

# Implementation of Static and Semi-Static Versions of a $24+8\times 8$ Quad-Rail NULL Convention Multiply and Accumulate Unit

S. R. Mallepalli, S. Kakarla, S. Burugapalli, S. Beerla, S. Kotla, P. Sunkara, W. K. Al-Assadi, and S. C. Smith  
University of Missouri - Rolla, Department of Electrical and Computer Engineering  
1870 Miner Circle, Rolla, MO 65409  
Email: {[srmqm9](mailto:srmqm9@umr.edu), [sk9qd](mailto:sk9qd@umr.edu), [sbz99](mailto:sbz99@umr.edu), [sk7v7](mailto:sk7v7@umr.edu), [sbf4b](mailto:sbf4b@umr.edu), [pvs647](mailto:pvs647@umr.edu), [waleed](mailto:waleed@umr.edu), [smithsco](mailto:smithsco@umr.edu)} [@umr.edu](mailto:@umr.edu)

**Abstract**—This paper focuses on implementing an unsigned  $24+8\times 8$  quad-rail (i.e., accumulator consists of 12 quad-rail signals, while the multiplier and multiplicand are each 4 quad-rail signals) Multiply and Accumulate (MAC) unit using the asynchronous NULL Convention Logic (NCL) paradigm. The design utilizes the array-structured algorithm for partial product summation, and is implemented and simulated in VHDL, the transistor level, and the physical level, using a 1.8V 0.18 $\mu\text{m}$  TSMC CMOS process. The MAC is realized using both static and semi-static versions of the NCL gates; and these two implementations are compared in terms of area, power, and speed.

## I. INTRODUCTION

For the past few decades, the development of synchronous circuits has dominated the semiconductor design industry. However, with clock speed nearing the gigahertz range, doubt has been cast over the suitability of synchronous designs for next generation chips. Asynchronous designs are a potential solution for problems like clock skew, clock jitter, power consumption, noise, etc. A variety of techniques exist for design and implementation of asynchronous circuits, NULL Convention Logic (NCL) [1] being one of the promising design paradigms.

NCL is a delay-insensitive asynchronous paradigm, which means that NCL circuits will operate correctly regardless of when circuit inputs become available; therefore NCL circuits are said to be correct-by-construction (i.e., no timing analysis is necessary for correct operation). NCL circuits utilize dual-rail or quad-rail logic 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. 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-hot delay-insensitive codes, requiring two wires per bit.

NCL uses threshold gates as its basic logic elements [2]. The primary type of threshold gate, shown in Fig. 1, is the  $TH_{mn}$  gate, where  $1 \leq m \leq n$ .  $TH_{mn}$  gates have  $n$  inputs, where at least  $m$  of the  $n$  inputs must be asserted before the output will become asserted. 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.

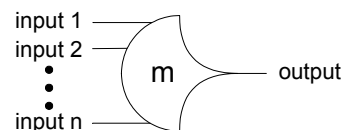


Fig. 1.  $TH_{mn}$  threshold gate.

Another type of threshold gate is referred to as a weighted threshold gate, denoted as  $TH_{mn}W_{w_1}W_{w_2}\dots W_{w_R}$ . 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  $w_1, w_2, \dots, w_R$ , each  $> 1$ , are the integer weights of  $input1, input2, \dots, inputR$ , respectively. For example, consider the  $TH_{34}W_2$  gate shown in Fig. 2, whose  $n = 4$  inputs are labeled  $A, B, C,$  and  $D$ . The weight of input  $A$ ,  $W(A)$ , is therefore 2. Since the gate's threshold,  $m$ , is 3, this implies that in order for the output to be asserted, either inputs  $B, C,$  and  $D$  must all be asserted, or input  $A$  must be asserted along with any other input,  $B, C,$  or  $D$ .

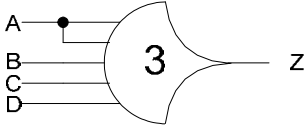


Fig. 2. TH34w2 threshold gate:  $Z = AB + AC + AD + BCD$ .

