5-2012

Ultra-Low Power and Radiation Hardened Asynchronous Circuit Design

Liang Zhou
University of Arkansas, Fayetteville

ULTRA-LOW POWER AND RADIATION HARDENED ASYNCHRONOUS CIRCUIT DESIGN

ULTRA-LOW POWER AND RADIATION HARDENED ASYNCHRONOUS CIRCUIT DESIGN

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

By

Liang Zhou
University of Arkansas
Master of Science in Electrical Engineering, 2012
Hubei University
Bachelor of Engineering in Telecommunication Engineering, 2007

May 2012
University of Arkansas

ABSTRACT

This dissertation proposes an ultra-low power design methodology called bit-wise MTNCL for bit-wise pipelined asynchronous circuits, which combines multi-threshold CMOS (MTCMOS) with bit-wise pipelined NULL Convention Logic (NCL) systems. It provides the leakage power advantages of an all high-Vt implementation with a reasonable speed penalty compared to the all low-Vt implementation, and has negligible area overhead. It was enhanced to handle indeterminate standby states. The original MTNCL concept was enhanced significantly by sleeping Registers and Completion Logic as well as Combinational circuits to reduce area, leakage power, and energy per operation.

This dissertation also develops an architecture that allows NCL circuits to recover from a Single Event Upset (SEU) or Single Event Latchup (SEL) fault without any data loss. Finally, an accurate throughput derivation formula for pipelined NCL circuits was developed, which can be used for static timing analysis.

This dissertation is approved for recommendation
to the Graduate Council.

Dissertation Director:

_____
Dr. Scott C. Smith

Dissertation Committee:

_____
Dr. Jia Di

_____
Dr. Randy Brown

_____
Dr. Alan Mantooth

DISSERTATION DUPLICATION RELEASE

I hereby authorize the University of Arkansas Libraries to duplicate this dissertation when needed for research and/or scholarship.

Agreed _____
Liang Zhou

Refused _____
Liang Zhou

ACKNOWLEDGEMENTS

Special thanks are due to Dr. Scott C. Smith, Dr. Jia Di, Dr. Randy Brown, and Dr. Alan Mantooth for all of their help.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Objectives

This Ph.D. dissertation has 2 main innovations. First, a design methodology is developed that combines Multi-Threshold CMOS (MTCMOS) and NULL Convention Logic (NCL) to reduce power consumption in standby mode. Second, an architecture is proposed that allows NCL circuits to recover from a Single Event Upset (SEU) and Single Event Latchup (SEL) fault without any data loss. Analytical and experimental results are discussed to validate the proposed schemes.

1.2 Design Challenges

With the current trend of semiconductor devices scaling into the deep submicron region, design challenges that were previously minor issues have now become increasingly important. In the past, dynamic switching power has been the predominant factor in CMOS digital circuit power dissipation. Recently, with the dramatic decrease of supply and threshold voltages, a significant growth in leakage power demands new design methodologies for digital integrated circuits (ICs). The main component of leakage power is sub-threshold leakage, caused by current flowing through a transistor even if it is supposedly turned off. Sub-threshold leakage increases exponentially with decreasing transistor feature size.

Semiconductor devices are becoming susceptible to particle strikes as they shrink to the nano-scale. There are 2 major negative effects caused by particle strikes: single event upset (SEU) and single event latchup (SEL).

When radiation-induced particles with sufficient energy hit the silicon substrate of a CMOS chip, a large number of electron-hole pairs are generated and an undesired short-duration

current may be formed, which can change the output of a logic gate. This is called a soft error or single event upset (SEU). SEU can cause deadlock or can cause incorrect data to be output.

Adjacent n-type and p-type regions in CMOS circuits may form a parasitic thyristor composed of two pairs of parasitic bipolar transistors. A spurious current spike induced by an ionizing particle in one of these transistors may be amplified by the large positive feedback of the thyristor. This will cause a virtual short between power and ground, resulting in a single-event latchup (SEL). SEL can cause permanent damage by huge current, or the outputs of multiple logic gates may be changed.

1.3 Previous Work

Multi-Threshold CMOS (MTCMOS) [1] is a very promising technology to control or minimize leakage power in deep submicron technology. It incorporates transistors with two or more different threshold voltages (Vt) in a circuit. Low-Vt transistors offer fast speed but have high leakage, whereas high-Vt transistors have reduced speed but far less leakage current. MTCMOS combines these two types of transistors by utilizing low-Vt transistors for circuit switching to preserve performance and high-Vt transistors to gate the circuit power supply to significantly decrease sub-threshold leakage.

Quasi-delay-insensitive (QDI) NULL Convention Logic (NCL) circuits [2] designed using CMOS exhibit an inherent idle behavior since they only switch when useful work is being performed; however, there is still significant leakage power during idle mode. MTNCL [3-7] combines the MTCMOS technique with NCL to sleep the NCL circuit during idle mode, in lieu of the NULL cycle, to yield a fast ultra-low power asynchronous circuit design methodology. MTNCL requires less area than the original NCL circuit. Other authors [8, 9] proposed schemes

integrating MTCMOS and QDI asynchronous circuits to reduce leakage power at the cost of more area and more active power than the original asynchronous circuits. MTNCL is better because it not only reduces leakage power, but also active power and area. NCL systems can be optimized for speed by partitioning the combinational circuitry and inserting additional NCL registers and corresponding completion components, utilizing either the full-word or bit-wise completion strategy [10]. MTNCL only utilizes full-word completion and can only be applied to asynchronous circuits in which the values of all signals can be determined in the standby state.

An SEU resistant design methodology [11-14] is proposed for asynchronous circuits implemented with Pre-Charged Half Buffers (PCHB). This methodology can make a PCHB circuit immune to SEU all the time. The idea is to double the circuit and use TH22 gates to stop the propagation of corrupted data and acknowledge/request signals. This makes the circuit completely SEU resistant when outputs from 2 copies are both observable. Other methodologies [15-17] have been proposed to design NCL circuits with reduced possibility that SEU may cause deadlock or wrong data, but none of them can make NCL circuits completely SEU resistant.

An SEL resistant design methodology [18-19] is proposed for memories. The circuit is divided into several redundant groups in such a way that if the states of one group are corrupted by SEL, they can be recovered from states of other groups by error correcting codes. The global power supply line is connected to many virtual power supply lines by current-limiting transistors. Each virtual power supply line is used by only one redundant group. When SEL happens, it will not cause permanent damage because the large current will be detected by a comparator and limited by gating the current-limiting transistors. No schemes were proposed to make NCL circuits completely SEL resistant.

A throughput estimation formula for pipelining NCL systems was proposed in the literature [10]. However, it ignores register delays and is therefore not accurate enough to be used for static timing analysis, especially when applied to finely pipelined NCL systems.

1.4 Dissertation Overview

This dissertation is organized into 10 chapters. Chapter 2 introduces NULL Convention Logic. Chapter 3 introduces Multi-Threshold CMOS. Chapter 4 introduces MTCMOS NCL. These chapters provide the basis for the rest of the dissertation. Chapter 5 presents a technique for utilizing bit-wise completion with MTNCL to produce a fast ultra-low power bit-wise pipelined asynchronous circuit design methodology. This methodology will be called bit-wise MTNCL. Chapter 6 provides significant enhancements to the original MTNCL concept; and Chapter 7 extends bit-wise MTNCL to handle indeterminate standby states. Chapter 8 develops an architecture that allows NCL circuits to recover from a SEU or SEL fault without any data loss. Chapter 9 derives an accurate throughput derivation formula for pipelined NCL circuits. And Chapter 10 concludes the dissertation.

2. INTRODUCTION TO NCL

NCL circuits utilize multi-rail logic, such as dual-rail, to achieve delay-insensitivity. A dual-rail signal, *D*, consists of two wires, D^0 and D^1, which may assume any value from the set {DATA0, DATA1, NULL}. The DATA0 state ($D^0 = 1$, $D^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state ($D^0 = 0$, $D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0$, $D^1 = 0$) corresponds to the empty set meaning that the value of *D* is not yet available. The two rails are mutually exclusive, such that both rails can never be asserted simultaneously; this state is defined as an illegal state. Dual-rail logic is a space optimal 1-hot delay-insensitive code, requiring two wires per bit.

NCL circuits are comprised of 27 fundamental gates [20]. These 27 gates constitute the set of all functions consisting of four or fewer variables. Here, a variable refers to one rail of a multi-rail signal; hence, a four variable function is not the same as a function of four literals, which would consist of eight variables, assuming dual-rail logic. The primary type of threshold gate, shown in Fig. 1, is the THmn gate, where $1 \leq m \leq n$. THmn gates have *n* inputs. At least *m* of the *n* inputs must be asserted before the output will become asserted. NCL threshold gates are designed with hysteresis state-holding capability such that all asserted inputs must be de-asserted before the output will be de-asserted. Hysteresis ensures a complete transition of inputs back to NULL before asserting the output associated with the next wavefront of input data. Therefore, a THnn gate is equivalent to an n-input C-element [21] and a TH1n gate is equivalent to an n-input OR gate. In a THmn gate, each of the *n* inputs is connected to the rounded portion of the gate; the output emanates from the pointed end of the gate; and the gate's threshold value, *m*, is written inside of the gate. NCL threshold gates may also include a reset input to initialize the

output. These resettable gates are used in the design of Delay-Insensitive (DI) registers [10]; an *N* inside the gate denotes it as being reset to 0, and a *D* as reset to 1.

Another type of threshold gate is referred to as a weighted threshold gate [20], denoted as THmnW$w_1w_2\ldots w_R$. Weighted threshold gates have an integer value, $m \geq w_R > 1$, applied to *inputR*. Here $1 \leq R < n$; where *n* is the number of inputs; *m* is the gate's threshold; and w_1, w_2, $\ldots w_R$, each > 1, are the integer weights of *input1*, *input2*, … *inputR*, respectively.

As shown in Fig. 2 (a), an NCL gate can be implemented with 4 equations: *set*, *reset*, *hold0*, *hold1*. The *set* equation is the condition when the gate will be asserted and the *reset* equation is the condition when the gate will be de-asserted. The *hold0* equation is the complement of the *set* equation and the *hold1* equation is the complement of the *reset* equation. The TH23 gate is implemented in Fig. 2 (b). Its *set* equation is AB + BC + AC and its *reset* equation is A'B'C'. Its *hold0* equation is A'B' + B'C' + A'C' and its *hold1* equation is A + B + C.

NCL systems contain at least two DI registers, one at both the input and at the output, and can be finely pipelined by inserting additional registers, as shown in Fig. 3. Two adjacent register stages interact through their request and acknowledge signals, *Ki* and *Ko*, respectively, to prevent the current DATA wavefront from overwriting the previous DATA wavefront, by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. The acknowledge signals are combined in the Completion Detection circuitry to produce the request signal(s) to the previous register stage, utilizing either the full-word or bit-wise completion strategy [10].

To ensure delay-insensitivity, NCL circuits must adhere to the following criteria: Input-Completeness [22] and Observability [22]. Input-Completeness requires that 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 outputs of a combinational circuit may not transition from

DATA to NULL until all inputs have transitioned from DATA to NULL. In circuits with multiple outputs, it is acceptable according to Seitz's "weak conditions" of DI signaling [23], for some of the outputs to transition without having a complete input set present, as long as all outputs cannot transition before all inputs arrive. Observability requires that no orphans may propagate through a gate [24]. An orphan is defined as a wire that transitions during the current DATA wavefront, but is not used in the determination of the output. Orphans are caused by wire forks and can be neglected through the isochronic fork assumption [25, 26], as long as they are not allowed to cross a gate boundary. This observability condition, also referred to as indicatability or stability, ensures that every gate transition is observable at the output, which means that every gate that transitions is necessary to transition at least one of the outputs.

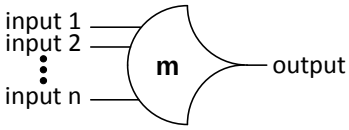
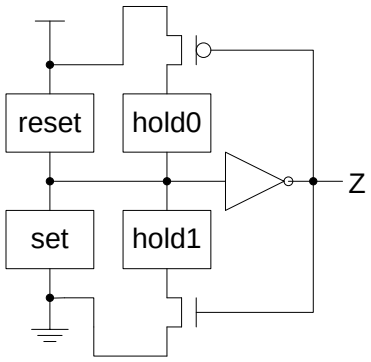
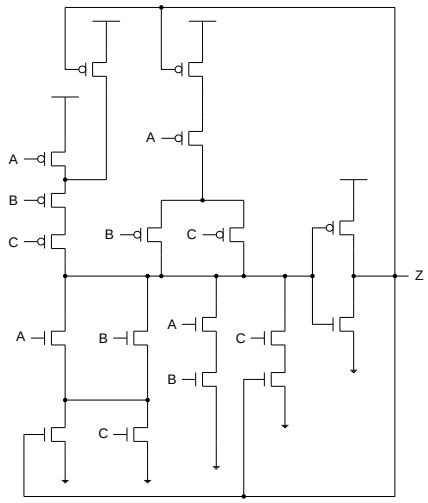


Figure 1. THmn threshold gate.



a) General implementation



b) TH23 implementation

Figure 2. NCL threshold gate design.

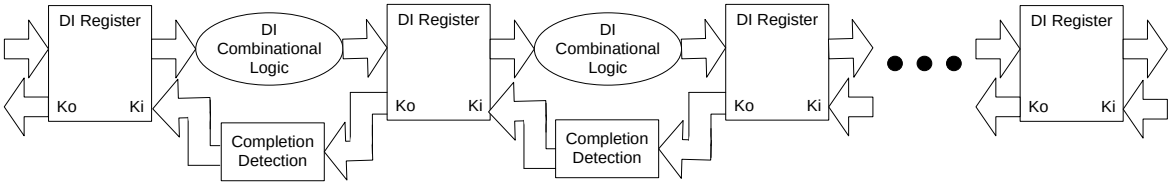


Figure 3. NCL system framework.

3. INTRODUCTION TO MTCMOS

MTCMOS reduces leakage power by disconnecting the power supply from the circuit during idle (or sleep) mode while maintaining high performance in active mode by utilizing different transistor threshold voltages (Vt) [1]. Low-Vt transistors are faster but have high leakage, whereas high-Vt transistors are slower but have far less leakage current. MTCMOS combines these two types of transistors by utilizing low-Vt transistors for circuit switching to preserve performance and high-Vt transistors to gate the circuit power supply to significantly decrease sub-threshold leakage.

One MTCMOS method uses low-Vt transistors for critical paths to maintain high performance, while using slower high-Vt transistors for the non-critical paths to reduce leakage. Besides this path replacement methodology, there are two other architectures for implementing MTCMOS. A course-grained technique investigated in [27] uses low-Vt logic for all circuit functions and gates the power to entire logic blocks with high-Vt sleep transistors, denoted by a dotted circle, as shown in Fig. 4. The sleep transistors are controlled by a Sleep signal. During active mode, the Sleep signal is de-asserted, causing both high-Vt transistors to turn on and provide a virtual power and ground to the low-Vt logic. When the circuit is idle, the Sleep signal is asserted, forcing both high-Vt transistors to turn off and disconnect power from the low-Vt logic, resulting in a very low sub-threshold leakage current. One major drawback of this method is that partitioning the circuit into appropriate logic blocks and sleep transistor sizing is difficult for large circuits. An alternative fine-grained architecture, shown in Fig. 5, incorporates the MTCMOS technique within every gate [28], using low-Vt transistors for the Pull-Up Network (PUN) and Pull-Down Network (PDN) and a high-Vt transistor to gate the leakage current

between the two networks. Two additional low-Vt transistors are included in parallel with the PUN and PDN to maintain nearly equivalent voltage potential across these networks during sleep mode (i.e., X1 is approximately V_{DD} and X2 is approximately GND). Implementing MTCMOS within each gate solves the problems of logic block partitioning and sleep transistor sizing, since each gate within the gate library is sized separately; however, this results in a large area overhead.

