Low Complexity and Power Efficient Asynchronous Design Based on NCL Logic

S.Karthikeyan[1], A.Santhosh Kumar[2]
[1]Dept of E.C.E.
[2]Assistant Professor, Dept of E.C.E.
[1, 2] SNS College of Technology, Coimbatore-641035

Abstract- *NULL convention logic (NCL) is a promising design paradigm for constructing low-power robust asynchronous circuits.NCL has been employed for a variety of applications, including low-power circuit design, fault-attack-resistant cryptographic circuits, ternary logic, and robust circuit design for operating in space environment The conventional NCL paradigm requires pipeline registers for separating two neighboring logic blocks, and those registers can account for up to 35% of the overall power consumption of the NCL circuit. This brief presents the register-less NULL convention logic (RL-NCL) design paradigm, which achieves low power consumption by eliminating pipeline registers, simplifying the control circuit, and supporting fine-grain power gating to mitigate the leakage power of sleeping logic blocks. Further, to improve dual-rail complexity and to achieve high-throughput pipeline design method, targeting to latch-free and extremely fine-grain is constructed. The data paths are composed of a mixture of dual-rail and single-rail gates. Dual-rail gates are limited to construct a stable critical data path. Compared with the conventional NCL counterpart, the Hybrid–Rail RL-NCL implementation of an eight-bit five-stage pipelined Kogge-Stone adder can reduce power dissipation for the input data rate ranging from 10 MHz to 900 MHz. Moreover, the Hybrid–Rail RL-NCL implementation can reduce the transistor count of the adder which results in optimized area compared to RL-NCL Kogge-Stone adder.*

Keywords- Asynchronous circuits, low-power electronics, Hybrid-rail, null convention logic, power gating, Kogge-Stone adder.

I. INTRODUCTION

The semiconductor industry is an important consideration for asynchronous circuit technology. Up until now, asynchronous circuits have been applied commercially only as small sub circuits, often as peripherals to controllers. Examples include counters, timers, wakeup circuits, arbiters; interrupt controllers, first-in first-out (FIFO), bus controllers, and interfaces (e.g., RS-232, SCSI, UART). The requirement for asynchronous circuits is mainly from asynchronous specifications. In last decade there will be recovery asynchronous circuits. The importance from asynchronous-in-the-small to asynchronous very large scale integration (VLSI) circuits and systems. Asynchronous VLSI is now progressing from a fashionable academic research topic to a viable solution to number of digital VLSI design challenges. Asynchronous circuits for the signal are binary have been assumed but remove the assumption that time is discrete. This has several possible benefits:

No clock skew - Clock skew is the difference in arrival times of the clock signal at different parts of the circuit. By definition asynchronous circuits have no globally distributed clock, there is no need to worry about clock skew. In contrast, synchronous systems often slow down their circuits to accommodate the skew. As feature sizes decrease, clock skew becomes a much greater concern.

Lower power - Standard synchronous circuits have clock lines, and may be recharge and discharge signals[1]. For example, even though a floating point unit on a processor might not be used in a given instruction stream, the unit still must be operated by the clock.

Today, all digital circuits are synchronous. Global clock is used to define the moment, when all data is ready and can be latched [7]. This seems like a good solution. So, all CAD tools and technologies are oriented to synchronous design. We can see a rapid increasing of complexity of Systems-On-Chips and integration degree of crystals. Now, it's a problem to distribute a clock all over the crystal and ensure suitable clock skew in all points. For this purpose, it is necessary to use special buffers. So, synchronization can takes up to 30 percent of the crystal area and power consumption. This is not the only disadvantage of the synchronous circuits. Global synchronization artificially limits the performance of the circuit. The clock period time can't be less then maximum combination delay all over the circuit. Moreover, synchronization makes all triggers switching, even if there aren't needs to latch new data. This leads to extra power consumption. Now, institutes and companies are searching for principles and methods of digital computing without clock.

