

# Speedup of NULL convention digital circuits using NULL cycle reduction

S.C. Smith \*

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

Received 19 September 2003; accepted 12 December 2005  
Available online 3 February 2006

## Abstract

A *NULL Cycle Reduction (NCR)* technique is developed to increase the throughput of NULL Convention Logic systems, by reducing the time required to flush complete DATA wavefronts, commonly referred to as the *NULL* cycle. The NCR technique exploits parallelism by partitioning input wavefronts, such that one circuit processes a DATA wavefront, while its duplicate processes a NULL wavefront. A NCR architecture is developed for both dual-rail and quad-rail circuits, using either full-word or bit-wise completion. To illustrate the technique, NCR is applied to case studies of a dual-rail non-pipelined 4-bit  $\times$  4-bit unsigned multiplier using full-word completion, a quad-rail non-pipelined 4-bit  $\times$  4-bit unsigned multiplier using full-word completion, and a dual-rail optimally-pipelined 4-bit  $\times$  4-bit unsigned multiplier using bit-wise completion. The application of NCR yields a speedup of 1.57, 1.55, and 1.34, respectively, over the standalone versions, while maintaining delay-insensitivity. Furthermore, NCR is applied to a single slow stage of two pipelined designs to boost the pipelines' overall throughput by 20% and 26%, respectively.

© 2005 Elsevier B.V. All rights reserved.

*Keywords:* Asynchronous logic design; Delay-insensitive circuits; Self-timed circuits; Dual-rail encoding; Quad-rail encoding

## 1. Introduction

Most multi-rail delay-insensitive logic paradigms employ both a DATA wavefront and a NULL wavefront in order to maintain delay-insensitivity [1–3]. The DATA wavefront realizes circuit functionality, while the NULL wavefront flushes the previous DATA wavefront. The NULL cycle accounts for approximately half of the total cycle time, thus decreasing attainable throughput by a factor of

two. The objective of this paper is to develop and illustrate a technique for reducing the NULL cycle time such that throughput does not depend as heavily on the DATA flush time, while still maintaining delay-insensitivity of the circuit.

The NULL Cycle Reduction technique is an adaptation of a *Wagging FIFO* [4], specifically targeted for the NULL Convention Logic paradigm. A *Ripple FIFO* [4] consists of a number of buffers, or storage elements, in series, such that the first input is loaded into the first buffer. The second input can then be loaded into the first buffer, only after the first input has been moved into the second

\* Tel.: +1 573 341 4232; fax: +1 573 341 4532.  
E-mail address: [smithsco@umr.edu](mailto:smithsco@umr.edu)

buffer. The inputs continue moving to the next adjacent buffer, until the last buffer in the chain is reached; at which point this final buffer produces the output of the FIFO. A *Wagging FIFO* [4] on the other hand, consists of a number of buffers in parallel, with an input Demultiplexer to control the sequential loading of the buffers, and an output Multiplexer to control the sequential outputting of the buffers. The Demultiplexer loads the first input into the first buffer, followed by the second input into the second buffer, and so on. Note that the loading of the second input does not depend on the first input being moved out of the first buffer, as in the case of the Ripple FIFO, thus increasing the FIFO's throughput. The Multiplexer then takes the first output from the first buffer, the second output from the second buffer, and so on.

The NCR technique described herein employs this Wagging technique to increase the throughput of NCL systems by decreasing a circuit's NULL cycle time without affecting its DATA cycle time. Successive input wavefronts are demultiplexed 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.

## 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 [5]. 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 [6].

### 2.1. Delay-insensitivity

NCL uses symbolic completeness of expression [1] 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 [7]. 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, so 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, so 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 [1–3], 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, like [5], differs from the other delay-insensitive paradigms [2,3] in that these other paradigms only utilize one type of state-holding gate, the

*C-element* [6]. 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 [8]. The primary type of threshold gate is the *TH $m$ n gate*, where  $1 \leq m \leq n$ , as depicted in Fig. 1. TH $m$ n 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 $m$ n 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 $m$ n 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.

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 for dual-rail inputs, or DATA0, DATA1, DATA2, and DATA3 for quad-rail inputs. Initially all circuit elements are reset to the NULL state. First, a DATA wavefront is presented to the circuit. Once

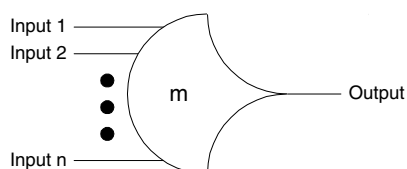


Fig. 1. TH $m$ n threshold gate.

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

The completeness of input criterion [1], which NCL combinational circuits and circuits developed from other delay-insensitive paradigms [2,3] 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.

Furthermore, circuits must also adhere to the completion-completeness criterion [9], which requires that completion signals only be generated such that no two adjacent DATA wavefronts can interact within any combinational component. This condition is only necessary when the bit-wise completion strategy is used with selective input-incomplete components, since it is inherent when using the full-word completion strategy and when using the bit-wise completion strategy with no input-incomplete components [9].

