

# Speedup of Self-Timed Digital Systems Using Early Completion

Scott C. Smith

University of Missouri – Rolla, Department of Electrical and Computer Engineering  
123 Emerson Electric Co. Hall, 1870 Miner Circle, Rolla, MO 65409  
Phone: (573) 341-4232, Fax: (573) 341-4532, E-mail: [smithsco@umr.edu](mailto:smithsco@umr.edu)

## Abstract

*An Early Completion technique is developed to significantly increase the throughput of NULL Convention self-timed digital systems without impacting latency or compromising their self-timed nature. Early Completion performs the completion detection for registration stage<sub>i</sub> at the input of the register, instead of at the output of the register, as in standard NULL Convention Logic. This method requires that the single-rail completion signal from registration stage<sub>i+1</sub>,  $Ko_{i+1}$ , be used as an additional input to the completion detection circuitry for registration stage<sub>i</sub>, to maintain self-timed operation. However, Early Completion does necessitate an assumption of equipotential regions, introducing a few easily satisfiable timing assumptions, thus making the design potentially more delay-sensitive. To illustrate the technique, Early Completion is applied to a case study of the optimally pipelined 4-bit by 4-bit unsigned multiplier utilizing full-word completion, presented in [1], where a speedup of 1.21 is achieved while self-timed operation is maintained and latency remains unchanged.*

## 1. Introduction

In this paper a new completion strategy for NULL Convention Logic (NCL) [2] is presented, which increases throughput of NCL systems without degrading latency or compromising their self-timed operation. The increased performance is due to a reduction of the impact of the handshaking overhead. The technique is based on anticipation, similar to that of a carry lookahead adder, in which future events are predicted, thus allowing overlapped computation time.

Most multi-rail delay-insensitive logic paradigms [3, 4, 5, 6, and 7] consist of combinational logic, latches or registration, and completion detection. As the standard, completion detection is performed at the output of

registration stage<sub>i</sub>; and the completion signal is fed back as an input to registration stage<sub>i-1</sub>. Assuming that each combinational logic block has the same delay and that each completion detection unit has the same delay, the basic cycle is as follows: NULL flows through combinational logic block<sub>i</sub> ( $N_i$ ), NULL flows through completion detection unit<sub>i</sub> ( $RFD_i$ ), DATA flows through combinational logic block<sub>i</sub> ( $D_i$ ), DATA flows through completion detection unit<sub>i</sub> ( $RFN_i$ ). These events are mutually exclusive, such that no two events overlap in time. However, with the application of Early Completion,  $N_i$  and  $RFD_i$  partially overlap in time and  $D_i$  and  $RFN_i$  partially overlap in time, thus decreasing the overall cycle time and increasing throughput.

The Early Completion technique presented herein is similar to the Early Done technique presented in the Previous Work section. Both increase throughput by moving the completion detection circuitry forward in the pipeline such that some sort of prediction can be preformed, allowing the overlapping of previously mutually exclusive events. The contribution of this paper is the application of the concept of early completion to multi-rail delay-insensitive paradigms, specifically NCL, where the solutions to the specific obstacles of this task are presented, the impact on self-timed operation is analyzed, and a test circuit is simulated to give an analytical measure of the technique's effectiveness.

## 2. Previous Work

In [8] an *Early Done* technique was developed to speedup Williams' PS0 pipeline [9], resulting in a 79% increase in throughput when applied to a 4-bit FIFO. Williams' PS0 pipeline is based on precharge logic and consists of stages composed of a dual-rail function block and a completion detector. The completion component detects the completion of every functional evaluation and precharge at the output of its corresponding function block. The output of the completion detector for stage<sub>i</sub> is connected to the precharge/evaluate control input of stage<sub>i-1</sub>, as shown in Figure 1. The basic cycle for a PS0 pipeline is:  $T_{PS0} = (3 \cdot t_{Eval}) + (2 \cdot t_{CD}) + t_{Prech}$ , assuming that each stage has the same functional evaluation delay

---

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

( $t_{Eval}$ ), precharge delay ( $t_{Prech}$ ), and completion delay ( $t_{CD}$ ) [8].

