Speedup of Delay-Insensitive Digital Systems Using NULL Cycle Reduction

S. C. Smith, R. F. DeMara, J. S. Yuan
School of Electrical Engineering and Computer Science
Box 162450, University of Central Florida
Orlando, FL 32816-2450
Tel: 407-823-5916 / Fax: 407-823-5385
E-mail: demara@mail.ucf.edu

M. Hagedorn and D. Ferguson
Theseus Logic, Inc.
3501 Quadrangle Blvd., Suite 100
Orlando, FL 32817
Tel: 407-380-9008 / Fax: 407-380-9509
E-mail: dferguson@theseus.com

Abstract

A *NULL Cycle Reduction (NCR)* technique is developed to increase the throughput of delay-insensitive digital systems. NCR reduces the time required to flush complete DATA wavefronts, commonly referred to as the *NULL* or *Empty* 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. To illustrate the technique, NCR is applied to a case study of a dual-rail non-pipelined 4-bit by 4-bit unsigned multiplier, yielding a speedup of 1.61 over the standalone version, while maintaining delay-insensitivity. NCR is also applied to a single slow stage of a pipeline to boost the pipeline's overall throughput by 21%.

Introduction

Most multi-rail delay-insensitive logic paradigms employ both a DATA wavefront and a NULL wavefront in order to maintain delay-insensitivity [1, 2, 3, 4, 5]. 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, yet still maintains delay-insensitivity.

Many architectures and algorithms employ the well-known divide and conquer strategy. This technique partitions a problem into smaller sub-problems that can be solved simultaneously, then merges their outputs to construct the solution to the original problem, thus reducing computation time. In asynchronous circuits this technique has been used by Molnar et. al. to increase the throughput of a *micropipelined* First-In-First-Out (FIFO) data buffer [6] and by Ebergen to decrease the latency of a GasP FIFO by *Squaring* it [7]. However, neither of these applications are delay-insensitive, since both micropipelines and GasP circuits rely heavily on delay analysis.

The NCR technique described herein employs the divide and conquer strategy to increase the throughput of delay-insensitive systems by decreasing a circuit's NULL cycle time without affecting its DATA cycle time. Successive input wavefronts are partitioned 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 delay-insensitive methodology used is NULL Convention Logic (NCL) [1].

Overview of NCL

NCL offers a delay-insensitive logic paradigm where control is inherent with each datum. It follows the so-called "weak conditions" of Seitz's delay-insensitive signaling scheme [3]. As with other delay-insensitive logic methods, the NCL paradigm assumes that forks in wires are *isochronic*.

A. *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 the time of evaluation. In particular, dual-rail signals with three logic states (NULL, DATA0, and DATA1) can be used to rid NCL of the implicit time reference of Boolean circuits and achieve symbolic completeness of expression. A dual-rail signal named Z has two rails denoted Z^0 and Z^1. The DATA0 state of NCL ($Z^0 = 1$, $Z^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state of NCL ($Z^0 = 0$, $Z^1 = 1$) corresponds to a Boolean logic 1, and the NULL state of NCL ($Z^0 = 0$, $Z^1 = 0$) corresponds to the empty set, meaning that the result is not yet available. The two rails of a dual-rail NCL signal are mutually exclusive, so both rails can never be asserted simultaneously; this state is defined as an illegal state.

All NCL systems have at least two register stages, one at both the input and output. These two register stages interact through their K_i and K_o lines to prevent DATA set_i from overwriting DATA set_{i-1} by ensuring that the two DATA sets are always separated by a NULL set.

B. *Logic Gates*

NCL uses *threshold gates* for its basic logic gates. The primary type of threshold gate is the *THmn gate*, where $1 \le m \le n$, as depicted in Figure 1. THmn 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. In a THmn gate, each of the n inputs is connected to the rounded portion of the gate; the output emanates from the pointed end of the gate; and the gate's threshold value, m, is written inside of the gate. NCL threshold gates may also include a reset to initialize the output. Resetable 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.