In asynchronous design, the choice of handshake protocols affects the circuit implementation (area, speed, power, robustness, etc.). The four-phase bundled-data protocol and the four-phase dual-rail protocol are two famous protocols that are used in most asynchronous circuits. The four phase bundled-data protocol design most closely resembles the design of synchronous circuits. Handshake circuits generate local clock pulses and use delay matching to indicate valid signal. It normally leads to the most efficient circuits due to the extensive use of timing assumptions. On the other hand, the four-phase dual-rail protocol design is implemented in an elaborate way that the handshake signal is combined with the dual-rail encoding of data. Handshake circuits are well informed of valid data by finding the encoded handshake signal.

II. RELATED WORK

Compared with conventional synchronous design paradigms, asynchronous design paradigms, such as NULL convention logic (NCL) [1], [2] offer several significant advantages: better modularity and composability, reduced electromagnetic interference, higher security, no clock distribution problems, exhibiting average-case instead of worst-case performance, and improved reliability towards variations in PVT (fabrication process parameters, supply voltage, and temperature). NCL is a quasi-delay insensitive (QDI) asynchronous logic style in which control is inherent in each datum, and thus it provides correct-by-construction designs, requiring no worst-case delay analysis [2]. NCL has been successfully employed in a number of commercial products, including microcontrollers, embedded medical products, and encryption engines for smart card applications [3]. Besides, several electronic design automation (EDA) tools have been developed for NCL [4], [5]. In recent years, NCL has been employed for a variety of applications, including low-power circuit design [6]-[9], fault-attack-resistant cryptographic circuits [10], ternary logic [11], and robust circuit design for operating in space environment [12].

An asynchronous system comprises a set of autonomous functional modules, each of which communicates with others via handshaking only when it needs to send/receive data to/from its neighboring peers. Therefore, an asynchronous module is inherently data-driven and becomes active only when it needs to perform useful operations. Although an inactive asynchronous module consumes no dynamic power, it still suffers from static leakage power dissipation. Lately, a number of techniques have been proposed for utilizing fine-grain power gating to diminish the static leakage power of asynchronous circuits [6]-[9],[13]-[15].Multi-Threshold NCL is a normal NCL that combine both multi-threshold CMOS and fine-grain power gating. In the MTNCL pipeline, a pipeline stage becomes active only when performing useful operations, and enters the sleep mode (i.e., being power-gated) when having no useful work to perform. Both NCL and MTNCL need pipeline registers for dividing into two neighboring logic modules. However, pipeline registers can account for up to 35% of overall power dissipation of the NCL/MTNCL circuit. It describes register-less NCL achieves low power consumption by eliminating the pipeline registers and fine-grain power gating.

NCL LOGIC:

NCL is a self-timed logic paradigm in which control is inherent in each datum. NCL follows the so-called weak conditions of Seitz's delay-insensitive signaling scheme. The NCL model assumes that forks in wires are isochoric. Various aspects of the paradigm, including the NULL (or spacer) logic state from which NCL derives its name, have origins in Muller's work on speed-independent circuits in the 1950s and 1960s.

NCL uses symbolic expression to perform delay insensitive behavior. A symbolically complete expression depends only on the relationships of the symbols present in the expression without reference to their time of evaluation. In dual-rail and quad-rail signals can combine data and control information into one mixed-signal path. A dual-rail signal consists of two mutually exclusive wires and which may assume any value from the set likewise, a quad-rail signal consists of four mutually exclusive wires that represent two bits. Consider for NCL will be isochoric wire forks, they must meet the observability criteria. It requires all outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all the outputs of a combinational circuit. According to Seitz's weak conditions some of the outputs are not having a complete input as long as all outputs cannot change. Observability requires that no orphans may propagate through a gate [7].

