5-2018

Comparison of Data Transfer Alternatives in Asynchronous Circuits

Mark Howard

Comparison of Data Transfer Alternatives in Asynchronous Circuits

A thesis submitted in partial fulfillment
of the requirements for the award of
Honors for Bachelor of Science in Computer Engineering

by

Mark Howard
University of Arkansas

May 2018
University of Arkansas

This thesis is approved for recommendation to the Honors College.

_____
Jia Di, Ph.D.
Honors Advisor and Committee Member

_____
Patrick Parkerson, Ph.D.
Committee Member

_____
David Andrews, Ph.D.
Committee Member

Abstract

Digital integrated circuits (ICs) have become progressively complex in their functionality. This has sped up the demand for asynchronous architectures, which operate without any clocking scheme, considering new challenges in the timing of synchronous systems. Asynchronous ICs have less stringent environmental constraints and are capable of maintaining reliable operation in extreme environments, while also enjoying potential benefits such as low power consumption, high modularity, and improved performance. However, when the traditional bus architecture of synchronous systems is applied to asynchronous designs, handshaking protocols required for asynchronous circuit operation result in significantly increased power consumption, offsetting the low power benefit of asynchronous designs. In this thesis, NULL Convention Logic is used to implement two data transfer alternatives to the bus, and their performance is compared to that of the prevailing bus architecture. According to the results, both of these proposed architectures demonstrate power-saving qualities while sacrificing area, indicating potential utilization in power-constrained applications where speed is not a prioritized design constraint, as in Internet of Things (IoT) devices.

Acknowledgements

This research is supported by both NSF Award ECCS-1607285 and a University of Arkansas Honors College Research Grant. Additionally, I would like to thank Dr. Jia Di for his guidance in this research project and my fellow lab colleagues that assisted me in the completion of this thesis. To my family and friends, I cannot thank you all enough for your support through my undergraduate career, despite the obstacles that were presented along the way.

Table of Contents

List of Tables

List of Figures

I. Introduction

A. Problem Statement

Increasingly complex functionality of digital integrated circuits, issues with clock skews, power consumption, and variabilities in process, voltage, and temperature have arisen out of tightening design constraints and have led to an increased demand in circuits that are designed with asynchronous logic. Asynchronous ICs are capable of dependable operation in less than ideal environments and present opportunities for power consumption reduction, thanks to the handshaking protocols that are implemented in lieu of the typical synchronous clock architecture [1]. Because of the natural design differences between synchronous and asynchronous systems, some design commonalities for synchronous systems have qualities that hinder performance when utilized in asynchronous systems, such as the typical bus architecture for intercomponent data communication. Bus data transfer architecture is applied regularly in synchronous systems as a means for efficient data transfers, but this method is costly in terms of power consumption when implemented in asynchronous systems that utilize multi-rail encodings. This negative characteristic is the result of a network of pull-down resistors that are attached to bus wires to allow propagation of NULL or spacer cycles between data transfers.

B. Thesis Statement

In response to the issues presented above, which arise from applications of synchronous design methodologies in asynchronous systems, this thesis presents and implements two new data transfer architectures as alternative communication methods for asynchronous systems designed with NULL Convention Logic (NCL). These architectures are all constructed in the IBM 130nm CMOS semiconductor process, and while there is additional overhead in terms of performance,

simulation results for the alternative architectures demonstrate serious improvements in power consumption when compared to the standard bus architecture.

C. Thesis Organization

Chapter 2 elaborates on background information pertinent to the understanding of key components and characteristics of asynchronous designs: NULL Convention Logic and specifically multi-rail encodings, hysteresis, and threshold gates. Chapter 3 explains the proposed architectures, MUX bus and STAR, and the drawbacks of the prevailing bus architecture. Chapter 4 discusses the results of simulations of the three aforementioned architectures and indicates situations where these specific architectures could capitalize on their strengths. Chapter 5 summarizes the extrapolations that can be made based off of the simulation results and places responsibility on chip designers to weigh in on characteristics of each architecture when deciding on a data communication method.

II. Background

A. NULL Convention Logic (NCL)

Distinct from traditional synchronous systems that utilize a global clocking architecture to implement basic circuit operation, systems that are designed asynchronously make use of handshaking protocols to accomplish data communication, data requests, and normal circuit operation. Implementation of normal circuit operation and communication without the overhead clocking architecture present in synchronous systems provides a number of advantages to circuits with asynchronous design conventions. These benefits range from more flexible timing constraints, increased energy efficiency, improved modularity, and average-case performance.

