Low Power and Area efficient asynchronous NCL design based on optimized datapath

Shruthi BS
M.Tech in VLSI design & Embedded Systems,
Dr. Ambedkar Institute of Technology,
Outer Ring Road ,
Bengaluru 560056, Karnataka ,India

Dr. H. Umadevi
Professor, ECE Dept,
Dr. Ambedkar Institute of Technology,
Outer Ring Road,
Bengaluru 560056, Karnataka ,India

Abstract: —

The concept of NULL convention logic (NCL) is introduced for constructing low-power robust asynchronous circuits. The presence of registers in conventional NCL can account up to 35 % overall power consumption. This brief presents the Combination of single rail and dual rail encoding in 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. And also here introducing the concepts of single rail encoding and dual rail encoding. Data paths are composed of a mixture of dual-rail and single-rail logic gates. Dual-rail logic gates are limited to construct a stable critical data path. Based on this critical data path, the handshake circuits are greatly simplified, which offers the pipeline high throughput as well as low power consumption.

Index Terms — **NCL, RL-NCL, Kogga-Stone**

I. INTRODUCTION

During the last decade, there has been a revival in research on asynchronous technology. Along with the continued CMOS technology scaling, VLSI systems become more and more complex. The physical design issues, such as global clock tree synthesis and top-level timing optimization, become serious problems. Even if technology scaling offers more integration possibilities, modularity and scalability are difficult to be realized at the physical level. Asynchronous design is considered as a promising solution for dealing with these issues that relate to the global clock, because it uses local handshake instead of externally supplied global clock . The attractive properties are listed as follows:

- low power consumption;
- high operating speed;
- no clock distribution and clock skew
- better composability and modularity;
- less emission of electromagnetic noise;

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 .

Both the conventional NCL and FPGNCL 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 hybrid register-less NCL (RL-NCL) design paradigm, which achieves low power by both eliminating the pipeline registers and supporting fine-grain power gating

This paper presents a novel design method of asynchronous logic pipeline, which focuses on improving the circuit efficiency and making asynchronous logic pipeline design more practical for a wide range of applications. The novel design method combines the benefits of the four-phase dual-rail protocol and the four-phase bundled-data protocol, which achieves an area-efficient and ultralow-power asynchronous Register-less Null Convention Logic.

This reminder of this brief is organized as follows. In section II, introduce two related NCL design paradigms: conventional NCL and FPGNCL. Section III describes the proposed Combination of dual rail encoding and single rail 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.

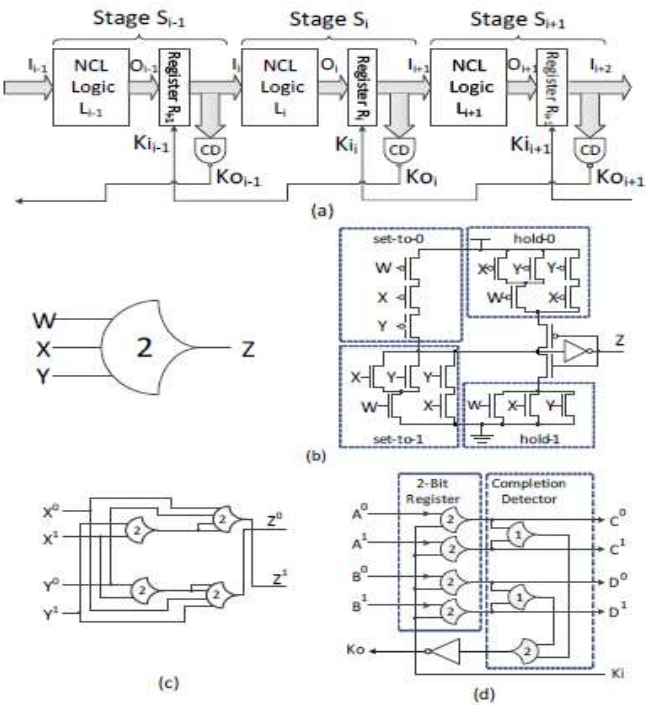

Fig 1 (a) NCL pipeline structure. (b) Symbol and structure of threshold gate *TH23*. (c) Implementation of logic function *Z* = *X* XNOR *Y*. (d) 2-bit register and completion detector.

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 *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.