An orphan is described as wire that change during the current DATA but not used in the determination of the output. Orphans are caused by wire forks and can be neglected through the isochoric fork assumption as long as they are not allowed to cross a gate boundary. An observability condition are also referred to as indicatability or stability provide that every gate transition is observable at the output. The observability condition can be relaxed through orphan analysis and still achieve self-timed behavior; however, this requires some delay analysis. For further the circuits use bit-wise

completion strategy with selective input incomplete components. It requires that completion signals can be generated such that no two adjacent DATA wave fronts can interact within any combinational component. In multi-rail delay insensitive systems are having NCL at least two register stages one at both the input and the output. Two adjacent register stages combine by request and acknowledge lines and, respectively.

The Conventional NCL Paradigm

The conventional NCL design paradigm is illustrated in Fig.1. In the NCL pipeline (see Fig. 1(a)), a pipeline stage, denoted by Si, comprises three components: a logic block Li, a data register Ri, and a completion detector CDi.

NCL uses delay-insensitive codes, such as dual-rail and quad-rail encodings, for data communication. In the dual-rail data encoding, n pairs of wires are required to encode n-bit data. For instance, one-bit data, denoted by D, can be encoded with a pair of wires D1 and D0. If (D1, D0) = (0, 1), the codeword (D1, D0) denotes the DATA0 state corresponding to a logic 0; if (D1, D0) = (1, 0), the codeword (D1, D0) denotes the DATA1 state corresponding to a logic 1; if (D1, D0) = (0, 0), the codeword (D1, D0) denotes the NULL state signifying that the value of D is not yet available. The codeword (D1, D0) = (1, 1) is not used.

NCL uses threshold gates as the primitive building blocks for constructing larger circuits. An m-of-n threshold gate, denoted by THmn, has n inputs and a threshold value of m, where $1 \leq m \leq n$. Threshold gates exhibit hysteresis state-holding capability.

Namely, the output of THmn does not transit from 0 to 1 until at least m of the n inputs have become 1, and the output of THmn does not transit from 1 to 0 until all the n inputs have become 0. Fig. 1(b) gives an example showing the structure of threshold gate TH23. As depicted in this figure, the static CMOS implementation of a threshold gate comprises four function blocks (i.e., 'set-to-1', 'set-to-0', 'hold-1', and 'hold-0') and an output inverter with feedback.

Threshold gates can be combined to build NCL logic blocks, registers, and completion detectors. In general, an n-bit NCL register comprises 2n TH22 gates; an n-bit completion detector comprises n 2- input OR gates (i.e., TH12) and an n-input C-element (i.e., THnn). Fig. 3.1(c) depicts the implementation of an NCL logic block Z = X XNOR Y using threshold gates. Fig. 3.1(d) illustrates the structures of a 2-bit NCL register and a 2-bit completion detector.

The completion detector CDi in stage Si is employed to sense whether the output of register Ri is DATA or NULL. The output of CDi transits from 0/1 to 1/0 when all bits of register Ri have become DATA/NULL.

NCL Gate	Function
TH12	A+B
TH13	A+B+C
TH22	AB
TH23	AB+AC+BC
TH33	ABC

Table: 1. 27Fundamental NCL Gates

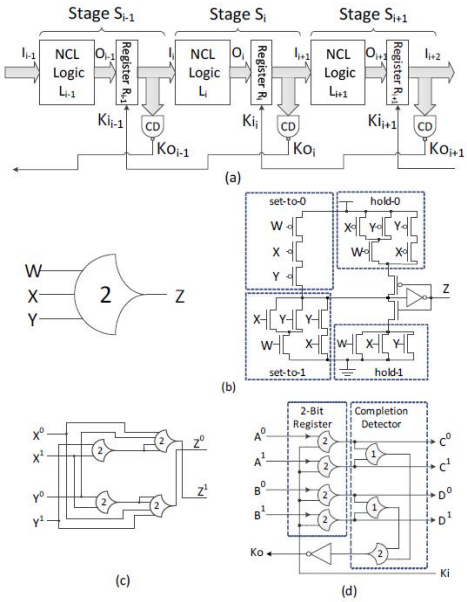


Fig 1. (a) NCL pipeline structure. (b) Symbol and structure of threshold gate TH23. (c) Implementation of logic function Z = X XNOR Y. (d) 2-bit register and completion detector