Multi-Rail Encoding

NULL Convention Logic (NCL) is a quasi-delay-insensitive (QDI) design methodology for asynchronous circuits. Where synchronous designs encode a single bit of data with one wire, NCL implements multi-rail encodings, meaning multiple wires are used to represent a single bit of data. The most common encoding method in NCL is dual-rail encoding, which uses two wires to represent the equivalent logic '1' and logic '0' Boolean values of synchronous systems [2]. These individual wires are referenced as either *rail0* or *rail1* of an NCL signal, which represent Boolean logic '0' and logic '1,' respectively. The value of a single rail is manipulated by referencing individual rails of an NCL signal. For example, *signal(i).rail1* references *rail1* of bit *i* in an NCL dual-rail vector named *signal*. Additionally, rails of an NCL signal are mutually exclusive, meaning only one of the two rails can be asserted at a time, and this allows for the representation of 3 valid states and 1 invalid state in dual-rail logic as described in Table 1. Asserting *rail1* of an NCL signal corresponds to a DATA1 output, and asserting *rail0* of an NCL signal corresponds to DATA0. Asserting neither of the rails results in a NULL output. Since these rails are mutually exclusive, asserting both rails equates to an INVALID state that should not occur during normal circuit operation of an asynchronous system designed with dual-rail NCL [3].

Table 1. Dual-Rail Encoding

	NULL	**DATA0**	**DATA1**	**INVALID**
rail0	**0**	**1**	**0**	**1**
rail1	**0**	**0**	**1**	**1**

NCL Gates

NCL can be implemented structurally in the VHDL hardware description language by using 27 fundamental NCL logic gates. These gates are called threshold gates, and the Boolean functionality that each gate implements is depicted in Table 2. These gates are named for the method in which the gate output is evaluated, implying that a certain amount of a gate's inputs must be asserted before the gate output will transition from logic '0' to logic '1.' These threshold gates range from two to four inputs, and each gate executes a unique Boolean operation such that they comprise all combinations of four or fewer inputs [4]. Each threshold gate uses a naming format that simultaneously identifies the number of inputs, the threshold value, and any input weights for that specific gate. The format is THmn, where the threshold value, *m*, ranges from 1 to *n*, *n* being the number of single-bit inputs. Therefore, these gates do not necessarily accept a full dual-rail signal as an input, but rather, a gate can accept *rail1*, *rail0*, or both rails of an NCL dual-rail signal. As mentioned previously, inputs to threshold gates may be weighted differently, resulting in a naming change that reflects the alteration. When a threshold gate has weighted inputs, a *w* follows the *n* value in the gate nomenclature, and it is succeeded by the weight values. These values identify the appropriate weight for each input, beginning at input A. For example, a TH34w32 gate will operate similarly to a TH34 gate with the caveat that input *A* will have a weight of 3, and input *B* will have a weight of 2. Now, instead of requiring 3 of the 4 inputs to be asserted before output evaluation, as in the TH34 gate, the *m* value can be met with any combination of the weighted or non-weighted inputs. Asserting input *A* for a TH34w32 gate will assert the output since the weight of input *A* matches the threshold value of the gate. Likewise, assertion of inputs *B* and *C* would also cause output evaluation. It is important to note that surpassing the threshold value does not adversely affect the functionality of a threshold gate.

Table 2. Fundamental NCL Gates

Gate	Boolean Function
TH12	A+B
TH22	AB
TH13	A+B+C
TH23	AB+AC+BC
TH33	ABC
TH23w2	A+BC
TH33w2	AB+AC
TH14	A+B+C+D
TH24	AB+AC+AD+BC+BD+CD
TH34	ABC+ABD+ACD+BCD
TH44	ABCD
TH24w2	A+BC+BD+CD
TH34w2	AB+AC+AD+BCD
TH44w2	ABC+ABD+ACD
TH34w3	A+BCD
TH44w3	AB+AC+AD
TH24w22	A+B+CD
TH34w22	AB+AC+AD+BC+BD
TH44w22	AB+ACD+BCD
TH54w22	ABC+ABD
TH34w32	A+BC+BD
TH54w32	AB+ACD
TH44w322	AB+AC+AD+BC
TH54w322	AB+AC+BCD
THxor0	AB+CD
THand0	AB+BC+AD
TH24comp	AC+BC+AD+BD