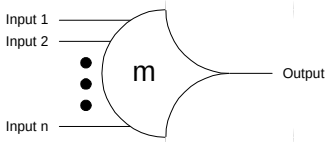

Figure 1. THmn threshold gate.

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.

C. Completeness of Input

The completeness of input criterion [1], which NCL combinational circuits 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 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.

D. Observability

There is one more condition that must be met in order for NCL to retain delay-insensitivity. No *orphans* may propagate through a gate. 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, as long as they are not allowed to cross a gate boundary. This *observability* condition 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.

NULL Cycle Reduction Technique

The technique for reducing the NULL cycle, thus increasing throughput for any NCL system is shown in Figure 2. *NCL Circuit #1* and *NCL Circuit #2* have identical functionality and are both initialized to output NULL and request DATA upon reset. Both have an asynchronous NCL register at the input and output. The combinational functionality can be designed using the TCR method described in [8]. These circuits may also be pipelined as described in [9], 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*

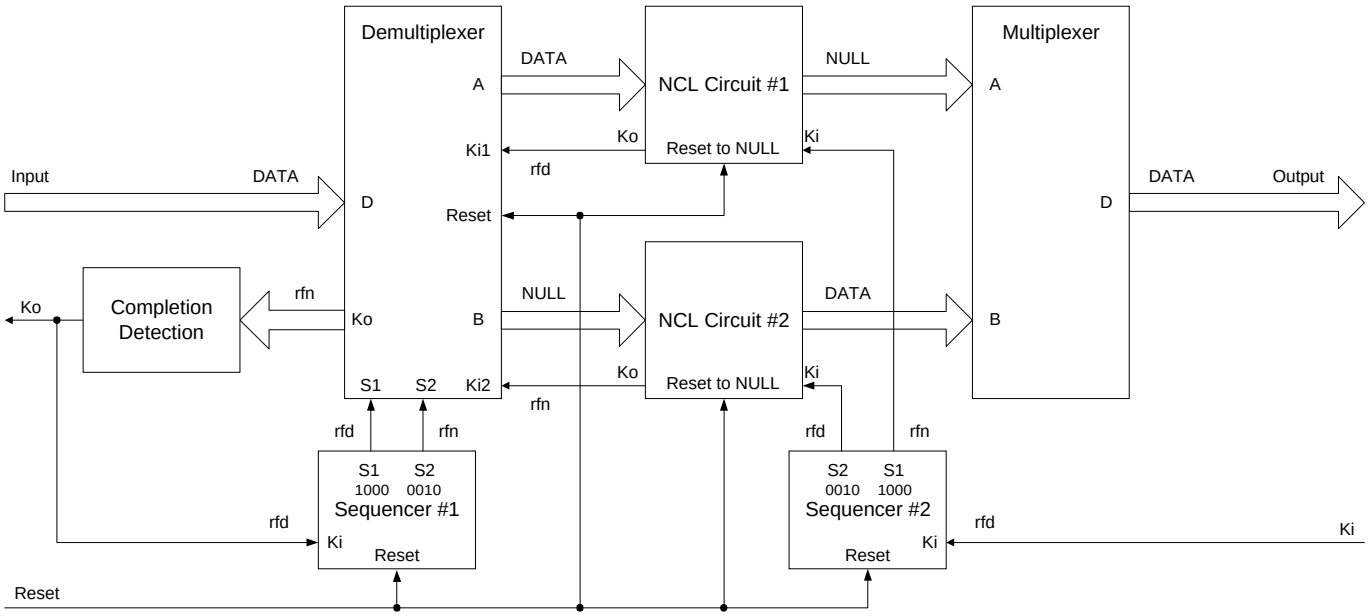

Figure 2. NCR architecture.

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 NCL Circuit #1 when requested by *Ki1*; and output *B* of the Demultiplexer is input to NCL Circuit #2 when requested by *Ki2*. The outputs of NCL Circuit #1 and NCL Circuit #2 are allowed to pass through their respective 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. Figure 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.