NCL threshold gates are designed with hysteresis state-holding capability, such that all asserted inputs must be deasserted before the output will be deasserted. 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 $n$ n gate is equivalent to an  $n$ -input C-element and a TH $1$  $n$  gate is equivalent to an  $n$ -input OR gate. There are 27 fundamental NCL gates, constituting the set of all functions consisting of four or fewer variables [2], as shown in Table 1. Since each rail of an NCL signal is considered a separate variable or value, a four variable function is not the same as a function of four literals, which would normally consist of eight variables or values, assuming dual-rail signals.

TABLE I  
27 FUNDAMENTAL NCL GATES

NCL Gate	Boolean Function	Transistors (static)	Transistors (semi-static)
TH12	$A + B$	6	6
TH22	$AB$	12	8
TH13	$A + B + C$	8	8
TH23	$AB + AC + BC$	18	12
TH33	$ABC$	16	10
TH23w2	$A + BC$	14	10
TH33w2	$AB + AC$	14	10
TH14	$A + B + C + D$	10	10
TH24	$AB + AC + AD + BC + BD + CD$	26	16
TH34	$ABC + ABD + ACD + BCD$	24	16
TH44	$ABCD$	20	12
TH24w2	$A + BC + BD + CD$	20	14
TH34w2	$AB + AC + AD + BCD$	22	15
TH44w2	$ABC + ABD + ACD$	23	15
TH34w3	$A + BCD$	18	12
TH44w3	$AB + AC + AD$	16	12
TH24w22	$A + B + CD$	16	12
TH34w22	$AB + AC + AD + BC + BD$	22	14
TH44w22	$AB + ACD + BCD$	22	14
TH54w22	$ABC + ABD$	18	12
TH34w32	$A + BC + BD$	17	12
TH54w32	$AB + ACD$	20	12
TH44w322	$AB + AC + AD + BC$	20	14
TH54w322	$AB + AC + BCD$	21	14
THxor0	$AB + CD$	20	12
THand0	$AB + BC + AD$	19	13
TH24comp	$AC + BC + AD + BD$	18	12

NCL threshold gate variations include *resetting* TH $n$ n and *inverting* TH $1$  $n$  gates. Circuit diagrams designate resettable gates by either a  $d$  or an  $n$  appearing inside the gate, along with the gate's threshold.  $d$  denotes the gate as being reset to logic 1;  $n$ , to logic 0. Both resettable and inverting gates are used in the design of delay-insensitive registers [1].

NCL systems contain at least two delay-insensitive registers, one at both the input and at the output. Two adjacent register stages interact through their request and acknowledge

signals,  $K_i$  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. The acknowledge signals are combined in the Completion Detection circuitry to produce the request signal(s) to the previous register stage. NCL registration is realized through cascaded arrangements of single-bit dual-rail registers or single-signal quad-rail registers. These registers consist of TH22 gates that pass a DATA value at the input only when  $K_i$  is *request for data* ( $rfd$ ) (i.e., logic 1) and likewise pass NULL only when  $K_i$  is *request for null* ( $rfn$ ) (i.e., logic 0). They also contain a NOR gate to generate  $K_o$ , which is  $rfn$  when the register output is DATA and  $rfd$  when the register output is NULL.

An  $N$ -bit register stage, comprised of  $N$  single-bit dual-rail or  $\frac{N}{2}$  single-signal quad-rail NCL registers, requires  $N$  or  $\frac{N}{2}$  completion signals, respectively, one for each dual-rail or quad-rail register. The NCL completion component uses these  $K_o$  lines to detect complete DATA and NULL sets at the output of every register stage and request the next NULL and DATA set, respectively. In full-word completion, the single-bit output of the completion component is connected to all  $K_i$  lines of the previous register stage. Since the maximum input threshold gate is the TH44 gate, the number of logic levels in the completion component for an  $N$ -bit register is given by  $\lceil \log_4 M \rceil$ , where  $M=N$  or  $M=\frac{N}{2}$ , for dual-rail or quad-rail registers, respectively.

