

# Design and Characterization of NULL Convention Arithmetic Logic Units

Satish K. Bandapati & Scott C. Smith

University of Missouri – Rolla, Department of Electrical and Computer Engineering  
133 Emerson Electric Co. Hall, 1870 Miner Circle, Rolla, MO 65409

Phone: (573) 341-4232, Fax: (573) 341-4532, E-mail: [smithsco@umr.edu](mailto:smithsco@umr.edu)

**Keywords:** Asynchronous logic design, delay-insensitive circuits, self-timed circuits, computer arithmetic, ALU

## Abstract

*In this paper a number of 4-bit, 8-operation arithmetic logic units (ALUs) are designed using the delay-insensitive NULL Convention Logic (NCL) paradigm, and are characterized in terms of speed and area. Both dual-rail and quad-rail, pipelined and non-pipelined versions are developed, and the tradeoffs and design considerations for each are discussed. Comparing the various architectures shows that the fastest dual-rail and quad-rail ALUs achieve average speedups of 1.72 and 1.59, respectively, over their non-pipelined counterparts, while requiring 133% and 119% more area, respectively. Overall, the dual-rail designs are both faster and require less area than their respective quad-rail counterparts; however, the quad-rail versions are expected to consume less power.*

## 1. Introduction

Presently, the development of synchronous circuits dominates the semiconductor industry. However, there are major limiting factors to this design approach, including clock distribution, increasing clock rates, decreasing feature size, excessive power consumption, and timing closure effort. Correct-by-construction asynchronous circuits, such as NCL, have been demonstrated to require less power, generate less noise, produce less EMI, and allow for easier reuse of components, compared to their synchronous counterparts, without compromising performance [1]. Furthermore,

correct-by-construction asynchronous paradigms should allow for much greater flexibility when designing complex circuits, like SoCs, since the circuits are self-timed; thus the effort required to ensure correct operation under all timing scenarios should be drastically reduced, compared to equivalent synchronous designs. Also, the self-timed nature of correct-by-construction SoCs should allow for previously designed and verified functional blocks to be reused in subsequent designs, without necessitating any significant modifications or retiming effort within a reused functional block, and may provide for simpler interfacing between the digital core and non-traditional functional blocks. Therefore one of the first tasks that needs to be undertaken in order to help integrate NCL into the semiconductor design industry is to design and characterize key components, in order to form a library of reusable designs. Of fundamental importance are arithmetic circuits, including the ALUs developed and compared in this paper. An overview of the NULL Convention Logic (NCL) paradigm is provided in the next section.

## 2. Overview of NCL

NCL offers a self-timed logic paradigm where control is inherent with each datum. NCL follows the so-called “weak conditions” of Seitz’s delay-insensitive signaling scheme [2]. As with other self-timed logic methods discussed herein, the NCL paradigm assumes that forks in wires are isochronic [3]. The origins of various aspects of the paradigm, including the NULL (or spacer) logic state from which NCL derives its name, can be traced back to Muller’s work on speed-independent circuits in the 1950s and 1960s [4].

---

<sup>†</sup> I would like to thank the University of Missouri Research Board for their funding that has made this work possible.

## 2.1 Delay-Insensitivity

NCL uses symbolic completeness of expression [5] to achieve delay-insensitive behavior. A symbolically complete expression is defined as an expression that only depends on the relationships of the symbols present in the expression without a reference to their time of evaluation. In particular, dual-rail signals, quad-rail signals, or other *Mutually Exclusive Assertion Groups* (MEAGs) can be used to incorporate data and control information into one mixed signal path to eliminate time reference [6]. 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. A quad-rail signal,  $Q$ , consists of four wires,  $Q^0, Q^1, Q^2$ , and  $Q^3$ , which may assume any value from the set {DATA0, DATA1, DATA2, DATA3, NULL}. The DATA0 state ( $Q^0 = 1, Q^1 = 0, Q^2 = 0, Q^3 = 0$ ) corresponds to two Boolean logic signals,  $X$  and  $Y$ , where  $X = 0$  and  $Y = 0$ . The DATA1 state ( $Q^0 = 0, Q^1 = 1, Q^2 = 0, Q^3 = 0$ ) corresponds to  $X = 0$  and  $Y = 1$ . The DATA2 state ( $Q^0 = 0, Q^1 = 0, Q^2 = 1, Q^3 = 0$ ) corresponds to  $X = 1$  and  $Y = 0$ . The DATA3 state ( $Q^0 = 0, Q^1 = 0, Q^2 = 0, Q^3 = 1$ ) corresponds to  $X = 1$  and  $Y = 1$ , and the NULL state ( $Q^0 = 0, Q^1 = 0, Q^2 = 0, Q^3 = 0$ ) corresponds to the empty set meaning that the result is not yet available. The four rails of a quad-rail NCL signal are mutually exclusive, such that no two rails can ever be asserted simultaneously; these states are defined as illegal states. Both dual-rail and quad-rail signals are space optimal 1-out-of- $N$  delay-insensitive codes, requiring two wires per bit. Other higher order MEAGs are not wire count optimal; however, they can be more power efficient due to the decreased number of transitions per cycle.