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 *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 FPG-NCL Paradigm

As depicted in Fig. 2(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*i- = 1/0, logic block *Li* is in the active/sleep mode.

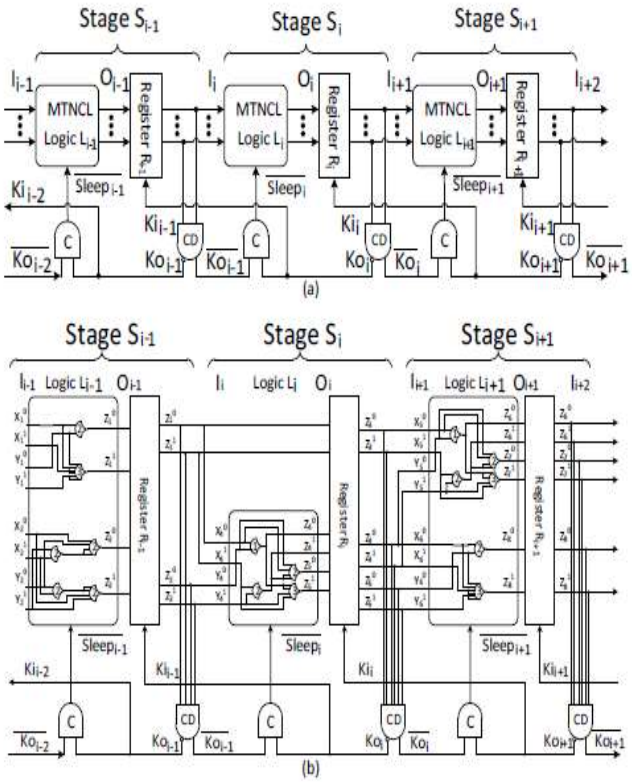

Fig 2 FPG-NCL. (a) The FPG-NCL pipeline. (b) An example of FPG-NCL

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) *Koi*-1 $\overline{}$ = 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) *Koi*-1 $\overline{}$ = 0 (i.e., the next NULL token, denoted by *N*k, has arrived at the input *Ii* of stage *Si*).

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 , 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., *Kii*) 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.

C. Existing Method and Drawbacks

- Both the conventional NCL and MTNCL paradigms require pipeline registers for separating two neighbouring logic modules
- Pipeline registers accounts for up to 35% of overall power dissipation of the NCL/MTNCL circuit.
- Dual-rail encoding overhead that consumes a lot of silicon area and power consumption.

D. *The RL-NCL Paradigm*

The proposed RL-NCL requires no pipeline registers and is able to support fine-grain power gating. Fig. 3(a) shows the structure of the RL-NCL pipeline.

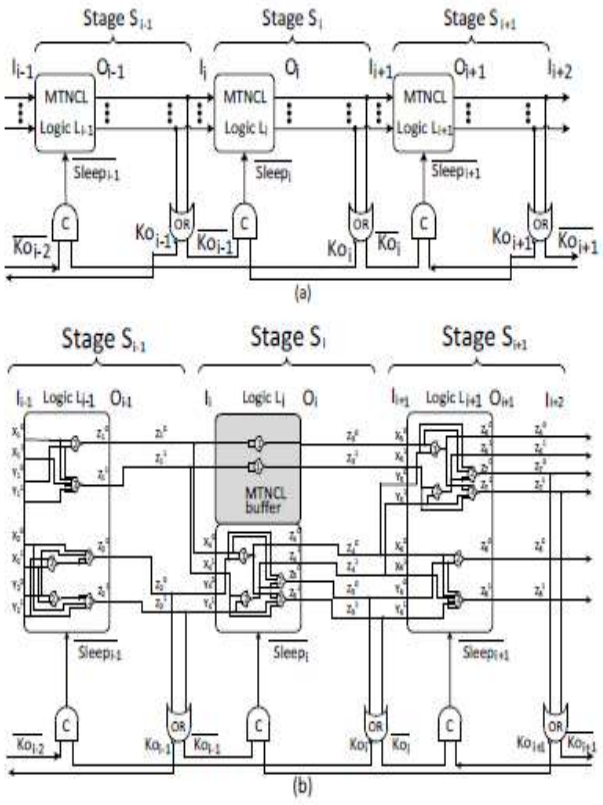

Fig 3 RL-NCL. (a) The RL-NCL pipeline. (b) An example of RL-NCL.