There are two additional threshold gate modifications, output inversion and reset functionality, that alter both the naming convention and functionality of an NCL threshold gate. A gate that inverts its output is named similarly to traditional threshold gates, but the *n* value is followed with a *b*. A threshold gate that is resettable will have either a *d* or an *n* follow the *n* value to identify whether the gate will reset to DATA or NULL. A gate name that ends in an *n* will output a logic '0' when the reset is asserted, and a gate that ends in a *d* will output a logic '1' when the reset is

asserted. Symbols for a normal, weighted, inverting, and DATA reset NCL threshold gates are found in Fig. 1.

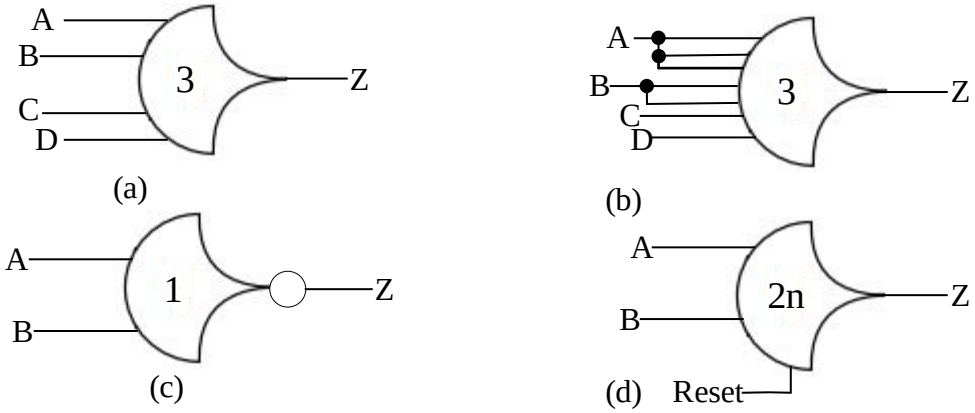

Figure 1. NCL Threshold Gates (a) TH34 Gate (b) TH34w32 Weighted Gate (c) TH12b Inverting Gate (d) TH22n Reset Gate

Hysteresis

A key characteristic of NCL threshold gates is their implementation of hysteresis. Hysteresis enforces the paradigm that gate output is dependent on both the assertion of inputs and the current value of the gate output. In other words, hysteresis requires that all inputs de-assert before the gate output may de-assert. Proper functionality of the NCL handshaking protocol requires this hysteresis quality of NCL threshold gates. Communication among components of an NCL design is accomplished through an NCL handshaking protocol that alternates between DATA and NULL wavefronts which stem from the valid three-state encoding method {DATA0, DATA1, NULL} of dual-rail NCL signals. Handshaking results from intercomponent requests for DATA and NULL wavefronts in an alternating manner such that operations can complete without exact timing or clocking [5]. Handshaking is implemented through the utilization of completion detection components, multi-rail encodings, and standard logic completion signals [4].

NCL Register

Crucial to the understanding of DATA/NULL wavefront propagation in an NCL design is comprehension of a basic NCL register. These asynchronous registers are resettable to DATA or NULL, and they store single dual-rail signal values. These registers operate such that DATA and NULL wavefronts are accepted in an alternating manner like any NCL component. However, an NCL register also accepts and outputs a standard logic completion signal. The output completion signal, k_o, is determined by the current contents of an NCL register. When the register is storing a NULL wavefront, the k_o value is '1,' indicating a register's ability to accept a DATA wavefront. When the k_o value is '1,' this is considered a request for data (*rfd*). Conversely, when the register is storing a DATA wavefront, the k_o value is '0,' and this is considered a request for NULL (*rfn*), indicating that a register can output its DATA wavefront in exchange for an incoming NULL wavefront. A gate level schematic of a NULL reset NCL register is exhibited in Fig. 2.