Most multi-rail delay-insensitive systems, including NCL, have at least two register stages, one at both the input and at the output. Two adjacent register stages interact through their request and acknowledge lines,  $K_i$  and  $K_o$ , 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.

## 2.2 Logic Gates

NCL differs from many other delay-insensitive paradigms in that these other paradigms only utilize one type of state-holding gate, the *C-element* [4]. A C-element behaves as follows: when all inputs assume the same value then the output assumes this value, otherwise the output does not change. On the other hand, all NCL gates

are state-holding. Thus, NCL optimization methods can be considered as a subclass of the techniques for developing delay-insensitive circuits using a pre-defined set of more complex components, with built-in *hysteresis* behavior.

NCL uses *threshold gates* for its basic logic elements [7]. The primary type of threshold gate is the *TH $mn$  gate*, where  $1 \leq m \leq n$ , as depicted in Figure 1. TH $mn$  gates have  $n$  inputs. At least  $m$  of the  $n$  inputs must be asserted before the output will become asserted. Because NCL threshold gates are designed with hysteresis, 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 TH $nn$  gate is equivalent to an  $n$ -input C-element and a TH1 $n$  gate is equivalent to an  $n$ -input OR gate. In a TH $mn$  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. Resettable gates are denoted by either a  $D$  or an  $N$  appearing inside the gate, along with the gate's threshold, referring to the gate being reset to logic 1 or logic 0, respectively.

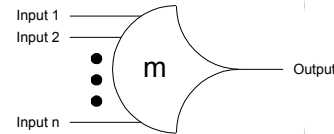


Figure 1: TH $mn$  threshold gate.

By employing threshold gates for each logic rail, NCL is able to determine the output status without referencing time. Inputs are partitioned into two separate wavefronts, the NULL wavefront and the DATA wavefront. The NULL wavefront consists of all inputs to a circuit being NULL, while the DATA wavefront refers to all inputs being DATA, some combination of DATA0 and DATA1. Initially all circuit elements are reset to the NULL state. First, a DATA wavefront is presented to the circuit. Once all of the outputs of the circuit transition to DATA, the NULL wavefront is presented to the circuit. Once all of the outputs of the circuit transition to NULL, the next DATA wavefront is presented to the circuit. This DATA/NULL cycle continues repeatedly. As soon as all outputs of the circuit are DATA, the circuit's result is valid. The NULL wavefront then transitions all of these DATA outputs back to NULL. When they transition back to DATA again, the next output is available. This period is referred to as the DATA-to-DATA cycle time, denoted

as  $T_{DD}$ , and has an analogous role to the clock period in a synchronous system.

### 2.3 Completeness of Input

The completeness of input criterion [5], which NCL combinational circuits and circuits developed from other delay-insensitive paradigms must maintain in order to be delay-insensitive, requires that:

1. all the outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and
2. all the 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 [2], 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.

### 2.4 Observability

There is one more condition that must be met to ensure delay-insensitivity for NCL circuits and other delay-insensitive circuits. No *orphans* may propagate through a gate [8]. 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 [3], 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.