In general, three serious drawbacks hinder the widespread usage of MTCMOS in synchronous circuits [27]: 1) the generation of Sleep signals is timing critical, often requiring complex logic circuits; 2) synchronous storage elements lose data when the power transistors are turned off during sleep mode; and 3) logic block partitioning and transistor sizing is very difficult for the course-grained approach, which is critical for correct circuit operation, and the fine-grained approach requires a large area overhead. However, all three of these drawbacks are eliminated by utilizing NCL in conjunction with the MTCMOS technique.

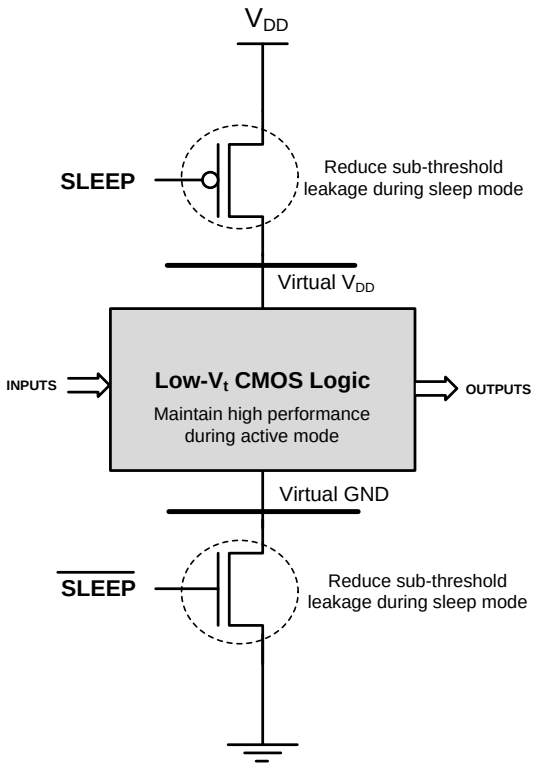


Figure 4. General MTCMOS circuit architecture [27].

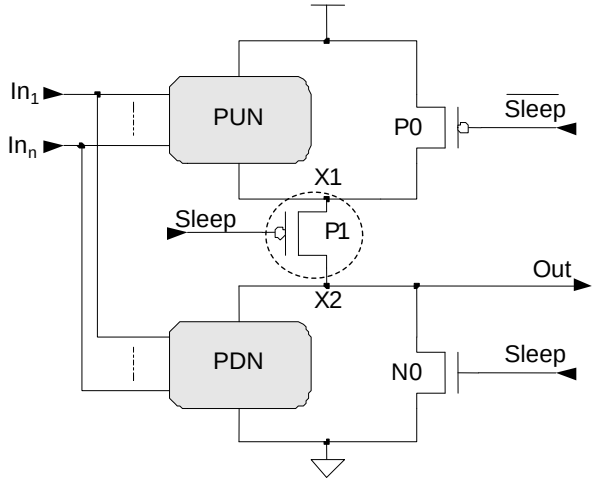


Figure 5. MTCMOS applied to a Boolean gate [28].

4. INTRODUCTION TO MTNCL

MTNCL was originally developed in [3-7], as summarized below, while Chapter 5 provides significant enhancements to the original MTNCL concept.

4.1 Early-Completion Input-Incomplete (ECII) MTNCL Architecture

NCL threshold gates are larger and implement more complicated functions than basic Boolean gates, such that fewer threshold gates are normally needed to implement an arbitrary function compared to the number of Boolean gates; however, the NCL implementation often requires more transistors. Therefore, incorporating MTCMOS inside each threshold gate facilitates easy sleep transistor sizing without requiring as large of an area overhead. Since floating nodes may result in substantial short circuit power consumption at the following stage, an MTCMOS structure similar to the one shown in Fig. 5 is used to pull the output node to ground during sleep mode. When all MTNCL gates in a pipeline stage are in sleep mode, all gate outputs are logic 0. This condition is equivalent to the pipeline stage being in the NULL state. Hence, after each DATA cycle, all MTNCL gates in a pipeline stage can be forced to output logic 0 by asserting the sleep control signal instead of propagating a NULL wavefront through the stage, such that data is not lost during sleep mode.

Since the completion detection signal, *Ko*, indicates whether the corresponding pipeline stage is ready to undergo a DATA or NULL cycle, *Ko* can be naturally used as the sleep control signal, without requiring any additional hardware, in contrast to the complex Sleep signal generation circuitry needed for synchronous MTCMOS circuits. Unfortunately, the direct implementation of this idea using regular NCL completion compromises delay-insensitivity [6].

To solve this problem, Early Completion [29] can be used in lieu of regular completion, as shown in Fig. 6, where each completion signal is used as the sleep signal for all threshold gates in the subsequent pipeline stage. Early Completion utilizes the inputs of register$_{i-1}$ along with the *Ki* request to register$_{i-1}$, instead of just the outputs of register$_{i-1}$ as in regular completion, to generate the request signal to register$_{i-2}$, Ko_{i-1}. The combinational logic will not be put to sleep until all inputs are NULL and the stage is requesting NULL; therefore the NULL wavefront is ready to propagate through the stage, so the stage can instead be put to sleep without compromising delay-insensitivity. The stage will then remain in sleep mode until all inputs are DATA and the stage is requesting DATA, and is therefore ready to evaluate. This Early Completion MTNCL architecture, denoted as ECII, ensures input-completeness and observability through the sleep mechanism (i.e., the circuit is only put to sleep after all inputs are NULL, when all gates are then simultaneously forced to logic 0, and only evaluates after all inputs are DATA), such that input-incomplete logic functions can be used to design the circuit, which decreases area and power and increases speed.

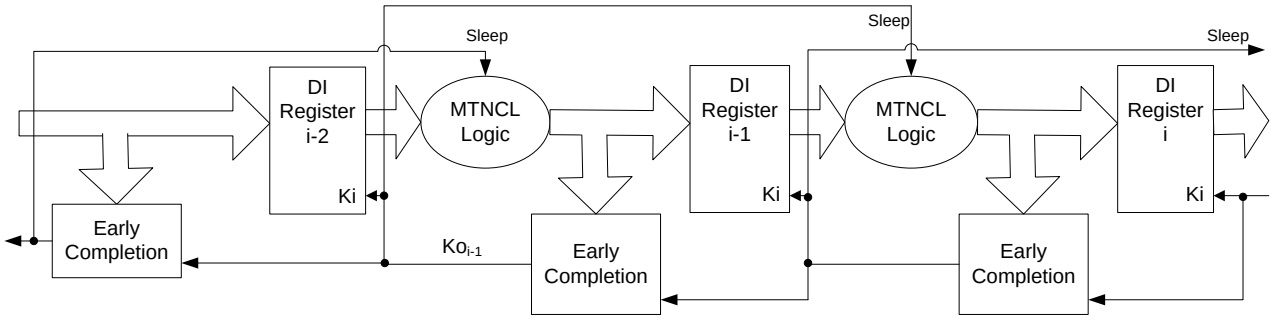


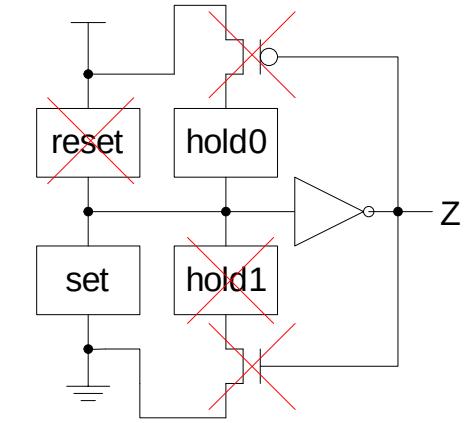
Figure 6. MTNCL pipeline architecture using Early Completion.

4.2 MTNCL Threshold Gate Design for ECII Architecture

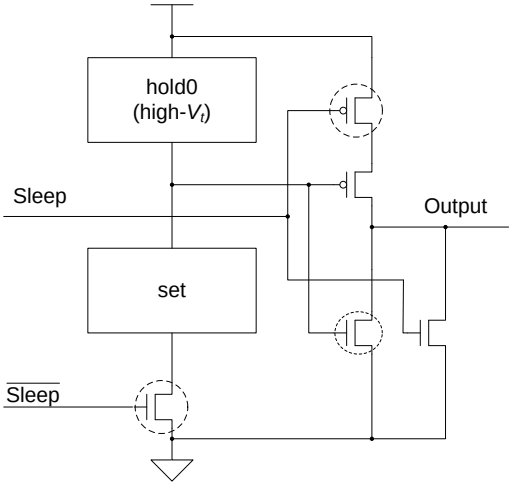
The MTCMOS structure is incorporated inside each NCL threshold gate, and actually results in a number of the original transistors no longer being needed. As shown in Fig. 7a, the reset circuitry is no longer needed, since the gate output will now be forced to NULL by the MTCMOS sleep mechanism, instead of by all inputs becoming logic 0. hold1 is used to ensure that the gate remains asserted, once it has become asserted, until all inputs are de-asserted, in order to guarantee input-completeness with respect to the NULL wavefront; however, since the ECII architecture guarantees input-completeness through the sleep mechanism, as explained in Chapter 4.1, it follows that NCL gate hysteresis is no longer required. Hence, the hold1 circuitry and corresponding NMOS transistor are removed, and the PMOS transistor is removed to maintain the complementary nature of CMOS logic (i.e., set and hold0 are complements of each other), such that the gate is never floating.

Improved from the direct MTCMOS NCL threshold gate implementation [3] similar to the structure shown in Fig. 5, a modified Static MTNCL threshold gate structure, referred to as SMTNCL, is shown in Fig. 7b. This modification eliminates the output wake-up glitch by moving the power gating high-Vt transistor to the PDN, and removing the two bypass transistors. All PMOS transistors except the output inverter are high-Vt, because they are only turned on when the gate enters sleep mode and the inputs become logic 0, and remain on when the gate exits sleep mode, until the gate's set condition becomes true. In both cases, the gate output is already logic 0; therefore, the speed of these PMOS transistors does not affect performance, so high-Vt transistors are used to reduce leakage current. During active mode, the Sleep signal is logic 0 and $\overline{Sleep}$ is logic 1, such that the gate functions as normal. During sleep mode, Sleep is

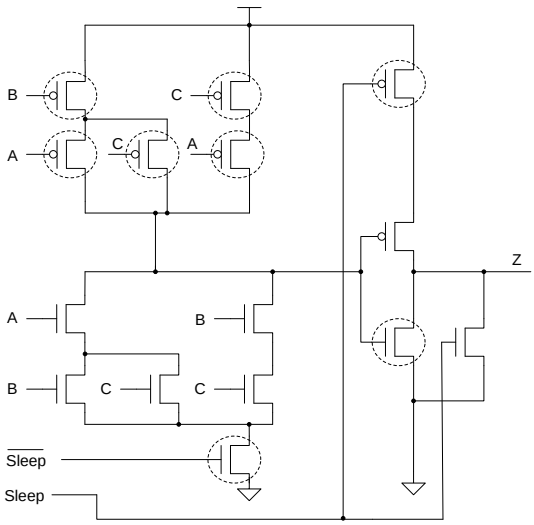
logic 1 and $\overline{Sleep}$ is logic 0, such that the output low-Vt pull-down transistor is turned on quickly to pull the output to logic 0, while the high-Vt NMOS gating transistor is turned off to reduce leakage. Note that since the internal node, between set and hold0, is logic 1 during sleep mode and the output is logic 0, the NMOS transistor in the output inverter is no longer on the critical path and therefore can be a high-Vt transistor. As an example, this SMTNCL implementation of the static TH23 gate is shown in Fig. 7c.



a)



b)



c)

Figure 7. (a) Incorporating MTCMOS into NCL threshold gates,

(b) SMTNCL gate structure, and (c) TH23 implementation.

4.3 Delay-Insensitivity Analysis

Combining the ECII architecture with the SMTNCL gate structure, results in a delay-sensitivity problem, as shown in Fig. 8. After a DATA cycle, if most, but not all, inputs become NULL, this Partial NULL (PN) wavefront can pass through the stage's input register, because the subsequent stage is requesting NULL, and cause all stage outputs to become NULL, before all inputs are NULL and the stage is put to sleep, because the hold1 logic has been removed from the SMTNCL gates. This violates the input-completeness criteria, discussed in Chapter 2, and can cause the subsequent stage to request the next DATA while the previous stage input is still a partial NULL, such that the preceding wavefront bits that are still DATA will be retained and utilized in the subsequent operation, thereby compromising delay-insensitivity, similar to the problem when using regular completion instead of Early Completion for MTNCL [6].

There are two solutions to this problem, one at the architecture level and the other at the gate level. Since the problem is caused by a partial NULL passing through the register, this can be fixed at the architecture-level by ensuring that the NULL wavefront is only allowed to pass through the register after all register inputs are NULL, which is easily achievable by using the stage's inverted sleep signal as its input register's *Ki* signal. This Fixed Early Completion Input-Incomplete (FECII) architecture is shown in Fig. 9. Compared to ECII, FECII is slower because the registers must wait until all inputs become DATA/NULL before they are latched. Note that a partial DATA wavefront passing through the register does not pose a problem, because the stage will remain in sleep mode until all inputs are DATA, thereby ensuring that all stage outputs will remain NULL until all inputs are DATA.

This problem can also be solved at the gate level by adding the hold1 logic back into each SMTNCL gate, to ensure input-completeness with respect to NULL, such that a partial NULL wavefront cannot cause all outputs to become NULL. Note that this requires the PMOS transistor between hold0 and V_{DD} to be added back to prevent a direct path from V_{DD} to ground when both hold1 and hold0 are simultaneously asserted. Also note that the hold1 transistors not shared with the set condition can be high-Vt transistors, since they are not on the critical path. This Static MTNCL implementation with hold1 is shown in Fig. 10, and is denoted as SMTNCL1.

To summarize, the ECII architecture only works with SMTNCL1 gates, which include the hold1 function. The FECII architecture works with SMTNCL and SMTNCL1 gates; however, SMTNCL gates would normally be used with FECII since they require fewer transistors. Additionally, the ECII architecture is faster than FECII, when both use the same MTNCL gates.

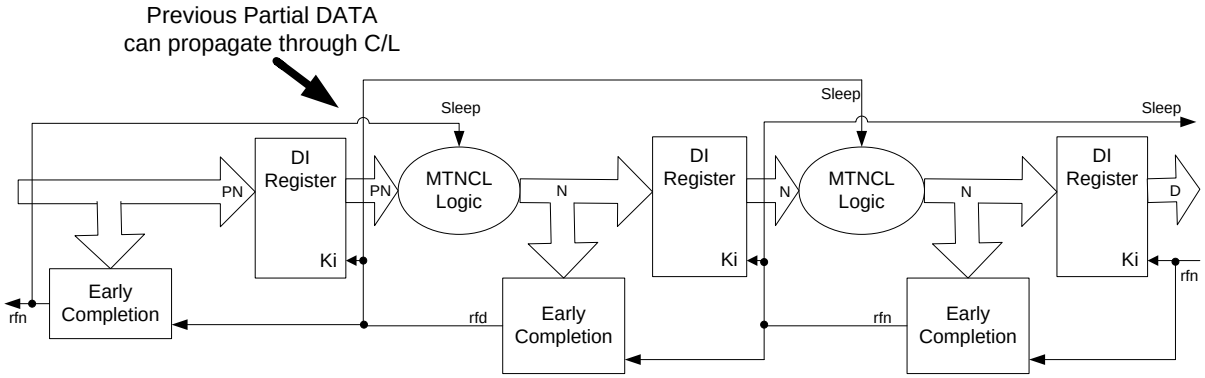


Figure 8. Delay-sensitivity problem combining ECII architecture with SMTNCL gates.