In the NCL pipeline, the data stream comprises a sequence of alternating NULL and DATA wave fronts. Namely, there is always a NULL/DATA wave front between two consecutive DATA/NULL wave fronts in the data stream. As depicted in Fig.1(a), the inverse output Koi of completion detector CDi in stage Si is wired to the control signal Kii-1 of register Ri-1 in stage Si-1. When a DATA/NULL token has passed through logic block Li and been successfully latched in register Ri, both Koi and Kii-1 will become 0/1, which enables register Ri-1 in the upstream stage to latch the succeeding NULL/DATA token.

III. REGISTER LESS NCL

The proposed RL-NCL requires no pipeline registers and is able to support fine-grain power gating.

RL-NCL differs from FPG-NCL as follows:

1) RL-NCL requires no pipeline registers.

2) In the RL-NCL pipeline, Koi+1, instead of Koi (as in the case of FPG-NCL), is used as one input of the C-element generating signal Sleepi. Namely, in RL-NCL, logic block Li in stage Si cannot begin evaluation/nullification for generating DATA/NULL token Dk/Nk at Ii+1 until the preceding NULL/DATA token Nk-1/Dk has safely arrived at the input Ii+2 of stage Si+2. This restriction prevents a DATA/NULL token Dk/Nk from overriding its preceding NULL/DATA token Nk-1/Dk.

3) In RL-NCL, it is not viable for an input bit of the logic block to be directly wired to an output bit without MTCMOS threshold gates placed between them, because pure wires themselves cannot operate in the sleep mode. If a logic block does contain pure wires in its input-output network (e.g., signals Z10 and Z11 of logic block Li i), every pure wire must be replaced with an MTNCL buffer (see signals Z10 and Z11 of logic block Li), which is a 2-input OR gate (i.e., MTCMOS threshold gate TH12) with the two inputs tied together.

4) In the RL-NCL pipeline, all MTCMOS threshold gates of a logic block begin evaluation/nullification at the same time, so the output bit on the critical path of the logic block becomes DATA/NULL after all the other output bits have already become DATA/NULL. Therefore, RL-NCL can employ an OR gate, whose two inputs are connected to the pair of wires associated with the output bit on the critical path of the logic block (e.g., Z50 and Z51 in Fig. 2(b)), to replace the completion detector for detecting whether the output of a logic block is DATA or NULL.

The RL-NCL pipeline operates as follows:

Precondition. Assume that logic block Li just entered the sleep mode (i.e., Sleepi= 0), causing the output of Li to become NULL (i.e., a NULL token, denoted by Nk-1, is now at Oi).

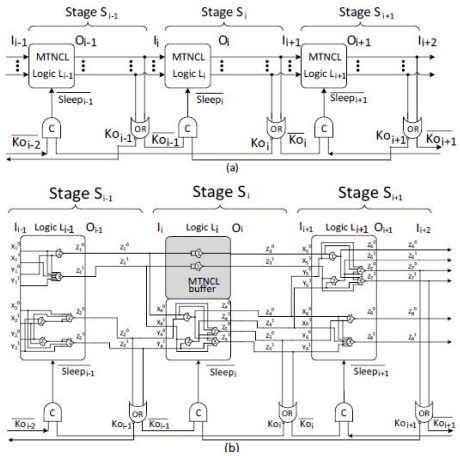


Fig 2. RL-NCL. (a) The RL-NCL pipeline. (b) An example of RL-NCL.