### 2.4. Observability

There is one more condition that must be met to ensure delay-insensitivity for NCL and other delay-insensitive circuits [2,3]. No *orphans* may propagate

through a gate [10]. 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 [5], 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. NULL cycle reduction technique

The technique for reducing the NULL cycle, thus increasing throughput for any delay-insensitive circuit developed according to the paradigms [1–3], is shown in Fig. 2. The NCR architecture in Fig. 2 is specifically designed for dual-rail circuits utilizing full-word completion, where all bits at the output of a registration stage are conjoined to form one completion signal. However, the underlying concept can also be applied to quad-rail logic and bit-wise completion [11], with little modification, as will be discussed in Sections 4 and 5. Bit-wise completion only sends the completion signal from bit  $b$  in register $_i$  back to the bits in register $_{i-1}$  that took part in the calculation of bit  $b$ . This method may therefore require fewer logic levels in the completion circuitry than that of full-word completion, thus increasing throughput.

*Circuit #1* and *Circuit #2* are both dual-rail delay-insensitive combinational circuits utilizing full-word completion, developed from one of the following delay-insensitive paradigms [1–3], with at least an input and output registration stage (additional registration stages may be present, thus further partitioning the combinational circuitry). Both circuits have identical functionality and are both initialized to output NULL and request DATA upon reset. In the case of the NCL paradigm, the combinational functionality can be designed using the Threshold Combinational Reduction method described in [12]; and the resulting circuit can also be pipelined, as described in [11], to further increase throughput. The *Demultiplexer* partitions the input,  $D$ , into two outputs,  $A$  and  $B$ , such that  $A$  receives the first DATA/NULL cycle and  $B$  receives the second DATA/NULL cycle. The input continuously alternates between  $A$  and  $B$ . The *Completion Detection* circuitry detects when either a complete DATA or NULL wavefront has propagated through the *Demultiplexer*, and requests the next NULL or DATA wavefront, respectively. *Sequencer #1* is controlled by the output of the *Completion Detection* circuitry and is used to select either output  $A$  or  $B$  of the *Demultiplexer*. Output  $A$  of the *Demultiplexer* is input to *Circuit #1*, when requested by  $Ki1$ ; and output  $B$  of the *Demultiplexer* is input to *Circuit #2*, when requested by  $Ki2$ . The outputs of *Circuit #1* and *Circuit #2* are allowed to pass through their respec-

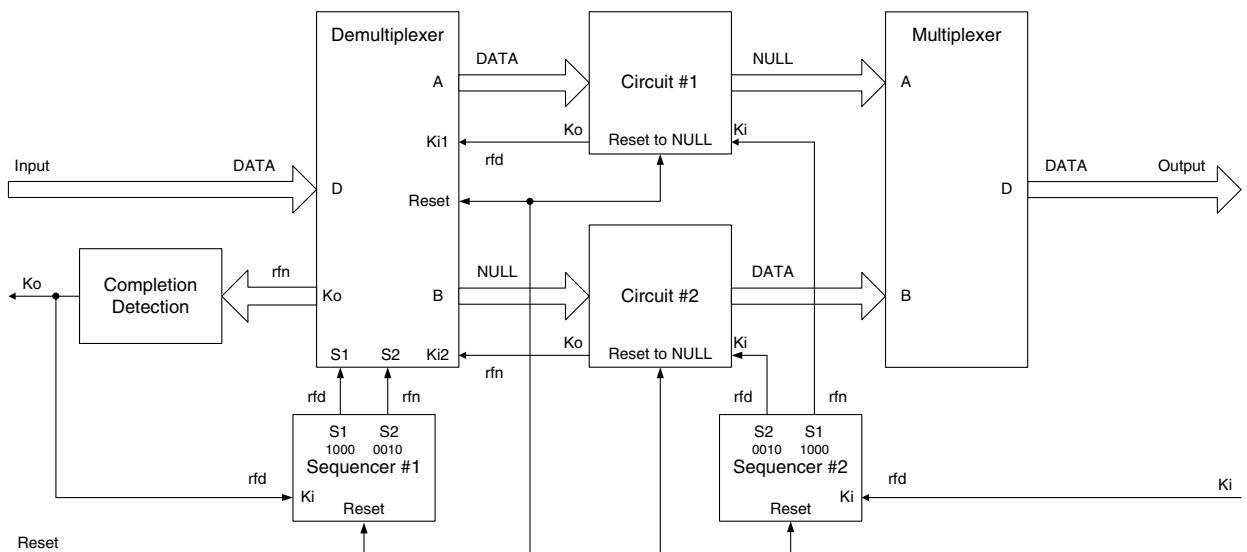


Fig. 2. NCR architecture.

tive output registers, as determined by *Sequencer #2*, which is controlled by the external request, *Ki*. The Multiplexer rejoins the partitioned datapath by passing a DATA input on either *A* or *B* to the output, or asserting NULL on the output when both *A* and *B* are NULL. Fig. 2 shows the state of the system when a DATA wavefront is being input, before its acknowledge flows through the Completion Detection circuitry, and when a DATA wavefront is being output, before it is acknowledged by the receiver.