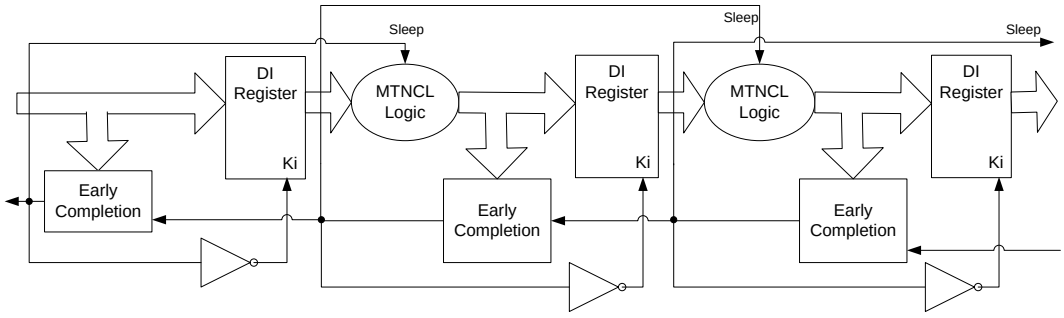


Figure 9. Fixed Early Completion Input-Incomplete (FECII) architecture.

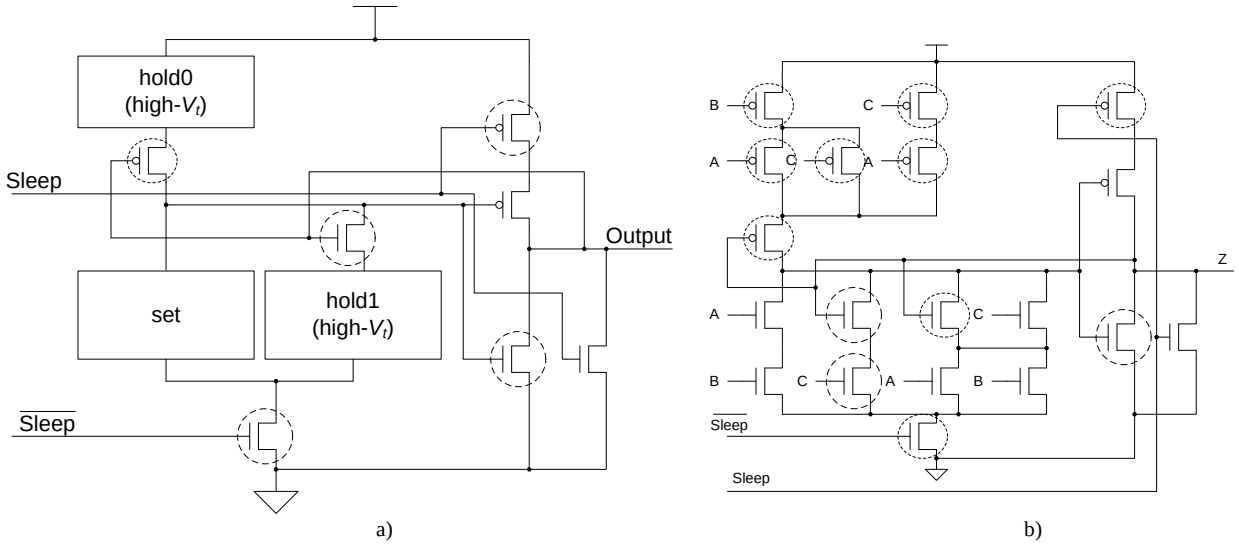


Figure 10. (a) SMTNCL1 gate structure, and (b) TH23 implementation.

5. BIT-WISE MTNCL (BWMTNCL)

5.1 Design Methodology

In NCL systems without feedback loops, the inputs of each gate while in the standby state are determinate, since all circuit inputs will be NULL, which causes all Combinational Logic gates and the data inputs and outputs of all registers to be de-asserted, which in turn causes all Completion Logic gates to be asserted (i.e., request-for-data or rfd), as shown in Fig. 11.

The leakage path is defined as the path formed by "on" transistors and "off" low-Vt transistors in standby state. To substantially reduce leakage power while degrading speed as little as possible, the following rules should be utilized to determine which transistors should be high-Vt and which transistors should be low-Vt [7]:

1. Determine threshold gate input and output values in standby state.

2. All transistors "on" in standby state should be low-Vt.

3. Replace the minimal number of "off" transistors with high-Vt transistors to eliminate leakage path, and replace the rest with low-Vt transistors.

Fig. 12 shows the standby states of a 1-bit dual-rail NCL register, which consists of two TH22 resettable to '0' gates with A = '0', B = '1', reset = '0' and one inverted TH12 gate with both '1' inputs in standby state. After applying those 3 rules, the schematic of the TH22 gate is given in Fig. 13, in which circled transistors are high-Vt and not circled are low-Vt transistors. T0, T1, and T2 are high-Vt because they do not switch except for initialization.

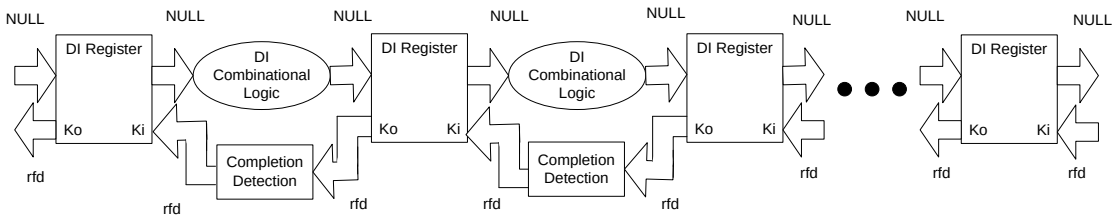


Figure 11. NCL system framework without feedback loops in standby state.

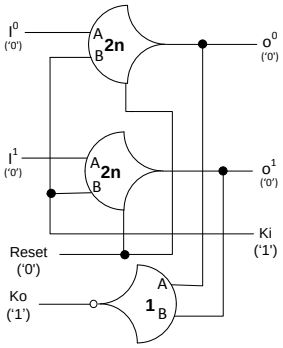


Figure 12. 1-Bit NCL register in standby state.

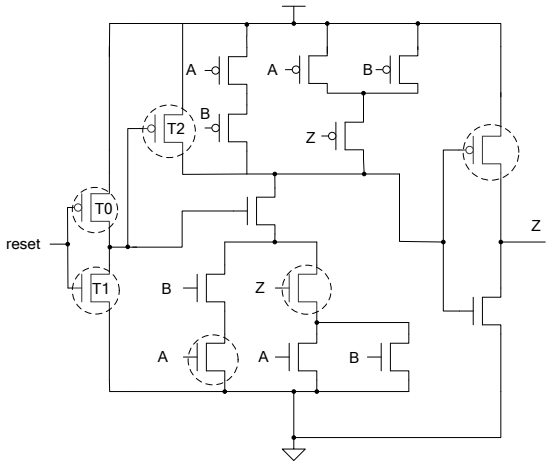


Figure 13. BWMTNCL applied to TH22 resettable to '0' gate with 1 standby state:

reset = '0', A = '0', B = '1', Z = '0'.

Compared to original NCL circuits implemented with all low-Vt and high-Vt transistors, respectively, BWMTNCL provides the leakage power advantages of the all high-Vt NCL

implementation with a reasonable speed penalty compared to the all low-Vt design, and has negligible area overhead.

5.2 Simulation Results

Three implementations of a 4-bit×4-bit pipelined multiplier utilizing bit-wise completion [5] were simulated at the transistor level, using the 1.2V IBM 8RF-LM 130nm CMOS process, and compared in terms of number of transistors, average cycle time (T_{DD}), energy per operation, and leakage power, as listed in Table I. As expected, the original bit-wise NCL circuit using all high-Vt transistors has the lowest leakage power, but is more than twice as slow as the all low-Vt implementation; while the all low-Vt implementation is fastest, but has 2 orders of magnitude more leakage power than the other circuits. BWMTNCL has 2 orders of magnitude less leakage power than the all low-Vt implementation, with the same area and approximately the same energy/operation, while only being 30% slower. BWMTNCL is 77% faster than the all high-Vt implementation, while requiring the same area, and only slightly more leakage power.

Table I. Bit-wise MTNCL comparisons.

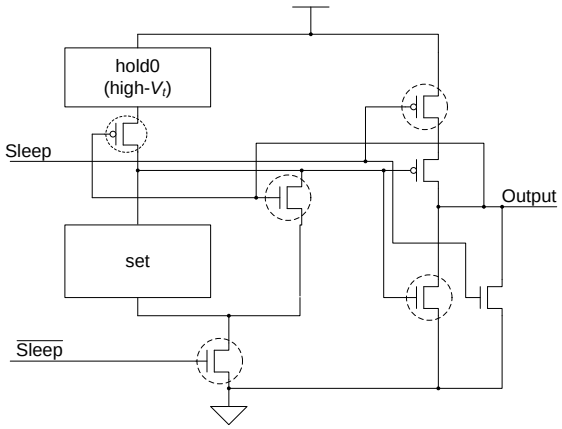
	# Transistors	T_{DD} (ns)	Energy/Operation (pJ)	Leakage Power (nW)
Low-Vt NCL	4570	2	1.39	571
High-Vt NCL	4570	4.6	1.44	5.22
BWMTNCL	4570	2.6	1.37	5.48

6. MTNCL ENHANCEMENTS

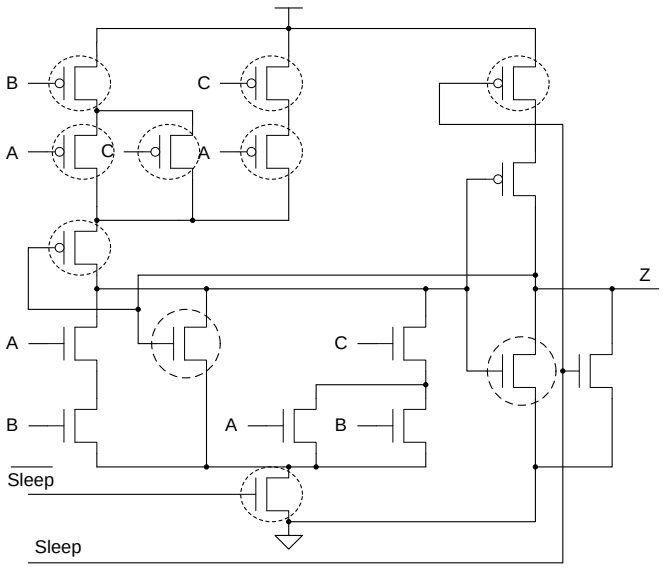
The previous SMTNCL1 gate, shown in Fig. 10, requires a significant number of additional transistors to implement the hold1 functionality; however, the number of additional transistors can be significantly reduced. Additionally, the previous MTNCL architecture only allows for the Combinational Logic (C/L) to be slept, whereas this chapter develops two modified MTNCL architectures, where 1) the completion logic can also be slept, and 2) both the registration and completion logic can be slept.

6.1 New SMTNCL1 Gate

Fig. 14 shows the new SMTNCL1 gate, which only requires 2 additional transistors vs. the SMTNCL gate. The difference between the new SMTNCL1 gate in Fig. 14 and the previous version in Fig. 10 is that the hold1 logic has been removed. The feedback NMOS transistor is sufficient to hold the output at logic 1, without the hold1 circuitry, because this ensures that once the gate output has been asserted due to the current DATA wavefront, that it will only be de-asserted when the gate is put to sleep (i.e., when all circuit inputs are NULL), and will not be de-asserted due to a partial NULL wavefront.



a)



b)

Figure 14. (a) New SMTNCL1 gate structure, and (b) TH23 implementation.

6.2 Sleep Completion and Registration Logic

Chapter 4 described the MTNCL architecture where an NCL circuit's C/L was slept in lieu of the NULL cycle to significantly reduce leakage power. However, during sleep mode the circuit's completion and registration logic remains active, which for a fine grain pipelined circuit may be a significant portion of the logic. Therefore, it would be very beneficial to be able to sleep the completion and registration logic in addition to the C/L. The completion logic can be slept by modifying the ECII architecture, shown in Fig. 6, to include a sleep input to the completion logic and use SMTNCL1 gates to implement the completion logic, as shown in Figs. 15 and 16, respectively. Note that the final inverting TH22 gate is a regular NCL gate, which is not slept. This is consistent with the NULL cycle, where the internal completion component gates are all logic 0, except for the final inverting TH22 gate.

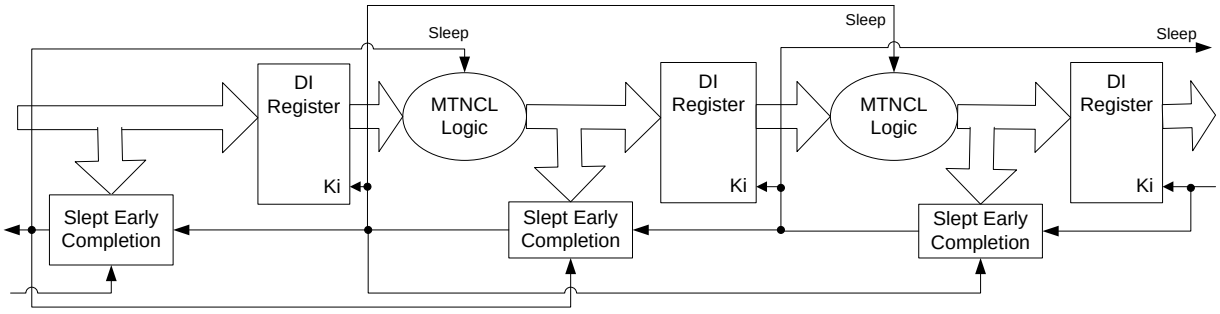


Figure 15. SECII architecture with Completion Logic slept.

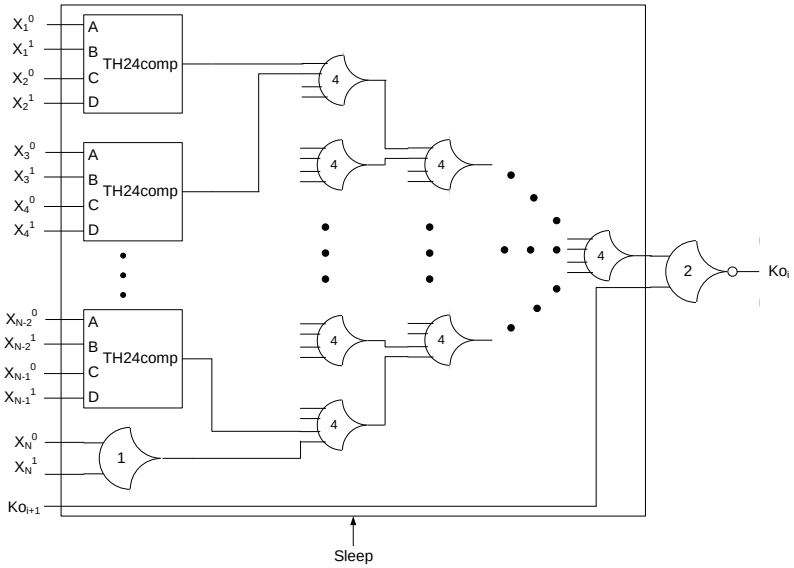


Figure 16. Early completion component with Sleep input.

During a NULL cycle, the register output is also NULL, so it too can be slept, as shown in Fig. 17. Instead of using two SMTNCL1 TH22 gates to implement the register, the sleep transistors for each rail can be combined, such that a dual-rail register is implemented as a single component in order to reduce area, as shown in Fig. 18. Note that this architecture is similar to the FECII architecture shown in Fig. 9, which does not allow a partial NULL to propagate through the register, such that the C/L can be implemented with the smaller SMTNCL gates instead of SMTNCL1 gates.

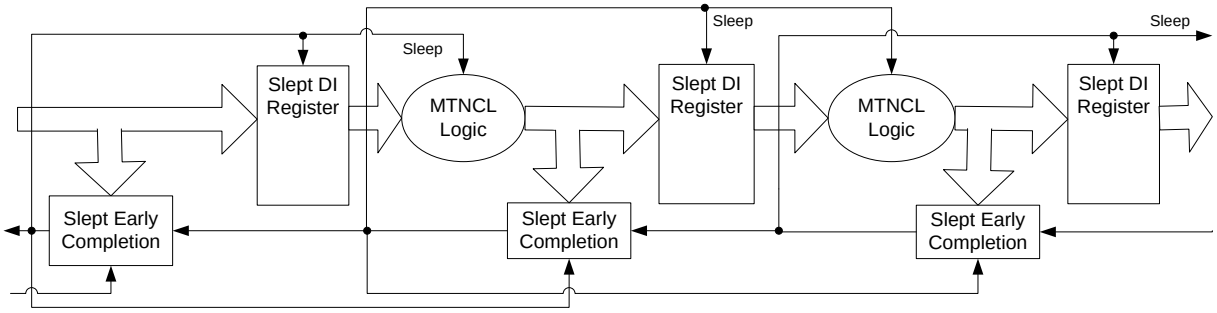


Figure 17. SECRII architecture with Completion Logic and Registration slept.