A. Demultiplexer

A logic diagram for one bit of the *Demultiplexer* is shown in Figure 3. Upon reset both *A* and *B* are initialized to NULL. When *S1* is asserted and *Ki1* is *rfd* (request for DATA), 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) 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.

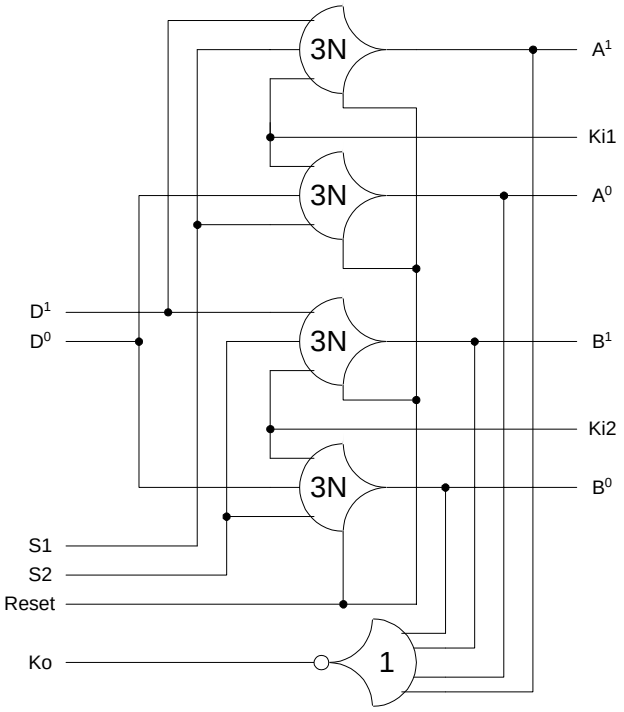

Figure 3. 1-Bit Demultiplexer.

B. Completion Detection Circuitry

The *Completion Detection* circuitry shown in Figure 4 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 $\lceil Log_4\ N \rceil$. The number of *Ko* lines from the Demultiplexer is also determined by the width of the input datapath.

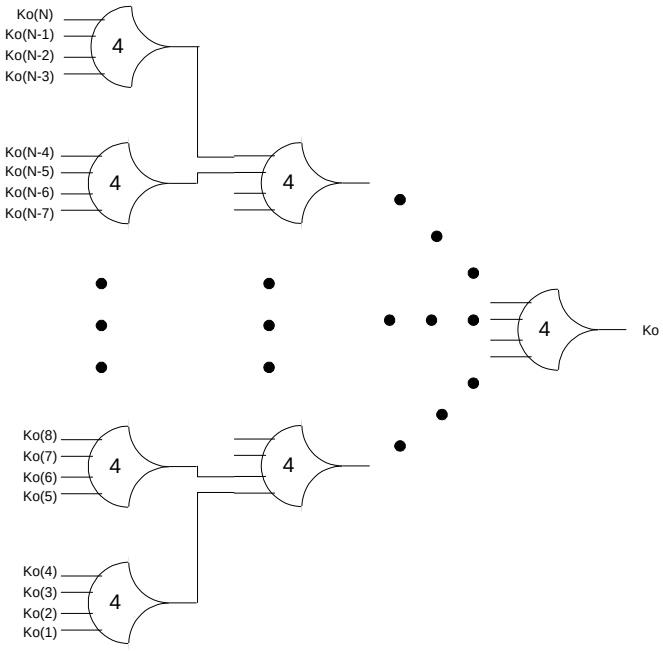

Figure 4. Completion Detection circuitry.

C. 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 Figure 5. This is a 4-stage single-rail ring structure with 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 [10]. 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 *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 I. The cycle for *S1* is 1000, while the cycle for *S2* is 0010.