This paper concentrates on implementation of a quad-rail NCL Multiply and Accumulate (MAC) unit at all levels of abstraction (i.e., from VHDL to layout). Section II describes the overall MAC design; Section III presents the implementation at the various levels of abstraction; Section IV includes the results and compares the static and semi-static versions in terms of power, area, and delay; and Section V concludes the paper.

## II. NCL MAC DESIGN

A MAC unit takes in two input vectors, generates their product, and adds this to the previous accumulated value. This cycle repeats continuously. The basic components are an input register, a multiplier unit, an accumulator unit, and an output register to store the accumulated value. The multiplier can be implemented using various partial product generation and summation algorithms. The accumulator unit adds the current product to the previous accumulated value, which is stored in the output register. Additionally, an NCL MAC requires three registers in the accumulator feedback loop to allow the DATA and NULL wavefronts to flow properly [1].

The quad-rail NCL MAC presented in this paper consists of the following main components, as shown in Fig. 3:

- 1) 4-signal quad-rail registers for the inputs,  $X$  and  $Y$
- 2) Partial product generation circuitry

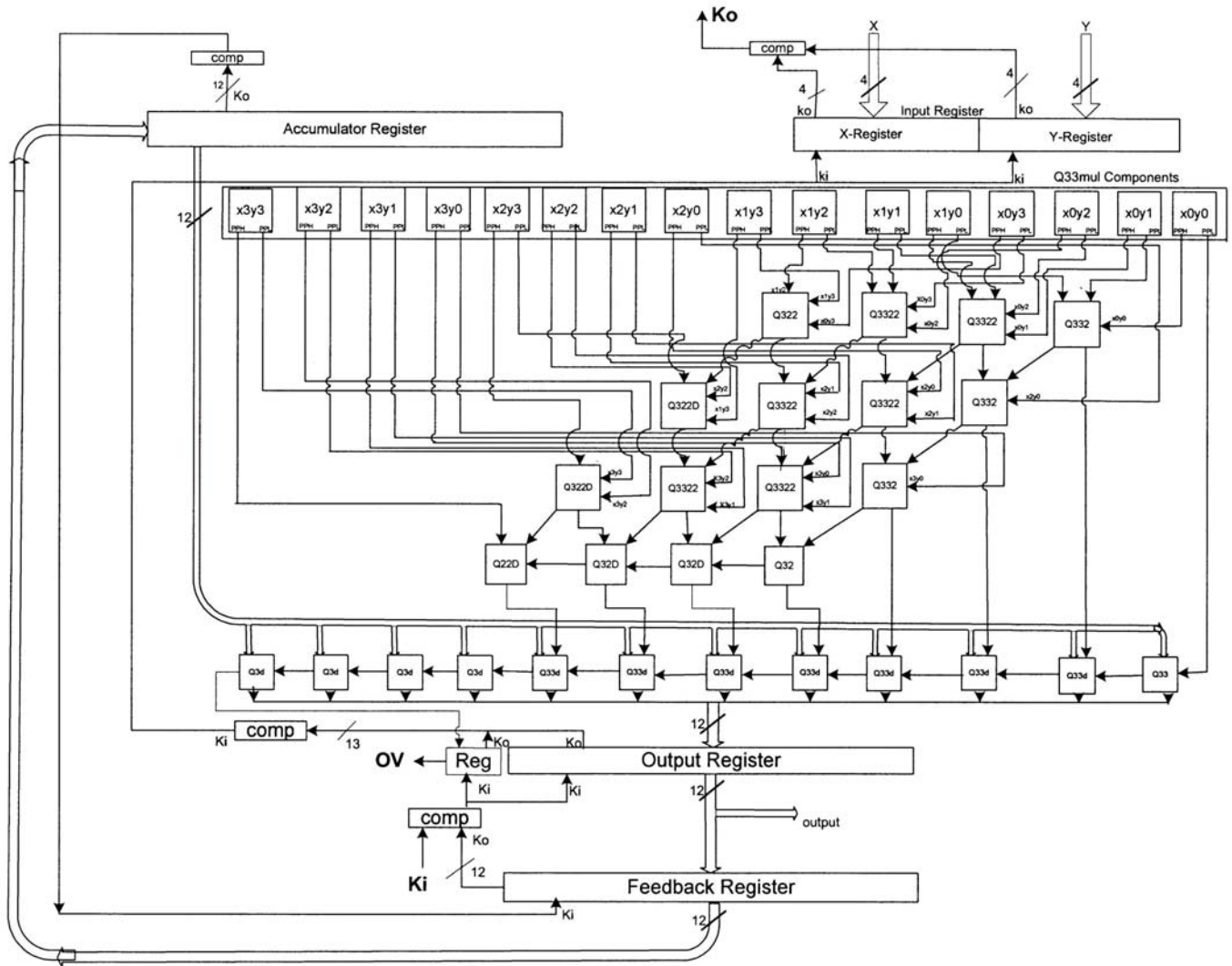


Fig. 3. Block diagram of non-pipelined quad-rail 24+8x8 NCL MAC.

- 3) An array-structured multiplier using quad-rail Carry-Save Adders (CSAs)
- 4) A quad-rail Ripple-Carry Adder (RCA)
- 5) Three 12-signal quad-rail registers for the accumulation feedback loop (i.e., Output Register, Feedback Register, and Accumulator Register)
- 6) Completion logic for the various registers
- 7) A dual-rail register that holds the value of OV, which is asserted for one iteration if the accumulated result cannot be stored in the 12-signal quad-rail output register (i.e., when an overflow occurs).

#### A. Partial Product Generation

The partial products are generated using Q33mul [3], which generates the product of two quad-rail signals. Multiplication of two quad-rail signals results in two quad-rail product signals,  $PPL$  and  $PPH_i$ .  $PPL$  has a range of 0-3; however,  $PPH$  only has a range of 0-2, since the maximum value a quad-rail signal can represent is  $DATA_3$  (i.e.,  $11_2$ ); and therefore the product of two quad-rail signals can be at most  $3 \times 3 = 1001_2$ , which when converted to quad-rail,