### 3.1. Demultiplexer

A logic diagram for one bit of the *Demultiplexer* is shown in Fig. 3. It consists of TH33n gates to latch the input and an inverting TH14 gate to generate the completion signal. Upon reset, both *A* and *B* are initialized to NULL. When *S1* is asserted and *Ki1* is *rfd* (request for DATA, i.e. logic 1), a DATA input on *D* will be passed to output *A*. Likewise, when *S2* is asserted and *Ki2* is *rfd*, a DATA input on *D* will be passed to output *B*. *Ko* becomes *rfd* when both *A* and *B* are NULL, and becomes *rfn* (request for NULL, i.e. logic 0) when either *A* or *B*

is DATA. When *A* becomes DATA, it will return to NULL only after *S1* is de-asserted, *Ki1* becomes *rfn*, and the input, *D*, becomes NULL. Likewise, when *B* becomes DATA, it will return to NULL only after *S2* is de-asserted, *Ki2* becomes *rfn*, and the input, *D*, becomes NULL. Therefore, *A* and *B* can never both be DATA since *S1* and *S2* can never be simultaneously asserted and both *A* and *B* must be NULL before the next DATA wavefront is requested. Each bit of the Demultiplexer is the same, and the number of bits is determined by the width of the input datapath (i.e. for an *N*-bit input, *N* 1-bit Demultiplexers are required).

### 3.2. Completion Detection circuitry

The *Completion Detection* circuitry [11] is a tree structure of THnn gates that uses the *N Ko* lines from the Demultiplexer to detect complete DATA or NULL sets, and then request the next NULL or DATA set, respectively. Since the maximum input threshold gate currently supported is the TH44 gate, the number of logic levels in the Completion Detection circuitry for *N Ko* lines is given by  $\lceil \log_4 N \rceil$  [11]. The number of *Ko* lines from the Demultiplexer is also determined by the width of the input datapath (i.e. there are *N Ko* lines for an *N*-bit input).

### 3.3. Sequencer #1

*Sequencer #1* is controlled by the output of the Completion Detection circuitry and is used to select either output *A* or *B* of the Demultiplexer. Upon reset, it selects output *A* to receive the first DATA/NULL cycle, after *Ki* becomes *rfd*. It then selects output *B* to receive the second DATA/NULL cycle. Sequencer #1 continuously alternates the DATA/NULL cycles between outputs *A* and *B*. A logic diagram of Sequencer #1 is shown in Fig. 4. This is a 4-stage single-rail ring structure consisting of TH33n gates to latch the wavefront as it is passed around the loop, and inverters to generate the internal completion signals. It has one token, where a token is defined as a DATA wavefront with corresponding NULL wavefront, and two bubbles, where a bubble is defined as either a DATA or NULL wavefront occupying more than one neighboring stage [13]. When *Ki* becomes *rfd*, the DATA wavefront moves through the two NULL bubbles ahead of it, creating two DATA bubbles in its wake. Likewise, when *Ki* becomes

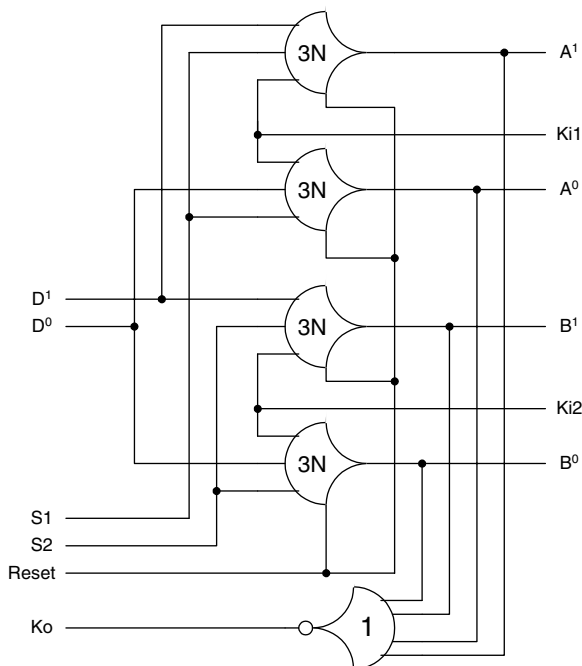


Fig. 3. 1-Bit dual-rail Demultiplexer.

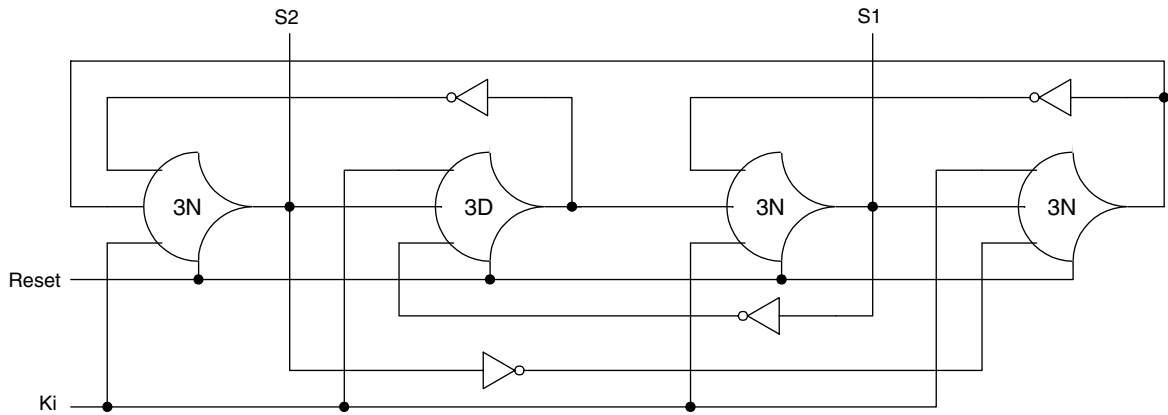


Fig. 4. Sequence generator.