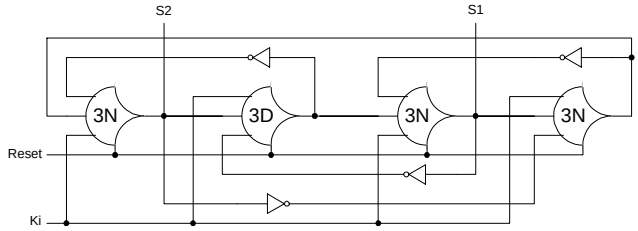

Figure 5. Sequence generator.

Table I. Sequencer output.

Cycle #	Initial State	1	2	3	4	5	6	7	8
Reset	1	0	0	0	0	0	0	0	0
Ki	X	1	0	1	0	***1***	***0***	***1***	***0***
S1	0	1	0	0	0	***1***	***0***	***0***	***0***
S2	0	0	0	1	0	***0***	***0***	***1***	***0***

D. Multiplexer

A logic diagram for one bit of the *Multiplexer* is shown in Figure 6. It simply consists of two 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 NCL Circuit #1 and NCL Circuit #2. Each bit of the Multiplexer is the same, and the number of bits is determined by the width of the output datapath.

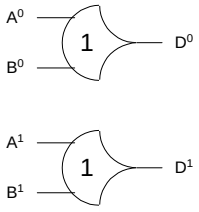

Figure 6. 1-Bit Multiplexer.

E. 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 NCL Circuit #1 and NCL Circuit #2. Upon reset it selects NCL Circuit #1 to output the first DATA/NULL cycle, after *Ki* becomes *rfd*. It then selects NCL Circuit #2 to receive the second DATA/NULL cycle. Sequencer #2 continuously alternates the DATA/NULL cycles between NCL Circuit #1 and NCL Circuit #2. When *S1* is asserted, DATA will be output from NCL Circuit #1. Likewise, when *S2* is asserted, DATA will be output from NCL Circuit #2. When the output of NCL Circuit #1 becomes DATA, it will return to NULL only after *S1* is de-asserted. Likewise, when the output of NCL Circuit #2 becomes DATA, it will return to NULL only after *S2* is de-asserted. Therefore, NCL Circuit #1 and NCL 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 Figure 5.

Simulation Results

A case study of a dual-rail non-pipelined 4-bit by 4-bit multiplier, shown in Figure 7, has been evaluated to assess the impact of the NCR technique on throughput. The specifications for this multiplier were simply to perform an unsigned multiply of the two 4-bit input vectors, *X* and *Y*, and then output their 8-bit product, *S*. A full NCL interface with request and acknowledge signals labeled *Ki* and *Ko*, respectively, is provided for requesting and acknowledging complete DATA and NULL wavefronts. The technology used is a 0.25μm CMOS process operating at 3.3V. From Synopsys simulation it was determined that the standalone version of the dual-rail non-pipelined 4-bit by 4-bit multiplier had an average DATA-to-DATA cycle time of 8.75 ns with approximately equal DATA and NULL cycles. When the NCR technique was applied to this design, the NULL cycle was reduced to approximately ¼ of the DATA cycle. This resulted in an overall average DATA-to-DATA cycle time of only 5.43 ns, which corresponds to a 61% increase in throughput. Values for average throughput were obtained from the arithmetic mean of throughputs corresponding to all 256 possible pairs of input operands.

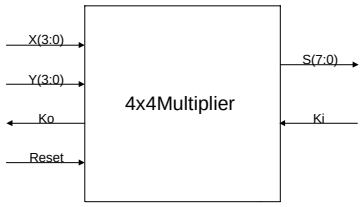

Figure 7. 4×4 multiplier block diagram.

The previous example 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 pipelined more finely due to the completeness of input criterion. 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 [9]. 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 Figure 8. Multiplier #1 and Multiplier #3 are both 2-stage unsigned multipliers with a worse-case stage delay of 5 gate delays, and have the same

interface as depicted in Figure 7. Multiplier #2 is the non-pipelined unsigned multiplier used in the first case study, 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 Figure 8 operates with T_{DD} = 8.42 ns; however, with NCR only applied to Multiplier #2, T_{DD} is decreased to 6.96 ns, a speedup of 1.21. Henceforth, 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 Figure 8, since each multiplier contains both an input and output register.