## 3. Dual-Rail ALU

A block diagram of the dual-rail 4-bit ALU is shown in Figure 2.  $S$  selects which operation is to be performed on the 4-bit inputs,  $A$  and  $B$ , and the carry/borrow,  $C_{in}/B_{in}$ , determined by the function table in Figure 3.  $F$  is the 4-bit output and  $C_{out}/B_{out}$  is the carry/borrow output. This circuit, as do all NCL systems, contains a complete request/acknowledge interface, including  $K_o$  for requesting the inputs, and  $K_i$  for acknowledging the outputs, and a *reset* input to initialize the NCL registers to NULL. Notice that  $C_{in}/B_{in}$  is not used in operations 0-3, and that  $B$  is not used in operations 3-5. However, to ensure delay-insensitivity the ALU must still be input-complete with respect to these inputs, even for the operations where the  $C_{in}/B_{in}$  and/or  $B$  input(s) are not used. This ensures that the unused inputs are received before the output can transition.

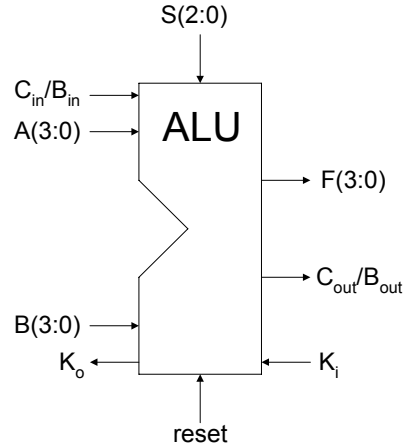


Figure 2. Dual-rail ALU block diagram.

$S_2$	$S_1$	$S_0$	F	$C_{out}/B_{out}$
0	0	0	A OR B	0
0	0	1	A AND B	0
0	1	0	A XOR B	0
0	1	1	NOT A	0
1	0	0	SHR $C_{in}$ , A	$A_0$
1	0	1	SHL A, $C_{in}$	$A_3$
1	1	0	$A-B-1+B_{in}$	$B_{out}$
1	1	1	$A+B+C_{in}$	$C_{out}$

Figure 3. ALU function table.

### 3.1 Non-Pipelined Version

The logic diagram of the non-pipelined dual-rail ALU is shown in Figure 4. It consists of dual-rail registers [5], completion components, denoted as COMP [9], a Convert to MEAG function, a Demultiplexer, NCL OR, AND, XOR, invert, shift right, and shift left functions, a ripple-carry subtractor and adder, two Multiplexers, and Carry Logic.

The Convert to MEAG function is comprised of eight TH33 gates that convert  $S$ , which consists of three dual-rail signals, into an 8-rail MEAG. The OR, AND, and XOR function are all input-complete versions [9], consisting of two gates and one gate delay. The invert, shift right, and shift left function are performed by renaming signals, hence they have no logic delay. The ripple-carry subtractor and adder consist of four full-adders [10], while the Multiplexers consist of TH14 and TH12 gates, which OR each rail of the demultiplexed results together to form a single result. The Carry Logic is required to generate a  $C_{out}$  of logic 0, when operation 0-3 is chosen, and to ensure input-completeness with respect to  $C_{in}$  in these cases, in order to maintain delay-insensitivity, as shown in Figure 5.