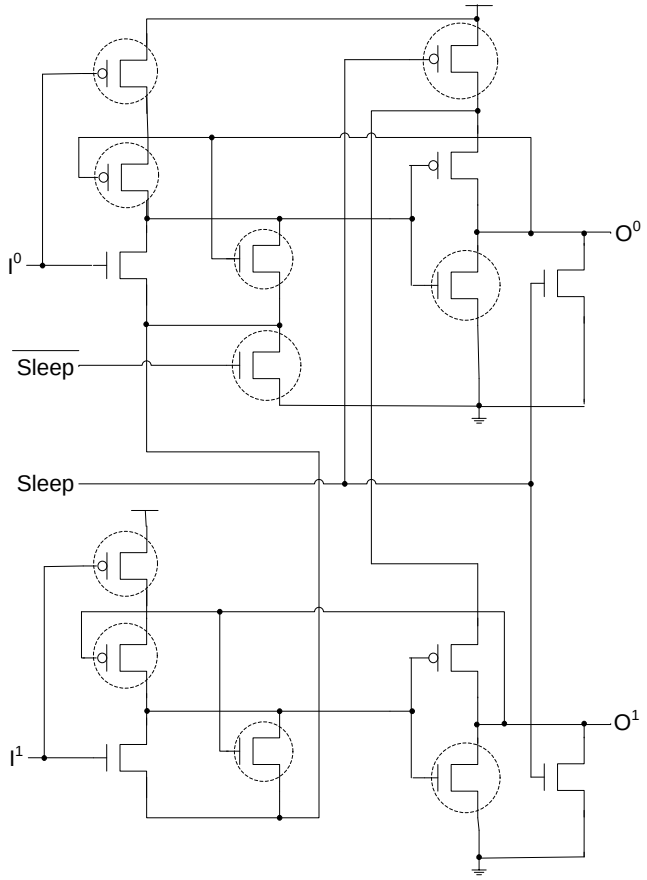


Figure 18. DI register with Sleep input.

6.3 Combine SECRII with BWMTNCL

The SMTNCL gates utilized in the SECRII architecture require both a sleep and nsleep input, each of which necessitates a large buffer tree. Hence, eliminating one of these inputs would decrease area and energy. The nsleep input can be eliminated from the SMTNCL gate by combining the SMTNCL architecture in Fig. 7 with the BWMTNCL architecture in Fig. 13, as shown in Fig. 19. Instead of utilizing a high-Vt transistor to gate the set logic from ground, the set logic is implemented in BWMTNCL fashion utilizing the minimum number of high-Vt transistors such that all paths through the set function to ground contain a high-Vt transistor.

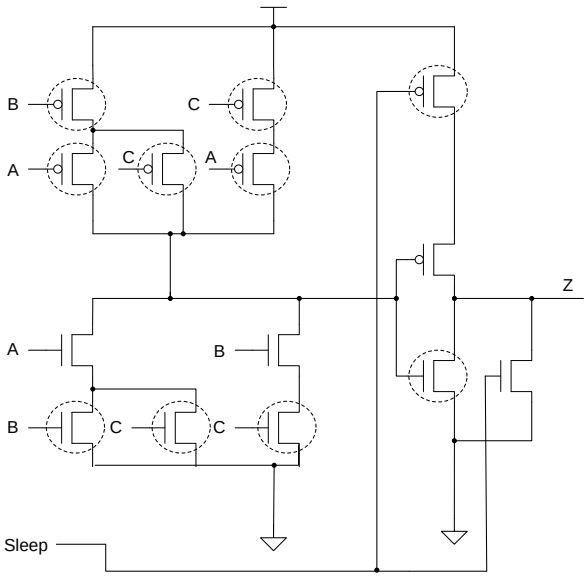


Figure 19. SMTNCL without nsleep applied to TH23 gate.

6.4 Simulation Results

To compare the various MTNCL architectures, a number of 4-stage pipelined IEEE single-precision floating-point co-processors, which perform addition, subtraction, and multiplication [30], were designed using the 1.2V IBM 8RF-LM 130nm CMOS process, and were simulated at the transistor level after inserting buffers using Cadence's UltraSim simulator

running a VerilogA controller in mixed-signal mode. The input patterns were randomized and the same input patterns were used for different designs. Note that all transistors are minimum sized except for the buffers. Table II lists the MTNCL results and also compares to the regular NCL implementation using all low-Vt transistors and all high-Vt transistors. The floating-point co-processor has two distinct datapaths, the add/subtract unit and the multiplier, which each have different throughput, so the data for each is presented separately, and can be averaged to yield the combined results. T_{DD} is the average DATA plus NULL processing time, which is comparable to the synchronous clock period. T_{DD} and Energy/Operation are calculated while the circuit is operating at its maximum speed, while Leakage Power is calculated using DC analysis after the pipeline is flushed with all NULL inputs. For the asynchronous circuits, leakage power doesn't depend on the previous type of operations (i.e., either add/sub or mult), since the following sleep state is the same (i.e., both pipelines are all NULL).

Comparing the various MTNCL designs shows that the new MTNCL gate with hold1 (SMTNCL1) requires less area, energy, and power than the previous version in [6], and is slightly faster. Sleeping the completion logic along with the C/L slightly reduces area, energy, and leakage power, and significantly increases speed, while sleeping the C/L, completion logic, and registers significantly decreases area, energy, and leakage power, and slightly increases speed. The SMTNCL with SECRII without nsleep design that combines the SMTNCL with SECRII and BWMTNCL architectures further reduces area and energy while increasing speed, at the cost of a slight increase in leakage power. Note that the FECII circuit is faster than the ECII circuit because the FECII design utilizes the faster SMTNCL gates.

The best MTNCL design, SMTNCL with SECRII without nsleep, requires 43% less area, 34% less energy, 2 orders of magnitude less leakage power, is 19% faster than the regular low-Vt

NCL design, and has 46% less leakage power than the regular high-Vt NCL design. Hence, the SMTNCL with SECRII without nsleep architecture presented herein vastly outperforms traditional NCL in all aspects: area, speed, energy, and leakage power.

Table II. MTNCL comparisons.

		TDD(ns)		Energy /Operation(pJ)		Leakage Power (nW)	
	#Transistors	add/sub	mult	add/sub	mult	add/sub	mult
NCL Low-Vt	158059	14.1	14.4	27.4	23.7	12300	12300
NCL High-Vt	158059	32.7	33.4	28.5	25.1	208	208
BWMTNCL	158059	17.9	16.2	27.1	23.7	190.7	190.7
SMTNCL with FECII	111506	11.6	15.3	14.9	27.5	115.9	115.9
Original SMTNCL1 with ECII	130476	12.5	16.7	16	27.8	140.8	140.8
New SMTNCL1 with ECII	119706	12.1	15.7	14.7	26.1	121.9	121.9
SMTNCL1 with SECII	119244	10.7	15.4	14.6	26	121.1	121.1
SMTNCL with SECRII	96640	11.1	14.8	13.5	25.3	111.2	111.2
SMTNCL with SECRII without nsleep	90041	10	13.9	12.1	21.8	112.1	112.1

7. BIT-WISE MTNCL ENHANCEMENTS

7.1 Asynchronous Circuits with Indeterminate Standby States

As an example of an asynchronous circuit with indeterminate standby states, an NCL unsigned 32+16×16 MAC is developed. As shown in Fig. 20, it consists of 3 parts: full-word pipelined 7-stage partial product generation and Wallace tree summation circuit (PP1, PP2), a 4-stage feedback loop which feeds back the accumulator as 2 partial products in carry-save form (A1, A2), and full-word pipelined 15-stage 30-bit Ripple-Carry-Adder. The architecture is elaborated in [32], except that it is full-word pipelined and extra control functions are removed for simplicity.

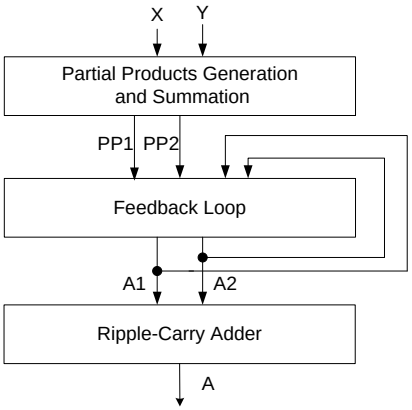


Figure 20. NCL MAC architecture.

Fig. 21 shows the standby states of the feedback loop, which has 2-level carry save adders to sum up new partial products PP1, PP2 and old accumulator A1, A2 to generate new accumulator in carry-save form. Partial DATA in Fig. 21 means that some bits of the register are DATA while the other bits are NULL.

In standby state, the old accumulator values are stored in register REG0 in Fig. 21. Each bit of REG0 can either be DATA0 or DATA1, which cannot be determined at design time. Similarly, register REG1, Combinational Logic COMB1, COMB2 and Completion Logic COMP0 also have indeterminate standby states. Therefore, BWMTNCL cannot be applied to this feedback loop.

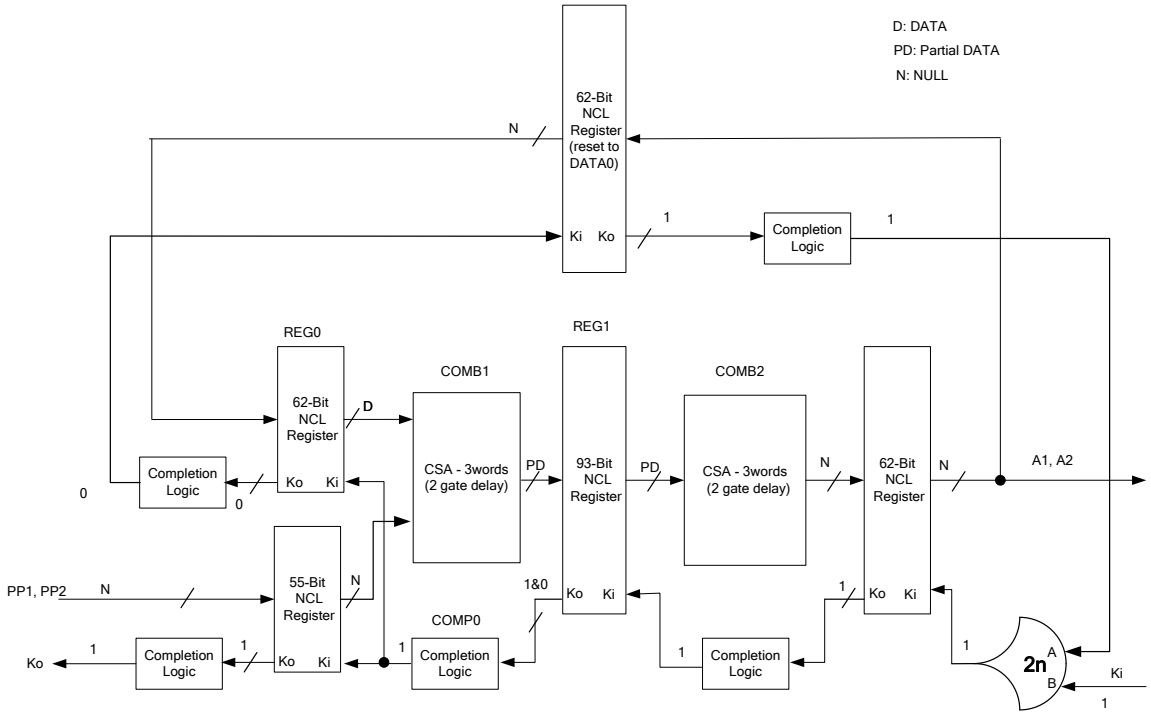


Figure 21. Standby states of the feedback loop without indeterminate states reduction.

7.2 Techniques to Handle Indeterminate Standby States

In asynchronous circuits with indeterminate standby states, usually not all of the inputs and outputs of a threshold gate are indeterminate in the standby state. For example, TH22 resettable to '0' gates used in REG0 in Fig. 21 have determinate standby states for *A*, *B*, and *reset* inputs (A = '0', B = '1', reset = '0'), but indeterminate standby states for output *Z* (Z = '1' or Z = '0'). In other words, it has 2 possible standby states. Instead of using an all high-Vt

implementation, those 2 standby states can be analyzed to use the minimal number of high-Vt transistors to eliminate the leakage paths in those 2 standby states, in order to substantially reduce leakage power while degrading speed as little as possible. The rules used in BWMTNCL can be enhanced as follows:

1. Determine the number of standby states and threshold gate input and output values in each standby state.

2. Replace the minimal number of transistors with high-Vt transistors to eliminate leakage paths in any standby state, and low-Vt transistors for the rest.

After applying these 2 rules, the schematic of the TH22 gate with 2 possible standby states is given in Fig. 22.

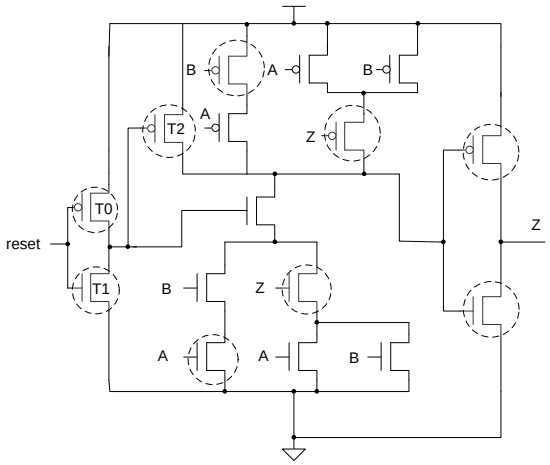


Figure 22. Enhanced BWMTNCL applied to TH22 resettable to ‘0’ gate with 2 standby states:

reset = ‘0’, A = ‘0’, B = ‘1’, Z = ‘0’ or Z = ‘1’

By comparing Fig. 13 with Fig. 22, it can be observed that Fig. 22 has 3 more high-Vt transistors than Fig. 13. These 3 high-Vt transistors are required by the extra standby state and make the TH22 gate in Fig. 22 slower than the one in Fig. 13. Therefore, it is beneficial to reduce

the number of gates with indeterminate standby states so that less high-Vt transistors can be used to degrade speed as little as possible.

To reduce the number of gates with indeterminate standby states, an inverter U0 and an asymmetric [32] TH22 gate U1 are added in Fig. 23 to control the *Ki* input of register REG0. The input *B* with '+' of U1 only takes effect in asserting the asymmetric TH22 gate. In other words, U1 will be asserted if both inputs are '1', and de-asserted if *A* is '0' regardless of the value of *B*. The preceding partial product generation and Wallace tree summation circuit has output register REG3, whose *Ko* output is asserted if PP1, PP2 are NULL and de-asserted if they are DATA.

