Low Complexity And Low Power Asynchronous Circuit Design Based on Null Convention Logic

R.Mohan[1], S.Abinaya[2]
[1, 2] HOD, Dept of ECE
[2]M.E.VLSI DESIGN
[1, 2] M.P.Nachimuthu M.Jaganathan Engineering College, Erode, Tamil Nadu.

Abstract- *NULL convention logic (NCL) is a promising design paradigm for constructing low-power robust asynchronous circuits. 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. Compared with the conventional NCL counterpart, the RL-NCL implementation of an eight-bit five-stage pipelined Kogge-Stone adder can reduce power dissipation by 56.4-72.5% . Moreover, the RL-NCL implementation can reduce the gates count of the adder by 49.5%.*

Keywords- Asynchronous circuits, low-power electronics, null convention logic, power gating.

I. INTRODUCTION

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]. Especially, Multi-Threshold NCL (MTNCL) [6]-[9] is a variant of the conventional NCL paradigm that incorporates both multi-threshold CMOS (MTCMOS) 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 the conventional NCL and MTNCL paradigms require pipeline registers for separating two neighboring logic modules, in order to prevent a DATA/NULL token from overriding its preceding NULL/DATA token because of latency difference between pipeline stages. However, pipeline registers can account for up to 35% of overall power dissipation of the NCL/MTNCL circuit. This brief presents the register-less NCL (RL-NCL) design paradigm, which achieves low power consumption by both eliminating the pipeline registers and supporting fine-grain power gating.

The remainder of this brief is organized as follows. In Section II, we introduce two related NCL design paradigms: conventional NCL and MTNCL. Section III describes the proposed RL-NCL design paradigm. Section IV presents the simulation results. Section V concludes this brief.

II. RELATED WORK

A. 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 D^1 and D^0. If (D^1, D^0) = (0, 1), the codeword (D^1, D^0) denotes the DATA0 state corresponding to a logic 0; if (D^1, D^0) = (1, 0), the codeword (D^1, D^0) denotes the DATA1 state corresponding to a logic 1; if (D^1, D^0) = (0, 0), the codeword (D^1, D^0) denotes the NULL state signifying that the value of *D* is not yet available. The codeword (D^1, D^0) = (1, 1) is not used.

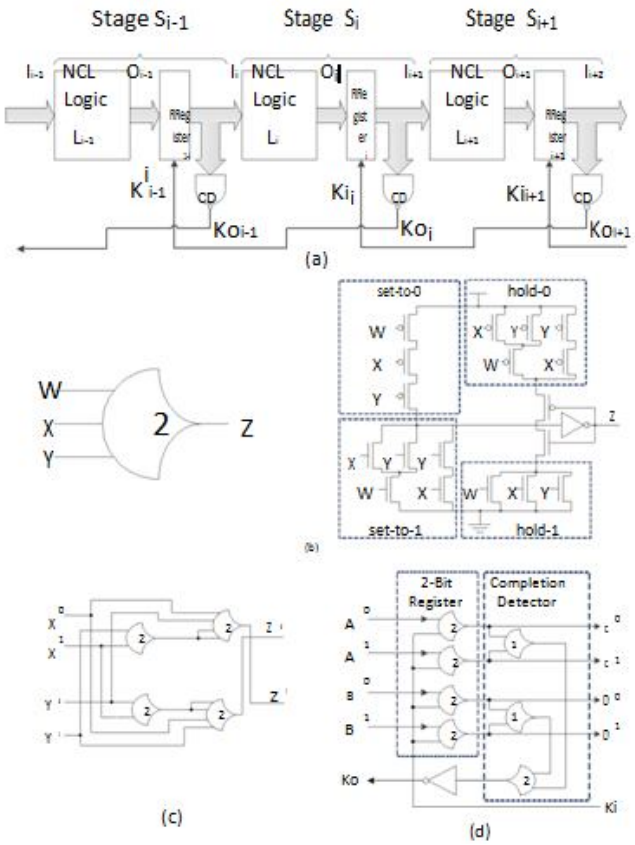

Fig 1. (a) NCL pipeline structure. (b) Symbol and structure of threshold gate TH_{23}. (c) Implementation of logic function *Z* = *X* XNOR *Y*. (d) 2-bit register and completion detector.