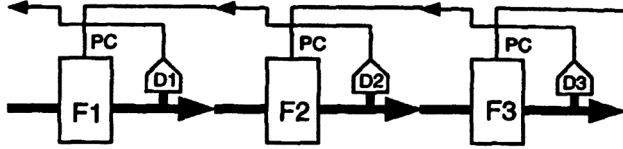


Figure 1. Block diagram of a PS0 pipeline [8]

The Early Done technique modifies the PS0 pipeline by moving the completion detectors in front of their corresponding functional blocks rather than after them. This allows the current stage to signal the previous stage when it is about to evaluate or precharge, instead of after the action has been completed; thus allowing the completion detection signal to be produced in parallel with the precharge or evaluation of its corresponding functional block, instead of after it. This new LP2/2 pipeline requires the completion detectors to be modified such that they require an additional input, the stage's PC control input, as shown in Figure 2. The basic cycle for a LP2/2 pipeline is:  $T_{LP2/2} = (2 \cdot t_{Eval}) + (2 \cdot t_{CD})$ , which is  $t_{Eval} + t_{Prech}$  shorter than that of the PS0 pipeline [8]. Furthermore, this throughput optimization does not impact latency, since the forward path is unchanged.

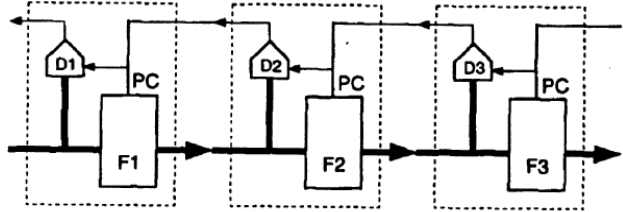


Figure 2. Block diagram of a LP2/2 pipeline [8]

### 3. 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* [10, 11].

#### 3.1 Delay-Insensitivity

NCL uses symbolic completeness of expression [2] 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 request and acknowledge lines,  $K_i$  and  $K_o$ , respectively, to prevent DATA wavefront $_i$  from overwriting DATA wavefront $_{i-1}$  by ensuring that the two DATA wavefronts are always separated by a NULL wavefront.

#### 3.2 Logic Gates

NCL uses *threshold gates* for its basic logic gates. The primary type of threshold gate is the *TH $mn$  gate*, where  $1 \leq m \leq n$ , as depicted in Figure 3. TH $mn$  gates have  $n$  inputs. At least  $m$  of the  $n$  inputs must be asserted before the output will become asserted. Because NCL threshold gates are designed with *hysteresis*, all asserted inputs must be de-asserted before the output will be de-asserted. Hysteresis ensures a complete transition of inputs back to NULL before asserting the output associated with the next wavefront of input data. 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. A TH $nn$  gate is equivalent to an  $n$ -input C-element [12], while a TH1 $n$  gate is equivalent to an OR gate.

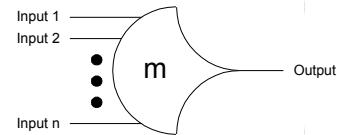


Figure 3. TH $mn$  threshold gate

By employing threshold gates for each logic rail, NCL is able to determine the output status without referencing time. Inputs are partitioned into two separate wavefronts, the NULL wavefront and the DATA wavefront. The NULL wavefront consists of all inputs to a circuit being NULL, while the DATA wavefront refers to all inputs being DATA, some combination of DATA0 and DATA1. Initially all circuit elements are reset to the NULL state. First, a DATA wavefront is presented to the circuit. Once all of the outputs of the circuit transition to DATA, the NULL wavefront is presented to the circuit. Once all of

the outputs of the circuit transition to NULL, the next DATA wavefront is presented to the circuit. This DATA/NULL cycle continues repeatedly. As soon as all outputs of the circuit are DATA, the circuit's result is valid. The NULL wavefront then transitions all of these DATA outputs back to NULL. When they transition back to DATA again, the next output is available. This period is referred to as the DATA-to-DATA cycle time, denoted as  $T_{DD}$ , and has an analogous role to the clock period in a synchronous system.

### 3.3 Completeness of Input

The completeness of input criterion [2], 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.

### 3.4 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.

### 4. Early Completion Technique

The standard NCL pipeline is shown in Figure 4, where each registration stage consists of multiple single bit registers, shown in Figure 5, and the gate-level structure of the completion components is shown in Figure 6. The Combinational Circuit is an input-complete, fully observable functional block, designed using Threshold Combinational Reduction, as described in [13].  $TD$  denotes the time when any DATA bits are propagating through the combinational circuit,  $TN$  denotes the time when any NULL bits are propagating through the combinational circuit,  $TRFD$  is the time associated with the request for DATA generation, and

$TRFN$  is the time associated with the request for NULL generation.