Table 1  
Sequencer output

Cycle #	Initial state	1	2	3	4	5	6	7	8
Reset	1	0	0	0	0	0	0	0	0
Ki	<i>X</i>	1	0	1	0	<b>1</b>	<i>0</i>	<b>1</b>	<i>0</i>
S1	0	1	0	0	0	<b>1</b>	<i>0</i>	<b>0</b>	<i>0</i>
S2	0	0	0	1	0	<b>0</b>	<i>0</i>	<b>1</b>	<i>0</i>

*rfn*, the NULL wavefront moves through the two DATA bubbles ahead of it, creating two NULL bubbles in its wake. The DATA/NULL wavefront restricts the forward propagation of the NULL/DATA wavefront, respectively, for each change of *Ki*, limiting the forward propagation to only the two bubbles. A complete cycle of the Sequencer is shown in boldface and italics in Table 1. The cycle for *S1* is 1000, while the cycle for *S2* is 0010, which corresponds to *Ki* requesting two DATA and NULL wavefronts.

### 3.4. Multiplexer

A logic diagram for one bit of the Multiplexer is shown in Fig. 5. It consists of two TH12 gates (2-input OR gates) that pass a DATA input on either *A* or *B* to the output, *D*, or assert NULL on the output when both *A* and *B* are NULL. The Multiplexer does not require any select signals, since *A* and *B* can never simultaneously be DATA. This mutual exclusion is ensured by Sequencer #2, which controls the outputs of Circuit #1 and Circuit #2. Each bit of the Multiplexer is the same, and the number of bits is determined by the width of the output datapath (i.e. for an *M*-bit output, *M* 1-bit Multiplexers are required).

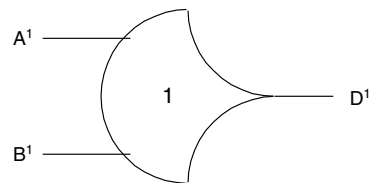
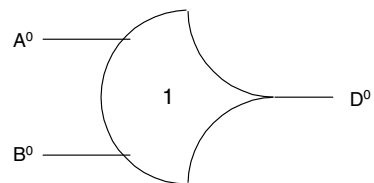


Fig. 5. 1-Bit dual-rail Multiplexer.

### 3.5. Sequencer #2

Sequencer #2 is controlled by the external request, *Ki*, and is used to allow DATA and NULL wavefronts to flow through the output register of Circuit #1 and Circuit #2. Upon reset, it selects Circuit #1 to output the first DATA/NULL cycle, after *Ki* becomes *rfd*. It then selects Circuit #2 to output the second DATA/NULL cycle. Sequencer #2 continuously alternates the DATA/NULL cycles between Circuit #1 and Circuit #2. When *S1* is asserted, DATA will be output from Circuit #1. Likewise, when *S2* is asserted, DATA will be output from Circuit #2. When the output of Circuit #1 becomes DATA, it will return to NULL only after *S1* is de-asserted. Likewise, when the output of Circuit #2 becomes DATA, it will return to NULL

only after  $S2$  is de-asserted. Therefore, Circuit #1 and Circuit #2 can never both output DATA since  $S1$  and  $S2$  can never be simultaneously asserted and the outputs of both circuits must be NULL before the next DATA wavefront is requested by asserting either  $S1$  or  $S2$ . The structure of Sequencer #2 is the same as that of Sequencer #1, shown in Fig. 4 and explained in Section 3.3.

#### 4. NULL cycle reduction technique for quad-rail logic

Applying the NCR method to quad-rail logic circuits requires a few modifications to the dual-rail NCR architecture. Of course, Circuit #1 and Circuit #2 must be designed using quad-rail logic

instead of dual-rail logic. Also, the Demultiplexer, Completion Detection circuitry, and Multiplexer must be redesigned to handle quad-rail signals.

##### 4.1. Quad-rail Demultiplexer

A logic diagram for a single-signal quad-rail Demultiplexer is shown in Fig. 6. It consists of TH33n gates to latch the input and two TH14 gates along with an inverting TH12 gate to generate the completion signal. Its basic operation is the same as that of the 1-bit dual-rail Demultiplexer described in Section 3.1. Each single-signal Demultiplexers is determined by the width of the input