Step: 1. The next DATA token, Dk, arrives at the input Ii of stage Si, causing Koi-1to become 1. If Koi+1 is still 0, logic block Li will remain in the sleep mode and will not take Dk as its input.

Step: 2. After Koi+1 becomes 1 (i.e., the downstream logic block Li+1 has entered the sleep mode and the preceding NULL token Nk-1 has successfully arrived at Ii+2), logic block Li enters the active mode (i.e., Sleep i= 1) and takes Dk as its input, beginning evaluation.

Step: 3. Eventually, logic block Li completes evaluation and the output of Li becomes DATA. That is, DATA token, Dk, now arrives at the input Ii+1 of stage Si+1.

Step: 4. The next NULL token, Nk, arrives at the input Ii of stage Si, causing Koi-1to become 0.

Step: 5. After Koi+1 becomes 0 (i.e., the downstream logic block Li+1 has entered the active mode and the preceding DATA token Dk has successfully arrived at Ii+2), logic block Li enters the sleep mode (i.e., Sleepi= 0), beginning nullification.

Step: 6. Eventually, logic block Li completes nullification and the output of Li becomes NULL. That is, NULL token, Nk, now arrives at the input Ii+1 of stage Si+1.

Step: 7. k ← k + 1. Go to Step 1.

IV. STRUCTURE OF APCDP

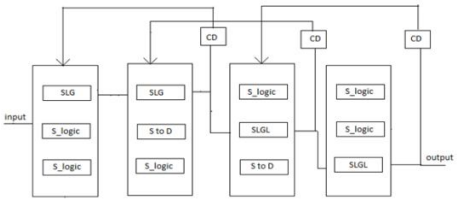


Fig. 3 Structure of APCDP.

SLG – Synchronization Logic Gate
SLGL- Synchronization Logic Gate with Latch
S to D-Single-rail to Dual-rail Converter
CD-Completion Detector
S_Logic –Single –rail Logic

Fig. 3 shows the structure of APCDP. The solid arrow indicate a critical data path (dual-rail data path), the dotted arrow indicate the noncritical data paths (singlerail data paths), and the dashed arrow indicate the output of single-rail to dual-rail encoding converter. In each pipeline stage, a static NOR gate is used as 1-bit completion detector to generate a total done signal for the entire data paths by detecting the constructed critical data path. Driving buffers deliver total signal to the precharge/evaluation control port of the previous stage. Since the completion detector only detects the constructed critical data path, the noncritical data paths do not have to transfer encoded handshake signal anymore. In single rail domino gates are used in the noncritical data path to save logic overhead. Encoding converter is used to bridge the connection between single-rail domino gate and dual-rail domino gate.

V. CONSTRUCTION OF THE CRITICAL DATA PATH

It is difficult to create a stable critical data path using traditional logic gates for their gate-delay problem. The critical signal transition varies from one data path to others according to different input data patterns. In past SLGs have solved the gate-delay problem, a stable critical data path can be easily constructed by the following steps:

1) Finding a gate (named as Lin gate) that has the largest number of inputs in each pipeline stage;
2) Changing these Lin gates to SLGs;
3) Linking SLGs together to form a stable critical data path.

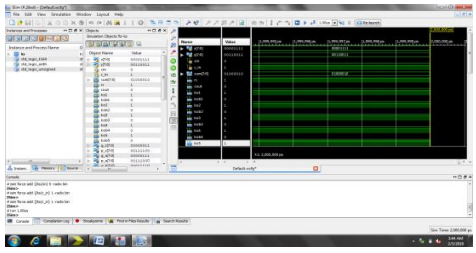
The basic idea of finding the critical signal transition is that embedding an SLG in each pipeline stage and making the SLG to be the last gate to start and finish evaluation. At first the embedded SLG has the largest gate delay in a pipeline stage.

The reasons are as follows.

