

Completion-Completeness for NULL Convention Digital Circuits Utilizing the Bit-wise Completion Strategy

Scott C. Smith

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

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

Keywords: NULL Convention Logic (NCL), Bit-wise Completion, Input-Completeness, Delay-Insensitive Circuits, Dual-Rail Logic

Abstract

Delay-Insensitive paradigms, such as NULL Convention Logic (NCL), require an additional condition, referred to herein as Completion-Completeness, in order to ensure delay-insensitivity when the bit-wise completion strategy is used along with one or more components that are not complete with respect to all of their inputs. Completion-completeness requires that completion signals only be generated such that no two adjacent DATA wavefronts can interact within any combinational component. Correctly applying the bit-wise completion strategy to a circuit with input-incomplete components does not ensure completion-completeness; instead this may require either modifying one or more completion sets or changing the input-completeness of one or more input-incomplete components. Examples of completion-incompleteness, and their solutions, are shown when applying the bit-wise completion strategy to circuits utilizing input-incomplete components, for an array of AND functions, the final stage of an unsigned multiplier, and the partial product generation for an unsigned multiplier.

1. Introduction

NULL Convention Logic (NCL) [1] 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

delay-insensitive logic methods, the NCL paradigm assumes that forks in wires are isochronic [3]. The origins of various aspects of the paradigm, including the NULL (or spacer) logic state from which NCL derives its name, can be traced back to Muller’s work on speed-independent circuits in the 1950s and 1960s [4].

In order for NCL combinational circuits to maintain delay-insensitivity, they must adhere to the completeness of input criterion [1], which requires that:

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

In circuits with multiple outputs, it is acceptable, according to Seitz’s weak conditions [2], for some of the outputs to transition without having a complete input set present, as long as all outputs cannot transition before all inputs arrive.

2. Previous Work

In [5] bit-wise completion was mentioned as one way to reduce the completion logic in order to speedup asynchronous circuits. However, the circuits in [5] to which bit-wise completion was applied were composed of dynamic logic; therefore they were not delay-insensitive and did not have to adhere to the strict delay-insensitive rules, such as input-completeness, that NCL and other delay-insensitive paradigms must. Therefore completion-completeness was not an issue for the asynchronous circuits in [5]. However, when the bit-wise completion strategy is applied to delay-insensitive circuits, completion-completeness must be ensured in order to maintain delay-insensitivity.

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

3. Completion Strategy

Full-word completion requires that the completion signal of each bit in register_{*i*} be conjoined by the completion component [6], whose single-bit output is connected to all K_i lines of register_{*i-1*}. On the other hand, 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 than that of full-word completion, thus increasing throughput [6]. Bit-wise completion will never reduce throughput, since in the worse case all bits of register_{*i-1*} are used to calculate each bit of register_{*i*}, such that the completion logic and therefore throughput does not change by selecting bit-wise completion rather than full-word completion. Bit-wise completion may or may not require more logic gates and therefore transistors than full-word completion; thus bit-wise completion will be used if it increases throughput, or if throughput is the same as for full-word completion but area is reduced.

Figure 1 shows full-word completion for a combinational stage of six 2-input AND functions, generating all combinations of the 4-bit input X . Figure 2 shows bit-wise completion for the same six AND functions. There is only one level of logic in the completion components for the bit-wise completion approach versus two levels of logic in the completion component for the full-word completion approach. Also notice that four gates are required for bit-wise completion

verses three gates for full-word completion, a difference of 8 additional transistors. To maximize throughput in this case, bit-wise completion would be selected in spite of its larger size since it reduces the completion logic path from two gate delays down to only one gate delay, which translates to an increase in throughput as calculated by Algorithm 4.2 in [6].

4. Completion-Completeness

Completion-completeness, which circuits developed from most delay-insensitive paradigms [1, 2, 7, 8, 9, 10] must maintain in order to be delay-insensitive, 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, since it is inherent when using the full-word completion strategy.