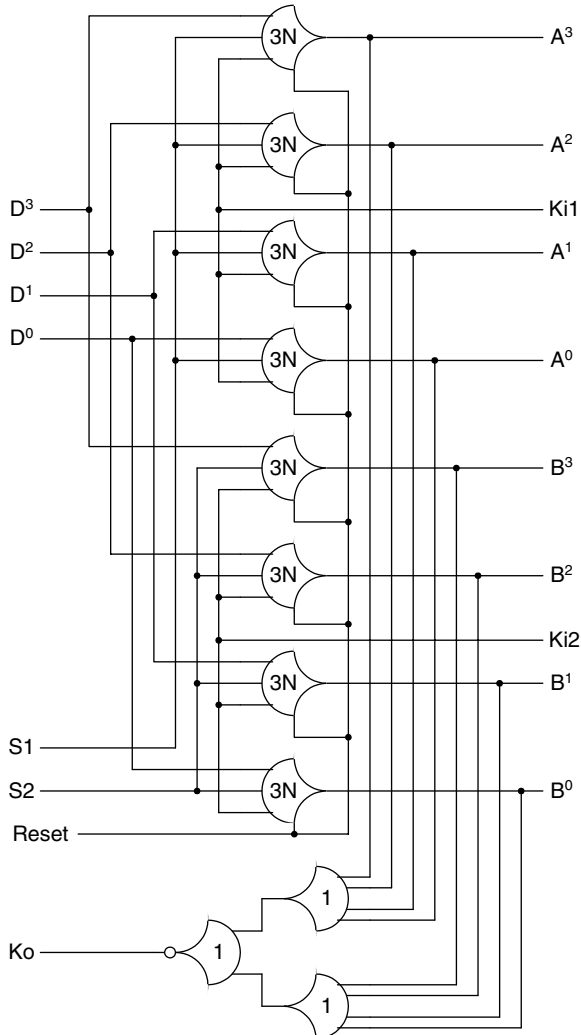


Fig. 6. Single-signal quad-rail Demultiplexer.

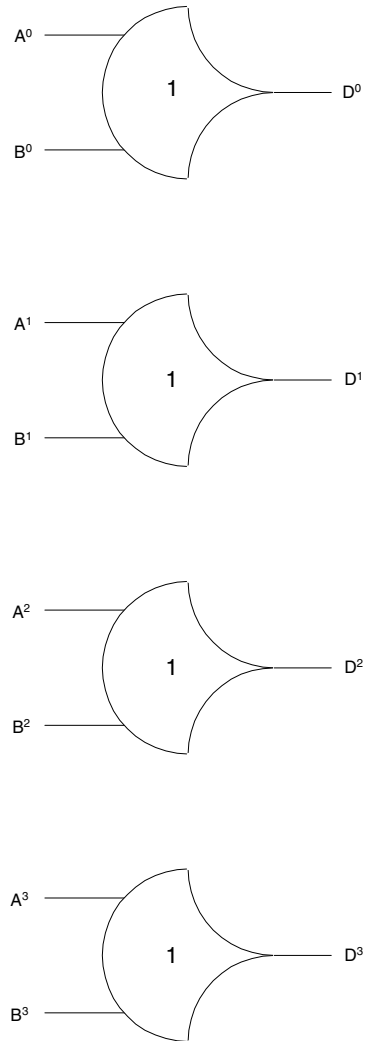


Fig. 7. Single-signal quad-rail Multiplexer.

datapath. Since each quad-rail signal represents 2 bits, for an  $N$ -bit input,  $N/2$  single-signal quad-rail Demultiplexers are required.

#### 4.2. Quad-rail Completion Detection circuitry

The *Completion Detection* circuitry for quad-rail NCR is the same structure as that for dual-rail NCR, as explained in Section 3.2. The only difference is in the number of  $Ko$  lines. Since only  $N/2$  quad-rail Demultiplexers are required for an  $N$ -bit input, as explained in Section 4.1, there will only be  $N/2$   $Ko$  lines input to the quad-rail Completion Detection circuitry.

#### 4.3. Quad-rail Multiplexer

A logic diagram for a single-signal quad-rail *Multiplexer*, consisting of four TH12 gates, is shown in Fig. 7. Its basic operation is the same as that of the dual-rail Multiplexer described in Section 3.4. Each single-signal Multiplexer is the same, and the number of Multiplexers is determined by the width of the output datapath. Since each quad-rail signal represents 2 bits, for an  $M$ -bit output,  $M/2$  single-signal quad-rail Multiplexers are required.

### 5. NULL cycle reduction technique for bit-wise completion

To apply the NCR method to circuits utilizing bit-wise completion requires a slight modification to either the dual-rail or quad-rail NCR architectures. A circuit with an  $N$ -bit input and an  $M$ -bit output utilizing bit-wise completion requires  $N$   $Ko$  outputs and  $M$   $Ki$  inputs for dual-rail logic, or requires  $N/2$   $Ko$  outputs and  $M/2$   $Ki$  inputs for quad-rail logic; since each dual-rail or quad-rail input/output signal is requested/acknowledged independently. Therefore, in order to retain the independent requesting/acknowledging, each Demultiplexer/Circuit #1 and Circuit #2 must be independently controlled. This is achieved by replicating the Sequencer #1/Sequencer #2 circuits, such that each single-signal Demultiplexer/Circuit #1 and Circuit #2 output has its own corresponding Sequencer. Each Sequencer #1 will be controlled by its corresponding Demultiplexer's  $Ko$  line, and in turn will control that specific Demultiplexer's input signal flow. Likewise, each Sequencer #2 will be controlled by its corresponding  $Ki$  line, and will control the flow of its corresponding Circuit #1 and Circuit #2 outputs.

### 6. Higher order NULL cycle reduction

Since putting two duplicate circuits in parallel increased throughput, it would follow that putting more than two duplicate circuits in parallel would further increase throughput. This is not the case. In fact, placing more than two duplicate circuits in parallel actually decreases throughput, versus the two-Circuit NCR technique, as will be shown in Section 7.5. This is because the NCR technique overlaps DATA and NULL wavefront processing. While Circuit #1 is processing NULL wavefront <sub>$i$</sub> , Circuit #2 may begin processing DATA wavefront <sub>$i+1$</sub> . However, if there were more Circuits in parallel, Circuit #3 could not process NULL wavefront <sub>$i+1$</sub> , while Circuit #2 processes DATA wavefront <sub>$i+1$</sub> , due to delay-insensitivity, which requires the NULL wavefront to flush its corresponding DATA wavefront, meaning that the Circuit which processes DATA wavefront <sub>$j$</sub>  must also process NULL wavefront <sub>$j$</sub> . In the above example, Circuit #2 must therefore process NULL wavefront <sub>$i+1$</sub> , at which time Circuit #1 is free to begin processing DATA wavefront <sub>$i+2$</sub> . Hence, an additional circuit in parallel is not needed.