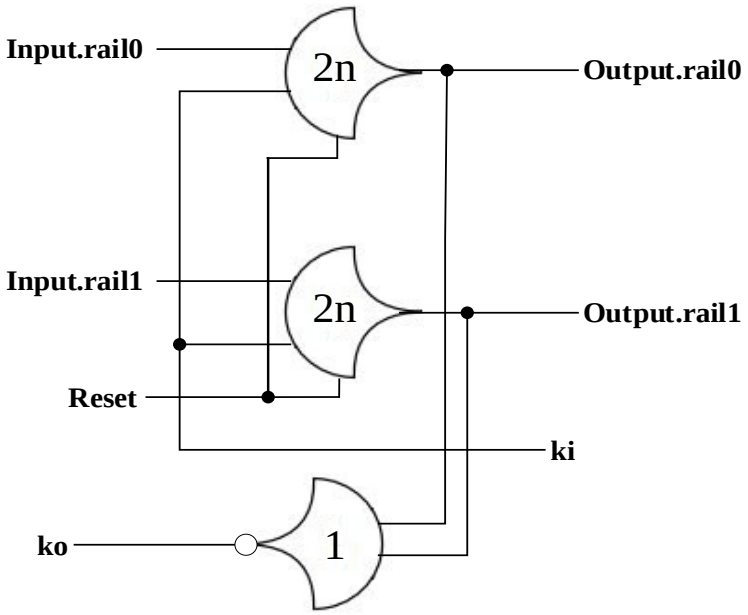

Figure 2. NCL Dual-Rail Register

Single NCL registers are grouped together to handle the processing of dual-rail vectors, and in doing so, a single k_o value must be generated by grouping together the individual register k_o signals [2]. The completion detection component for these NCL vector registers are THnn gates, where *n* is the number of k_o signals. In instances where more than four of these NCL registers are used to store a dual-rail vector, the completion detection component is comprised of THnn gates that are staggered to eventually generate a single k_o value for the register set. According to the previous NCL threshold gate description, using the THnn gates ensures that the final k_o value for a register set will not be *rfd* until all internal k_o values are also *rfd*. Similarly, the final k_o value will not become *rfn* until all internal k_o values are also *rfn*. There is another critical completion signal named k_i that is also a standard logic type, and this signal value is the k_o value generated by an NCL register in a successive pipeline stage. NCL registers function such that a DATA wavefront is only accepted when the subsequent register is requesting DATA (i.e., k_i = '1'). Similarly, NCL registers can only accept a NULL wavefront when the subsequent register is requesting NULL (i.e., k_i = '0'). It is the interaction of completion signals between NCL register stages from which the NCL handshaking protocol arises.

Single-Stage NCL Pipeline

As depicted in Fig. 3, two sets of NCL registers and an intermediate combinational logic block comprise the single-stage pipeline for an NCL design. In such a single-stage NCL pipeline, the only external signal required other than the expected DATA or NULL dual-rail wavefront is the k_i value for the output register that would normally be generated by the k_o value of a subsequent register. Within the pipeline stage, the output register's k_o becomes the input register's k_i signal. DATA and NULL wavefronts, indicated by the grey arrows in Fig. 3, flow in the opposite direction of the completion signals in an NCL design, identified by the black arrows. NCL systems are

usually initialized with a reset that causes NCL registers to output NULL, generating a k_o value of '1,' or *rfd* [2]. This allows NCL registers to accept a DATA wavefront upon system initialization. When DATA is presented at the input of this NCL input register, the DATA wavefront will be latched by the register since its k_i value is '1.' Once this DATA wavefront is latched, the input register's k_o becomes a '0,' or *rfn*. After being latched by the input register, the DATA wavefront propagates through the NCL combinational logic block to the input of the output NCL register. Once the external k_i is asserted, the DATA wavefront will be latched by the output NCL register, and both registers in the single-stage pipeline will have k_o values of '0.' This means that both registers will now be requesting a NULL wavefront. According to the NCL handshaking protocol, a NULL wavefront should now be delivered to the input of the NCL input register. Since the internal completion signal (k_i of input register; k_o of output register) is *rfn*, this NULL wavefront will be latched by the input register. As before, this NULL wavefront will then be propagated throughout the combinational logic block and presented to the input of the NCL output register. Only when the external k_i is de-asserted will the output register latch this NULL wavefront, completing a DATA-NULL cycle [4]. When this occurs, both registers in this single-stage pipeline will be requesting a DATA wavefront, as they were just after initialization.