Completion-completeness for full-word completion is inherent due to the input-completeness criterion. For full-word completion there is only one completion signal, K_j , composed of the completion signals of each bit in register_{*j*}; therefore all bits of register_{*j*} must become DATA/NULL before K_j transitions to *rfd* (request for NULL, i.e. logic 0)/*rfd* (request for DATA, i.e. logic 1), respectively. Due to the input-completeness criterion, all outputs of the combinational circuit will not become DATA/NULL until all inputs become DATA/NULL, respectively; therefore K_j will not transition until all bits

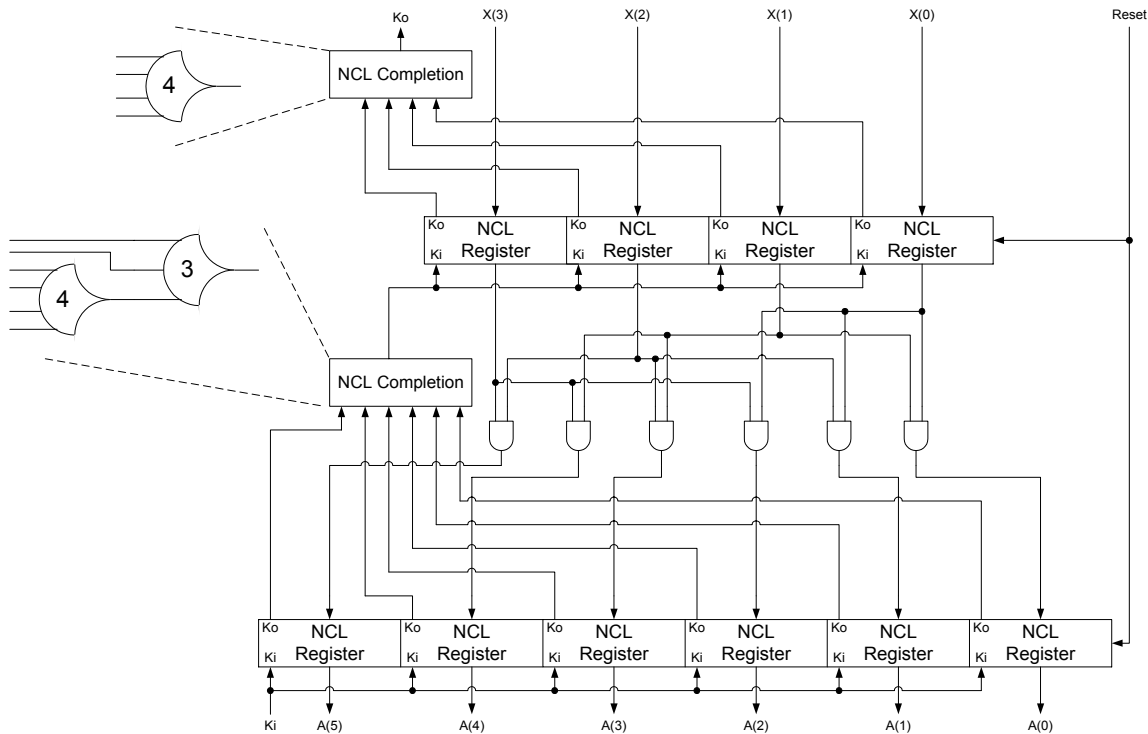


Figure 1. Full-word completion.