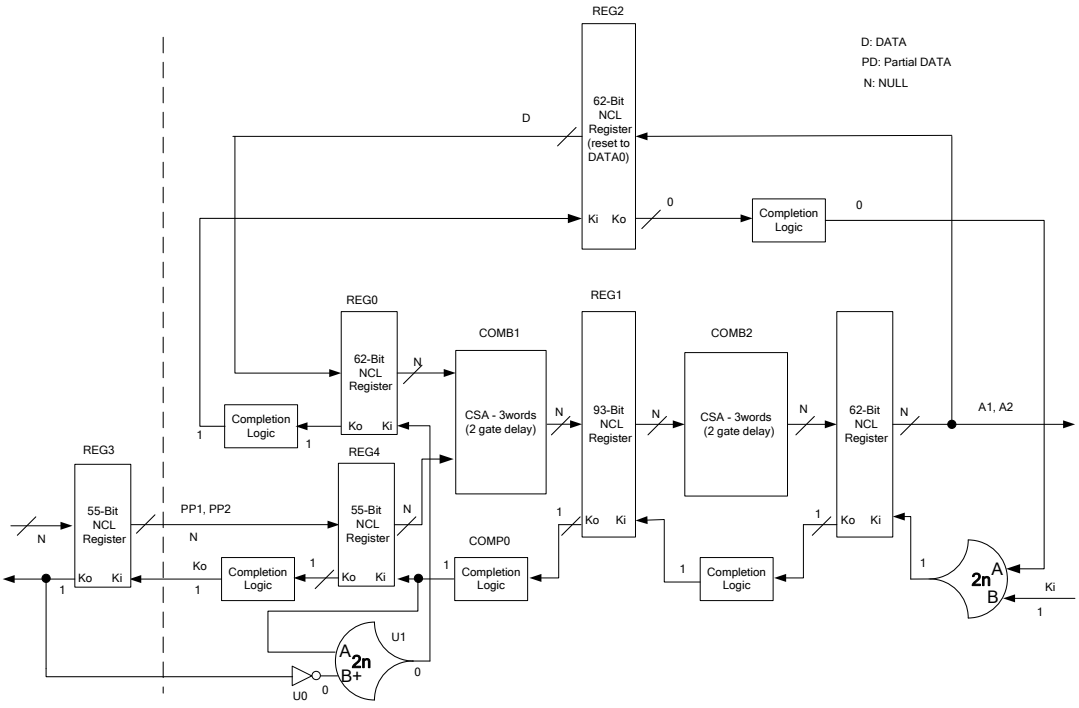


Figure 23. Standby states of the feedback loop with indeterminate states reduction.

As a result, the condition for the *Ki* input of REG0 to be de-asserted remains that Completion Logic COMP0 is de-asserted, but the condition to assert it is changed from that COMP0 is asserted to that COMP0 is asserted and PP1, PP2 are DATA. The function of the

feedback loop in active state is not changed, because the added condition that PP1, PP2 are DATA is always met in active state. But in standby state PP1, PP2 are NULL so that *Ki* input of REG0 cannot be asserted, and the old accumulator values are stored in REG2 instead of REG0. In Fig. 23, only REG2 and REG0 have determinate standby states, compared to REG0, REG1, COMB1, COMB2, and COMP0 in Fig. 21. As the number of gates with indeterminate standby states is significantly reduced, less high-Vt transistors are required, which degrades speed as little as possible.

The area overhead is negligible as only two gates are added. The added condition is equivalent to adding Combinational Logic with 1 gate delay between REG3 and REG4. As the speed of the whole circuit is limited by the slowest stage [10], which is the feedback loop which has much longer delay than the stage between REG3 and REG4, the speed of the whole circuit will not be degraded by adding those two gates.

Fig. 23 requires 4 types of gates with indeterminate standby states. First, REG2 requires TH22 gates resettable to ‘0’ with standby states reset = ‘0’, A = ’0’, B = ’1’, Z=’0’ or Z = ‘1’, which is already shown in Fig. 22. Second, REG2 requires TH22 gates resettable to ‘1’ with standby states reset = ‘0’, A =’0’, B =’1’, Z=’0’ or Z = ‘1’, which is shown in Fig. 24. Third, REG2 requires inverted TH12 gates with standby states A = ‘1’, B= ‘0’ or A = ‘0’, B = ‘1’, which is shown in Fig. 25. Finally, REG0 requires TH22 gates resettable to ‘0’ with standby states reset = ‘0’, A = ’0’ or A = ’1’, B = ‘1’, Z = ‘0’, which is shown in Fig. 26.

In summary, two techniques to handle indeterminate standby states are proposed. First, add extra gates to reduce indeterminate standby states as much as possible. Second, analyze the standby states of threshold gates and replace the minimal number of transistors with high-Vt

transistors to eliminate leakage paths in any standby state, and utilize low-Vt transistors for the rest.

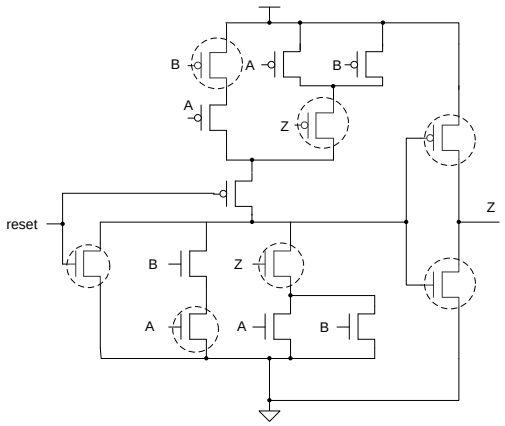


Figure 24. Enhanced BWMTNCL applied to TH22 resettable to ‘1’ gate with 2 standby states: reset = ‘0’, A = ‘0’, B = ‘1’, Z = ‘0’ or Z = ‘1.

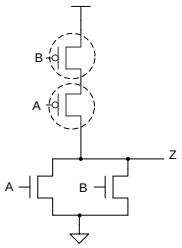


Figure 25. Enhanced BWMTNCL applied to inverted TH12 gate with 2 standby states: A = ‘0’, B = ‘1’ or A = ‘1’ or B = ‘0’.

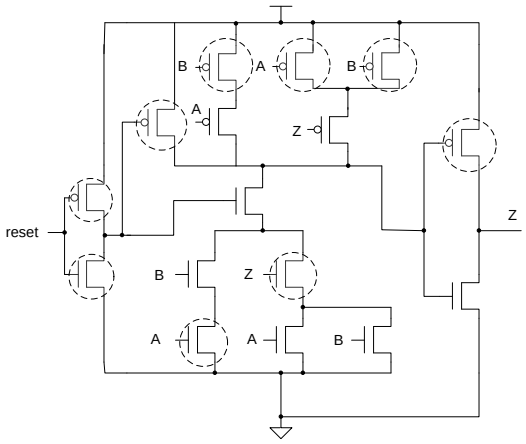


Figure 26. Enhanced BWMTNCL applied to TH22 resettable to ‘0’ gate with 2 standby states: reset = ‘0’, A = ‘0’ or A = ‘1’, B = ‘1’, Z = ‘0’.

7.3 Simulation Results

To compare the proposed BWMTNCL with indeterminate state reduction design to the regular design without MTCMOS, the 32+16×16 unsigned MAC with feedback loop in Fig. 23 was implemented with enhanced BWMTNCL and the MAC with feedback loop in Fig. 21 was implemented with regular-Vt transistors using the 1.2V IBM 8RF-LM 130nm CMOS process, and was simulated at transistor level using Cadence’s UltraSim simulator running a VerilogA controller in mixed-signal mode. Note that all transistors are minimum sized except for the buffers used for high fanout signals.

The first two rows of Table III show the results, in which T_{DD} is the average DATA plus NULL processing time, which is comparable to the synchronous clock period. T_{DD} and Energy/Operation are calculated while the circuit is operating at its maximum speed, while Leakage Power is calculated using Cadence Spectre DC analysis after the pipeline is flushed with all NULL inputs. The results show that the BWMTNCL with indeterminate state reduction

design has 36× standby power reduction over the regular design, with 20% speed penalty and 0.02% area overhead.

The 2 MACs were also implemented with all high-Vt and all low-Vt minimum sized transistors, respectively, as all high-Vt implementation will give the lower bound on standby power and all low-Vt implementation will give the lower bound on T_{DD}. The results are shown from row 3 to row 6 of Table III, which prove that adding extra gates to reduce indeterminate states does not increase T_{DD}. It also shows that the proposed design provides the leakage power advantages of all high-Vt implementations with a reasonable speed penalty compared to all low-Vt implementations.

Table III. Simulation results.

	Transistor#	T_{DD} (ns)	Energy/ Operation (pJ)	Standby Power (nW)
Enhanced BWMTNCL with indeterminate states reduction	118176	6.2	36	159.312
Regular-Vt without indeterminate states reduction	118158	4.9	37.5	5761.8
All high-Vt with indeterminate states reduction	118176	9.3	35.3	129.096
All high-Vt without indeterminate states reduction	118158	9.3	35.3	130.56
All low-Vt with indeterminate states reduction	118176	4	34.5	13123.8
All low-Vt without indeterminate states reduction	118158	4	34.5	13255.8

8. SEU/SEL HARDENED NCL CIRCUITS

8.1 Design Methodology

The original NCL architecture is shown in Fig. 3. As shown in Fig. 27, the following 4 steps are required to make it SEU resistant:

1. Double the original circuit.

2. Replace TH22n gates in NCL register (Fig. 12) with TH33n gates and accept acknowledge signals from both copies.

3. Insert TH22 gates at the output of NCL registers and accept register outputs from both copies.

4. Move the inverted TH12 gate in the original NCL register (Fig. 12) and Completion Logic from the outputs of NCL registers to the outputs of the added TH22 gates.

As explained in Chapter 1.3, a circuit is SEL resistant if it can be divided into several groups with separate virtual power supplies, and has the property that if the power supply of one group is cut off and resumed later, the information stored in this group can be recovered from other intact groups without causing wrong output or deadlock. The fewer groups the circuit is divided into, the smaller the area overhead, as less power gating transistors and current comparators are required.

The inserted TH22 gates, the following Completion Logic, Combinational Circuit, and NCL register can be put into one group, as shaded in Fig. 27. Chapter 8.2 proves that this division scheme is SEL resistant and automatically proves that it is SEU resistant also, because SEL fault model assumes errors in a whole group while SEU fault model only assumes error in one gate.

As mentioned in Chapter 1.3, the power management unit designed in [18-19] can be used along with the proposed architecture and grouping scheme to achieve a SEL resistant design.

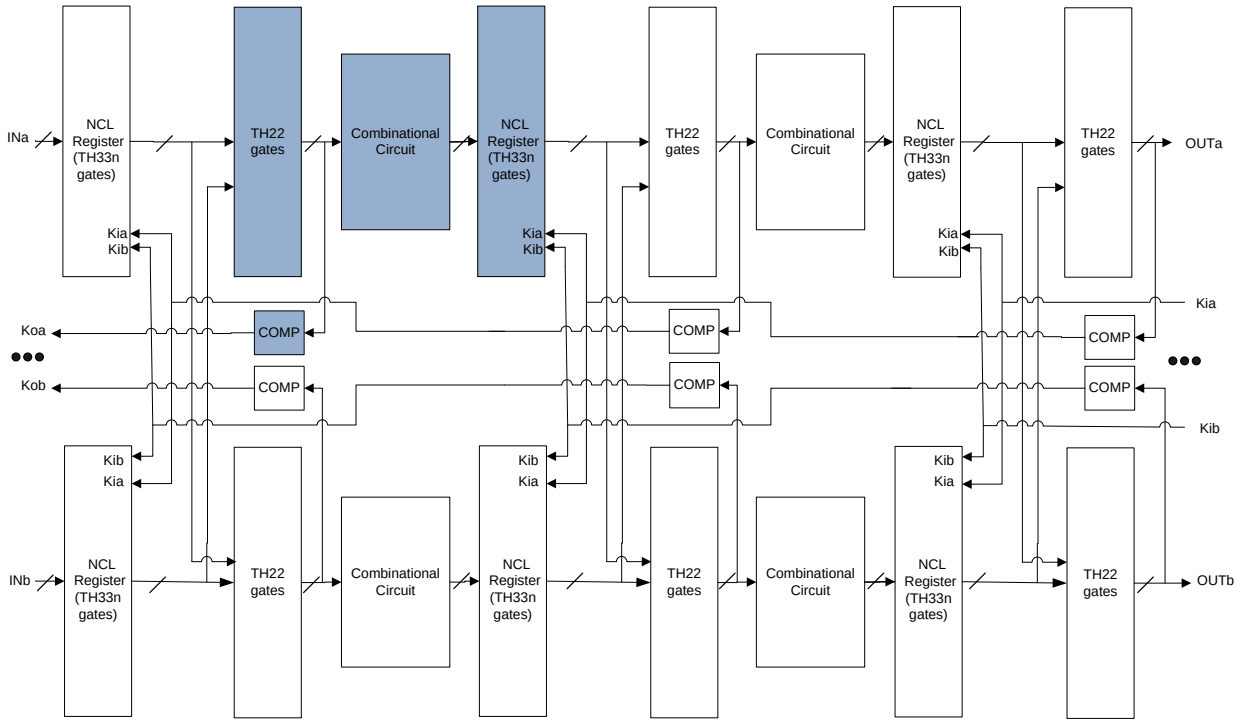


Fig.27 SEL/SEU resistant NCL circuits.

8.2 SEL/SEU Resistance Proof

As explained in Chapter 1.3, when SEL happens, the power management unit will cut off the power supply of one group and resume it later. Then the outputs of that group will be unknown (either logic 1 or 0) and will be re-evaluated according to the outputs of other intact groups. In the following analysis, the unknown state is referred as X. DATAX refers to the state in which a rail that should be 0 changes to X because of SEL. For example, if the correct DATA state of a dual-rail signal is (0, 1), then DATAX of this signal means it becomes (X, 1).

Subscripts are used to show timing in the following analysis. For example, t_{1a} and t_{1b} are both earlier than t_2, but for t_{1a} and t_{1b} which one is earlier than the other does not matter.

First, we prove that SEL will not cause deadlock. In the four-phase handshaking protocol used by NCL, registers should only allow a DATA/NULL wavefront to pass through when the previous NULL/DATA wavefront has arrived at the outputs of subsequent registers. Otherwise, deadlock will happen.

As shown in Fig. 28, at t_0 a register let NULL pass through when both *Kia* and *Kib* become RFN (logic 0). SEL can only change the acknowledge signal of one copy but not both. If *Kib* is not affected by SEL, it can be deduced that at t_{-1} the outputs of copy$_b$'s inserted TH22 gates became DATA. It can be further deduced that at t_{-2} the outputs of the two copies of the subsequent register both became DATA, which corresponds to the handshaking protocol.

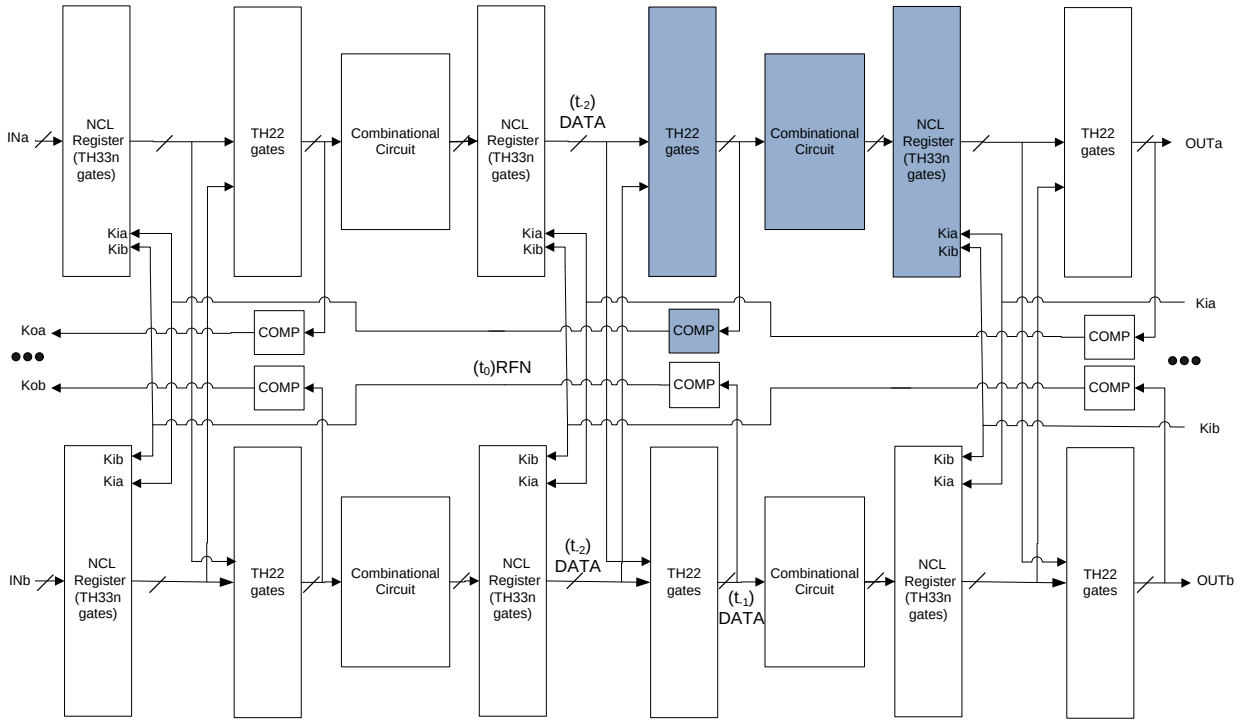


Fig. 28 SEL will not cause deadlock.