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. Fig. 1(c) depicts the implementation of an NCL logic block *Z* = *X* XNOR *Y* using threshold gates. Fig. 1(d) illustrates the structures of a 2-bit NCL register and a 2-bit completion detector. In general, an *n*-bit NCL register comprises 2*n* *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*). The completion detector *CDi* in stage *Si* is employed to sense whether the output of register R_i is DATA or NULL. The output of *CDi* transits from 0/1 to 1/0 when all bits of register R_i have become DATA/NULL.

In the NCL pipeline, the data stream comprises a sequence of alternating NULL and DATA wavefronts. Namely, there is always a NULL/DATA wavefront between two consecutive DATA/NULL wavefronts in the data stream. As depicted in Fig.

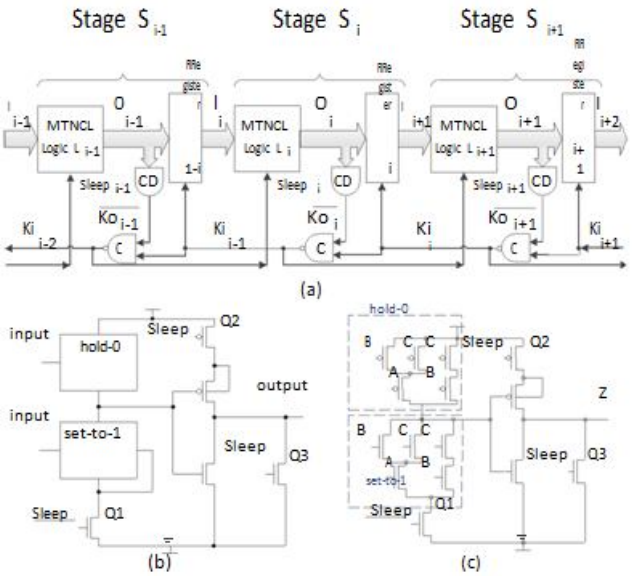

Fig 2. (a) The MTNCL pipeline. (b) Structure of an MTCMOS threshold gate(c) MTCMOS threshold gate TH_{23}.

1(a), the inverse output *Ko*i of completion detector *CD*i in stage *S*i is wired to the control signal *Ki*i-1 of register *R*i-1 in stage *S*i-1. When a DATA/NULL token has passed through logic block *L*i and been successfully latched in register *R*i, both *Ko*i and *Ki*i-1 will become 0/1, which enables

register *R*i-1 in the upstream stage to latch the succeeding NULL/DATA token.

B. The Multi-Threshold NCL Paradigm

As depicted in Fig. 2(a), in the Multi-Threshold NCL (MTNCL) pipeline, a pipeline stage, denoted by *Si*, comprises four components: a logic block *Li*, a data register *Ri*, a completion detector *CDi*, and an extra C-element (i.e., *TH22*).

In MTNCL, the logic blocks are built with MTCMOS threshold gates. Fig. 2(b) illustrates the structure of an MTCMOS threshold gate [7], which comprises two function blocks (i.e., 'hold-0' and 'set-to-1'), two high-*VT* sleep transistors (Q1 and Q2) for power gating, and an output inverter with a pull-down transistor Q3. An example of the MTCMOS threshold gate, *TH23*, is shown in Fig. 2(c).

An MTCMOS threshold gate can operate either in the active mode (with control signal *Sleep* deasserted) or in the sleep mode (with control signal *Sleep* asserted). If the present input of logic block *Li* is DATA and *Sleepi* is 0, transistors Q1 and Q2 in Fig. 2(b) are turned on, transistor Q3 is turned off, the MTCMOS threshold gates in *Li* begin to evaluate their outputs, and eventually the output of logic block *Li* becomes DATA. If the present input of logic block *Li* is NULL and *Sleepi* is 1, transistors Q1 and Q2 in Fig. 2(b) are turned off, causing all MTCMOS threshold gates in *Li* to be power-gated, and transistor Q3 in Fig. 2(b) is turned on, forcing the outputs of all MTCMOS threshold gates in *Li* to become 0 (i.e., forcing the output of logic block *Li* to become NULL).

As shown in Fig. 2(a), in the MTNCL pipeline, completion detector *CDi* is placed before register *Ri* rather than after, so that the evaluation of *CDi* can overlap with the latching of *Ri*, leading to a shorter cycle time. More details on the operation of the MTNCL pipeline can be found in [6]-[9], [15].