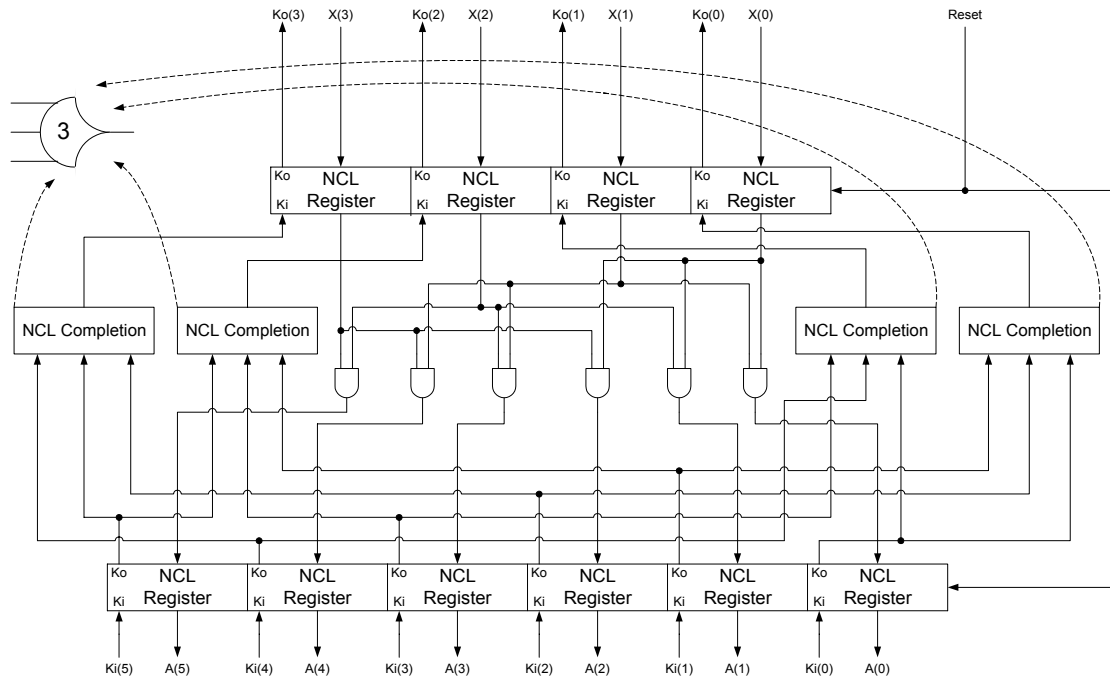


Figure 2. Bit-wise completion.

of register_{*i*} are DATA/NULL. Hence, a circuit utilizing full-word completion is inherently completion-complete, since the inputs follow the sequence, all DATA then all NULL, continuously, making it impossible for two adjacent DATA wavefronts to interact in the combinational logic.

Input-completeness does not require all components in the combinational circuit to be input-complete as long as the output set as a whole is input-complete. Take for example the six AND functions of Figure 1. In order for the combinational logic to be input-complete, only two of the AND functions need be input-complete circuits, shown in Figure 3, for example AND functions $X(3) \bullet X(2)$ and $X(1) \bullet X(0)$. The other four AND functions can be input-incomplete circuits, shown in Figure 4. The benefit to using input-incomplete AND functions is a reduction of 13 transistors per function.

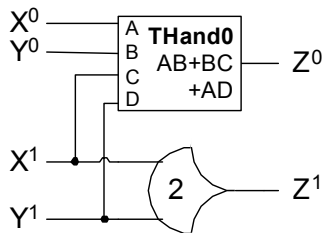


Figure 3. Complete AND function: $Z = X \bullet Y$.

On the other hand, completion-completeness with bit-wise completion is not inherent. If the combinational circuit is input-complete and the completion component is

partitioned correctly, such that the completion signal from bit b in register_{*i*} is only sent back to the bits in register_{*i-1*} that took part in the calculation of bit b , the resulting circuit is not necessarily completion-complete and therefore is not necessarily delay-insensitive. This may be the case when any of the components in the combinational circuit are not input-complete with respect to all of their inputs. If all combinational components are input-complete with respect to all of their inputs, then completion-completeness is guaranteed, otherwise it may need to be ensured by either modifying one or more completion sets or by changing the input-completeness of one or more input-incomplete components.

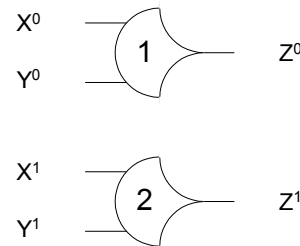


Figure 4. Incomplete AND function: $Z = X \bullet Y$.