$PPH = 10_2 = DATA_2$  and  $PPL = 01_2 = DATA_1$ . Since  $PPH$  has a maximum value of  $DATA_2$ , it can be represented using only 3 wires (i.e., a 3-rail MEAG (Mutually Exclusive Assertion Group)).

#### B. Multiplication

We are using an array-structured CSA-type partial product summation. This results in a regular, repetitive structure and is therefore easier to layout at the physical level. Although other partial product summation methods, such as a Wallace tree using CSAs, are faster, they do not yield a regular, repetitive structure and would therefore be much more difficult to layout.

Various quad-rail adders are used in the MAC for summing the partial products and adding to the accumulator, as detailed below:

- 1) Q3Dadd: This component adds a quad-rail and dual-rail signal, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 2) Q32add: This component adds a quad-rail signal and a 3-rail MEAG, resulting in a quad-rail *sum* and a dual-rail *carry*.

- 3) Q33add: This component adds two quad-rail signals, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 4) Q22Dadd: This component adds two 3-rail MEAGs and a dual-rail signal, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 5) Q32Dadd: This component adds a quad-rail signal, a 3-rail MEAG, and a dual-rail signal, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 6) Q33Dadd: This component adds two quad-rail signals and a dual-rail signal, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 7) Q322add: This component adds a quad-rail signal and two 3-rail MEAGs, resulting in a quad-rail *sum* and a dual-rail *carry*.
- 8) Q332add: This component adds two quad-rail signals and a 3-rail MEAG, resulting in a quad-rail *sum* and a 3-rail MEAG *carry*.
- 9) Q322Dadd: This component adds a quad-rail signal, two 3-rail MEAGs, and a dual-rail signal, resulting in a quad-rail *sum* and a 3-rail MEAG *carry*.
- 10) Q3222add: This component adds a quad-rail signal and three 3-rail MEAGs, resulting in a quad-rail *sum* and a 3-rail MEAG *carry*.
- 11) Q3322add: This component adds two quad-rail signals and two 3-rail MEAGs, resulting in a quad-rail *sum* and a 3-rail MEAG *carry*.

These circuits were designed using the Threshold Combinational Reduction method [4] for NCL combinational logic design, with area optimization being the primary goal.