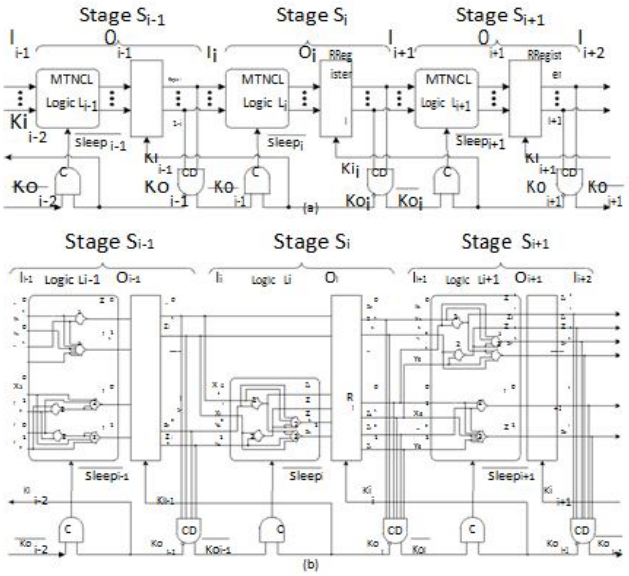

Fig 3. FPG-NCL. (a) The FPG-NCL pipeline. (b) An example of FPG-NCL

III. THE PROPOSED REGISTER-LESS NULL CONVENTION LOGIC

In this section, we will begin with the fine-grain power gating NCL (FPG-NCL) design paradigm, which supports fine-grain power gating but still requires pipeline registers. Then, we will use FPG-NCL to derive the proposed register-less NCL (RL-NCL), which achieves low power consumption by both supporting fine-grain power gating and eliminating pipeline registers.

A. The FPG-NCL Paradigm

As depicted in Fig. 3(a), in the FPG-NCL paradigm, a pipeline stage, denoted by *Si*, comprises four components: 1) a logic block *Li*, which is built from MTCMOS threshold gates, 2) a data register *Ri*, 3) a completion detector *CDi*, and 4) a C-element, which is used to control the operating mode (i.e., active or sleep) of logic block *Li*. If *Sleep*$\overline{\text{i}}$ = 1/0, logic block *Li* is in the active/sleep mode.

The data stream in FPG-NCL comprises a sequence of alternating DATA and NULL tokens, denoted by *D*0, *N*0, *D*1, *N*1, *D*2, *N*2, and so on, where *D*k/*N*k is the *k*-th DATA/NULL token. The operation of FPG-NCL is very similar to that of conventional NCL except that the logic blocks in FPG-NCL can be in the active or sleep mode.

In FPG-NCL, logic block *Li* enters the active mode when the following two conditions have both been fulfilled: 1) *Koi* = 1 (i.e., the preceding NULL token, *N*k-1, has successfully passed through the input *I*i+1 of stage *S*i+1), and

2) $Ko\overline{\ }_{i-1} = 1$ (i.e., the next DATA token, denoted by *D*k, has arrived at the input *Ii* of stage *Si*). Similarly, logic block *Li* enters the sleep mode when the following two conditions have both been fulfilled: 1) *Koi* = 0 (i.e., the preceding DATA token, *D*k, has successfully passed through the input *I*i+1 of stage *S*i+1), and 2) $Ko\overline{\ }_{i-1} = 0$ (i.e., the next NULL token, denoted by *N*k, has arrived at the input *Ii* of stage *Si*).