5. Completion-Completeness Examples

5.1 Six 2-Input AND Functions

Take for example the six 2-input AND functions, generating all combinations of the 4-bit input X , and

utilizing bit-wise completion, as shown in Figure 2. Only AND functions $X(3) \bullet X(2)$ and $X(1) \bullet X(0)$ are input-complete circuits, shown in Figure 3, while the other four AND functions are input-incomplete circuits, as shown in Figure 4. This circuit is not delay-insensitive since it contains input-incomplete components and is not completion-complete.

To show that this circuit is not delay-insensitive, let $X_j(3)$ and $X_j(2)$ initially be DATA0. Let $X_j(1)$ initially be DATA1, and $X_j(0)$ be NULL initially, where the subscript, j , refers to the wavefront. This causes outputs $A_j(5:1)$ to become DATA0, assuming their respective K_i signals are *rfd*, while $A(0)$ remains NULL, which allows *rfn* to pass through the completion components to the $X(3)$ and $X(2)$ input registers, allowing NULL to flow through these two input registers. This in turn causes outputs $A_j(5:1)$ to become NULL, assuming their respective K_i signals are *rfn*, which allows *rfd* to pass through the completion components to the $X(3)$ and $X(2)$ input registers, allowing the next DATA wavefront, wavefront _{$j+1$} , to flow through these two input registers; therefore causing two adjacent DATA wavefronts, $X_{j+1}(3:2)$ and $X_j(1)$, to interact within the combinational logic, which violates the completion-completeness criterion, thus causing the circuit to no longer be delay-insensitive.

To show that this circuit is therefore not correct, let $X_{j+1}(2)$ be DATA1 and $X_{j+1}(1)$ be DATA0. $X_{j+1}(2)$ and $X_j(1)$ then interact in the combinational logic to produce $A_{j+1}(3) = \text{DATA1}$, which is incorrect, since the correct value should be $X_{j+1}(2) \bullet X_{j+1}(1) = \text{DATA0}$.

To ensure delay-insensitivity of this circuit, either the incomplete AND functions would have to be replaced with complete AND functions, which would not increase the number of logic levels but would require 52 additional transistors, or the completion sets would have to be modified. This can be done by adding the completion signal from output register $A(0)$ to the left two completion components and by adding the completion signal from output register $A(5)$ to the right two completion components. This alternate approach requires the four completion components to use a TH44 gate instead of a TH33 gate, which would also not increase the number of logic levels and would only require an additional 16 transistors. Therefore the second approach of modifying the completion sets would be chosen, since it would not reduce throughput and would require less additional logic.

Assume the same scenario as above: $X_j(3)$ and $X_j(2)$ are DATA0, $X_j(1)$ is DATA1, and $X_j(0)$ is NULL. This causes outputs $A_j(5:1)$ to become DATA0, while $A(0)$ remains NULL. However, now *rfn* will not be sent to any of the input registers until $A(0)$ becomes DATA, which will not occur until $X_j(0)$ becomes DATA, thus ensuring that the completion-completeness criterion holds and that the circuit therefore remains delay-insensitive.

5.2 Final Stage of Unsigned Multiplier

Take for example the last stage of a dual-rail pipelined 4-bit \times 4-bit unsigned multiplier using bit-wise completion [6], shown in Figure 5. The GEN_S7 component is input-complete only with respect to input C , as shown in Figure 6. The full-adder (FA) is input-complete with respect to the other inputs, X , Y , and Z . Therefore the combinational circuit as a whole is input-complete with respect to all four inputs. The circuit also correctly utilizes the bit-wise completion strategy as explained in Section 3. However, this circuit is not delay-insensitive since it contains an input-incomplete component and is not completion-complete.

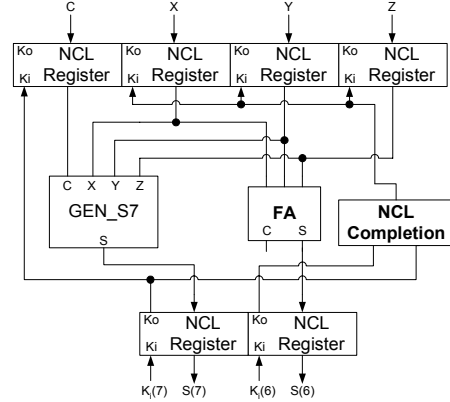


Figure 5. Completion-incomplete circuit [6].