### III. DESIGN IMPLEMENTATION

#### A. VHDL Implementation

The partial product generation component and various adders were implemented as gate-level structural designs, and were combined with generic registration and completion components [5] to form a hierarchical design of the entire  $24+8 \times 8$  MAC. The complete gate-level structural VHDL version of the MAC was simulated using a VHDL testbench comprised of numerous test cases, and was verified to be functionally correct.

#### B. Transistor-Level Implementation

As explained in Section I, NCL threshold gates are designed with *hysteresis* state-holding capability, such that after the output is asserted, all inputs must be deasserted before the output will be deasserted. Therefore, NCL gates have both *set* and *hold* equations, where the *set* equation determines when the gate will become asserted and the *hold* equation determines when the gate will remain asserted once it has been asserted. The *set* equation determines the gate's functionality as one of the 27 NCL gates, as listed in Table I, whereas the *hold* equation is the same for all NCL gates, and is simply all inputs ORed together. The general equation for an NCL gate with output  $Z$  is:  $Z = \text{set} + (Z \cdot \text{hold})$ , where  $Z$

is the previous output value and  $Z$  is the new value. Take the TH23 gate for example. The *set* equation is  $AB + AC + BC$ , as given in Table I, and the *hold* equation is  $A + B + C$ ; therefore the gate is asserted when at least 2 inputs are asserted and it then remains asserted until all inputs are deasserted.

To implement an NCL gate using CMOS technology, an equation for the complement of  $Z$  is also required, which in general form is:  $Z' = \text{reset} + (Z' \cdot \text{set}')$ , where *reset* is the complement of *hold* (i.e., the complement of each input, ANDed together), such that the gate is deasserted when all inputs are deasserted and remains deasserted while the gate's *set* condition is false. For the TH23 gate, the *reset* equation is  $A'B'C'$  and the simplified *set'* equation is  $A'B' + B'C' + A'C'$ . Directly implementing these equations for  $Z$  and  $Z'$ , after simplification, yields the static transistor-level implementation of an NCL gate, as shown in Fig. 4 for the TH23 gate. This requires the output,  $Z$ , to be feedback as an input to the NMOS and PMOS logic to achieve hysteresis behavior.

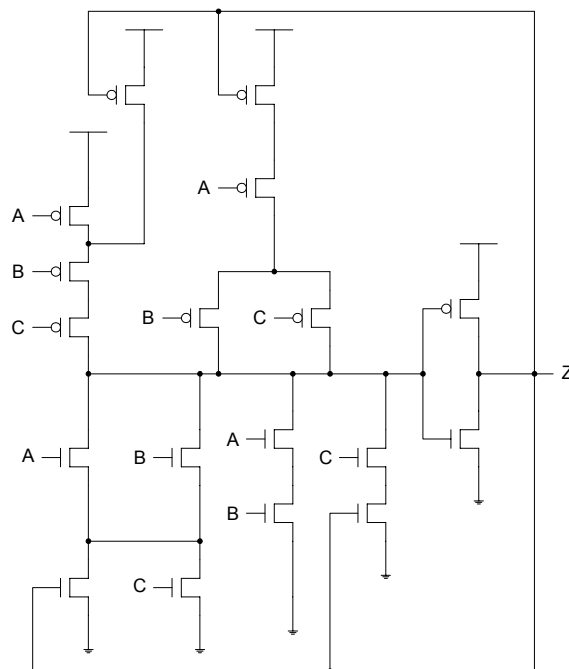


Fig. 4. Static CMOS implementation of a TH23 gate:  $Z = AB + AC + BC$ .

NCL gates can also be implemented in a semi-static fashion, where a weak feedback inverter is used to achieve hysteresis behavior, which only requires the *set* and *reset* equations to be implemented in the NMOS and PMOS logic, respectively. The semi-static TH23 gate is shown in Fig. 5. In general, the semi-static implementation requires fewer transistors, but is slightly slower because of the weak inverter. Note that TH1n gates are simply OR gates and do not require any feedback, such that their static and semi-static implementations are exactly the same.