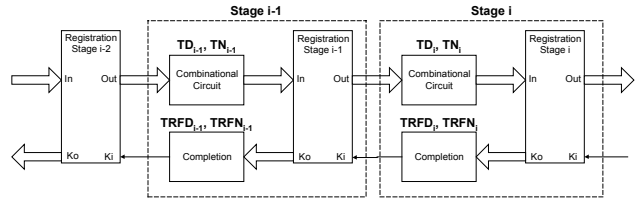
As described in [1], the worse-case throughput for an N-stage NCL pipeline is based on the following three equations:  $TRFD_{i-1} + TD_{i-1} + TD_i + TRFN_i$ ,  $TRFN_{i-1} + TN_{i-1} + TN_i + TRFD_i$ , and  $TRFD_i + TD_i + TRFN_i + TN_i$ , corresponding to the case of adjacent DATA propagation delays and request times, the case of adjacent NULL propagation delays and request times, and the case of NULL and DATA propagation delays and request times for a single registration stage, respectively. The worse-case cycle time for the entire pipeline is then calculated by finding the maximum of these three equations for every adjacent stage pair in the pipeline, as listed in the following algorithm:

```

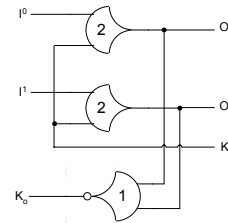
max_cycle_time = TRFD1 + TD1 + TRFN1 + TN1
for (i = 2 to N) loop
    temp_cycle_time = MAX(TRFDi + TDi + TRFNi + TNi,
                          TRFDi-1 + TDi-1 + TDi + TRFNi,
                          TRFNi-1 + TNi-1 + TNi + TRFDi)
    if (temp_cycle_time > max_cycle_time) then
        max_cycle_time = temp_cycle_time
    end if
end loop
worst_case_throughput = 1 / max_cycle_time

```

**Algorithm 1. Calculation of worst-case throughput for an N-stage NCL pipeline [1]**



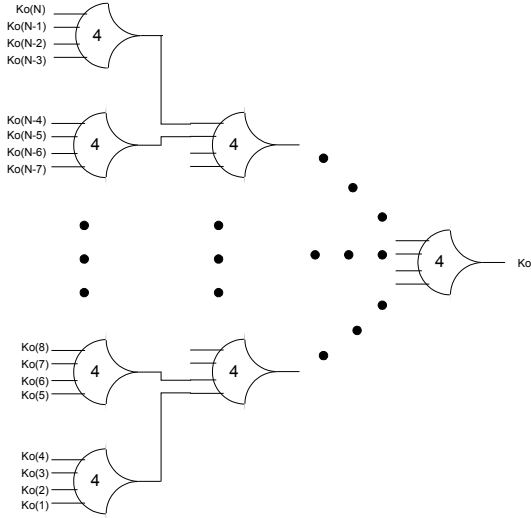
**Figure 4. Standard NCL pipeline**



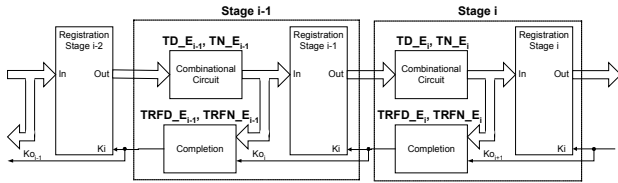
**Figure 5. Single-bit dual-rail NCL register [2]**

Notice that in the standard NCL pipeline the completion detection is performed at the output of the registration stage. The inputs to completion component<sub>i</sub> are the  $Ko$  outputs from registration stage<sub>i</sub>. On the other hand, a NCL pipeline utilizing Early Completion, shown in Figure 7, also uses the inputs to registration stage<sub>i</sub> as the inputs to completion component<sub>i</sub>; however this

necessitates the completion component for Early Completion to require an additional input, the completion signal from stage<sub>*i*+1</sub>,  $Ko_{i+1}$ , in order to maintain self-timed operation. The registration stage is also slightly modified, by removing the inverting TH12 gate for each single-bit register, since the  $Ko$  output is no longer required.