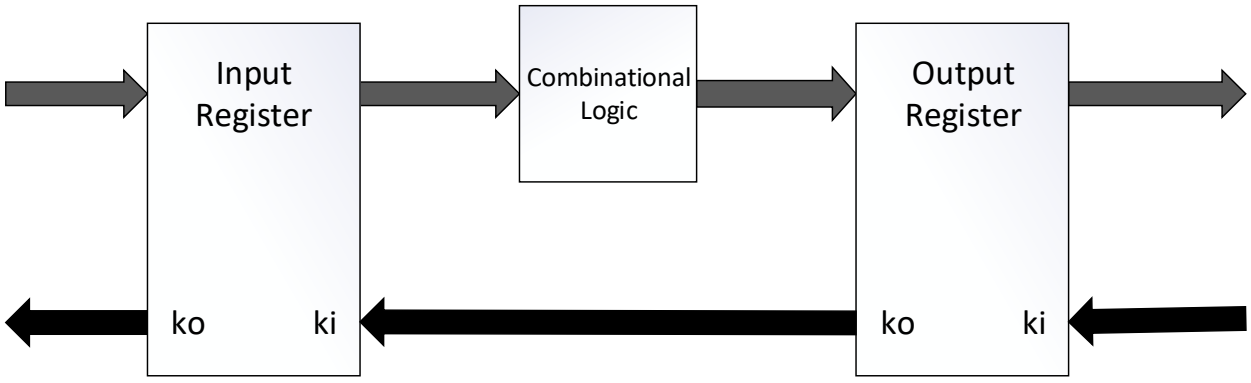

Figure 3. Single-Stage NCL Pipeline

Asynchronous systems that are designed with NCL do not have clock cycles as traditional synchronous systems do, so their performance is measured by DATA-to-DATA time, or the time between two successive DATA wavefronts. Since this DATA-to-DATA cycle is generated by the NCL handshaking protocol, operations are event-driven, not clock-driven. This quality of NCL results in average-case performance, contrasting the worst-case performance of synchronous designs [4]. Additionally, this protocol facilitates normal circuit operation in extreme environments that would negatively impact the functionality of synchronous systems.

III. Approach

A. Drawbacks of NCL Bus

Just as in synchronous IC designs, asynchronous architectures can utilize a bus for intercomponent data transfers and any other component communication in a system, but this traditional communication method has characteristics that become cumbersome in asynchronous systems designed with NCL.

The most impactful difference between the bus communication architecture in synchronous versus asynchronous designs is that in synchronous systems, the bus is only responsible for data delivery, whereas an asynchronous bus is required to deliver DATA and NULL wavefronts to system components [2]. In the common implementation of an NCL bus, it is necessary to attach pull-down resistors to each bus wire so that a '0' can be delivered to system components from each bus wire, resulting in delivery of a NULL wavefront. Without this implementation, the bus wires would be floating during NULL wavefront delivery, resulting in a number of issues including large short circuit power consumption and even disruption of circuit operation. Delivery of DATA wavefronts to asynchronous components does not require additional overhead to support it because

either *rail0* or *rail1* of an NCL dual-rail signal will be assigned logic ‘1’ during a DATA wavefront. This communication architecture can be identified in Fig. 4(a), and its structure is not complicated with regards to the number of components. Additionally, this architecture demonstrates efficient performance in combination with little area requirements. However, the pull-down resistors associated with each bus wire result in higher power consumption when driving these wires to logic ‘1’ because the current in a bus wire present during a DATA wavefront will sink through the pull-down resistors. Upon delivery of a DATA wavefront, the successive NULL wavefront that is expected by NCL components will be presented by discharging asserted bus wires with the pull-down resistor network. A secondary drawback of this bus communication architecture is that the pull-down resistor network causes rise and fall delays when the bus wires are asserted and de-asserted. These architectural disadvantages lay the foundation for further research into new methods of communication among asynchronous components that improve upon bus disadvantages while maintaining the inherent advantages of circuits designed with NCL.

