

[15] Alaghi, A.; Hayes, J. STRAUSS: Spectral Transform Use in Stochastic Circuit Synthesis. IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst. 2012, 34, 1770–1783.

[16] Chen, T.H.; Hayes, J.P. Equivalence among Stochastic Logic Circuits and its Application to Synthesis. IEEE Trans. Emerg. Top. Comput. 2016, 7, 67–79.

[17] Kwok, S.H.; Lam, E.Y. FPGA-based high-speed true random number generator for cryptographic applications. In Proceedings of the TENCON 2006—2006 IEEE Region 10 Conference, Hong Kong, China, 14–17 November 2006; pp. 1–4.

[18] Majzoobi, M.; Koushanfar, F.; Devadas, S. FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control; CHES; Springer: Berlin, Germany, 2011; pp. 17–32.

[19] 9. Alaghi, A.; Hayes, J.P. Survey of stochastic computing. ACM Trans. Embed. Comput. Syst. (TECS) 2013, 12, 92.

[20] Manohar, R. Comparing Stochastic and Deterministic Computing (accessed on 15 November 2019).

[21] Alaghi, A.; Hayes, J.P. Fast and accurate computation using stochastic circuits. In Proceedings of the Conference on Design, Automation & Test in European Design and Automation Association, Dresden, Germany, 24–28 March 2014; p. 76.

[22] Wikipedia. Low-Discrepancy Sequence. Available online: (accessed on 15 November 2019).

[23] Hsieh, T.Y.; Peng, Y.H.; Ku, C.C. An Efficient Test Methodology for Image Processing Applications Based on Error-Tolerance. In Proceedings of the 2013 22nd Asian Test Symposium, Jiaosi Township, Taiwan, 18–21 November 2013; pp. 289–294.

SECTION

2. CONCLUSION

In this research work methodologies to to improve reliability of 3D heterogeneous processors (3DHP) and to reduce the area overhead of asynchronous designs have been presented. In the first work, a novel adaptive multi-path BCH decoder design approach is proposed and validated to address the bit error variation issue caused by hotspots in 3DHP. The proposed design has multiple decoding paths with variable decoding latency and area trade-off. For each word read from DRAM, thermal gradient data from the on-chip temperature sensors is utilized to estimate the expected number of error bits. Then, the fastest possible decoding path which is able to correct the expected number of error bits is adaptively selected to reduce the overall decoding time. Also, a parallel decoding approach leveraging the multiple independent decoding paths of the proposed decoder design is also proposed and validated in this work.

The next part of this work summarizes how NCL designs realized using convention static CMOS technique causes a large area overhead compared to its synchronous counterparts. To address this limitations, two novel approaches, HYBRID and GNCL were proposed. Both approaches shows a 7% - 14% reduction in the transistor count. Furthermore, GNCL methodology shows a 14% - 30% decrement in the dynamic power consumption when compared to the conventional CMOS NCL counterpart. This considerable enhancement in terms of area and power will further increase the use of NCL in asynchronous digital designs, competing with conventional synchronous designs.

BIBLOGRAPHY

[1] Andrade, Hugo, and Ivica Crnkovic. "A review on software architectures for heterogeneous platforms." In *2018 25th* Asia-Pacific Software Engineering Conference (APSEC), pp. 209-218. IEEE, 2018.

[2] S. Borkar, “Getting Gigascale Chips: Challenges and Opportunities in Continuing Moore’s Law,” Queue, vol. 1, no. 7, pp. 26–33, 2003.

[3] M. Horowitz, “Scaling, Power and the Future of CMOS,” in Proceedings of the 20th International Conference on VLSI Design held jointly with 6th International Conference: Embedded Systems. Washington, DC, USA: IEEE Computer Society, 2007, p. 23.

[4] Lee, Seok-Hee. "Technology scaling challenges and opportunities of memory devices." In *2016 IEEE International Electron Devices Meeting (IEDM)*, pp. 1-1. IEEE, 2016.

[5] B. M. Rogers et al., “Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling,” in ISCA’09: Proceedings of the 36th Annual International Symposium on Computer Architecture. New York, NY, USA: ACM, 2009, pp. 371–382.

[6] S. Mittal and J. S. Vetter, “A survey of cpu-gpu heterogeneous computing techniques,” ACM Comput. Surv. vol. 47, no. 4, pp. 69:1–69:35, Jul. 2015.

[7] R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, “State-of-the-art in heterogeneous computing,” Sci. Program., vol. 18, no. 1, pp. 1–33, Jan. 2010.