Similarly, it can be proved that deadlock will not happen during the propagation of the DATA wavefront by exchanging NULL with DATA, RFN with RFD, and logic 0 with logic 1.

Second, we prove that SEL will not cause an incorrect output by examining the worse case situation when the pipeline stores the largest number of DATA/NULL tokens, meaning that the outputs of registers alternate between DATA and NULL. We examine the following 2 scenarios respectively: when the register in the group affected by SEL stored NULL and when it stored DATA.

Fig. 29 and Fig. 30 show the scenario when the register in the group affected by SEL stored NULL. The power supply of that group was resumed at t_0 so that all states of that group became X. X cannot pass through the inserted TH22 gates if the output of the register of the other copy is still NULL. At t_1, the output of the TH22 gates in the affected group changes to DATAX. At t_{2a}, the output of the Combinational Logic in the affected group changes to DATAX, because of the monotonic property of NCL function [15]. At t_{2b}, the output of the Completion Logic in the affected group changes to RFN (logic 0), which corresponds to the handshaking protocol. It can be expected that when at t_3 DATA in $stage_i$ propagates to $stage_{i+1}$, DATAX will be filtered to be correct DATA by the inserted TH22 gates.

Fig. 31 shows the scenario when the register in the group affected by SEL stored DATA. The power supply of that group was resumed at t_0 so that all states of that group became X. X cannot pass through the inserted TH22 gates if the output of the register of the other copy is still correct DATA. At t_1, the output of the TH22 gates in the affected group changes to NULL. At t_{2a}, the output of the Combinational Logic in the affected group changes to NULL, because of the monotonic property of NCL function. At t_{2b}, the output of the Completion Logic in the affected group changes to RFD (logic 1), which corresponds to the handshaking protocol.

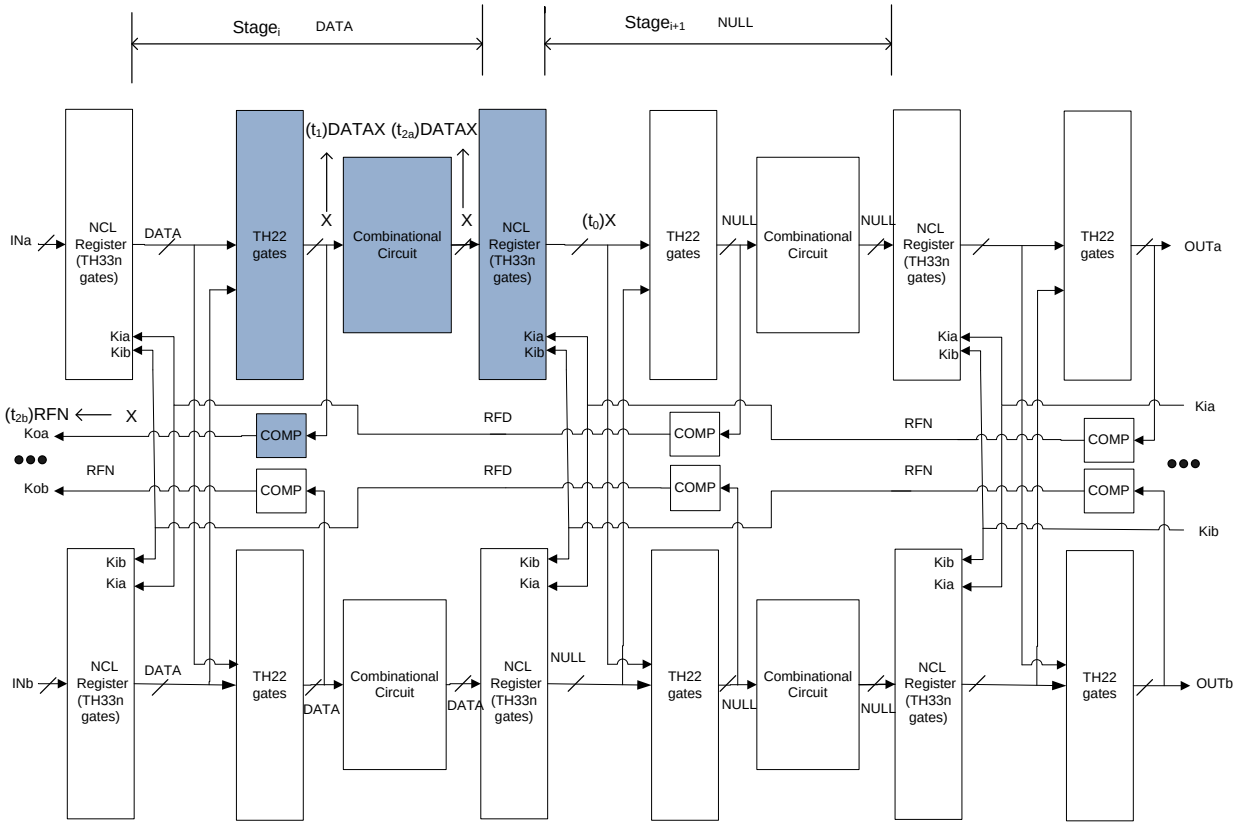


Fig. 29 SEL will not cause wrong output when the register in the group affected by SEL stored NULL.

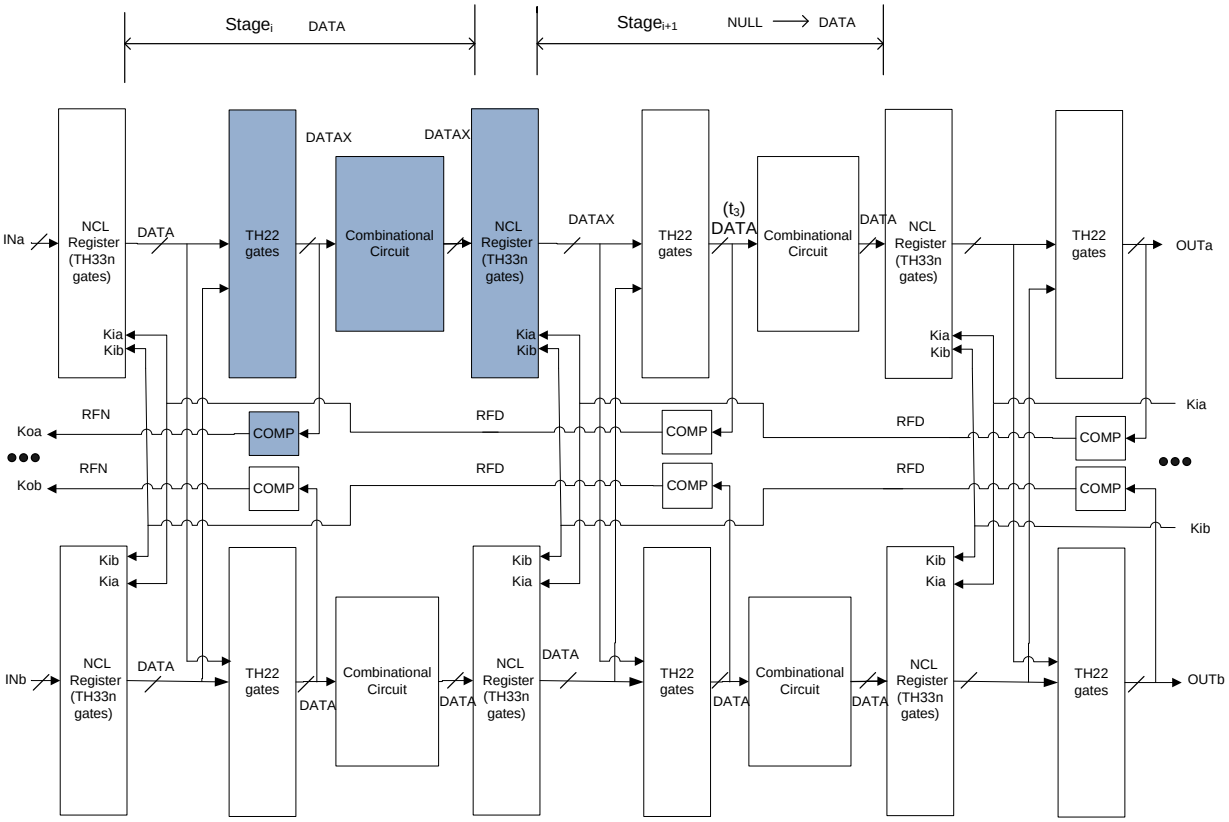


Fig. 30 SEL will not cause wrong output when the register in the group affected by SEL stored NULL (continued).

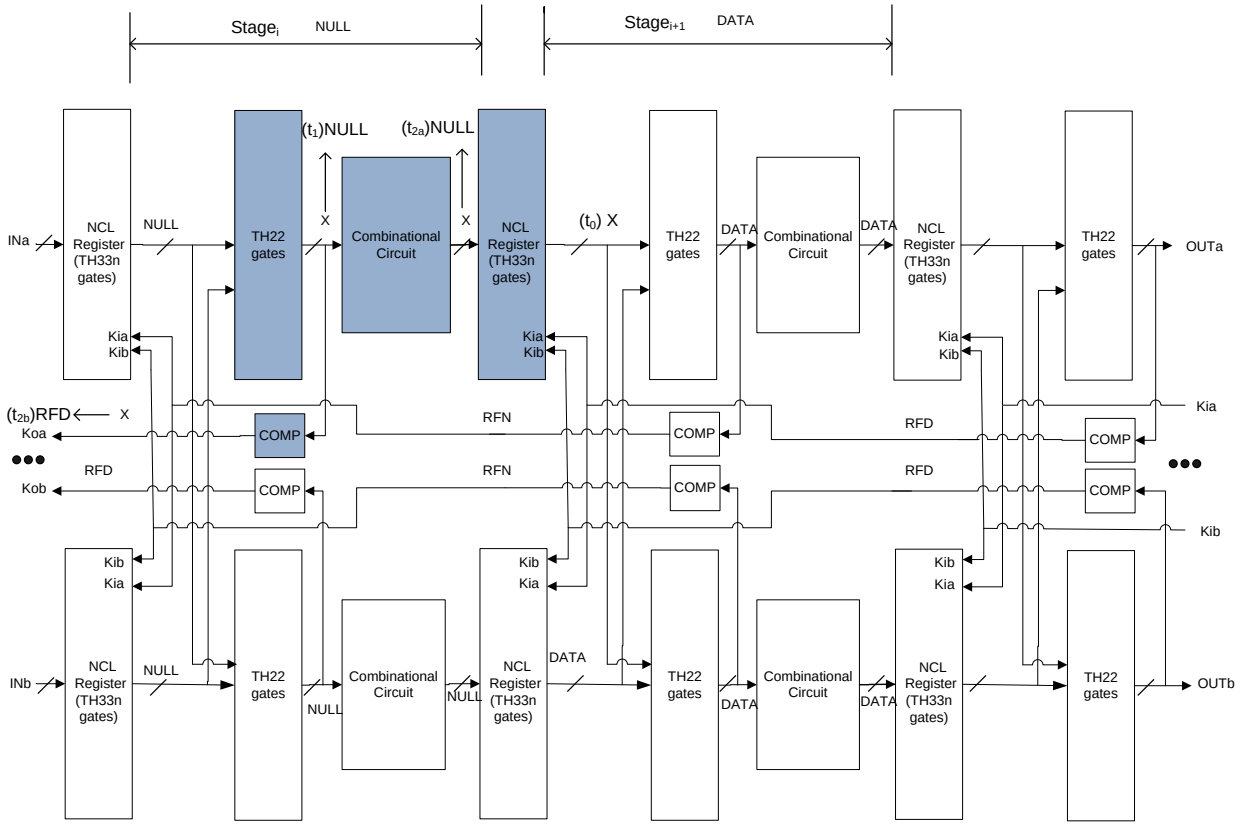


Fig. 31 SEL will not cause wrong output when the register in the group affected by SEL stored DATA.

8.3 Optimal 4-Group Division

The data flow graph of the group division scheme proposed in Chapter 8.1 is shown in Fig. 32 in which forward arrows represent data signals. It can be observed from the proof in Chapter 8.2 that the recovery from SEL of one group (e.g., $group_{ia}$) only depends on the intact states of the previous stage ($group_{i-1a}$, $group_{i-1b}$), next stage ($group_{i+1a}$, $group_{i+1b}$), and the other copy of the same stage ($group_{ib}$). Therefore, all groups of every other stage in one copy (e.g. ..., $group_{i-2a}$, $group_{ia}$, $group_{i+2a}$, ...) can be combined into one group to yield the optimal 4-group division.

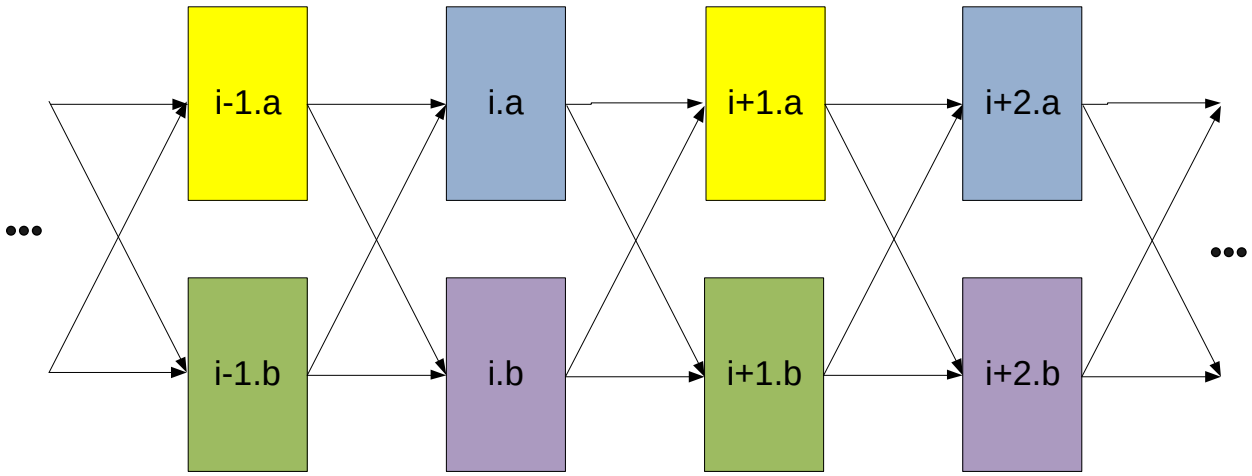


Fig. 32 Data flow graph of the group division scheme proposed in Chapter 8.1.

8.4 Simulation Results

A 3-stage full-word pipelined NCL multiplier [10] was transformed to a SEU/SEL resistant version with the 4-group optimal division scheme proposed in Chapter 8.3. The outputs of 2 copies must be observable to ensure correct interfacing when SEU/SEL happens. As a result, 54 pins are required. To reduce the pin number, as in Fig. 33, the combined *Ko* signal of the last register was made an output pin named ready, and only rail1 of the output dual rail signals were made output pins (e.g., pa[0] = Sa[0].rail1). This reduces the pin number to 40 without compromising performance.