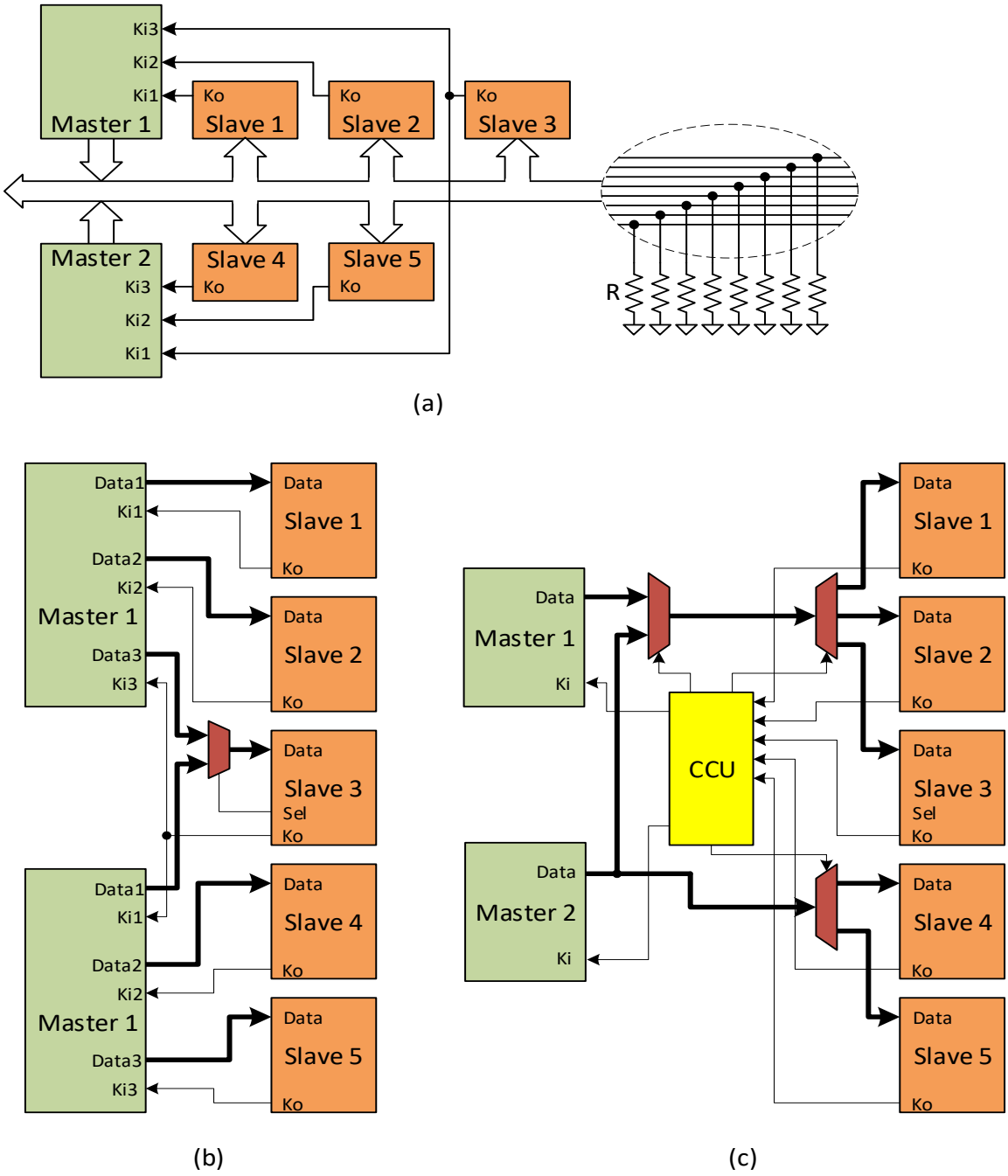

Figure 4. (a) NCL Bus (b) MUX Bus (c) STAR

B. Proposed Alternative Communication Architectures

In response to the drawbacks of the NCL bus communication architecture, MUX Bus and STAR asynchronous communication architectures are designed and tested. These proposed architectures promote improved rise and fall times for information transmission between system components, and both architectures feature multiplexers at the inputs of slave components. In both designs, *select* signals for these multiplexers are generated that identify which master and slave components are involved in DATA/NULL wavefront conveyance. Between the two new architecture proposals, STAR requires less control logic at the expense of increased NCL gate count and therefore area.