Transistor-level libraries have been created for both the static and semi-static versions of all the NCL gates used in the design. For the static version, minimum widths were used for

all transistors to maximize gate speed; however, for the semi-static version, larger transistors were required to overcome the weak feedback inverter to obtain proper gate functionality and reduce propagation delay.

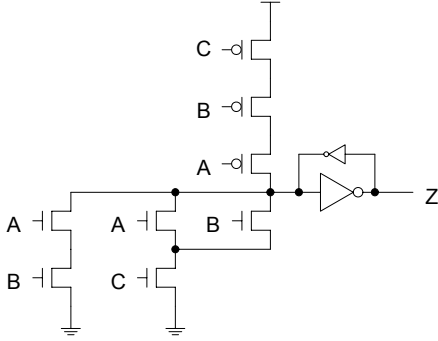


Fig. 5. Semi-static CMOS implementation of a TH23 gate:  $Z = AB + AC + BC$ .

### C. Physical-Level Implementation

Using Mentor Graphics IC Station, a physical-layout was created for both static and semi-static versions of all NCL gates used in the design, with assistance from the Schematic Driven Layout tool. All the NCL gate layouts were checked for functional correctness and Design Rules Check (DRC) errors, and were made into standard cells. Next, the layouts for both static and semi-static versions of all of the MAC components (i.e., partial product generation, various adders, registers, and completion components, as discussed in Section II) were generated using the NCL gate standard cells. Post-layout simulation was then performed to calculate propagation delay and power utilization for all MAC components, using Mentor Graphics Eldo and EZwave tools, in order to compare the static and semi-static versions, as presented in the next section.

The major steps in the physical-level design flow are listed and explained below:

- 1) A Verilog netlist is generated from the gate-level structural VHDL file, using Leonardo Spectrum.
- 2) The Verilog netlist is imported into IC station and the NCL and other appropriate libraries are loaded.
- 3) A floor plan is generated using the Auto Floor Plan option, and the cells are placed in order to optimize layout area.
- 4) The cells are routed using the Auto Route option after enabling Peek on View, so that the tool can recognize the metals used in the standard cells, thus avoiding shorts.
- 5) The complete layout is checked for any unrouted overflows, which are then connected manually.
- 6) I/O ports are then placed.
- 7) Using the Caliber tool, a `pex` netlist file is generated.
- 8) The `pex` netlist file is then modified to a `.CIR` file by adding input test patterns, which is used for Eldo simulation.
- 9) Eldo generates a `.COU` file after simulation, which is used to view and analyze the waveforms using EZwave.

## IV. RESULTS AND COMPARISON

The physical-level simulation results for the static and semi-static MAC components, designed using a 1.8V 0.18 $\mu\text{m}$  TSMC CMOS process, are listed in Tables II and III, respectively. Power is the average power reported by Eldo for an exhaustive test of all input combinations, using 10 ns for both the DATA and NULL wavefront widths. This shows that power dissipation and area is less for the semi-static implementation, as expected, since far fewer transistors are required. However, propagation delay is greater for the semi-static implementation (i.e., 840 vs. 797 on average), since the transistor widths had to be increased due to the weak inverter.

TABLE II  
ANALYSIS OF STATIC MAC COMPONENTS

Component	$T_{PLH}$ (ps)	$T_{PHL}$ (ps)	Area ( $\mu\text{m}^2$ )	Power (W)
Q33mul	235.81	554.14	4.01E+03	5.71E-10
Q3Dadd	222.95	481.41	4.00E+03	6.09E-10
Q32add	226.21	284.45	4.17E+03	5.50E-10
Q33add	442.02	815.15	4.84E+03	9.42E-10
Q22Dadd	1271.04	684.88	8.83E+03	1.37E-09
Q32Dadd	550.89	962.12	1.15E+04	1.92E-09
Q33Dadd	613.29	973.4	1.67E+04	2.71E-09
Q322add	783.51	1173.3	2.41E+04	2.83E-09
Q332add	577.25	1176.6	2.41E+04	4.04E-09
Q322Dadd	738.72	1249.5	3.33E+04	6.68E-09
Q3222add	765.83	1203.5	4.93E+04	1.03E-08
Q3322add	1211.7	1923.7	6.72E+04	1.36E-08