A VerilogA controller was made to pump in data patterns exhaustively and check the output automatically. The interface protocol is: When *Koa* and *Kob* are both RFD/RFN, pump in DATA/NULL; When *readya* and *readyb* are both RFD/RFN and *pa[7:0]* is identical to *pb[7:0]*, change *Ki* to RFD/RFN. The test setup is drawn in Fig. 34.

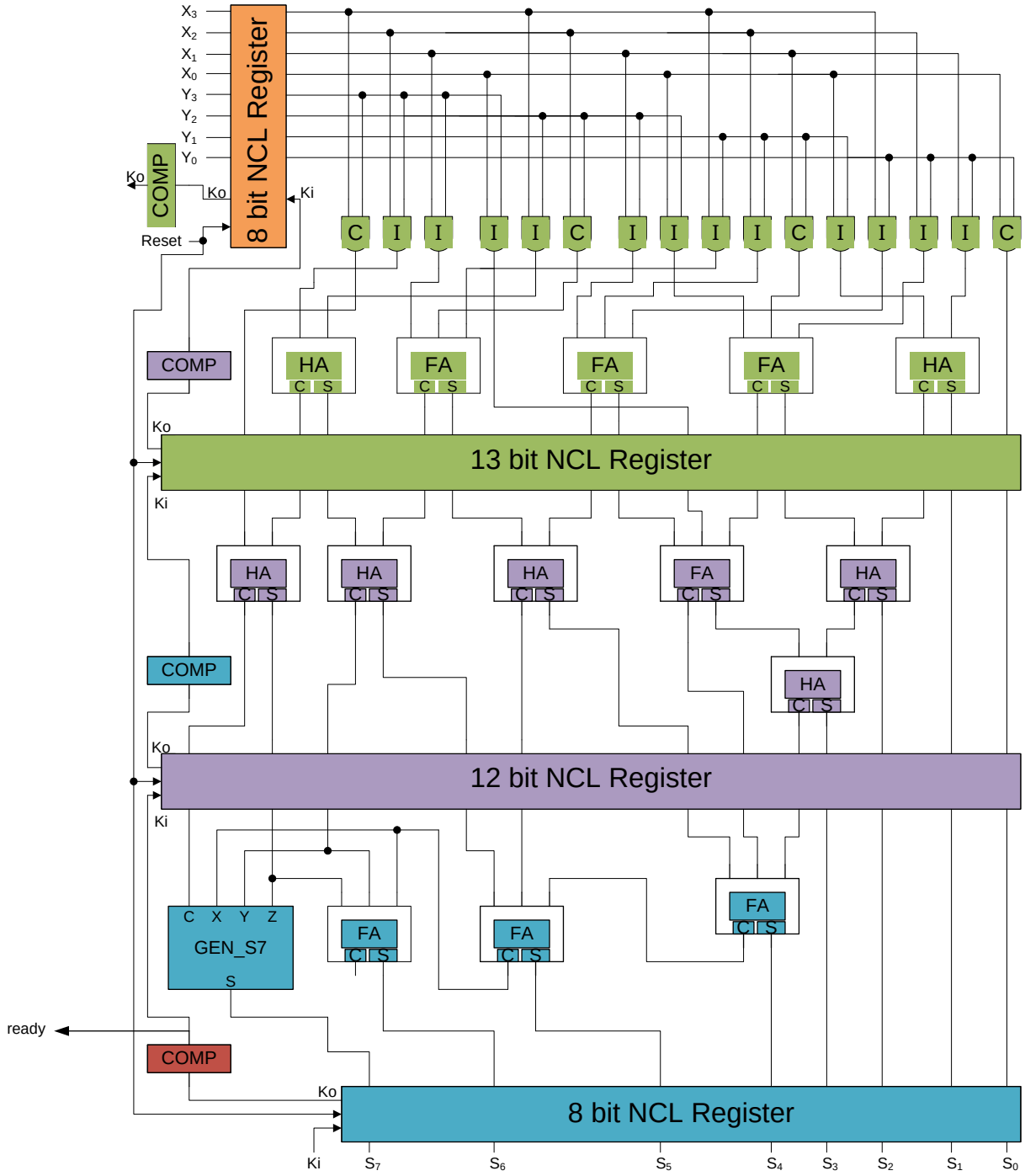


Fig. 33 3-stage NCL 4×4 multiplier with ready output.

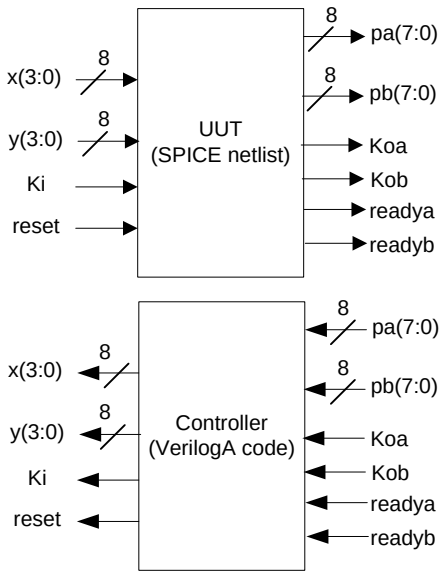


Fig. 34 test setup.

The designs were implemented with the IBM cmrf8sf 130nm process and simulated with Cadence Ultrasim simulator. SEL fault was injected by connecting the power supply of one group to ground for a short time periodically and 4 simulations were run for the 4 groups, respectively. SEL did not cause deadlock or wrong output in any of the 4 simulations. The waveform of one simulation is shown is Fig. 35. Signals from top to bottom are as follows: active high SEL fault injection signal, expected decimal output, active high signal if output is wrong, and observed decimal output.

The active high SEL fault injection signal is a pulse with 100 ns period and 10% duty cycle and is independent of input/output handshaking. When it becomes high, the power supply of the group under test will be forced to ground immediately by the VerilogA controller. It will keep high for 10 ns, which is long enough for all the internal nodes of the group under test to become logic 0. In a real power management unit, when SEL happens the power supply is

usually cut off for several microseconds to make sure SEL disappears. But for our fault injection purpose, it makes no difference to make the pulse high for that long.

When the pulse becomes low, the power supply of the group under test will be forced to V_{DD} immediately by the VerilogA controller. In the group under test, the gates driven by the outputs of intact groups will start re-evaluation. Other gates will first change to either logic 1 or logic 0 by the SPICE simulator and finally be re-evaluated in sequence.

The original multiplier and SEU/SEL hardened multiplier without power management unit were also simulated for speed, area, and power consumption. The results are shown in Table IV. Compared to the original version, the SEU/SEL hardened version has 1.31× speed overhead, 2.74× area overhead, and 2.79× energy/operation overhead.

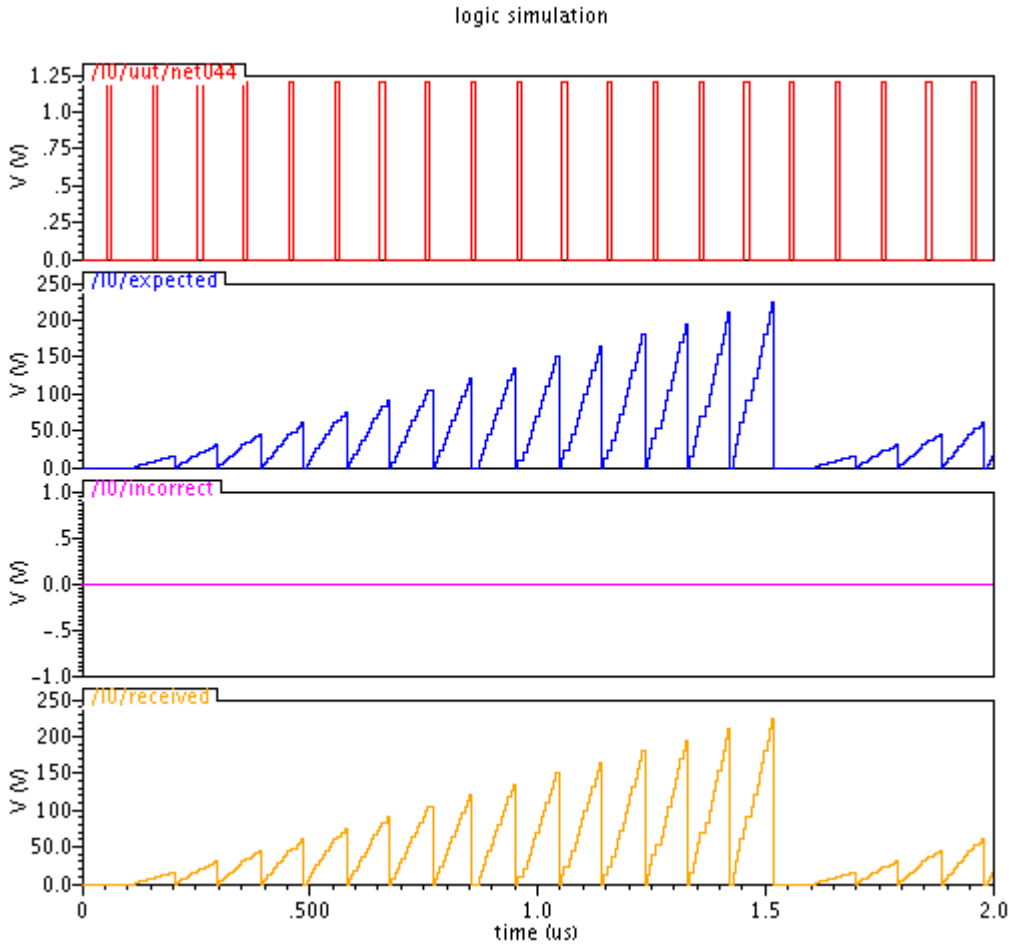


Fig. 35 Simulation with SEL fault injection.

Table IV. Simulation results

	Transistor#	T_{DD}(ns)	Energy/ Operation(pJ)
Original NCL	1695	7.2	1.05
SEU/SEL Hardened NCL	4646	9.4	2.93

9. ACCURATE THROUGHPUT DERIVATION OF PIPELINED NCL CIRCUITS

9.1 Throughput Derivation of Non-Pipelined NCL Systems

Non-pipelined NCL systems contain two DI registers, one at both the input and at the output. In NCL systems, the DATA-to-DATA cycle time (T_{DD}) [10] has an analogous role to the clock period in a synchronous system and is the reciprocal of throughput. To derive the throughput of NCL systems, the external interface is assumed to respond to *Ko* signals and circuit outputs immediately. Four system parameters are defined in the handshaking sequence, as shown in Figs. 36-39:

TD: DATA propagation time (from when all input register inputs become DATA and *Ki* signals become RFD, to when all inputs of the output register become DATA).

TRFN: NULL request time (from when all output register inputs become DATA and *Ki* signals become RFD, to when all *Ki* inputs of the input register become RFN).

TN: NULL propagation time (from when all input register inputs become NULL and *Ki* signals become RFN, to when all inputs of the output register become NULL).

TRFD: DATA request time (from when all output register inputs become NULL and *Ki* signals become RFN, to when all *Ki* inputs of the input register become RFD).

In non-pipelined NCL systems, the end of the DATA request time of the current cycle is the beginning of the DATA propagation time of the next cycle. Therefore, T_{DD} = TD + TRFN + TN + TRFD and throughput = 1 / TDD. The estimation in [10] is inaccurate as it ignores register delays in the definition of the four system parameters, since it was used to determine where to add pipeline stages, not to precisely calculate timing.

If a gate delay is used as the minimal delay unit, then TD = TN, TRFN = TRFD, and the formula can be simplified. The schematic of a 1-bit DI register is shown in Fig. 12. The gate delay from its input to output is 1. The gate delay from its input to *Ko* is 1.5, because the inverted TH12 gate is equivalent to a NOR gate, which doesn't have an output inverter, so has 0.5 gate delay compared to other types of threshold gates. As shown in Fig. 40, if the Combinational Logic has TCOMB gate delays and the Completion Logic has TCOMP gate delays, then TD = TN = 1 + TCOMB, TRFN = TRFD = 1.5 + TCOMP, and T_{DD} = 2 * (2.5 + TCOMB + TCOMP). The formula in [10] estimates T_{DD} as 2 * (TCOMB + TCOMP), which is imprecise.

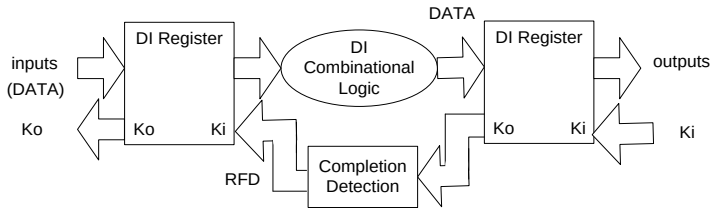


Fig.36 DATA propagation time.

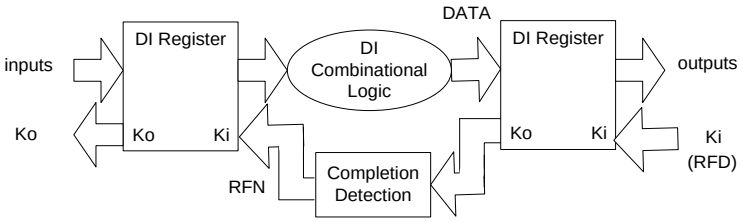


Fig.37 NULL request time.

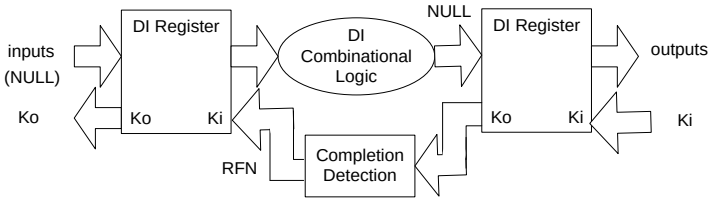


Fig.38 NULL propagation time.

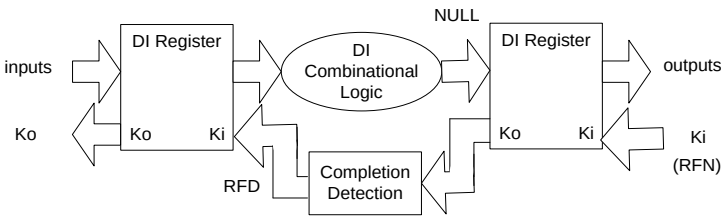


Fig.39 DATA request time.

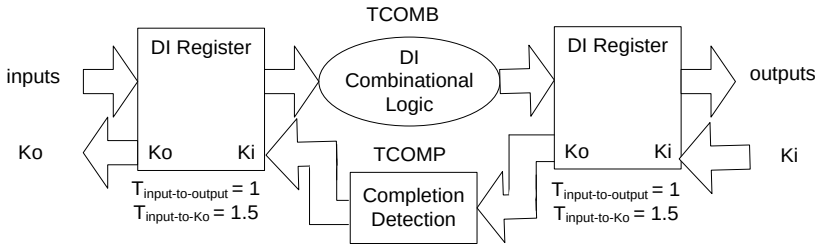


Fig.40 System parameters of non-pipelined NCL systems with gate delay.

9.2 Throughput Derivation of Pipelined NCL Systems

The throughput of pipelined NCL systems is determined by the stage with the largest T_{DD}. In this paper, we define a stage as shown in Fig. 41, where registers are shared by adjacent stages. The T_{DDi} of $stage_i$ has 3 possibilities [10]. As shown in Fig. 42, if $TN_i + TRFD_i < TRFD_{i-1} + TD_{i-1}$, then when all of the *Ki* signals of the input register of $stage_i$ become RFD, it has to wait for $(TRFD_{i-1} + TD_{i-1} - TN_i - TRFD_i)$ before all of the inputs of the input register of $stage_i$ become DATA. Therefore, $T_{DDi} = TD_i + TN_i + TRFD_i + TRFN_i + (TRFD_{i-1} + TD_{i-1} - TN_i - TRFD_i) = TRFD_{i-1} + TD_{i-1} + TD_i + TRFN_i$. Similarly, as shown in Fig. 43, if $TD_i + TRFN_i < TRFN_{i-1} + TN_{i-1}$, then when all of the *Ki* signals of the input register of $stage_i$ become RFN, it has to wait for $(TRFN_{i-1} + TN_{i-1} - TD_i - TRFN_i)$ before all of the inputs of the input register of $stage_i$ become NULL. Therefore, $T_{DDi} = TD_i + TN_i + TRFD_i + TRFN_i + (TRFN_{i-1} + TN_{i-1} - TD_i - TRFN_i) = TRFN_{i-1} + TN_{i-1} + TN_i + TRFD_i$. If none of the above two conditions are true, $T_{DDi} = TD_i + TN_i + TRFD_i + TRFN_i$. In summary,

T_{DD} = max ($TRFD_{i-1}$ + TD_{i-1} + TD_i + $TRFN_i$, $TRFN_{i-1}$ + TN_{i-1} + TN_i + $TRFD_i$, TD_i + TN_i + $TRFD_i$ + $TRFN_i$) for *i* in all of the stages.