**Figure 6. N-bit completion component**



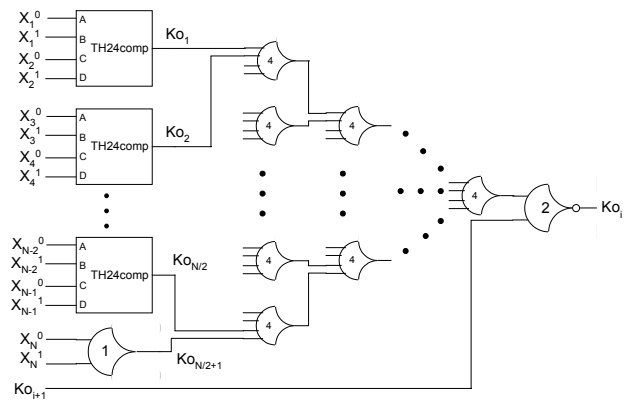
**Figure 7. NCL pipeline with Early Completion**

Early Completion allows the completion evaluation for stage<sub>*i*</sub> to begin before all bits have propagated through combinational circuit<sub>*i*</sub> and have been latched by registration stage<sub>*i*</sub>. Therefore,  $TRFN\_E_i$  overlaps with  $TD\_E_i$  and  $TRFD\_E_i$  overlaps with  $TN\_E_i$  such that  $TRFN\_E_i + TD\_E_i < TRFN_i + TD_i$  and  $TRFD\_E_i + TN\_E_i < TRFD_i + TN_i$ . However,  $TRFD_{i-1} + TD_{i-1}$  and  $TRFN_{i-1} + TN_{i-1}$  still do not overlap, such that  $TRFD_{i-1} + TD_{i-1} \cong TRFD_{i-1} + TD_{i-1}$  and  $TRFN_{i-1} + TN_{i-1} \cong TRFN_{i-1} + TN_{i-1}$ . By examining the equations for determining the worst-case throughput for an N-stage NCL pipeline,  $TRFD_i + TD_i + TRFN_i + TN_i$ ,  $TRFD_{i-1} + TD_{i-1} + TD_i + TRFN_i$ , and  $TRFN_{i-1} + TN_{i-1} + TN_i + TRFD_i$ , it can be seen that each equation contains at least one of the previously noted sums:  $TRFN_i + TD_i$  and  $TRFD_i + TN_i$ . Therefore, the cycle time for a NCL pipeline using Early Completion must be less than one using standard completion since  $TRFD_{i-1} + TD_{i-1} \cong TRFD_{i-1} + TD_{i-1}$ ,  $TRFN_{i-1} + TN_{i-1} \cong TRFN_{i-1} + TN_{i-1}$ ,

$TRFN\_E_i + TD\_E_i < TRFN_i + TD_i$ , and  $TRFD\_E_i + TN\_E_i < TRFD_i + TN_i$ , as previously shown. Furthermore, Early Completion does not impact latency, since the forward path is unchanged.

The completion component for stages 1 through M-1 for an M-stage NCL pipeline utilizing Early Completion is shown in Figure 8, where the TH24comp gate has the following functionality:  $(A + B) \cdot (C + D)$  [13]. This completion component is for a datapath of  $N$  bits, where  $N$  is an odd number. For a datapath with an even number of bits, the TH12 gate would not be required, so there would only be  $\frac{N}{2}$  intermediate  $Ko$  signals. Also, note that

the final gate, the inverting TH22 gate, can be incorporated into the tree structure of TH44 gates, depending on the width of the datapath, to reduce the number of logic levels by one. This component requests DATA when all inputs to register<sub>*i*</sub> are NULL and the request from register<sub>*i*+1</sub>,  $Ko_{i+1}$ , is requesting NULL (*rfn*). It requests NULL when all inputs to register<sub>*i*</sub> are DATA and  $Ko_{i+1}$  is requesting DATA (*rfd*). The completion component for the final stage, stage<sub>*M*</sub>, is slightly different. The inverting TH22 gate is no longer inverted; instead the final gate of the tree structure of TH44 gates is inverted. This causes the component to request DATA when the input to register<sub>*M*</sub> is NULL and the external request input line,  $Ki$ , is *rfd*; and to request NULL when the input to register<sub>*M*</sub> is DATA and  $Ki$  is *rfn*. This variation in the completion component for the last stage is required since  $Ki$  changes to *rfn* as soon as the output is DATA, and changes to *rfd* as soon as the output is NULL, to simulate an infinitely fast external interface. An alternative is to use the standard completion component, shown in Figure 6, for the last stage. However, this later approach produces a system with reduced throughput compared to that when the modified Early Completion component is used for stage<sub>*M*</sub>.