RL-NCL differs from FPG-NCL as follows:

1) RL-NCL requires no pipeline registers.

2) In the RL-NCL pipeline, *Ko*i+1, instead of *Ko*i (as in the case of FPG-NCL), is used as one input of the C-element generating signal *Sleep*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 signals, every pure wire must be replaced with an MTNCL buffer (see signals *Z*10 and *Z*11 of logic block *L*i in Fig. 3(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., *Z*50 and *Z*51 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*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*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*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*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*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.

III . Proposed Work – Optimised path

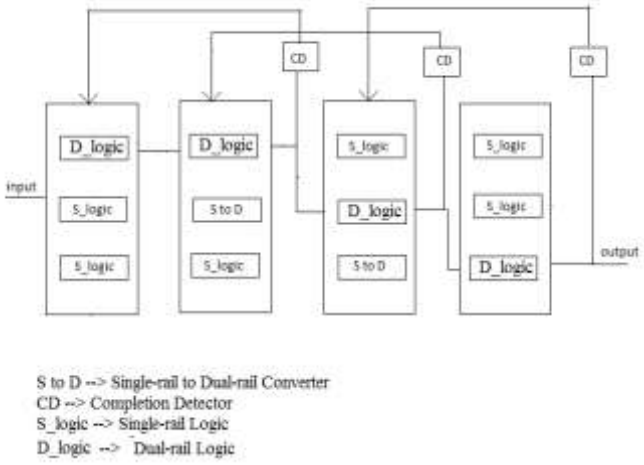

Fig 4a Optimised Hybrid NCL

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 each total done 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. Therefore, singlerail logic gates are used in the noncritical data path to save logic overhead. Encoding converter is used to bridge the connection between single-rail logic gate and dual-rail logic gate.

A. 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). Each logic block uses only one dual-rail gate(critical datapath), while all others being single-rail. The following steps describe the dual-rail logic functionality. The single-rail operation is similar to conventional logic gate operation with single-rail to dual-rail convertor added in the required places.

B. Construction of the Critical Data Path

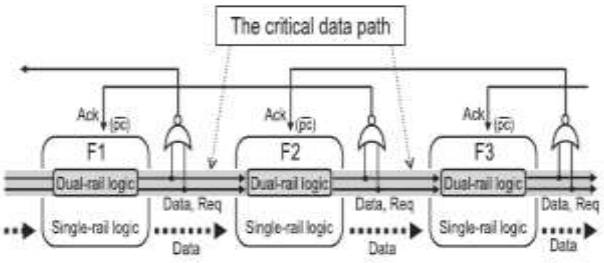

Fig 4b. critical data path

It is difficult to construct a stable critical data path using traditional logic gates for their gate-delay data-dependence problem. The critical signal transition varies from one data path to others according to different input data patterns. Since DRGs have solved the gate-delay data-dependence 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 DRGs;

3) Linking DRGs together to form a stable critical data path.

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

The reasons are as follows.

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

2) The dual rail has only one pull-down transistor path activated for each input data pattern.

Then, if all gates evaluate at the same time or the dual rail is the last gate to start evaluation in the pipeline stage, the critical signal transition would present on the output of the dual rail. In practice, making all gates evaluate at the same time is difficult, especially without the help of intermediate latches or registers. Therefore, we make the dual rail become the last gate to start evaluation by linking each pipeline stage's dual rail together. In the first pipeline stage, the critical signal transition is on the output of the dual rail because all gates evaluate at the same time for the input control of latches or registers. After linking each pipeline stage's dual rail together, the dual rail in the following pipeline stage would be the last gate to start evaluation since it always waits for the critical signal transition from the previous dual rail. As a result, the linked dual rail data path becomes a stable critical data path. Linking each pipeline stage's dual rail together is partially done in the process of selecting Lin gate in each pipeline stage. When searching Lin gate, there might be more than one option. It is best to select the Lin gate that is originally linked to the Lin gate in the following pipeline stage. After changing these Lin gates to DRGs, DRGs are naturally linked. For example, the linkage between Stage1 and Stage2 in Fig. However, if we cannot find the linked Lin gates in neighbor stages, dual rail needs to be used to solve the linking problem. The linkage between Stage2 and Stage3 is in such situation. The linkage is established by connecting the output of dual rail in Stage2 and the enable port of dual rail in Stage3.