### 7. Simulation results

Case studies of a dual-rail non-pipelined 4-bit  $\times$  4-bit unsigned multiplier using full-word completion, a quad-rail non-pipelined 4-bit  $\times$  4-bit unsigned multiplier using full-word completion, and a dual-rail optimally-pipelined 4-bit  $\times$  4-bit unsigned multiplier using bit-wise completion have been evaluated to assess the impact of the NCR technique on throughput. The specifications for the multipliers are to perform an unsigned multiply of two 4-bit input vectors,  $X$  and  $Y$ , and output their 8-bit product,  $P$ . A delay-insensitive interface with request and acknowledge signals, labeled  $Ki$  and  $Ko$ , respectively, is required for requesting and acknowledging DATA and NULL wavefronts. NCL was chosen as the delay-insensitive paradigm for circuit development; and the technology being simulated is a 0.5  $\mu\text{m}$  CMOS process operating at 3.3 V.

#### 7.1. Dual-rail non-pipelined multiplier with full-word completion

The logic diagram of this multiplier may be viewed in [11]. From Mentor Graphics simulation

it was determined that the standalone version of the dual-rail non-pipelined 4-bit  $\times$  4-bit multiplier had an average DATA-to-DATA cycle time ( $T_{DD}$ ) of 9.21 ns, with approximately equal DATA and NULL cycles. When the NCR technique was applied to this design, the NULL cycle was reduced to approximately 1/4 of the DATA cycle. This resulted in an overall  $T_{DD}$  of only 5.88 ns, which corresponds to a 57% increase in throughput. The average throughput was calculated as the arithmetic mean of the throughputs corresponding to all 256 possible pairs of input operands.

### 7.2. Quad-rail non-pipelined multiplier with full-word completion

The logic diagram of this multiplier may be viewed in [14]. The input vectors,  $X$  and  $Y$ , are only two signals each and the output vector,  $P$ , is only comprised of 4 signals (i.e.  $X(1:0)$ ,  $Y(1:0)$ , and  $P(3:0)$ ). This is because each quad-rail signal represents two bits. From Mentor Graphics simulation it was determined that the standalone version of the quad-rail non-pipelined 4-bit  $\times$  4-bit multiplier had a  $T_{DD}$  of 9.89 ns. When the NCR technique was applied to this design,  $T_{DD}$  was reduced to only 6.39 ns, which corresponds to a 55% increase in throughput.

### 7.3. Dual-rail optimally-pipelined multiplier with bit-wise completion

The logic diagram of this multiplier may be viewed in [11]. Since bit-wise completion is used, there are eight  $Ko/Ki$  lines, one for each input/output bit, respectively. From Mentor Graphics simulation it was determined that the standalone version of the dual-rail optimally-pipelined 4-bit  $\times$  4-bit multiplier utilizing bit-wise completion had a  $T_{DD}$  of 3.84 ns. When the NCR technique was applied to this design,  $T_{DD}$  was reduced to only 2.87 ns, which corresponds to a 34% increase in throughput. This increase in throughput was less than for the dual-rail and quad-rail models with full-word completion, because the initial cycle time

of the bit-wise design without NCR was much less than for the other two models (3.84 ns versus 9.21 ns and 9.92 ns), which lessened the impact of the NULL cycle time reduction. For all three cases the NULL cycle time was reduced to approximately 1/4 of the DATA cycle time.

### 7.4. Application of NCR to only one slow pipeline stage

The previous examples duplicated the entire circuit; however, it is not necessary to duplicate the entire circuit when applying the NCR technique. Rather, its benefits can be obtained without doubling area and power requirements by applying it to selective portions of a circuit, which cannot be more finely pipelined due to the completeness of input criterion (i.e. all inputs must be present before all outputs can be generated) [11]. This input-completeness criterion prohibits circuits from being arbitrarily divided for pipelining (as can be done in clocked Boolean circuits), thus possibly resulting in unbalanced stage delays.

If NCR was applied to stage $_i$  to boost throughput, both stage $_{i-1}$  and stage $_{i+1}$  may have to be non-functional stages to realize the full increase due to the adjacent DATA propagation delays in determining throughput, as explained in [11]. A non-functional stage can be easily added by inserting an additional asynchronous register. Thus, throughput of a pipelined design with a small number of slow stages can be readily boosted with relatively little cost by using NCR.