**Figure 8. Completion component for Early Completion**

## 5. Evaluation of Self-Timed Operation

Standard NCL systems are self-timed, assuming that wire forks are isochronic. However, the application of Early Completion changes the fundamental structure of the NCL handshaking system, thus necessitating the self-timed issue to be revisited. In the most delay-sensitive case,  $Ko_{i+1}$  and  $Ko_{i+2}$  are both *rfd* and all bits at the input of register<sub>*i*</sub> change to DATA within a very short period of time. The DATA wavefront at the input of register<sub>*i*</sub> would flow through register<sub>*i*</sub>, followed by combinational logic block<sub>*i+1*</sub>, and finally completion component<sub>*i+1*</sub>, in order to transition  $Ko_{i+1}$  to *rfn*. Simultaneously, the DATA wavefront at the input of register<sub>*i*</sub> would flow through completion component<sub>*i*</sub> in order to transition  $Ko_i$  to *rfn*. Therefore, in order for the system to function incorrectly, the DATA wavefront would have to travel through a set of TH22 gates (register<sub>*i*</sub>), combinational logic block<sub>*i+1*</sub>, and completion component<sub>*i+1*</sub>, before the same signal traveled through only completion component<sub>*i*</sub>. These paths are shown in boldface in Figure 9. Since the first path is normally much longer, the delay is well known and the system remains self-timed, through the assumption of equipotential regions [3]. This same argument can be made for the NULL wavefront, by replacing DATA with NULL, *rfn* with *rfd*, and *rfd* with *rfn*, yielding the same result. For the special case of a FIFO, the combinational logic delay would be zero, but the delay through completion component<sub>*i*</sub> and completion component<sub>*i+1*</sub> would be identical, so the above argument would still hold. For the generalized case, completion component<sub>*i*</sub> and completion component<sub>*i+1*</sub> normally have about the same delay, within one or two gate delays, such that the above analysis holds true.

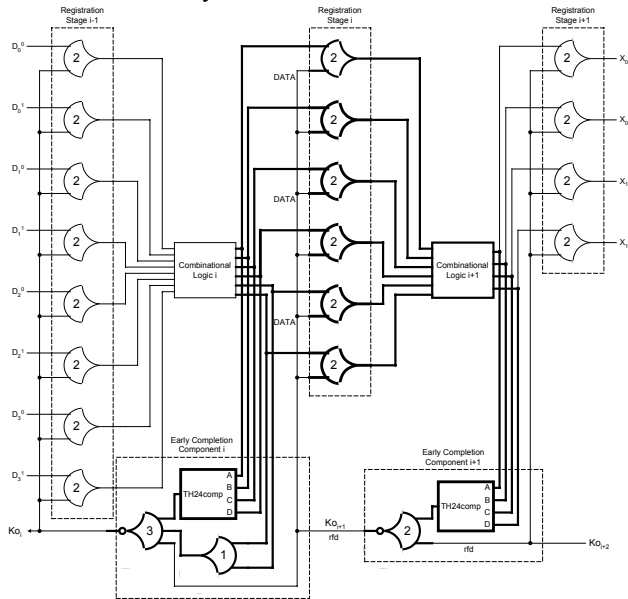


Figure 9. Delay-sensitive scenario #1

The other delay-sensitive scenario introduced by Early Completion is when  $Ko_{i+1}$  changes to *rfd* when all inputs to register<sub>*i*</sub> are already DATA and all inputs to register<sub>*i-1*</sub> are NULL. In this case the *rfd* has to pass through one gate (at the least an inverting TH22 gate) in order to transition  $Ko_i$  to *rfn*. Once  $Ko_i$  is *rfn*, the NULL wavefront at the input of register<sub>*i-1*</sub> can flow through the register's TH22 gates and overwrite the previous DATA wavefront at the input of register<sub>*i*</sub>. Simultaneously, the DATA wavefront at the input of register<sub>*i*</sub> has to pass through only one TH22 gate to be latched at the output of register<sub>*i*</sub>. Therefore, in order for the system to function incorrectly, a signal would have to travel through both an inverting and non-inverting TH22 gate before the same signal travels through only a single TH22 gate. Since the path through the two gates is obviously longer than the path through a single gate, the delays are well known and the system remains self-timed, through the assumption of equipotential regions [3]. This same argument can be made for the NULL wavefront by replacing DATA with NULL, *rfn* with *rfd*, and *rfd* with *rfn*, yielding the same result. Note that this example assumes that there is no combinational logic delay, as would be the case in a FIFO. For the generalized case the delay-sensitivity would be even less, since the path through an inverting TH22 gate, a TH22 gate, and combinational logic would have to be faster than the path through a single TH22 gate, as depicted in boldface in Figure 10, in order to adversely affect self-timed operation.