TABLE III  
ANALYSIS OF SEMI-STATIC MAC COMPONENTS

Component	$T_{PLH}$ (ps)	$T_{PHL}$ (ps)	Area ( $\mu\text{m}^2$ )	Power (W)
Q33mul	578.31	641.42	3.73E+03	5.31E-10
Q3Dadd	328.64	441.7	3.74E+03	5.43E-10
Q32add	645.89	690.95	3.01E+03	4.91E-10
Q33add	590.49	658.866	4.62E+03	8.46E-10
Q22Dadd	1939.4	601.65	6.60E+03	1.22E-09
Q32Dadd	645.64	717.64	8.13E+03	1.71E-09
Q33Dadd	739.57	911.28	1.02E+04	2.35E-09
Q322add	528.95	802.59	1.06E+04	2.52E-09
Q332add	800.99	988.77	1.42E+04	3.59E-09
Q322Dadd	1006.8	1243.6	2.42E+04	5.79E-09
Q3222add	957.5	1129.1	3.20E+04	8.90E-09
Q3322add	1230.04	1340.5	4.25E+04	1.17E-08

The overall system-level layout of the MAC has been completed, requiring 0.6994  $\text{mm}^2$  for the static version and 0.5377  $\text{mm}^2$  for the semi-static version. *ADVance MS* (ADMS), an extension of Mentor Graphics Eldo simulator, is currently being utilized to simulate the static and semi-static physical-level designs using a VHDL testbench, as described in [6]. This VHDL-controlled physical-level simulation

method is absolutely necessary for asynchronous circuits because the inputs do not change relative to a periodic clock pulse, but instead change value at various times based on handshaking signals. Power, energy per operation, and average propagation delay are automatically calculated using ADMS, but the system-level results have not been obtained in time for inclusion in this paper.

## V. CONCLUSION

We designed and implemented a non-pipelined unsigned  $24+8\times 8$  quad-rail NCL MAC at all levels of abstraction, from VHDL to layout, using both static and semi-static gates. The gate-level structural VHDL model of the entire system was successfully simulated and verified to be functionally correct. Furthermore, all of the major system components were implemented, simulated, and verified at the transistor-level and physical-level. The full system-level implementation at the physical level was completed for both the static and semi-static versions; however, the system-level power, energy per operation, and average propagation delay results have not been obtained in time for inclusion in this paper, most notably due to many unrouted overflows that needed to be manually connected, which is a major time consuming process.

From the physical-level simulation results of the various MAC components, the semi-static implementation was found to be better in terms of area and power dissipation, whereas the static implementation was faster. The overall system-level design also showed that the semi-static version required less area than the static version.

Future work includes completing the simulation of the system-level layout to obtain the power, energy per operation, and average propagation delay results, and optimizing the design for increased throughput. This can be achieved through pipelining and by merging the last stage of the partial product summation with the addition to the accumulator such that one of the two RCAs can be replaced with a CSA.

## REFERENCES

- [1] 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.
- [2] Gerald E. Sobelman and Karl M. Fant, "CMOS Circuit Design of Threshold Gates with Hysteresis," *IEEE International Symposium on Circuits and Systems (II)*, pp. 61-65, 1998.
- [3] S. K. Bandapati, S. C. Smith, and M. Choi, "Design and Characterization of NULL Convention Self-Timed Multipliers," *IEEE Design and Test of Computers: Special Issue on Clockless VLSI Design*, Vol. 30/6, pp. 26-36, November-December 2003.
- [4] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits," *Elsevier's Integration, the VLSI Journal*, Vol. 37/3, pp. 135-165, August 2004.
- [5] <http://web.umr.edu/~smithsco/VHDL.html> (available March 2007).
- [6] A. Singh and S. C. Smith, "Using a VHDL Testbench for Transistor-Level Simulation and Energy Calculation," *The 2005 International Conference on Computer Design*, pp. 115-121, June 2005.