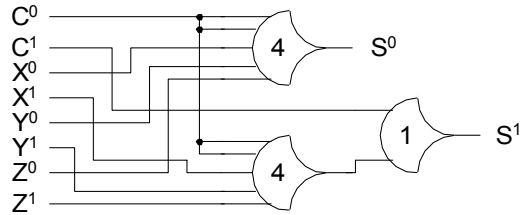


Figure 6. Original GEN S7 component [6].

To show that this circuit is not delay-insensitive, let C_j be DATA1 and X_j , Y_j , and Z_j all be DATA0 initially, where the subscript j refers to the wavefront. Now assume that the full-adder component has a much longer delay than does the GEN_S7 component. Since C_j is DATA1, S^j of the GEN_S7 component becomes asserted by way of the TH12 gate. This DATA value will then pass through the output register to produce $S_j(7)$, assuming $K_i(7)$ is *rfd*, causing the register's K_o line to become *rfn*. In turn this will allow NULL to flow through input register _{c} . Since only the TH12 gate was asserted in the GEN_S7 component, C_j changing to NULL will cause the output of the GEN_S7 component to become NULL, even though the other inputs, X_j , Y_j , and Z_j are still DATA. This NULL value will then pass through the output register to $S(7)$, assuming $K_i(7)$ is *rfn*, causing the register's K_o line to

become *rfd*. In turn this will allow the next DATA value, C_{j+1} , to flow through input register_C. The GEN_S7 component now has two adjacent DATA wavefronts, wavefront_j (X_j , Y_j , and Z_j) and wavefront_{j+1} (C_{j+1}) interacting to produce the output, which is incorrect. For example, if C_{j+1} is DATA0 and X_{j+1} , Y_{j+1} , and Z_{j+1} are all DATA1, then $S_{j+1}(7)$ should be DATA1. However, since C_{j+1} is interacting with X_j , Y_j , and Z_j to produce $S_{j+1}(7)$, $S_{j+1}(7)$ will be DATA0, which is not correct.

This circuit can be made completion-complete, and therefore delay-insensitive, by either changing the input-completeness of the GEN_S7 component or by changing the completion set for input C . The GEN_S7 component would have to be modified, such that it is not only complete with respect to input C , but also with respect to at least two of its other inputs. If the GEN_S7 component was made to be input-complete with respect to only one additional input, the resulting design would still not be completion-complete. To make this component complete with respect to input X would require the TH12 gate to become a TH34w32 gate, as shown in Figure 7, which would not increase the number of gate delays in the GEN_S7 component, but would require an additional 11 transistors. Assume the same scenario as before: C_j is DATA1 and X_j , Y_j , and Z_j are all DATA0 initially. Since C_j is DATA1 and X_j is DATA0, S^l of the GEN_S7 component will become asserted by way of the TH34w32 gate. This DATA value will then pass through the output register to produce $S_j(7)$, assuming $K_i(7)$ is *rfd*, causing the register's K_o line to become *rfn*. In turn this will allow NULL to flow through input register_C. However, the output of the GEN_S7 component will not become NULL until X_j also becomes NULL, and X_j will not become NULL until the output of the full-adder becomes DATA and passes through the output register to produce $S_j(6)$, causing the register's K_o line to become *rfn*, which in turn will allow NULL to flow through input register_X. However, X can become NULL while Y and Z remain DATA, which will cause $S_j(7)$ to become NULL, assuming $K_i(7)$ is *rfn*. This in turn will cause the register's K_o line to become *rfd*, thus allowing the next DATA value, C_{j+1} , to flow through input register_C. Therefore the GEN_S7 component again has two adjacent DATA wavefronts, wavefront_j (Y_j , and Z_j) and wavefront_{j+1} (C_{j+1}) interacting to produce the output, which is incorrect. For example, if C_{j+1} is DATA0 and Y_{j+1} and Z_{j+1} are both DATA1, then $S_{j+1}(7)$ should be DATA1. However, since C_{j+1} is interacting with Y_j and Z_j to produce $S_{j+1}(7)$, $S_{j+1}(7)$ will be DATA0, which is not correct.