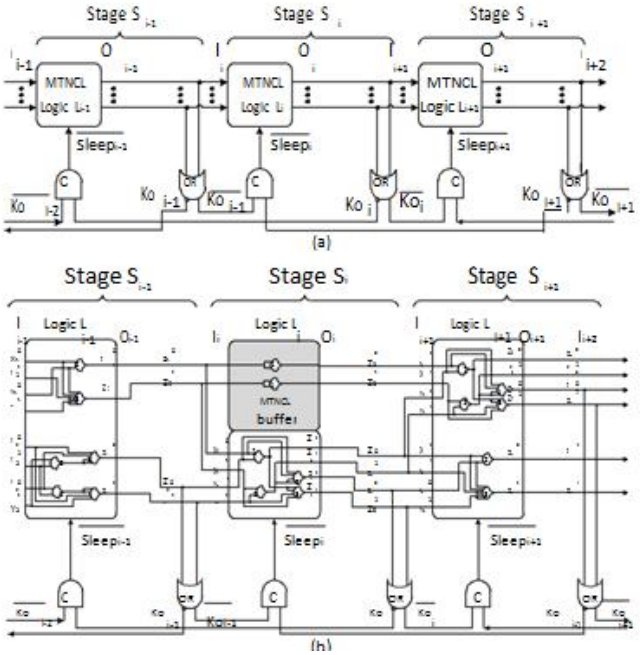

Fig 4. RL-NCL. (a) The RL-NCL pipeline. (b) An example of RL-NCL

The purpose of using pipeline registers in FPG-NCL is to prevent a DATA token, denoted by *D*k, from overriding its preceding NULL token *N*k-1 as well as to prevent a NULL token, denoted by *N*k, from overriding its preceding DATA token *D*k. As an example, let us assume that 1) a DATA token *D*k is now at *I*i (see Fig. 3(a)), 2) its preceding NULL token *N*k-1 is at *I*i+1, and 3) logic block *Li* is active. When *Li* finishes its evaluation and generates valid output, *D*k advances to *O*i. However, pipeline register *R*i cannot latch *D*k (i.e., the valid output of *Li*) until *N*k-1 has successfully arrived at *I*i+2, which event causes *Koi*+1 (i.e., *Ki*i) to become 1 and enables *R*i to latch *D*k. Therefore, pipeline registers in FPG-NCL can prevent a DATA/NULL token *D*k/*N*k from overriding its preceding NULL/DATA token *N*k-1/*D*k.

B. The RL-NCL Paradigm

The proposed RL-NCL requires no pipeline registers and is able to support fine-grain power gating. Fig. 4(a) shows the structure of the RL-NCL pipeline.

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 $Sleep\overline{\ }_i$. Namely, in RL-NCL, logic block *L*i in stage *S*i cannot begin evaluation/nullification for generating DATA/NULL token *D*k/*N*k at *I*i+1 until the preceding NULL/DATA token *N*k-1/*D*k has safely arrived at the input *I*i+2 of stage *S*i+2. This restriction prevents a DATA/NULL token *D*k/*N*k from overriding its preceding NULL/DATA token *N*k-1/*D*k.

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 $Z1^0$ and $Z1^1$ of logic block *L*i in Fig. 3(b)), every pure

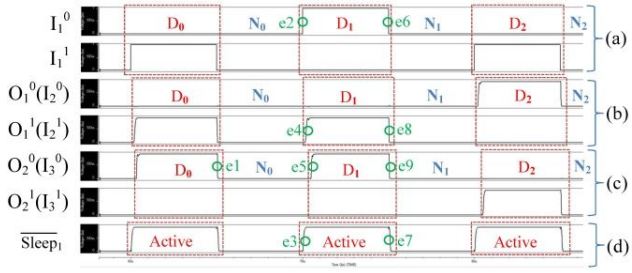

Fig 5. Simulation waveforms of the Kogge-Stone adder implemented with RL-NCL. (a) One bit of the input of stage S_1. (b) One bit of the output of stage S_1. (c) One bit of the output of stage S_2. (d) The sleep signal for stage S_1.

wire must be replaced with an MTNCL buffer (see signals $Z1^0$ and $Z1^1$ of logic block *L*i in Fig. 4(b)), which is a 2-input OR gate (i.e., MTCMOS threshold gate *TH*12) 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., $Z5^0$ and $Z5^1$ in Fig. 4(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., *Sleep*$\overline{}$i = 0), causing the output of *Li* to become NULL (i.e., a NULL token, denoted by *N*k-1, is now at *O*i).

Step 1. The next DATA token, *D*k, arrives at the input *Ii* of stage *Si*, causing *Ko*$\overline{}$i-1 to become 1. If *Ko*i+1 is still 0, logic block *Li* will remain in the sleep mode and will not take *D*k as its input.

Step 2. After *Ko*i+1 becomes 1 (i.e., the downstream logic block *Li+1* has entered the sleep mode and the preceding NULL token *N*k-1 has successfully arrived at *I*i+2), logic block *Li* enters the active mode (i.e., *Sleep*$\overline{}$i = 1) and takes *D*k as its input, beginning evaluation.

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

Step 4. The next NULL token, *N*k, arrives at the input *Ii* of stage *Si*, causing *Ko*$\overline{}$i-1 to become 0.

Step 5. After *Ko*i+1 becomes 0 (i.e., the downstream logic block *Li+1* has entered the active mode and the preceding DATA token *D*k has successfully arrived at *I*i+2), logic block *Li* enters the sleep mode (i.e., *Sleep*$\overline{}$i = 0), beginning nullification.