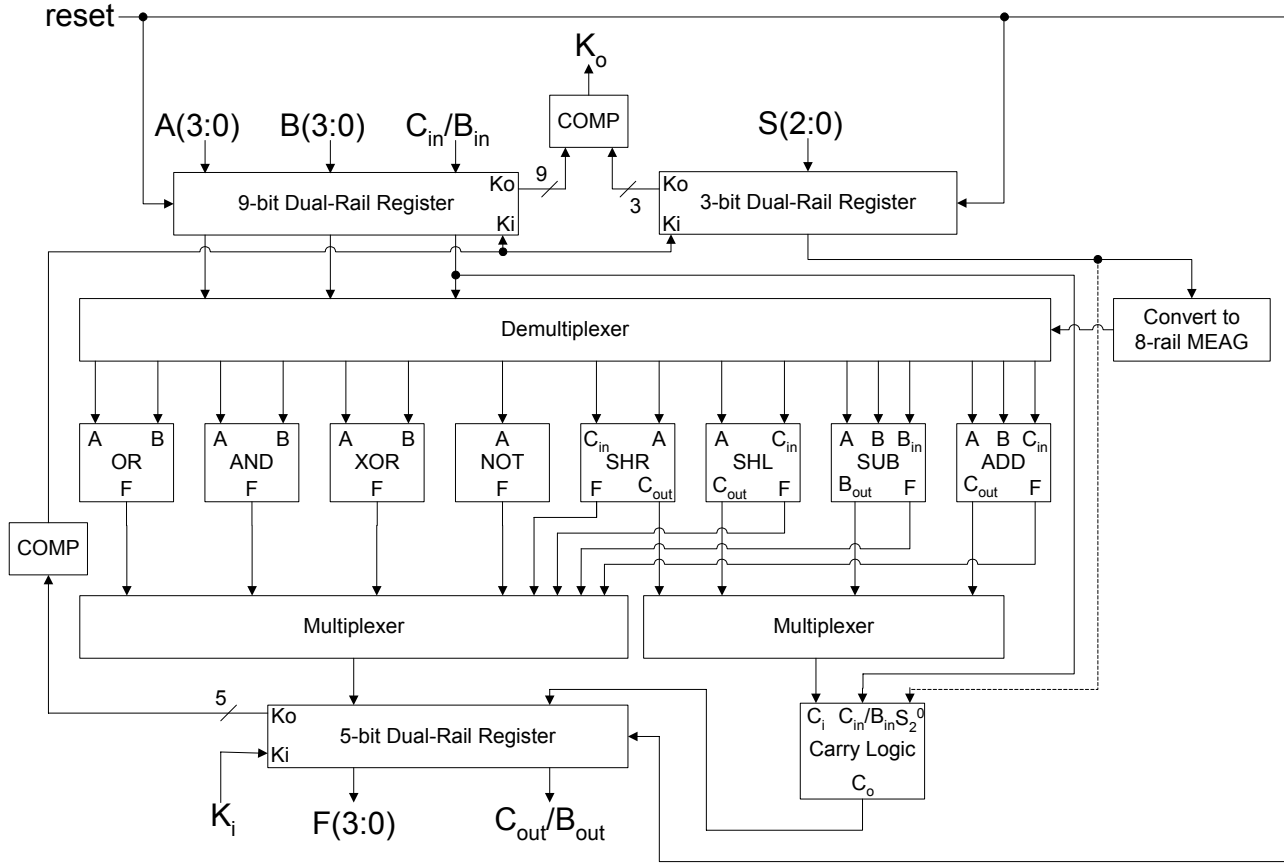


Figure 4. Logic diagram of dual-rail non-pipelined ALU.

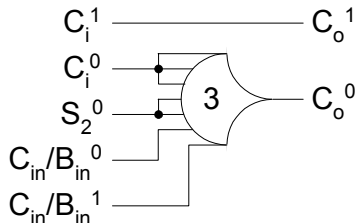


Figure 5. Carry logic.

The Demultiplexer consists of TH22 gates, which pass the  $A$  input, and the  $B$  and  $C_{in}/B_{in}$  inputs, when necessary, to the correct function, determined by the select MEAG. For the functions in which  $B$  is not required, the Demultiplexer consists of TH34 gates to pass  $A$ , while maintaining input-completeness with respect to  $B$ . Input-completeness of  $C_{in}/B_{in}$  is ensured in the carry logic. The alternative is to always pass  $B$  to every function, whether it is needed or not, and then ensure input-completeness of  $B$  within functions 3-5. However, this would require an extra gate delay for these three functions, which reduced average throughput by 3% and required an additional 24 gates; hence this alternative was not chosen.