Conclusion

The NCR method of partitioning delay-insensitive systems into two concurrent paths such that one circuit processes a DATA wavefront, while its duplicate processes a NULL wavefront can significantly increase throughput. A 4-bit by 4-bit multiplier case study indicates a speedup of 1.61 over the standalone design, while a case study where NCR was applied to only a slow stage of a pipeline resulted in a speedup of 1.21. 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 pipelined more finely due to the completeness of input criterion. Thus, throughput of a pipelined design with a small number of slow stages can be readily boosted with relatively little cost by using NCR. However, NCR will not increase throughput of a feedback loop with only one token, since this structure eliminates the ability to simultaneously process two consecutive DATA wavefronts, because $wavefront_{i+1}$ at least partially consists of the output from $wavefront_i$.

In this paper NCR was applied to a dual-rail NCL design utilizing full-word completion. However, it can also be applied to a quad-rail NCL design, by modifying the Demultiplexer and the Multiplexer to handle quad-rail signals, or to a design utilizing bit-wise completion by modifying the Demultiplexer and the Completion Detection circuitry. Furthermore, this technique could also be applied to other delay-insensitive methodologies [2, 3, 4, 5] as well.

References

[1] Karl M. Fant and Scott 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] T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technology Bulletin*, Sept. 1986.

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

[4] N. P. Singh, *A Design Methodology for Self-Timed Systems*, Master's Thesis, MIT/LCS/TR-258, Laboratory for Computer Science, MIT, 1981.

[5] Ilana David, Ran Ginosar, and Michael Yoeli, "An Efficient Implementation of Boolean Functions as Self-Timed Circuits," *IEEE Transactions on Computers*, Vol. 41, No. 1, pp. 2-10, 1992.

[6] C. E. Molnar, I. W. Jones, W. S. Coates, J. K. Lexau, S. M. Fairbanks, and I. E. Sutherland, "Two FIFO Ring Performance Experiments," *Proceedings of the IEEE*, Vol. 87, No. 2, pp. 297-307, 1999.

[7] J. Ebergen, "Squaring the FIFO in GasP," *Seventh International Symposium on Asynchronous Circuits and Systems*, pp. 194-205, 2001.

[8] S. C. Smith, R. F. DeMara, D. Ferguson, and D. Lamb, "Optimization of NULL Convention Self-Timed Circuits," submitted to *IEEE Transactions on Computers*, July 2000 (revision submitted February 2001).

[9] S. C. Smith, R. F. DeMara, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," submitted to *Integration, the VLSI Journal*, November 2000.

[10] Jens Sparso and Jorgen Staunstrup, "Design and Performance Analysis of Delay Insensitive Multi-Ring Structures," *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, Vol. 1, pp. 349-358, 1993.

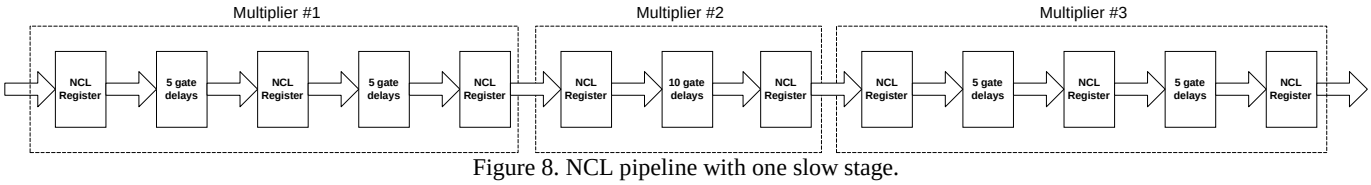

Figure 8. NCL pipeline with one slow stage.