[8] S. Damaraju, V. George, S. Jahagirdar, T. Khondker, R. Milstrey, S. Sarkar, S. Siers, I. Stolero, and A. Subbiah, “A 22nm IA multi-CPU and GPU System-on-Chip,” in Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International. IEEE, 2012, pp. 56–57.

[9] K. Bent, “AMD Touts Trinity APU,” CRN, no. 1321, p. 36, 2012.

[10] J. Kim, S. Seo, J. Lee, J. Nah, G. Jo, and J. Lee, “OpenCL as a unified programming model for heterogeneous CPU/GPU clusters,” in Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming. ACM, 2012, pp. 299–300.

[11] J. Lee, J. Kim, S. Seo, S. Kim, J. Park, H. Kim, T. Dao, Y. Cho, S. Seo, S. Lee et al., "An OpenCL framework for heterogeneous multicores with local memory," in Proceedings of the 19th international conference on Parallel architectures and compilation techniques. ACM, 2010, pp. 193–204.

[12] S. Naffziger, "Invited plenary talk: Technology impacts from the new wave of architectures for media-rich workloads," in IEEE VLSI Technology Symposium, 2011.

[13] P. Wilkerson, M.Furmanczyk, and M. Turowski, "Compact Thermal Modeling Analysis for 3D Intergrated Circuits," International Conference Mixed Design of Integrated Circuits and Systems, 2004.

[14] M. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," IBM Journal of Research and Development, vol. 14, no. 4, pp. 395–401, 1970.

[15] R. Bose and D. Ray-Chaudhuri, "On a class of error correcting binary group codes," Information and control, vol. 3, no. 1, pp. 68–79, 1960.

[16] S. M. T. Moreira, M. Arendt, F. G. Moraes, and N. L. V. Calazans, "Static differential ncl gates: Toward low power," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 62, no. 6, pp. 563–567, Jun. 2015.

[17] N. Nemati, P. Beckett, M. C. Reed, and K. Fant, "Clock-less DFT-less test strategy for null convention logic," IEEE Transactions on Emerging Topics in Computing, vol. 6, no. 4, pp. 460–473, Oct. 2018.

[18] J. Sudhakar, Y. Alekhya, and K. S. Syamala, "A dual-rail delay-insensitive ieee-754 single-precision null convention floating point multiplier for low-power applications," in Innovations in Electronics and Communication Engineering. Springer, Jan. 2018.

[19] M. Chang, P. Yang, and Z. Pan, "Register-less null convention logic," IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 64, no. 3, pp. 314–318, Mar. 2017.

[20] K. Fant and S. Brandt, "Null convention logicTM: a complete and consistent logic for asynchronous digital circuit synthesis," International Conference on Application Specific Systems, Architectures and Processors, pp. 261–273, 1996.

[21] K. M. Fant and S. A. Brandt, "Null convention logic: A complete and consistent logic for asynchronous digital circuit synthesis," in Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors, p. 261, 1996.

[22] S. C. Smith, "Speedup of null convention digital circuits using null cycle reduction," Journal of System Architecture, vol. 52, no. 7, pp. 411–422, 2006.

[23] J. McCardle and D. Chester, “Measuring an asynchronous processor’s power and noise,” Proceedings of the Synopsys User Group Conference, 2001.

[24] S. K. Bandapati and S. C. Smith, “Design and characterization of null convention arithmetic logic units,” Microelectron. Eng., vol. 84, pp. 280–287, Feb. 2007.

[25] S. C. Smith, “Design of a null convention self-timed divider,” in The International Conference on VLSI, vol. 1, pp. 447–453, 2004.

[26] L. Zhou and S. Smith, “Speedup of a large word-width high-speed asynchronous multiply and accumulate unit,” in 52nd IEEE International Midwest Symposium on Circuits and Systems, pp. 499 –502, Aug. 2009.

[27] P. A. Beerel, R. O. Ozdag, and M. Ferretti, A designer’s guide to asynchronous VLSI. Cambridge University Press, 2010.

VITA

Prashanthi Metku received her Bachelors Degree in Electronics and Communication Engineering from JNTUH, Hyderabad, India, in 2011. She earned her Masters of Technology Degree in Electronics Engineering from Pondicherry University, Pondicherry, India, in 2014. She recieved her Doctor of Philosophy in Computer Engineering from Missouri University of Science and Technology in May 2020. Her research interests included computer architecture, error correction codes, ASIC design, embedded design and stochastic computing.