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

Step 7. $k \leftarrow k + 1$. Go to Step 1.

IV. SIMULATION RESULTS

In order to evaluate the effectiveness of the proposed RL-NCL, we have employed four NCL design paradigms—conventional NCL (Fig. 1(a)), MTNCL (Fig. 2(a)), FPG-NCL (Fig. 3(a)), and RL-NCL (Fig. 4(a))—to implement

TABLE I

POWER DISSIPATION COMPARISON OF FOUR NCL DESIGN PARADIGMSIMPLEMENTING AN EIGHT-BIT FIVE-STAGE PIPELINED KOGGE–STONE ADDER

Input data rate (MHz)	Conv. NCL	MTNCL	FPG-NCL	RL-NCL	
	Power (µw) [A]	Power (µw) [B]	Power (µw) [C]	Power (µw) [D]	Savings (%) [(A-D)/A]
10	10.1	8.0	7.9	2.8	72.5
30	13.7	13.0	12.6	5.1	62.9
50	18.4	17.3	17.1	7.0	61.8
100	29.8	30.6	28.8	12.3	58.5
300	78.1	77.7	76.0	33.4	57.2
500	126.5	128.2	124.6	55.1	56.4
700	176.7	177.1	173.6	66.3	62.5
900	221.0	N/A	N/A	94.7	57.1

an eight-bit five-stage pipelined Kogge–Stone adder for performance comparison. The simulations were performed with HSPICE using the transistor models of PTM (predictive technology model) [16] 32-nm process technology at the typical PVT case (i.e., TT process corner, 25 °C, and *VDD* = 0.9 V).

Fig. 5 shows part of the Hspice simulation waveforms of the Kogge-Stone adder implemented with RL-NCL. The waveforms in Fig. 5 show a sequence of tokens, denoted by *D*0, *N*0, *D*1, *N*1, *D*2, and *N*2, passing through the RL-NCL pipeline. In the following, we will explain how stage *S*1 evaluates the value corresponding to DATA token *D*1. When NULL token *N*0 has safely arrived at *I*3 (i.e., *I*3 becomes NULL; this event is denoted by **e1** in Fig. 5), *Ko*2 becomes 1. When DATA token *D*1 has arrived at *I*1 (i.e., *I*1 becomes DATA; this event is denoted by **e2** in Fig. 5), *Ko*$\overline{}$0 becomes 1. The fulfillment of both events **e1** and **e2** causes stage *S*1 to enter the active mode (event **e3**). Stage *S*1 takes the value corresponding to *D*1 as its input, and then generates an output value at *I*2 (i.e., *I*2 becomes DATA and now *D*1 is said to have arrived at *I*2; this event is denoted by **e4**). When *D*1 has finally arrived at *I*3 (i.e., *I*3 becomes DATA; this event is denoted by **e5**), *Ko*2 becomes 0. When the succeeding NULL token *N*1 has arrived at *I*1 (i.e., *I*1 becomes NULL; this event is denoted by **e6**), *Ko*$\overline{}$0 becomes 0. The fulfillment of both events **e5** and **e6** causes stage *S*1 to enter the sleep mode (event **e7**). Stage *S*1 takes the NULL value corresponding to *N*1 as its input, and then generates a NULL output value at *I*2 (i.e., *I*2 becomes NULL and now *N*1 is said to have arrived at *I*2; this event is denoted by **e8**). *N*1 will finally arrive at *I*3 (event **e9**), causing *Ko*2 to become 1.

Table I gives a power dissipation comparison of the four NCL design paradigms—conventional NCL, MTNCL, FPG-NCL, and RL-NCL—implementing the pipelined Kogge–Stone adder with the input data rate ranging from 10 MHz to 900 MHz. As given in Table I, the maximum

sustainable throughput rates for the four implementations are 900 MHz, 700 MHz, 700 MHz, and 900 MHz, respectively. Because the latency of a logic block can be deteriorated by power gating, the MTNCL implementation has a lower maximum sustainable throughput rate than the conventional NCL counterpart. In contrast, the RL-NCL implementation can achieve the same maximum sustainable throughput rate as the conventional NCL counterpart by replacing completion detectors with much faster OR gates to countervail the latency

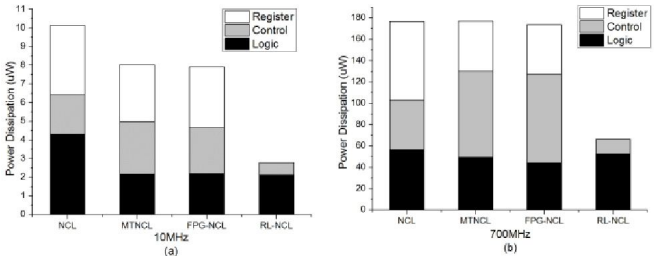

Fig 6. Power dissipation comparison of four NCL paradigms implementing a pipelined Kogge-Stone adder. (a) The input data rate is set to 10 MHz. (b) The input data rate is set to 700 MHz.