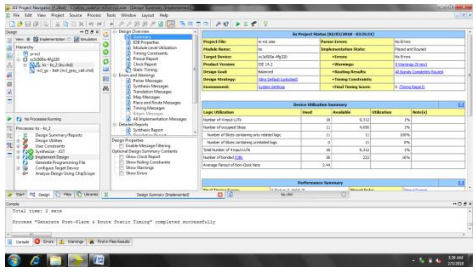
1) The SLG has the largest stack in the pull-down network compared with other gates.

2) The SLG has only one pull-down transistor path activated for each input data pattern. All gates evaluate at the same time or the SLG is the last gate to start evaluation in the pipeline stage, the critical signal transition would present on the output of the SLG. In practice, making all gates evaluate at the same time is difficult, especially without the help of intermediate latches or registers. We make the SLG is the last gate to start evaluation for pipeline stage. In the first pipeline stage, the critical signal transition is on the output of the SLG because all gates evaluate at the same time for the input control of latches or registers. After linking each pipeline stage the SLG in the following pipeline stage to start evaluation .According to the linked SLG data path becomes a stable critical data path. Linking each pipeline stage SLG together in the process of selecting Lin gate in each pipeline stage. They have been one more option. It is best to select the Lin gate that is originally linked to the Lin gate in the following pipeline stage. By later changing Lin gates to SLGs. For example, the linkage between Stage1 and Stage2 in Fig. We cannot find the linked Lin gates in neighbor stages, SLGL want to solve the linking problem. The linkage between Stage2 and Stage3 is in such situation. The linkage is established by connecting the output of SLG in Stage2 and the enable port of SLGL in Stage3.

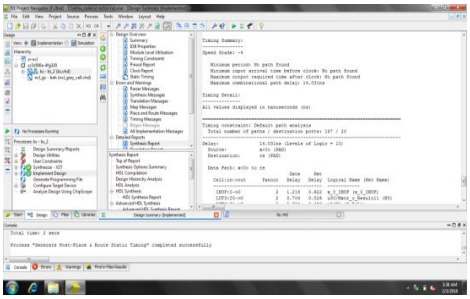
VI. RESULT



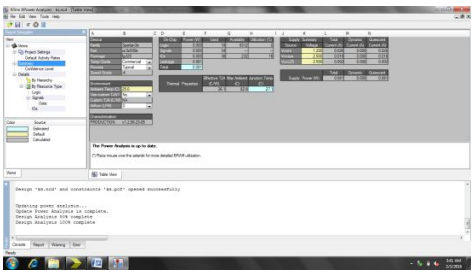
(a)



(b)



(c)



(d)

Fig. 4 (a) Simulation Result of RL NCL Hybrid rail Kogge Stone Adder (b) Calculation of Area for RL NCL Hybrid rail Kogge Stone Adder (c) Calculation of Delay for RL NCL Hybrid rail Kogge Stone Adder (d) Calculation of Power for RL NCL Hybrid rail Kogge Stone Adder.

VII. CONCLUSION & FUTURE WORK

The proposed hybrid-rail register-less NULL convention logic (HR-RLNCL) paradigm achieves low power consumption and low area by 1) eliminating pipeline registers, 2) replacing complex completion detectors with simpler OR gates, and 3) removing dual-rail logic for all gates except the one in the critical path. HR-RLNCL circuits are constructed by adapting FPG-NCL circuits and inserting proper dual rail logic in the critical logic paths. Table:2 shows the compared result from conventional NCL counterpart, the HR-RLNCL implementation of the Kogge–Stone adder adder produce better results in terms of area and power. Further, most of the switching activity in the non-critical path is also optimized.

As the transistor feature sizes and threshold voltages decrease, the standby power due to leakage current becomes comparable to dynamic power.

Table: 2 Comparison between RL NCL Kogge Stone Adder and RL NCL Hybrid Rail(HR) Kogge Stone Adder.