To make this circuit completion-complete by changing the input-completeness of the GEN_S7 component requires the component to be input-complete with respect to at least two other inputs besides C . To make this component input-complete with respect to both X and Y would require additional circuitry as shown in Figure 8,

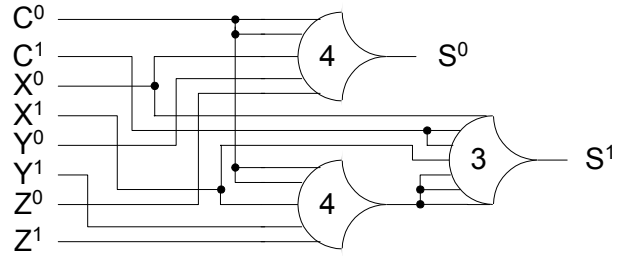


Figure 7. Revised GEN_S7 component input-complete with respect to X .

which would increase the number of gate delays for S^0 , and would require 2 additional gates, for an additional 38 transistors. The TH24comp gate is added to ensure input-completeness of X and Y for output, S^l , when C is DATA1. However, adding this gate for S^l necessitates it to also be required for S^0 in order to ensure observability, which is also necessary for delay-insensitivity. The observability condition states that any gate which transitions must be necessary to transition at least one of the outputs. Therefore if the circuitry for S^0 was not changed, so that it was the same as in Figure 7, then if all four inputs were DATA0, the TH24comp gate would be asserted and S^0 would be asserted; hence the TH24comp gate would be unobservable, since it would be asserted without causing any outputs to become asserted.

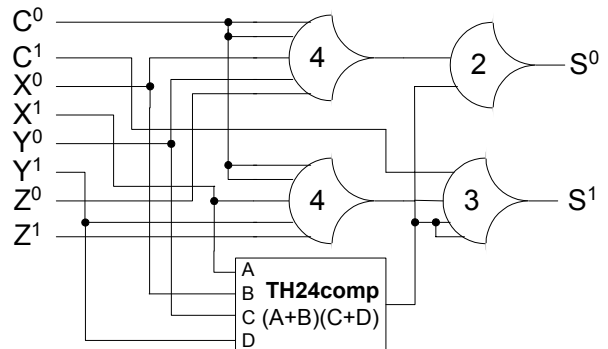


Figure 8. Revised GEN_S7 component input-complete with respect to X and Y .

Using this newly revised GEN_S7 component, assume the same scenario as before: C_j is DATA1 and X_j , Y_j , and Z_j are all DATA0 initially. Since C_j is DATA1 and X_j and Y_j are DATA0, S^l of the GEN_S7 component will become asserted by way of the TH24comp and TH33w2 gates. This DATA value will then pass through the output register to produce $S_j(7)$, assuming $K_i(7)$ is *rfd*, causing the register's K_o line to become *rfn*. In turn this will allow NULL to flow through input register_C. However, the output of the GEN_S7 component will not become NULL

until X_j also becomes NULL, and X_j will not become NULL until the output of the full-adder becomes DATA and passes through the output register to produce $S_j(6)$, causing the register's K_o line to become *rfn*, which in turn will allow NULL to flow through input register_X. Now X and Y must both transition to NULL to cause $S_j(7)$ to become NULL, assuming $K_i(7)$ is *rfn*. Z can remain DATA. This in turn will cause the register's K_o line to become *rfd*, thus allowing the next DATA value, C_{j+1} , to flow through input register_C. Now there are two adjacent DATA wavefronts, wavefront_j (Z_j) and wavefront_{j+1} (C_{j+1}), at the input of the GEN_S7 component; however, these wavefronts cannot interact within the component since to transition any gate requires at least two of the following inputs: X , Y , or Z . Therefore the GEN_S7 component must wait for either X_{j+1} or Y_{j+1} to become DATA, and these inputs cannot transition to DATA until Z_j first becomes NULL, causing the output of the full-adder to become NULL and pass through the output register, causing the register's K_o line to become *rfd*, which in turn will allow DATA to flow through input register_X and register_Y. Thus only DATA wavefront_{j+1} is now being input to the GEN_S7 component; hence this circuit is now completion-complete and therefore delay-insensitive.