C. MUX Bus

MUX Bus architecture is depicted in Fig. 4(b), and this implementation consists of multiplexer-based control to connect several master-slave combinations. Since *select* signals can exclude master component outputs from being delivered to certain slave components, there is a possibility in some instances that there could exist multiple non-overlapping master-slave combinations at a given time. *Select* signals are determined by the system control finite state machine (FSM) upon decoding and interpreting an instruction, and they are taken as inputs by the slave data input multiplexers to determine the data paths from master to slave components in the system. The *select*, k_o, and k_i signals necessary for proper handshaking protocol functionality originate from the slave component. For example, if a multiplier is attempting to send data to a register, the appropriate *select* signal is generated such that the multiplier output is received by the register's data input multiplexer. With respect to this specific example, the multiplier is the master that will be waiting on the register (the slave component) to change its k_o output from '1' to '0,' indicating that it has received the DATA wavefront and is now ready for a NULL wavefront. Since this

register k_o is sent to the master as the multiplier k_i input, the multiplier will send a NULL wavefront to the slave multiplexer once the *rfn* is received. The register multiplexer then propagates the NULL wavefront to the slave, changing the register k_o value to '1' from '0' to indicate that it is prepared to accept another DATA wavefront.

D. STAR

Different from MUX Bus architecture in STAR architecture is the Central Control Unit (CCU) that is responsible for *select* signal generation for all component data input multiplexers. STAR architecture is identified in Fig. 4(c). The CCU generates these *select* signals based on the k_o completion signals of the system and the communication instructions from the system FSM. As with other NCL components, the CCU requires a NULL wavefront after it propagates a DATA wavefront throughout a system for reinitialization to process successive DATA wavefronts. Since the CCU is dependent upon instructions from the system FSM, STAR does not maintain the MUX Bus architectural advantage of allowing non-overlapping instructions (multiple groups of slave-master components). Therefore, execution time for STAR is expected to be longer when compared to the two alternative architectures discussed. In addition, due to the decoder in the CCU that generates the system's internal handshaking and selection logic, there are specific instructions that may have a longer execution time. The instructions that will be affected most by this structural alteration are instructions that require multiple DATA-NULL cycles.

In lieu of accepting every component k_o signal, the CCU generates the required internal system signals based off a given instruction. These instructions are delivered to the CCU as previous instructions are completed, as identified by the completion logic. This aspect of STAR makes the architecture handle the least amount of input/output of the three transfer architectures discussed.

IV. Results and Comparisons

A. Tested Components

All three data transfer architectures utilized the same five NCL components for communication during simulation. The five components used were two 8-bit NCL registers, an 8-bit NCL program counter (PC), a 4 × 4 NCL multiplier, and an 8-bit NCL ripple-carry adder. Each of these components, regardless of the architecture, was designed to operate with 8-bit dual-rail signals. Furthermore, all components and data transfer architectures themselves were simulated with the IBM 130nm semiconductor process.

B. Test Setup

Each data communication architecture was functionally verified in VHDL, flattened into an NCL threshold gate netlist, imported into Cadence, and simulated at the transistor-level in the 130nm IBM semiconductor process so that power and timing data could be determined. The sole differences between the simulated transfer architectures were how DATA and NULL wavefronts were delivered to the NCL components of the design. Obtaining meaningful power and timing information was achieved by designing architecture-specific VerilogA controllers to be associated with their appropriate architecture symbols in the Cadence schematic environment so that inputs could be delivered to the architectures. Similarly, the outputs of each communication architecture were delivered to their respective VerilogA controller to allow the three systems to fully simulate and be self-dependent once the simulation was started.

Not only did each transfer architecture use the same components, but they also implemented the same instruction set to guarantee comparable results. The instruction set is as follows: (1) Load a register with a value; (2) Load data into the multiplier; (3) Load a second register with the

multiplier output; (4) Store a register value into input *A* of the adder; (5) Store the second register value into input *B* of the adder; (6) Store the adder output to a register; (7) Deactivate the program counter and store the final PC value into a register. The final NCL register contents should be the final PC value and the result of the addition step.

C. Simulation Results

Performance characteristics of the data transfer architectures that were deemed important for meaningful comparisons were speed, leakage power, size, and active energy. Active energy indicates the energy that an architecture consumes during execution of the aforementioned instruction set. Size was gauged on the quantity of NCL threshold gates utilized in each architecture implementation. Leakage power is a designation of the power consumed when architectures are idle. The results with regards to these performance indices are shown in Table 3.