C. Encoding Conversion

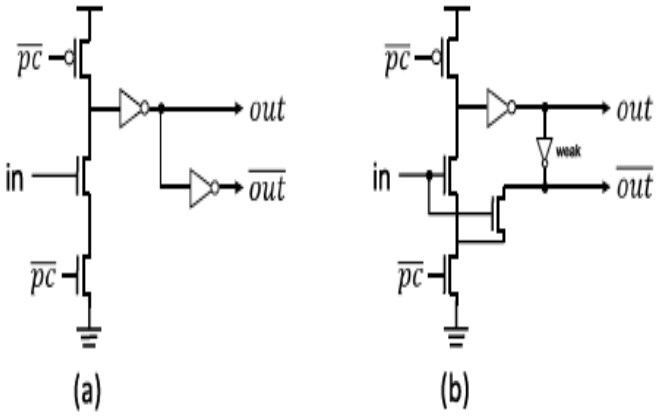

Fig5. Encoding converters. (a) Intuitive design. (b) Proposed design.

Since the completion detector detects only the constructed critical data path, the noncritical data paths do not have to transfer encoded handshake signal anymore. The logic overhead in the noncritical data paths can be reduced using single-rail logic gates instead of dual-rail logic gates. However, single-rail logic gate and dual-rail logic gate use different encoding schemes. It has encoding compatibility problem when a single-rail logic gate connects to a dual-rail logic gate. Encoding converter needs to be designed to solve the problem. Fig. shows two implementations of encoding converter. Table III shows the truth table. In precharge phase pc = 0, encoding converter outputs a dual-rail data0 (out, out) = (0, 1). In evaluation phase pc = 1, if the input is a single-rail data0 in = 0, the converter keeps the dual-rail data0. If the input is a single-rail data1 in = 1, the converter outputs a dual-rail data1 (out, out) = (1, 0). Since single-rail encoding only has two states that, respectively, represent data0 and data1, there is no other state that can be converted to spacer (out, out) = (0, 0). The disappearance of spacer violates the four-phase dual-rail protocol, which would cause data transfer error.

In practice, the robustness of the constructed critical path is affected by delay variations. As a matter of fact, it is a common problem in VLSI circuit design, same as the robustness of a clock signal in synchronous design and a match delay line in bundled-data asynchronous design. As we all know, these designs all suffer from delay variations. To resist the influence of delay variations, synchronous design enlarges the cycle time of a clock signal to get some margin. On the other hand, bundled-data asynchronous design adds extra delay margin on the matching delay line to match the worst case delay in combinational logic block. Same like these solutions, the delay variations problem in the proposed design can be solved by enlarging delay margin on the constructed critical data path. We supply four measures to enlarge the delay margin, which are listed as follows:

1) Sizing the pull-down transistors or the static inverters of DRGs and dual rail to increase gate delays;

2) Applying a low priority in circuit layout for the constructed critical path;

3) Improving the noncritical paths delay;

4) Adding delay elements on the critical path.

These measures have different impacts on the performance of circuits. It is better to choose a proper measure or multiple measures according to the practical design requirements. In measure

1) Reducing the transistor size of pull-down network or the static inverters can increase the gate delays of DRGs and dual rails. The increased delay slightly slows down pipeline speed, but smaller transistor size helps in saving power and silicon area.

2) Is layout optimization. Although this measure slightly decreases pipeline speed, it does not impose the extra overhead of transistors.

3) Does not degrade pipeline speed. The problem is that improving the noncritical path delay would increase the power consumption.

4) Is an intuitive way to enhance the critical data path. The drawback is the extra overhead of transistors.

In addition, the use of logic introduces many design risks because it is very sensitive to noise, circuit, and layout topologies. The solutions to alleviate these problems are not in the scope of this paper. We only recommend limiting the largest stack of logic when designing APCDP. The limitation depends on processing technology, practical problems, and actual conditions. Smaller stack design helps in alleviating the noise problems and increasing the pipeline speed. The tradeoff is the increased number of pipeline stages. More pipeline stages means larger silicon area and higher power consumption. On the other hand, larger stack design deteriorates the noise problems and the pipeline speed. The merit is the reduced number of pipeline stages. Less pipeline stages help in saving silicon area and power consumption.