If a gate delay is used as the minimal delay unit, then TD = TN, TRFN = TRFD, and the formula can be simplified as T_{DD} = max (2 * (2.5 + $TCOMB_i$ + $TCOMP_i$)) for *i* in all of the stages. The formula in [10] estimates T_{DD} as 2 * ($TCOMB_i$ + $TCOMP_i$), which is imprecise, especially when applied to finely pipelined NCL systems.

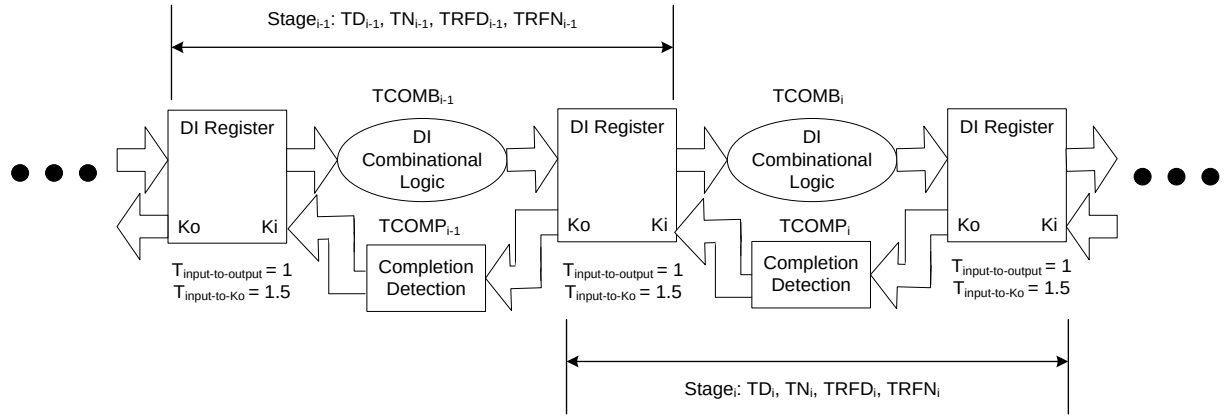


Fig.41 System parameters of pipelined NCL systems with gate delay.

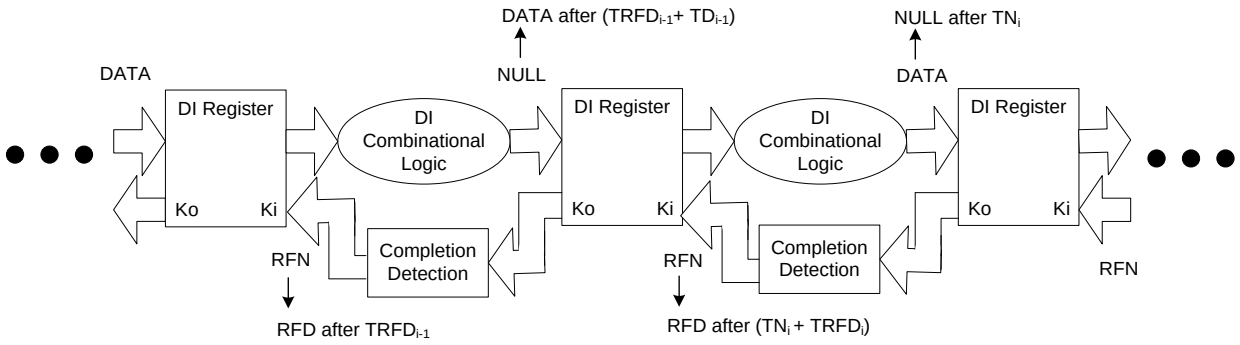


Fig.42 TDD_i derivation when TN_i + $TRFD_i$ < $TRFD_{i-1}$ + TD_{i-1}.

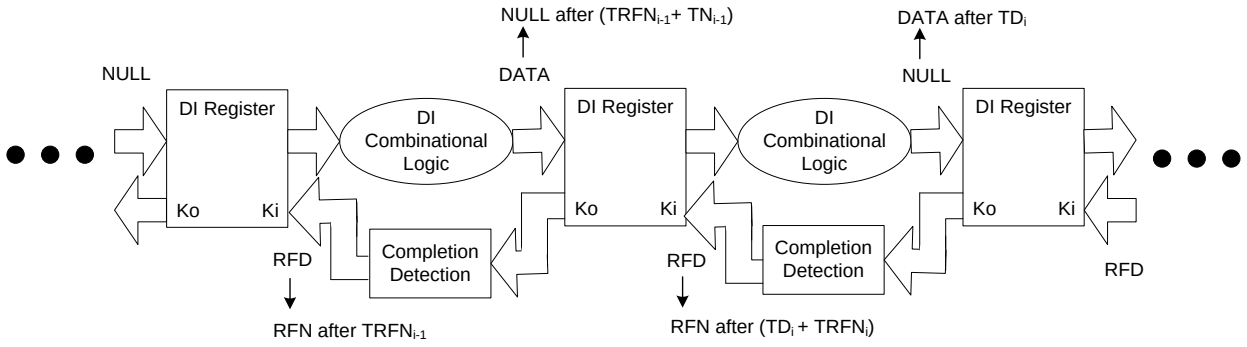


Fig.43 TDD_i derivation when $TD_i + TRFN_i < TRFN_{i-1} + TN_{i-1}$.

9.3 Simulation Results

To verify the formula proposed in Chapter 9.2, a 3-stage full-word pipelined 4×4 NCL multiplier [10] was implemented with the IBM cmos10lpe 65nm process at the transistor level and simulated with Cadence Spectre simulator. A VerilogA controller was designed to characterize the four system parameters of each stage and to measure T_{DD} with different input patterns. As shown in Table V, the measured T_{DD} was compared with the predicted T_{DD} calculated from the formula and they matched.

Table V. Simulation results

Characterized system parameters and derivations (ps)	Input patterns		
	X = 15, Y = 15	X = 0, Y = 0	X = 7, Y = 8
TD1	267	279	349
TN1	393	500	658
TRFD1	346	358	357
TRFN1	413	412	413
TD2	261	249	320
TN2	404	420	424
TRFD2	336	336	345
TRFN2	392	391	363
TD3	251	249	226
TN3	370	530	246
TRFD3	316	310	309
TRFN3	303	303	301
TD1 + TN1 + TRFD1 + TRFN1	1420	1551	1778
TRFD1 + TD1 + TD2 + TRFN2	1267	1279	1390
TRFN1 + TN1 + TN2 + TRFD2	1548	1669	1842
TD2 + TN2 + TRFD2 + TRFN2	1395	1397	1454
TRFD2 + TD2 + TD3 + TRFN3	1153	1138	1193
TRFN2 + TN2 + TN3 + TRFD3	1484	1652	1344
TD3 + TN3 + TRFD3 + TRFN3	1242	1393	1083
Predicted T_{DD}	1548	1669	1842
Measured T_{DD}	1548	1669	1842

9.4 Static Timing Analysis of Pipelined NCL Systems

The NCL gate library used in Chapter 9.3 was characterized with Synopsys NCX and Cadence Spectre simulator. Synopsys Primetime was used to calculate the worst case system parameters of the design. The results are shown in Table VI, and the worst case T_{DD} was calculated using the proposed formula.

Table VI. Static timing analysis results

Calculated worse case system parameters and derivations (ps)	
TD1	356
TN1	704
TRFD1	423
TRFN1	498
TD2	342
TN2	704
TRFD2	404
TRFN2	454
TD3	436
TN3	797
TRFD3	371
TRFN3	397
TD1 + TN1 + TRFD1 + TRFN1	1981
TRFD1 + TD1 + TD2 + TRFN2	1575
TRFN1 + TN1 + TN2 + TRFD2	2310
TD2 + TN2 + TRFD2 + TRFN2	1904
TRFD2 + TD2 + TD3 + TRFN3	1579
TRFN2 + TN2 + TN3 + TRFD3	2326
TD3 + TN3 + TRFD3 + TRFN3	2001
Predicted T_{DD}	2326

The design was also simulated with Modelsim using the characterized library in VITAL format with SDF annotation from Primetime. As shown in Table VII, the relative error caused by the characterized library is below 5%.

Table VII. Comparison between SPICE and VITAL simulations

Input patterns	TDD from SPICE simulation (ps)	TDD from VITAL simulation (ps)	Relative error (%)
X = 15, Y = 15	1548	1553	0.3
X = 0, Y = 0	1669	1598	4.4
X = 7, Y = 8	1842	1819	1.3

10. CONCLUSION

In this dissertation, a technique for utilizing bit-wise completion with MTNCL to produce a fast ultra-low power bit-wise pipelined asynchronous circuit design methodology was developed, and simulated at the transistor-level in Chapter 5. Significant enhancements to the original MTNCL concept were derived, and simulated at the transistor-level in Chapter 6. Bit-wise MTNCL was extended, and simulated at the transistor-level in Chapter 7 to handle indeterminate standby states. An architecture that allows NCL circuits to recover from an SEU or SEL fault without any data loss was developed, proved by deduction, and verified by Spice simulation with fault injection in Chapter 8. An accurate throughput derivation formula for pipelined NCL circuits was proposed, and verified by Spice simulation in Chapter 9.

REFERENCES

[1] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 30/8, August 1995, pp. 847-854.

[2] K. M. Fant, S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, 1996, pp. 261–273.

[3] A. D. Bailey, J. Di, S. C. Smith, and H. A. Mantooth, "Ultra-Low Power Delay-Insensitive Circuit Design," *IEEE Midwest Symposium on Circuits and Systems*, August 2008, pp. 503-506.

[4] A. D. Bailey, A. Al Zahrani, G. Fu, J. Di, and S. C. Smith, "Multi-Threshold Asynchronous Circuit Design for Ultra-Low Power," *Journal of Low Power Electronics*, Vol. 4/3, December 2008, pp. 337-348.

[5] A. Alzahrani, A. D. Bailey, G. Fu, and J. Di, "Glitch-Free Design for Multi-Threshold CMOS NCL Circuits," *2009 Great Lakes Symposium on VLSI*, May 2009.

[6] S. C. Smith and J. Di, *Designing Asynchronous Circuits using NULL Convention Logic (NCL)*, Synthesis Lectures on Digital Circuits and Systems, Morgan & Claypool Publishers (doi: 10.2200/S00202ED1V01Y200907DCS023), Vol. 4/1, July 2009.

[7] L. Zhou, S. C. Smith, and J. Di, "Bit-Wise MTNCL: An Ultra-Low Power Bit-Wise Pipelined Asynchronous Circuit Design Methodology," *IEEE Midwest Symposium on Circuits and Systems*, August 2010, pp. 217-220.

[8] M. Imai, K. Takada, T. Nanya, "Fine-grain leakage power reduction method for m-out-of-n encoded circuits using multi-threshold-voltage transistors," *IEEE Int. Symp. on Asynchronous Circuit and Systems*, May 2009, pp. 209 – 216.

[9] C. Ortega, J. Tse, and R. Manohar, "Static power reduction techniques for asynchronous circuits," *IEEE Int. Symp. on Asynchronous Circuit and Systems*, May 2010, pp. 52-61.

[10] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Elsevier's Integration, the VLSI Journal*, Vol. 30/2, October 2001, pp. 103-131.

[11] Wonjin Jang, and Alain J. Martin, "A Soft-error tolerant Asynchronous Microcontroller," *13th NASA Symposium on VLSI Design*, June 2007.

[12] Wonjin Jang, and Alain J. Martin, "Soft-error Robustness in QDI Circuits," *Workshop on System Effects of Logical Soft Errors - SELSE 1*, 05-06 April 2005.

[13] Wonjin Jang and Alain J. Martin, "SEU-tolerant QDI Circuits," *Proc. 11th IEEE International Symposium on Asynchronous Systems & Circuits (ASYNC)*, 14-16 March 2005.

[14] Wonjin Jang, and Alain J. Martin, "Soft-error tolerant Asynchronous FPGA," *Fast Abstract, Dependable System and Network 2005*, June 2005.

[15] Weidong Kuang, P. Zhao, J.S.Yuan, and R. DeMara, "Design of asynchronous circuits for high soft error tolerance in deep submicron CMOS circuits," *IEEE Trans. on VLSI systems*, vol.18, no.3, March 2010, pp.410-422.

[16] W.Kuang, Casto Manuel Ibarra, Peiyi Zhao, "Soft Error Hardening for Asynchronous Circuits," *the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, Rome, Italy, Sept., 2007.

[17] Weidong Kuang, Enjun Xiao, Casto Manuel Ibarra, Peiyi Zhao, "Design Asynchronous Circuits for Soft Error Tolerance," *IEEE International Conf. on Integrated Circuit Design and Technology*, Austin, Texas, June, 2007.

[18] Michael Nicolaidis, Kholdoun Torki, Federico Natali, Kader Belhaddad, Dan Alexandrescu, "A Low-Cost Single-Event Latchup Mitigation Scheme," *On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International*, July 2006

[19] Michael Nicolaidis, Kholdoun Torki, Federico Natali, Kader Belhaddad and Dan Alexandrescu, "Implementation and Validation of a Low-Cost Single-Event Latchup Mitigation Scheme," *2009 IEEE Workshop on Silicon Errors in Logic - System Effects*, March 2006

[20] Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)* , 1998, pp. 61-65.

[21] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, 1963, pp. 289-297.

[22] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits, Integration," *the VLSI Journal*, Vol. 37/3, August 2004, pp. 135-165.

[23] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, 1980, pp. 218-262.

[24] A. Kondratyev, L. Neukom, O. Roig, A. Taubin, and K. Fant**,** "Checking Delay-Insensitivity: 10^4 Gates and Beyond," *Eighth International Symposium on Asynchronous Circuits and Systems*, April 2002, pp. 149-157.

[25] A.J. Martin, "Programming in VLSI: From Communicating Processes to Delay-Insensitive Circuits," in *Developments in Concurrency and Communication*, UT Year of Programming Institute on Concurrent Programming, Addison-Wesley, 1990, pp. 1-64.

[26] K. Van Berkel, "Beware the Isochronic Fork," *Integration, the VLSI Journal*, Vol. 13/2, 1992, pp. 103-128.

[27] J. T. Kao and A. P. Chandrakasan, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE Journal of Solid-State Circuits*, Vol. 35/7, July 2000, pp. 1009-1018.

[28] P. Lakshmikanthan, K. Sahni, and A. Nunez, "Design of Ultra-Low Power Combinational Standard Library Cells Using a Novel Leakage Reduction Methodology," *IEEE International SoC Conference*, 2006.

[29] S. C. Smith, "Speedup of Self-Timed Digital Systems Using Early Completion," *IEEE Computer Society Annual Symposium on VLSI*, April 2002, pp. 107-113.

[30] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford University Press, New York, 2000.

[31] L. Zhou and S. C. Smith, "Speedup of a Large Word-Width High-Speed Asynchronous Multiply and Accumulate Unit," *IEEE Midwest Symposium on Circuits and Systems*, pp. 499-502, August 2009.

[32] S. B. Furber and P. Day, "Four-phase micropipeline latch control circuits," *IEEE Trans VLSI Syst.*, vol. 4, pp. 247–253, June 1996.