To illustrate this point, NCR was applied to only a single stage of the pipeline shown in Fig. 8. Multiplier #1 and Multiplier #3 are both 2-stage unsigned multipliers with a worse-case stage delay of 5 gate delays. Multiplier #2 is the non-pipelined unsigned multiplier used in the first case study, described in Section 7.1, and consists of 10 gate delays. Therefore, the 10 gate delays of Multiplier #2 is much longer than the 5 gate delays per stage of the other multipliers, making Multiplier #2 a good candidate for NULL Cycle Reduction. Without NCR, the pipeline of Fig. 8 operates with  $T_{DD} = 9.02$  ns;

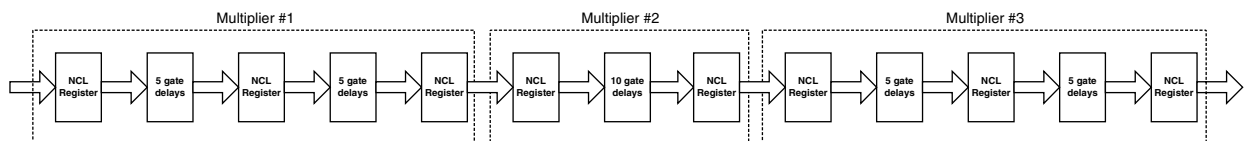


Fig. 8. Pipeline with one slow stage.

however, with NCR only applied to Multiplier #2,  $T_{DD}$  is decreased to 7.51 ns, a speedup of 1.20. Hence, applying NCR to only slow stages in a pipeline can boost throughput for the pipeline as a whole. Note that additional registration was not needed to form non-functional stages around the NCR stage, since these non-functional stages already existed when the multipliers were connected to form the pipeline of Fig. 8, since each multiplier contains both an input and output register.

Another example arises from the 4-bit  $\times$  4-bit quad-rail multiplier discussed in Section 7.2 and depicted in [14]. The optimally pipelined multiplier utilizes bit-wise completion and consists of 4 stages; where Stages 1, 3, and 4 have a worst-case combinational delay of 2 gates and Stage 2 has a worst-case combinational delay of 3 gates. Stages 1, 2, and 3 have a completion delay of 1 gate, while Stage 4 has zero completion delay. Stage 2 cannot be further reduced without violating the input-completeness criterion. When applying NCR to only Stage 2,  $T_{DD}$  is reduced from 6.22 ns to 4.93 ns, a speedup of 1.26.

An alternate solution to increasing the throughput of the 4  $\times$  4 quad-rail multiplier is to redesign the carry-save adders, such that their worst-case delay is only 2 gates. This requires adding at most two quad-rail signals per adder, instead of three quad-rail signals as before, resulting in additional stages and latency. This second method of summing the partial products would not be used for a non-pipelined design since it introduces added critical path delay. Optimally pipelining this alternate design results in a circuit with 5 stages, utilizing bit-wise completion, with a worst-case combinational delay of 2 gates and a completion delay of 1 gate for Stages 1–4, and a worst-case combinational delay of 1 gate and zero completion delay for Stage 5. This alternate design results in approximately the same average cycle time as the previous,  $T_{DD} = 4.94$  ns, and would actually be preferable due to its smaller area. However, this shows that applying NCR to only a slow stage can achieve the same throughput as re-designing the entire circuit.

As the number of stages in a pipeline increase, the application of NCR to only a few stages has less of an impact on area. Furthermore, while the 4  $\times$  4 quad-rail multiplier could be redesigned to achieve a maximal combinational stage delay of 2 gates, this may not be possible for other circuits. A good example is a dual-rail NCL Booth2 multiplier, which

requires 3 gate delays to generate the partial products in the first stage, but only 2 gate delays to add the partial products in each subsequent stage. Therefore the first stage will be the bottleneck when optimally pipelining the design. Thus applying NCR to only the first stage will increase the entire pipeline's throughput without significantly increasing area, when multiplying large word width operands.

### 7.5. Application of higher order NULL cycle reduction

There are two approaches to placing additional Circuits in parallel to create a higher order NCR architecture. The first consists of redesigning the Demultiplexer, Multiplexer, and Sequencers, to support additional Circuits in parallel. The second consists of replacing Circuit #1 and Circuit #2 with their equivalent NULL Cycle Reduced architectures, such that there are four Circuits operating in parallel, which does not require the redesign of any components.

The first approach was designed and simulated for four parallel Circuits, using the dual-rail non-pipelined multiplier with full-word completion as the test circuit. This first new NCR architecture had a  $T_{DD}$  of 5.98 ns, which is an increase in throughput over the stand alone multiplier (9.21 ns), but not as fast as the 5.88 ns of the original two-Circuit NCR architecture, described in Section 7.1. The reason for this is the added delay in the redesigned Demultiplexer and Multiplexer. The second approach of increasing parallelism by replacing the two Circuits with their NULL Cycle Reduced equivalents was also simulated using the same test circuit as above. This second new NCR architecture had a  $T_{DD}$  of 7.12 ns, which again is an increase in throughput over the stand alone multiplier, but not as fast as the 5.88 ns of the original two-Circuit NCR architecture. The reason for this is the extra overhead for two levels of NULL Cycle Reduction, consisting of two demultiplexers and two multiplexers in series. Thus, higher order NULL Cycle Reduction is shown to not further increase throughput over the original two-circuit NCR technique, as theorized in Section 6.