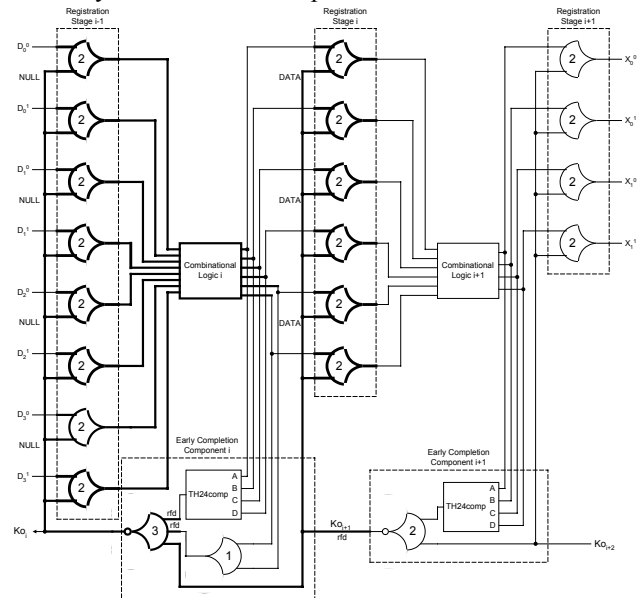


Figure 10. Delay-sensitive scenario #2

## 6. Initialization

In standard NCL, the system is initialized using a global reset to set the output of each register to either

DATA or NULL. Since each completion component only uses the outputs of its corresponding register as inputs, its output will become initialized in constant time after the global reset is applied, without requiring any reset circuitry itself. However, if this same initialization procedure was used with Early Completion, the reset time would be  $O(N)$ , where  $N$  is the number of stages in the pipeline, since  $K_i$  would have to trickle through each completion component in order to initialize  $K_o$ , because the early completion component for stage <sub>$i$</sub>  not only uses the inputs of register <sub>$i$</sub>  as its inputs; but it also requires the request output from stage <sub>$i+1$</sub>  as an input. To remedy this situation, the reset signal is also applied to the final gate of each early completion component such that completion component <sub>$i$</sub>  is reset to *rfd* when register <sub>$i$</sub>  is reset to NULL or completion component <sub>$i$</sub>  is reset to *rfn* when register <sub>$i$</sub>  is reset to DATA. This revised initialization strategy retains the constant time initialization of standard NCL, and is actually faster than standard NCL initialization. However, more reset circuitry is required, which could also be applied to standard NCL to attain the same reduced initialization time.

## 7. Results

This paper does not use a FIFO as an example system, since bit-wise completion [1] would obviously outperform any other completion strategy for an NCL FIFO. Instead, the optimally pipelined 4-bit by 4-bit unsigned multiplier utilizing full-word completion, presented in [1], was chosen as the case study. A functional block diagram of the 4×4 multiplier is shown in Figure 11, where *I* denotes an incomplete AND function [1], *C* denotes a complete AND function [1], *HA* denotes a half-adder [1], *FA* denotes a full-adder [1], *COMP* denotes a completion component, as shown in Figure 6, and *GEN\_S7* denotes a specialized component to produce the most significant bit of the result [1].

To assess the effectiveness of the Early Completion technique, both the 4×4 multiplier utilizing standard completion and Early Completion were simulated using Mentor Graphics, a commercial design tool. The Mentor Graphics technology library is based on Spice simulations of static 0.25  $\mu\text{m}$  CMOS gates, operating at 3.3V. The two systems were exhaustively tested and their average throughput calculated. The throughput for the multiplier using the standard completion technique was determined to be  $0.209 \text{ ns}^{-1}$  [1], while the application of Early Completion produced a throughput of  $0.256 \text{ ns}^{-1}$ , resulting in a speedup of 1.21.