TABLE II
COMPARISON Of GATES and POWER FOR RL-NCL and FPG-NCL

S.NO	TECHNIQUE	GATES	POWER
1	RL-NCL	1491	188mW
2	FPG-NCL	1683	227mW

TABLE III
AREA COST FOR COMPONENTS IN THE RL-NCL IMPLEMENTATION

	Logic blocks		Control	Total
	Power gating transistors	Other transistors		
Area (nm^2)	1089536	3342336	333312	4765184
Proportion (%)	22.9	70.1	7.0	100.0

Fig. 6 highlights the power dissipation comparison of the four NCL paradigms at input data rates of 10 MHz (Fig. 6(a)) and 700 MHz (Fig. 6(b)). In the conventional NCL implementation of the Kogge-Stone adder, pipeline stage *S*1/*S*2/*S*3/*S*4/*S*5 requires a 17-/24-/22-/18-/9-bit NCL register and a 17-/24-/22-/18-/9-bit completion detector; the NCL registers account for 36.5-41.7% of the overall power dissipation, and the control circuits (including completion detectors and C-elements) account for 20.8-26.2% of the overall power dissipation. When the input data rate is low (e.g. 10 MHz), the MTNCL implementation consumes less power than the conventional NCL counterpart because logic blocks in MTNCL have high probability of staying in the sleep mode and the leakage power of a sleeping logic block can be mitigated by power gating. However, when the input data rate is high (e.g., 700 MHz), the MTNCL implementation no longer exhibits the advantage of lower power because logic blocks in MTNCL have lower probability of being sleeping under high input data rate. In contrast, from Table I and Fig. 6, it can be seen that the proposed RL-NCL always achieves lower power consumption than conventional NCL both at low and at high input rates. The advantage of low power dissipation in the RL-NCL paradigm comes from 1) eliminating pipeline registers, 2) replacing complex completion detectors with simpler OR gates, and 3) mitigating the leakage power of sleeping logic blocks by fine-grain power gating. The RL-NCL implementation of the Kogge–Stone adder can reduce power dissipation by 72.5% and 62.5%, respectively, for the input data rate of 10 MHz and 700MHz, compared with the conventional NCL counterpart.

Table II gives a comparison of hardware cost for the four implementations of the Kogge–Stone adder. The transistor count/area for the RL-NCL implementation is 50.5%/45.8% of that for the conventional NCL counterpart.

Table III lists the area cost for the components in the RL-NCL implementation. The area for power-gating transistors accounts for 22.9% of the total area.

V. CONCLUSION

This brief has proposed the register-less NULL convention logic (RL-NCL) paradigm, which achieves low power consumption by 1) eliminating pipeline registers, 2) replacing complex completion detectors with simpler OR gates, and 3) mitigating the leakage power of sleeping logic blocks by fine-grain power gating. We have described how to construct RL-NCL circuits by adapting FPG-NCL circuits and inserting proper MTNCL buffers in logic blocks. Compared with the conventional NCL counterpart, the RL-NCL implementation of the Kogge–Stone adder can 1) reduce power dissipation by 56.4%-72.5% for the input data rate ranging from 10 MHz to 900 MHz, and 2) reduce the number of transistors required to implement the adder by 49.5%.

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 Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 743–753, Feb. 2016.
[10] 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.
[11] S. Andrawes and P. Beckett, "Ternary circuits for NULL convention logic," in *Proc. Int. Conf. Comput. Eng. Syst.*, Nov. 2011, pp. 3-8.
[12] 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.

S.ABINAYA received B.E degree in Electronics and Communication Engineering from Anna Universiy, Chennai. Tamil Nadu, India in 2014. She is currently doing M.E.VLSI DESIGN in Anna University, Chennai, Tamil Nadu, and India. Here areas of inerest include Digial circuit design, logic synthesis and test, variable latency designs, especially low power and high performance VLSI designs.