The alternative solution is to change the completion set for register_C, such that NULL is not requested from register_C until both the GEN_S7 component and the full-adder produce a DATA output. This would require the output of the completion component to not only be input to register_X, register_Y, and register_Z, as shown in Figure 5, but also to register_C. This approach would not require any additional logic. Again assume the same scenario as before, C_j is DATA1 and X_j , Y_j , and Z_j are all DATA0, initially. Since C_j is DATA1, S^l of the GEN_S7 component becomes asserted by way of the TH12 gate. This DATA value will then pass through the output register to produce $S_j(7)$, assuming $K_i(7)$ is *rfd*, causing the register's K_o line to become *rfn*. However, the completion signal to register_C will not become *rfn* until the output of the full-adder becomes DATA and passes through the output register to produce $S_j(6)$, causing the register's K_o line to become *rfn*. Only then will NULL be allowed to flow through input register_C. Therefore the two adjacent DATA wavefronts will never be able to interact within the GEN_S7 component and the circuit will function properly. Since this second approach does not require any additional logic and does not decrease throughput, it would be chosen in this case.

5.3 Unsigned Multiplier Partial Product Generation

Consider the partial product generation for a 4-bit \times 4-bit unsigned multiplier: $X(3:0) \times Y(3:0)$, consisting of sixteen AND functions [6]. Similar to the six AND functions in the example in Section 5.1, only

four of the sixteen AND functions, $X(0) \bullet Y(0)$, $X(1) \bullet Y(1)$, $X(2) \bullet Y(2)$, and $X(3) \bullet Y(3)$, need be complete versions, as shown in Figure 3, while the remaining twelve can be incomplete functions, shown in Figure 4. Using the bitwise completion strategy on this circuit will result in eight completion components, each consisting of one TH44 gate.

Following the same logic as in Section 5.1, it can be shown that this circuit is not completion-complete and is therefore not delay-insensitive. The circuit can be made to be delay-insensitive by either modifying the completion sets or by switching the incomplete AND functions with complete versions. In this case, modifying the completion sets would require more than four inputs to each of the completion components, thus requiring additional levels of logic; since 4-input threshold gates are the maximum currently used in the NCL paradigm. Hence, it would be best to replace the incomplete AND functions with their complete counterparts; since this modification would not increase the number of logic levels and would therefore not decrease throughput.

6. Conclusion

In this paper it was shown that besides the input-completeness criterion, which must be enforced for circuits to be delay-insensitive, an additional criterion, completion-completeness, is required to ensure delay-insensitivity for circuits utilizing the bit-wise completion strategy along with incomplete components. Incorrect operation was demonstrated for three circuits, which utilized bit-wise completion along with incomplete components without adhering to the completion-completeness criterion. The alternate solutions, modifying the completion sets and modifying the incomplete components, to make the incorrect circuits completion-complete and therefore delay-insensitive, were then compared in terms of decreased throughput and increased area. This comparison showed that for some circuits it is best to modify the completion sets, while for others modifying the incomplete component(s) is best.

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] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.
- [3] C. H. (Kees) van Berkel, M. Rem, and R. Saeijs, "VLSI Programming," *1988 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 152-156, 1998.

- [4] D. E. Muller, "Asynchronous Logics and Application to Information Processing," in *Switching Theory in Space Technology*, Stanford University Press, pp. 289-297, 1963.
- [5] 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.
- [6] 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.
- [7] N. P. Singh, *A Design Methodology for Self-Timed Systems*, Master's Thesis, MIT/LCS/TR-258, Laboratory for Computer Science, MIT, 1981.
- [8] T. S. Anantharaman, "A Delay Insensitive Regular Expression Recognizer," *IEEE VLSI Technology Bulletin*, Sept. 1986.
- [9] 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.
- [10] 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.