## 8. Conclusions

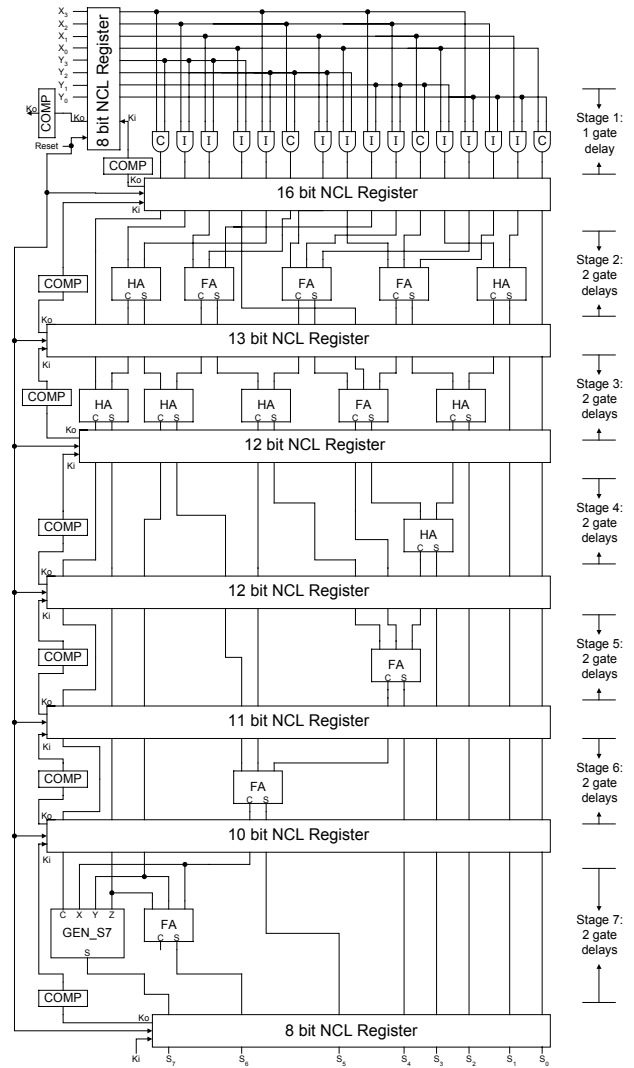
The technique of Early Completion that moves the completion detection for registration stage <sub>$i$</sub>  from the output of the register to its input can significantly increase throughput of self-timed systems without increasing latency. A NCL 4-bit by 4-bit multiplier case study indicates a speedup of 1.21 over the design utilizing standard completion. Furthermore, the technique could be applied to other self-timed paradigms [3, 4, 5, 6, and 7] as well, since they use the same handshaking scheme, with only differed combinational logic. However, since these other self-timed paradigms do not support the multitude of gates supported by NCL, each TH24comp gate of the early completion component in Figure 8 would have to be replaced by two TH12 gates, causing there to be  $N$  intermediate  $K_o$  signals instead of only  $\frac{N}{2}$  as for NCL,

thus necessitating more gates and logic levels, and therefore reducing throughput for the other self-timed paradigms.

## References

- [1] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Integration, The VLSI Journal*, Vol. 30/2, pp. 103-131, 2001.
- [2] 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.
- [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] T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technology Bulletin*, Sept. 1986.
- [6] 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.
- [7] J. Sparso, J. Staunstrup, M. Dantzer-Sorensen, Design of Delay Insensitive Circuits using Multi-Ring Structures. *Proceedings of the European Design Automation Conference*, pp. 15-20, 1992.
- [8] M. Singh and S. M. Nowick, "High-Throughput Asynchronous Pipelines for Fine-Grain Dynamic Datapaths," *Proceeding of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 198-209, 2000.
- [9] T. E. Williams, *Self-Timed Rings and Their Application to Division*, Ph.D. Thesis, CSL-TR-91-482, Department of Electrical Engineering and Computer Science, Stanford University, 1991.

- [10] A. J. Martin, "Programming in VLSI," in *Development in Concurrency and Communication*, Addison-Wesley, pp. 1-64, 1990.
- [11] K. Van Berkel, "Beware the Isochronic Fork," *Integration, The VLSI Journal*, Vol. 13, No. 2, pp. 103-128, 1992.
- [12] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
- [13] S. C. Smith, *Gate and Throughput Optimizations for NULL Convention Self-Timed Digital Circuits*, Ph.D. Dissertation, School of Electrical Engineering and Computer Science, University of Central Florida, 2001.



**Figure 11. Optimally pipelined 4x4 multiplier using full-word completion**