### 3.2 Pipelined Version

The interface for the pipelined dual-rail ALU is the same as for the non-pipelined version, shown in Figure 2; the logic diagram is shown in Figure 6. To pipeline the ALU, registration stages must be added within the combinational logic of the non-pipelined version, without violating the completeness of input criterion [5]. To minimize both delay and area, many embedded registration stages were used, where the combinational logic and registration are combined together (i.e. the Convert to MEAG Register, the Carry MEAG Register, the Demultiplexer Register, the Select (Sel) Registers, and the Multiplexer Register). The major differences between the pipelined and non-pipelined designs are the input-completeness of the  $B$  and  $C_{in}/B_{in}$  inputs, and the subtractor and adder circuits. Since a registration stage was to be added between the Demultiplexer and the functions, an extra level of logic would have been required for this registration if input-completeness of  $B$  was performed within the demultiplexer, as was done for the non-pipelined design, since this required maximum 4-input, TH34 gates; thus excluding the possibility of embedded registration due to the needed additional request input and the 4-input limitation. Therefore the

alternative to pass  $B$  to functions 3-5 and then ensure input-completeness of  $B$  within these functions was chosen; since this option only required 2-input, TH22 gates, for the demultiplexer, thus enabling the extra request input to be added, changing the TH22 gates to TH33 gates, in order to embed the registration stage within the combinational logic of the demultiplexer. Also, both the Demultiplexer Register and the Select Register require special completion components, since the inputs are only sent to one out of the eight output sets, whereas normally all outputs transition every DATA/NULL cycle.

Input-completeness of the  $C_{in}/B_{in}$  input was ensured within the Carry MEAG register, instead of at the end of the design, like in the non-pipelined version, since this would have required unnecessarily pipelining  $C_{in}/B_{in}$  throughout the design. This allowed for the carry output

of zero to be generated from the  $F_0$  output of functions 0-3; hence only one multiplexer component is shown in the diagram. Also, the adder and subtractor circuits have themselves been pipelined [9, 10], by inserting two registration stages between the four full-adders and utilizing bit-wise completion [9, 10]. A third registration stage could have been inserted, however this did not increase throughput. Finally, the select MEAG was pipelined by adding two additional registration stages, after the Carry MEAG Register, such that the ALU could store the select signals associated with the concurrent DATA/NULL wavefronts being processed within the ALU. Both more and less MEAG pipelining stages decreased throughput. These pipelining optimizations resulted in a 32% increase in average throughput, with a 151% increase in area.