## 8. Conclusion

The NCR method of partitioning delay-insensitive systems into two concurrent paths such that

Table 2  
Summary of simulation results

	Reference section	NCR application	Without NCR $T_{DD}$ (ns)	With NCR $T_{DD}$ (ns)	Speedup
Dual-rail full-word non-pipelined multiplier	7.1	Entire circuit	9.21	5.88	1.57
Quad-rail full-word non-pipelined multiplier	7.2	Entire circuit	9.89	6.39	1.55
Dual-rail bit-wise pipelined multiplier	7.3	Entire circuit	3.84	2.87	1.34
Dual-rail full-word 3-multiplier pipeline	7.4	Only Multiplier #2	9.02	7.51	1.20
Quad-rail bit-wise pipelined multiplier	7.4	Only Stage 2	6.22	4.93	1.26
Alternate quad-rail bit-wise pipelined multiplier	7.4	N/A	4.94	N/A	N/A

one circuit processes a DATA wavefront, while its duplicate processes a NULL wavefront can significantly increase throughput. A number of 4-bit  $\times$  4-bit multiplier case studies indicate significant speedup over their respective standalone designs, while two case studies where NCR was applied to only a slow stage of a pipeline also resulted in a significant increase in throughput, as summarized in Table 2. Therefore, the benefits of NCR can be obtained without doubling area and power requirements by applying it to selective portions of the circuit, which cannot be more finely pipelined due to the completeness of input criterion. Thus, throughput of a pipelined design with a small number of slow stages can be readily boosted without a significant increase in area by using NCR. NCR can also be used to increase the throughput of a feedback loop, which cannot be increased by any other means, as demonstrated in the redesign of the  $72 + 32 \times 32$  MAC in [15]. Furthermore, it was shown that the two-Circuit NCR architecture developed herein was superior to other NCR architectures employing more parallelism, due to increased overhead (area) without additional increased throughput.

In this paper NCR was applied to circuits designed using the NCL paradigm. However, NCR is readily applicable to other multi-rail delay-insensitive paradigms [2,3] as well, since all of the NCR components (Demultiplexer, Completion Detection, Multiplexer, and Sequencer) are comprised solely of C-elements, NOR gates, OR gates, and invertors, all of which are primitives for any of these delay-insensitive paradigms [1–3].

### Acknowledgements

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

### References

- [1] K.M. Fant, S.A. Brandt, NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis, in: International Conference on Application Specific Systems, Architectures, and Processors, 1996, pp. 261–273.
- [2] C.L. Seitz, System timing, in: Introduction to VLSI Systems, Addison-Wesley, 1980, pp. 218–262.
- [3] I. David, R. Ginosar, M. Yoeli, An efficient implementation of boolean functions as self-timed circuits, IEEE Transactions on Computers 41 (1) (1992) 2–10.
- [4] C.H. Kees van Berkel, M. Rem, R. Saeijs, VLSI programming, in: IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1998, pp. 152–156.
- [5] A.J. Martin, Programming in VLSI, in: Development in Concurrency and Communication, Addison-Wesley, 1990, pp. 1–64.
- [6] D.E. Muller, Asynchronous logics and application to information processing, in: Switching Theory in Space Technology, Stanford University Press, 1963, pp. 289–297.
- [7] T. Verhoff, Delay-insensitive codes—an overview, Distributed Computing 3 (1988) 1–8.
- [8] G.E. Sobelman, K.M. Fant, CMOS circuit design of threshold gates with hysteresis, in: IEEE International Symposium on Circuits and Systems (II), 1998, pp. 61–65.
- [9] S.C. Smith, Completion-completeness for NULL convention digital circuits utilizing the bit-wise completion strategy, in: The 2003 International Conference on VLSI, 2003, pp. 143–149.
- [10] A. Kondratyev, L. Neukom, O. Roig, A. Taubin, K. Fant, Checking delay-insensitivity:  $10^4$  gates and beyond, in: Eighth International Symposium on Asynchronous Circuits and Systems, 2002, pp. 137–145.
- [11] S.C. Smith, R.F. DeMara, M. Hagedorn, D. Ferguson, Delay-insensitive gate-level pipelining, Integration, The VLSI Journal 30 (2) (2001) 103–131.
- [12] S.C. Smith, R.F. DeMara, J.S. Yuan, D. Ferguson, D. Lamb, Optimization of NULL convention self-timed circuits, Integration, The VLSI Journal 37 (3) (2004) 135–165.
- [13] J. Sparso, J. Staunstrup, Design and performance analysis of delay insensitive multi-ring structures, in: Twenty-Sixth Hawaii International Conference on System Sciences, vol. 1, 1993, pp. 349–358.
- [14] S.K. Bandapati, S.C. Smith, M. Choi, Design and characterization of NULL convention self-timed multipliers, IEEE

Design and Test special issue on Clockless VLSI Design, 2003.

- [15] S.C. Smith, Development of a large word-width high-speed asynchronous multiply and accumulate unit, Integration, The VLSI Journal 39/1 (2005) 12–28.



**Scott C. Smith** is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Missouri—Rolla. He received B.S. degrees in both Electrical Engineering and Computer Engineering from the University of Missouri—Columbia in May of 1996. After a six-month co-op with Motorola's Paging Products Group in Boynton Beach, Florida, he returned to Missouri and received a M.S. degree

in Electrical Engineering with an emphasis in Computers and

Digital Systems from the University of Missouri—Columbia in May of 1998. To further his education, he then returned to Florida to pursue a Ph.D. degree in Computer Engineering with an emphasis in Computer Architecture and Digital Systems from the University of Central Florida, Orlando. He received the Ph.D. degree in May of 2001.

He has authored 7 refereed journal publications and 13 conference papers. He has also received 3 US patents and 2 European patents. He is a member of numerous professional societies including Sigma Xi, Eta Kappa Nu, Tau Beta Pi, IEEE, and ASEE. His research interests include Computer Architecture, Logic Design, Embedded System Design, and VLSI.