D.PROPOSED METHOD AND ITS ADVANTAGES

- Register-less NULL convention logic (RL-NCL) design paradigm, which achieves low power consumption by eliminating pipeline registers.
- Pipelined design based on dual-rail and single-rail gates.

- The overhead of function block logic is reduced by applying single-rail logic in noncritical data paths.

IV.RESULT AND DISCUSSION

In order to evaluate the effectiveness of the proposed Combination dual rail encoding and single encoding RL-NCL, I have employed Three NCL paradigm-FPG NCL, RL NCL and hybrid RL-NCL, VHDL language is used for the design of this parallel counters. Modelsim 6.3f is used as the simulating tool. Xilinx ISE 8.1 is used as the synthesis tool. Experimental results include the simulation result of 8 bit kogge stone adder implemented as fine-grain power gating NCL, RL-NCL, Combination of single rail and dual rail encoding used in RL-NCL and the comparison of various parameters. The 8 bit kogge stone adder is implemented in FPG-NCL. The adder has two input which we want to add, represented in dual rail encoding. And three other input one is carry input, other two are enabling signals. When carry input is zero and other enabling inputs are 1 then adder adds the inputs. An eight-bit five-stage pipelined Kogge–Stone adder is designed using VHDL language and simulated using Modelsim software. The overall gate count, Slices, LUT and power comparisons are performed using Xilinx ISE with SPARTAN 2E Family.

A. FPG-NCL:

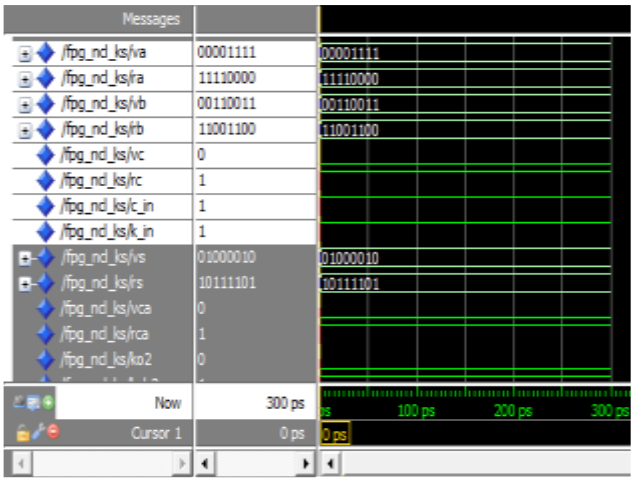

fig 6 .wave form of 8 bit kogge stone adder implemented as fpg-ncl

Fig6 shows the simulation result of FPG-NCL. The 8 bit kogge stone adder is implemented in FPG-NCL. The adder has two input which we want to add, represented in dual rail encoding. And three other input one is carry input, other two are enabling signals. When carry input is zero and other enabling inputs are 1 then adder adds the inputs. The data stream in FPG-NCL comprises a sequence of alternating DATA and NULL tokens, denoted by D0, N0, D1, N1, D2, N2, and so on, where Dk/Nk 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.

(va,ra) and (vb,rb) are the dual-rail input of Kogge stone adder, whereas (vc,rc) is the carry in. c in is the output of the completion detector with sum(vs,rs) as its final adder output. All the inputs and outputs are in dual rail logic.

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, Nk-1, has successfully passed through the input Ii+1 of stage Si+1), and 2) Koi-1 = 1 (i.e., the next DATA token, denoted by Dk, 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, Dk, has successfully passed through the input Ii+1 of stage Si+1), and 2) Koi-1= 0 (i.e., the next NULL token, denoted by Nk, has arrived at the input Ii of stage Si).

B.RL-NCL:

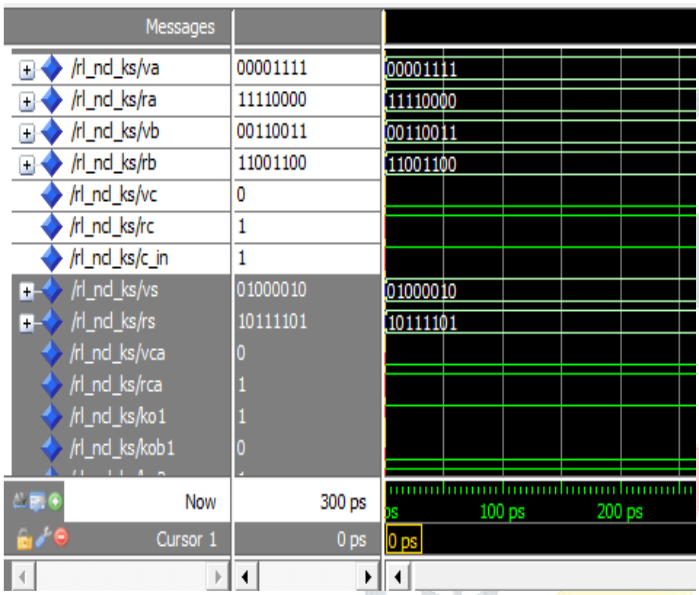

Fig7: simulation results of RL-NCL

Fig7. shows the simulation result of RL-NCL. The proposed RL-NCL requires no pipeline registers and is able to support fine-grain power gating (va, ra) and (vb,rb) are the dual-rail input of Kogge stone adder, whereas (vc,rc) is the carry in. c_in is the output of the completion detector with sum(vs,rs) as its final adder output. All the inputs and outputs are in dual rail logic.

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 i. 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.

C.HYBRID-RAIL NCL:

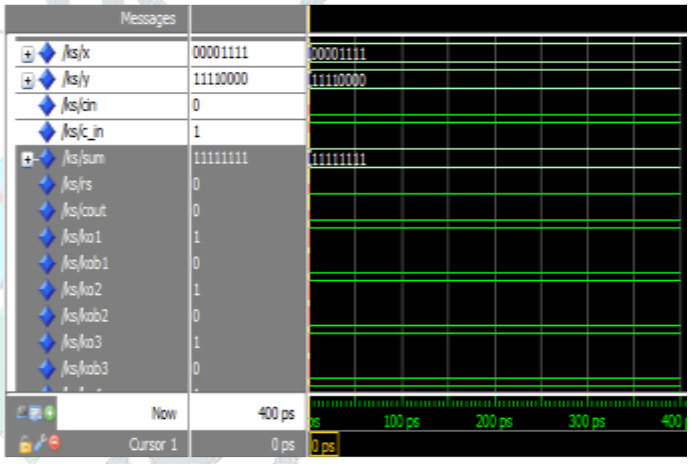

Fig8: Simulation Results Of Hybrid- Rail Ncl

Fig8. shows the simulation result of RL-NCL .X and Y is the input of the hybrid rail Kogge stone adder. Cin is the carry and and c_in is the output of the completion detector with sum as its final adder output. All the inputs are in single rail except X(7) and Y(7). The final out sum (7) will be the critical path output which is dual rail with all other outputs in single-rail.

FPGNCL	RLNCL	HYBRID NCL
Path delay: 38.162 ns	Path delay : 32.200 ns	Path delay: 19.043ns
4 i/p LUT : 275	4 i/p LUT : 247	4 i/p LUT : 234
Powerconsumption : 227mW	Powerconsumption :188mW	Powerconsumption :100mW

Table1:- Comparison results of existing & proposed work

Table I gives a power dissipation comparison of the three NCL design paradigm FPG-NCL,RL-NCL, Combination of single rail and dual rail encoding in RL-NCL implementing the pipelined kogge stone adder with input data ranging from 10MHz to 900MHz. The power dissipation comparison of the three NCL paradigms, FPG-NCL, RL-NCL, Combination of single rail and dual rail encoding in RL-NCL , the FPG includes 200mW and it reduces to 188mW and our proposed system reduces to 50% of FPG-NCL.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 blocks by fine-grain power gating.

V.CONCLUSION

This brief has proposed the combination of single rail and dual rail encoding in register-less Null Convention Logic paradigm, which achieves low power consumption by eliminating pipeline registers, replacing complex completion detector with simpler OR gates, and mitigating the leakage power of sleeping blocks by fine-grain power gating. Compared with the register-less NCL, the RL-NCL implementation of the kogge-stone adder can reduce the power dissipation by 50% for the input data range 100MHz.

VI. 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. Man tooth, "An asynchronous cell library for operation in wide-temperature & ionizing-radiation environments," in Proc. 2015 IEEE Aerospace Conf., Mar. 2015, pp. 1-10.

[13] 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.

[14] 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.

[15] 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.