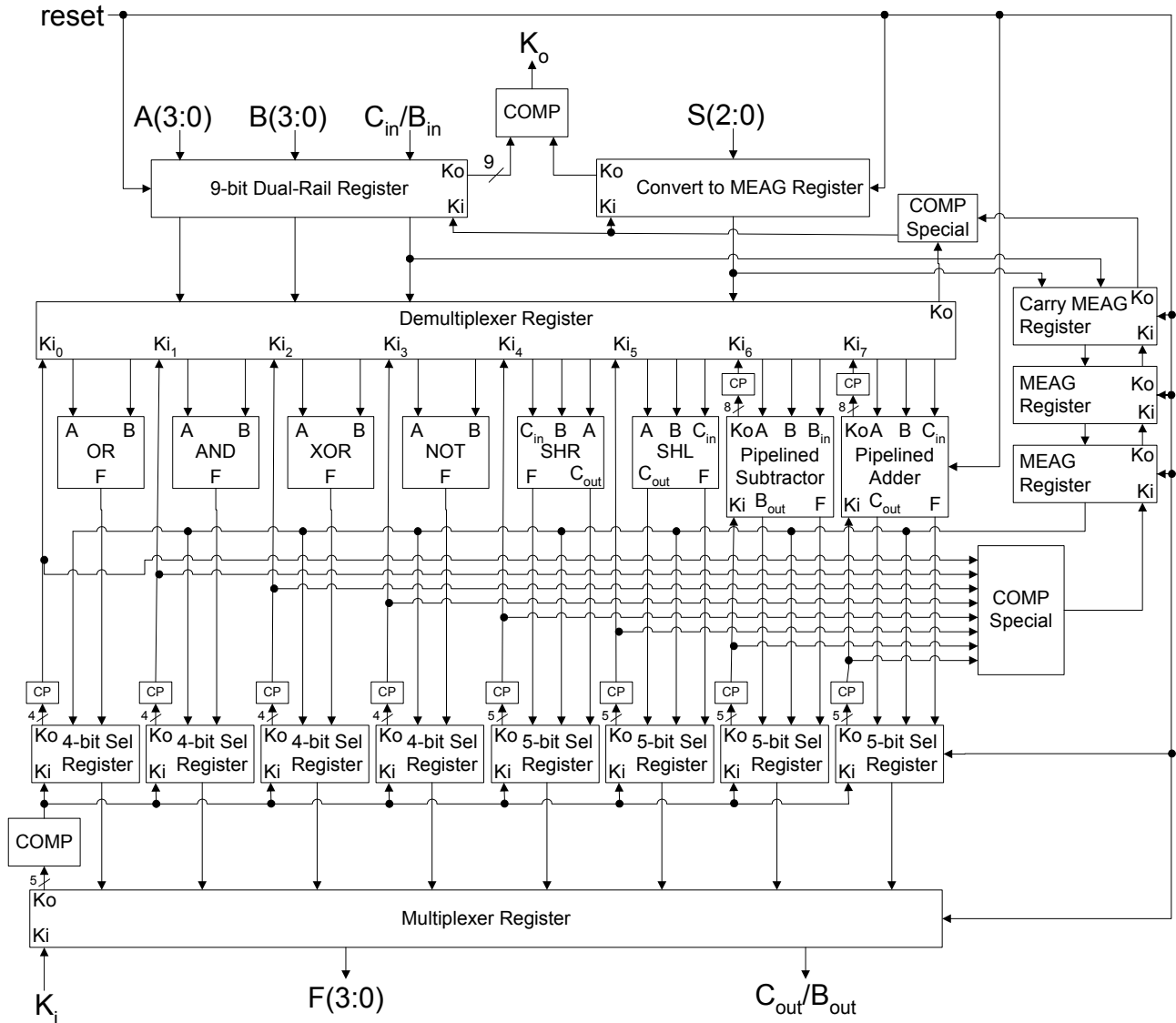


Figure 6. Logic diagram of dual-rail pipelined ALU.

### 3.3 NULL Cycle Reduced Version

Since pipelining the ALU resulted in more than twice the area of the non-pipelined design, with a speedup of only 1.32, a viable alternative was to use the NULL Cycle Reduction (NCR) technique [9, 11] on the non-pipelined design to increase its throughput. NCR demultiplexes successive input wavefronts such that one circuit processes a DATA wavefront, while its duplicate processes a NULL wavefront. The first DATA/NULL cycle flows through the original circuit, while the next DATA/NULL cycle flows through the duplicate circuit. The outputs of the two circuits are then multiplexed to form a single output stream. The application of NCR to the non-pipelined design resulted in a 63% increase in average throughput, with only a 136% increase in area, thus the NCR version was both faster than the pipelined version and it required less area.

To further increase throughput and decrease area, embedded registration was applied to the non-pipelined design. This resulted in the Convert to MEAG function becoming an embedded register, replacing the 3-bit input register, and the multiplexer for  $F$  and Carry Logic function becoming embedded registers, replacing the 5-bit output register. This optimization resulted in a speedup of 1.18 over the original non-pipelined design, and a 2% decrease in area. Now NCR could be applied to the non-pipelined design utilizing embedded registration, resulting in a speedup of 1.72 with only a 133% increase in area, verses the original non-pipelined design.

### 4. Quad-Rail ALU

The interface for the quad-rail ALU is similar to that of the dual-rail ALU, shown in Figure 2, except that  $A$ ,  $B$ , and  $F$  now consist of two quad-rail signals, instead of four dual-rail signals, and the select input,  $S$ , consists of one dual-rail signal,  $S_2$ , and one quad-rail signal, which represents  $S(1:0)$ . The logic diagram for the quad-rail ALU is also very similar to that of the dual-rail version, shown in Figure 4. It consists of both quad-rail and dual-rail registers [9], completion components, a Convert to MEAG function, a Demultiplexer, NCL OR, AND, XOR, invert, shift right, and shift left functions, a ripple-carry subtractor and adder, two Multiplexers, and Carry Logic. The main difference is that each component and each function was designed using quad-rail logic instead of dual-rail logic. This results in ripple-carry subtractor and adder circuits consisting of only two quad-rail adders (two quad-rail signals plus a dual-rail carry input, yielding a quad-rail sum and a dual-rail carry output). The other major difference was that  $B$  had to be passed to all functions through the demultiplexer and input-completeness of  $B$  had to be ensured within functions 3-5, due to the quad-rail logic and 4-input limitation for

gates. Input-completeness of  $C_{in}/B_{in}$  was ensured in the same manner as for the dual-rail design.