Table 3. Data Transfer Architecture Performance Comparison

Transfer Architecture	NCL Bus	MUX Bus	STAR
NCL Gate Count	1,292	1,621	1,962
Active Energy (picojoules)	74.33	10.41	18.7
Leakage Power (microwatts)	0.855	1.648	2.078
Execution Time (nanoseconds)	17	12.3	30

According to the data presented in Table 3, the two proposed data transfer alternatives showed improvements in some areas when compared to the NCL bus implementation, but these architectures also had some comparative disadvantages. Thus, there are specific situations where one might choose NCL bus over either of the proposed architectures and vice versa. Due to the absence of pull-down resistors on bus wires that would need to be asserted/de-asserted during execution, MUX bus achieves a shorter execution time than NCL bus. The active energy of MUX bus is less than that of NCL bus for the same logic (14% of NCL bus active energy). Since MUX

bus implemented multiplexers for master/slave component communication, this instinctively results in more NCL gates. The increased gate count in MUX bus is also the reason for increased idle-state leakage power when compared to NCL bus (92% increase). Thus, MUX bus is the advantageous architecture when importance is placed on execution time or active energy, as opposed to space and leakage power. However, NCL bus remains ideal in situations that are constrained by space requirements or that allow systems to idle for lengthy amounts of time.

The differences between STAR architecture and MUX bus architecture are minimal, so similar trends with regards to performance measurements are anticipated. STAR architecture and MUX bus architectures are quite alike, with the caveat that STAR utilizes a CCU to generate *select* signals for the multiplexers that are responsible for controlling component data inputs and the handshaking protocol through k_i selection. The CCU naturally resulted in more NCL threshold gates than MUX bus (around 300 more gates than MUX bus). This additional component, while reducing the number of inputs required for proper system functionality, did come with some expenses. The execution time of STAR architecture was nearly double that of the NCL bus architecture, and it had a 52% NCL gate increase over NCL bus (roughly 700 more NCL threshold gates). Again, the increased gate count naturally equates to increased leakage power when the system idles. The major advantage of STAR over the traditional NCL bus architecture is decreased active energy, as is the case with MUX bus. STAR had 25% of the active energy demanded by the NCL bus, and like MUX bus, this is because of the absence of bus wires that are tied to a pull-down resistor network. As the CCU is essentially a large decoder, the increased execution time of STAR over NCL bus can be attributed to the interpretation of data communication instructions and to the absence of multiple master-slave communication paths at once. In closure, utilization

of the STAR architecture seems to be limited by chip area requirements and leakage power constraints.

V. Conclusion

This research proposed, implemented, and analyzed the tradeoffs of two new data transfer architectures as alternatives to the typical NCL bus communication method. These new architectures succeeded in reducing the active energy requirements for asynchronous systems designed with NULL Convention Logic (NCL). On-chip area and leakage power are sacrificed in both implementations to achieve this advantage. Consequently, chip designers should consider all project-specific design constraints when choosing between any of these transfer architectures.

VI. References

[1] N. Kuhns, L. Caley, A. Rahman, S. Ahmed, J. Di, H. A. Mantooth, A. M. Francis, and J. Holmes, *Complex High-Temperature CMOS Silicon Carbide Digital Circuit Designs*, IEEE Transactions on Device and Materials Reliability, Vol. 16, Issue 2, pp. 105-111, February 2016

[2] N. Kuhns, Power Efficient High Temperature Asynchronous Microcontroller Design, University of Arkansas PhD. Dissertation, May 2017.

[3] J. Brady, A. M. Francis, J. Holmes, J. Di, and H. A. Mantooth, "An Asynchronous Cell Library for Operation in Wide-Temperature & Ionizing-Radiation Environments," 2015 IEEE Aerospace Conference

[4] Scott C. Smith, Jia Di, Designing Asynchronous Circuits using NULL Convention Logic (NCL), Morgan & Claypool Publishers, 2009.

[5] N. Kuhns, L. Caley, A. Rahman, S. Ahmed, J. Di, H. A. Mantooth, A. M. Francis, and J. Holmes, "High Temperature Testing Results of Synchronous and Asynchronous Digital Silicon Carbide Integrated Circuits," Government Microcircuit Applications & Critical Technology Conference (GOMACTech), 2015