PARAMETER	Delay (ns)	SLICES	LUTs	POWER (mW)
RL NCL KOGGE STONE ADDER	20.430	238	121	860
RL NCL HYBRID RAIL KOGGE STONE ADDER	14.031	11	18	820

Especially, in asynchronous design, the standby power is a serious problem because it has an enormously large number of transistors to achieve its functionality. The proposed work has just analyzed the linear pipeline structure.

In the future, more complex data paths, forks and joins which are needed in the non-linear structures can be analyzed.

REFERENCES

[1] K. M. Fant and S. A. Brandt, “NULL convention logic: A complete and consistent logic for asynchronous digital circuit synthesis,” in Proc. Int. Conf. Appl. Specific Syst. Archit. Process., Aug. 1996, pp. 261–273.

[2] K. M. Fant, Logically Determined Design: Clockless System Design with NULL Convention Logic. New York: Wiley, 2005.

[3] D. A. Edwards and W. B. Toms, “The status of asynchronous design in industry,” Information Society Technologies (IST) Programme, Tech. Rep. IST-1999-29119, Jun. 2004.

[4] M. T. Moreira, C. H. M. Oliveira, R. C. Porto, and N. L. V. Calazans, “Design of NCL gates with the ASCEnD flow,” in Proc. IEEE 4th Latin American Symp. Circuits Syst., Feb. 2013, pp. 1-4.

[5] F. A. Parsan, W. K. Al-Assadi, and S. C. Smith, “Gate mapping automation for asynchronous NULL convention logic circuits,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 1, pp. 99–112, Jan. 2014.

[6] A. Bailey, J. Di, S. C. Smith, and H. A. Mantooth, “Ultra-low power delay-insensitive circuit design,” in Proc. IEEE Midwest Symp. Circuits Syst., Aug. 2008, pp.503-506.

[7] A. Bailey, A. A. Zahrani, G. Fu, J. Di, and S. C. Smith, “Multi-threshold asynchronous circuit design for ultra-

low power," J. Low Power Electronics, vol. 4, Dec. 2008, pp. 1-12.
[8] A. A. Zahrani, A. Bailey, G. Fu, and J. Di, "Glitch-free design for multi-threshold CMOS NCL circuits," in Proc. Great Lake Symp. VLSI, 2009, pp.215-220.
[9] F. A. Parsan, S. C. Smith, and W. K. Al-Assadi, "Design for testability of sleep convention logic," IEEE Trans. Very Large
[10] Scale Integr. (VLSI) Syst., vol. 24, no. 2, pp. 743–753,Feb.2016.
[11] Q. Ou, F. Luo, S. Li, and L. Chen, "Circuit level defences against fault attacks in pipelined NCL circuits," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 9, pp. 1903–1913, Sep. 2015.
[12] S. Andrawes and P. Beckett, "Ternary circuits for NULL convention logic," in Proc. Int. Conf. Comput. Eng. Syst., Nov. 2011, pp. 3-8.
[13] J. Brady, A. M. Francis, J. Holmes, J. Di, and H. A. Mantooth, "An asynchronous cell library for operation in wide-temperature & ionizing-radiation environments," in Proc. 2015 IEEE Aerospace Conf., Mar. 2015, pp. 1-10.
[14] T. Lin, K.-S. Chong, B.-H. Gwee, and J. S. Chang, "Fine-grained power gating for leakage and short-circuit power reduction by using asynchronous-logic," in Proc. IEEE Int. Symp. Circuits Syst., May 2009, pp. 3162–3165.
[15] M.-C. Chang and W.-H. Chang, "Asynchronous fine-grain power-gated logic," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 6, pp. 1143–1153, Jun. 2013.
[16] M.-C. Chang, M.-H. Hsieh, and P.-H. Yang, "Low-power asynchronous NCL pipelines with fine-grain power gating and early sleep," IEEE Trans. Circuits. Syst. II, Exp. Briefs., vol. 61, no. 12, pp. 957-961, Dec. 2014.