Pipelining the quad-rail design followed in much the same manner as the dual-rail design, resulting in a 34% increase in average throughput, with a 116% increase in area. Again, NCR was applied to the non-pipelined design, resulting in a throughput increase of 53% with a 119% increase in area. Like the dual-rail design, embedded registration was applied to the non-pipelined design, resulting in a 10% increase in throughput with a minimal increase in area; and NCR was applied to the non-pipelined design with embedded registration, resulting in a speedup of 1.59 with an area increase of 119%, verses the original non-pipelined design.

## 5. Simulation Results

The circuits compared in this paper have all been simulated using a 0.25 $\mu$ m CMOS process operating at 3.3V. Table I summarizes the characterization of the various ALUs discussed herein, in terms of both speed and area. Gate count can be used as one measure of area for comparison purposes; however since the NCL gates vary greatly in size (i.e. from 2 transistors for an inverter to 26 transistors for a TH24 gate), transistor count provides a better means of comparison, especially when embedded registration is used, since this increases transistor count without increasing gate count. Also, since NCL circuits are delay-insensitive, speed is data dependent; therefore average cycle time,  $T_{DD}$ , is calculated and used for comparison.  $T_{DD}$  is calculated as the arithmetic mean of the cycle times corresponding to all 4096 possible pairs of input operands.

Table I. ALU Characterization.

ALU Architecture	Gate Count	Transistor Count	$T_{DD}$ (ns)
Dual-Rail Non-Pipelined	260	3757	7.75
Dual-Rail Pipelined	620	9926	5.89
Dual-Rail NCR	542	8878	4.76
Dual-Rail Non-Pipelined with Embedded Registration	245	3688	6.57
Dual-Rail NCR with Embedded Registration	512	8740	4.51
Quad-Rail Non-Pipelined	522	7179	8.85
Quad-Rail Pipelined	1027	15478	6.62
Quad-Rail NCR	1137	15726	5.80
Quad-Rail Non-Pipelined with Embedded Registration	514	7182	8.05
Quad-Rail NCR with Embedded Registration	1121	15732	5.57

## 6. Conclusions

Comparing the various architectures shows that the dual-rail versions of all designs outperform their quad-rail counterparts in terms of both speed and area. However, the quad-rail designs are expected to consume less power, due to the fact that only one quad-rail signal transitions for every two dual-rail signals that transition. The reason that pipelining the designs did not further increase throughput is due to the long completion delays in the special completion components required for both the Demultiplexer Register and Select Registers, described in Section 3.2. However, the application of embedded registration to the non-pipelined design, followed by applying NCR yielded a significant additional increase in throughput over the original non-pipelined design, versus the throughput increase achieved by pipelining the design for both dual-rail and quad-rail architectures. Furthermore, this NCR approach required less area than pipelining for the dual-rail architecture, and only slightly more area than pipelining for the quad-rail architecture.

## References

- [1] John McCardle and David Chester, "Measuring an Asynchronous Processor's Power and Noise," *Synopsys User Group Conference (SNUG)*, Boston, 2001.
- [2] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.
- [3] C. H. (Kees) van Berkel, M. Rem, and R. Saeijs, "VLSI Programming," *1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 152-156, 1998.
- [4] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
- [5] K. M. Fant and 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*, pp. 261-273, 1996.
- [6] T. Verhoff, "Delay-Insensitive Codes – An Overview," *Distributed Computing*, Vol. 3, pp. 1-8, 1988.
- [7] G. E. Sobelman and K. M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.
- [8] 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*, pp. 137-145, 2002.
- [9] S. C. Smith, *Gate and Throughput Optimizations for NULL Convention Self-Timed Digital Circuits*, Ph.D. Dissertation, School of Electrical Engineering and Computer Science, University of Central Florida, 2001.
- [10] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Integration, The VLSI Journal*, Vol. 30/2, pp. 103-131, 2001.
- [11] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Speedup of Delay-Insensitive Digital Systems Using NULL Cycle Reduction," *The 10<sup>th</sup> International Workshop on Logic and Synthesis*, pp. 185-189, June 2001.