NULL Convention Logic™: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis

Karl M. Fant
Theseus Logic, Inc
1080 Montreal Ave, Ste 200
St Paul, Minnesota, USA 55116

Scott A. Brandt
Dept of Computer Science
Campus box 430
University of Colorado
Boulder, Colorado, 80309

Abstract

NULL Convention Logic (NCL) is a symbolically complete logic which expresses process completely in terms of the logic itself and inherently and conveniently expresses asynchronous digital circuits. The traditional form of Boolean logic is not symbolically complete in the sense that it requires the participation of a fundamentally different form of expression, time in the form of the clock, which has to be very carefully coordinated with the logic part of the expression to completely and effectively express a process. We introduce NULL Convention Logic in relation to Boolean logic as a four value logic, and as a three value logic and finally as two value logic quite different from traditional Boolean logic. We then show how systems can be constructed entirely in terms of NULL Convention Logic.

1: Introduction

NULL Convention Logic [7] is derived directly from the Invocation Model of Process Expression. The Invocation Model is a conceptual model of general process expression in contrast to a model of computation. It transcends limiting mathematical notions of computation to provide a unifying conceptual framework which relates all forms of process expression from the simplest physical and chemical processes to the most complex natural and artificial processes. For instance, the processes of cell metabolism and of digital computers are characterized using the same concepts and relationships. They simply occupy different places in a single expression space defined by the Invocation Model.

A central concept of the Invocation Model is the **general notion of completeness**. Just how fundamental the notion of completeness is will become clear in the course of the discussion. It will appear in many guises in different aspects of process expression but always in service of providing necessary and sufficient conditions. We will begin with the notion of **symbolic completeness of expression**. A process expression is symbolically complete if it is expressed entirely in terms of symbolic dependency relationships. It integrates the expression of data transformation and the expression of control into a single symbolically determined expression.

Traditionally, in computer science, the expression of symbol transformation and the expression of control have been viewed as inherently independent forms of process expression which must, of necessity, be carefully coordinated. But, the data transformation and control expressions are not inherently independent. They can be integrated into a single expression using only symbolic-value-dependent relationships, making it a symbolically complete expression. A symbolically complete logic circuit has no time relationships and is **insensitive to the propagation time of symbols among its component elements**.

Boolean logic is not symbolically complete. Its circuits exhibit time dependent relationships as well as symbolic-value-dependent relationships. The symbolic-value-dependent relationships depend on the interconnection of the logic gates and their logical behavior. The time-dependent relationships depend on the propagation delays of the component elements. These two aspects of expression are independent because the time relationships can be expressed arbitrarily in relation to the expression of symbolic relationships and vice versa. These two partial expressions must be carefully and explicitly coordinated to provide a complete correctly resolvable expression of a process. A Boolean logic circuit with its clock is a complete expression, but it is not a symbolically complete expression.

There have been attempts to eliminate time dependencies in digital logic circuits since D. E. Muller, pioneered the pursuit in the late 1950s [1,2]. These attempts to eliminate time dependencies are called delay insensitive circuits, and are nearly always expressed within the traditional context of Boolean logic. They focus on designing Boolean logic circuits with appropriate switching behavior and surrounding them with Muller C-elements to express the control, then transmitting data between circuits with dual rail encoding. These structures of Boolean logic circuits and C-elements can be difficult to design and expensive to implement [3]. Recent work [4-6] has produced many interesting results, but has not produced a theoretically complete and economically feasible solution. The difficulty is that traditional Boolean logic is expressionally insufficient in that it is not symbolically complete. The traditional form of Boolean logic is simply the wrong conceptual context in which to consider the problem of delay insensitivity.

NULL Convention Logic is a theoretically complete and economically feasible approach to delay insensitive circuits. In this paper we first introduce the NULL Convention in the context of Boolean logic, showing how to make Boolean logic symbolically complete as a four value logic. Then, we show how the NULL Convention can be implemented as a two value logic, which will prove to be the most practical form. We conclude with a discussion of the properties of NULL Convention Logic.

2: Making Boolean Logic Symbolically Complete

To begin with there must be symbolic completeness in the most primitive form of expression. This means that the most primitive form of expression must sufficiently express both symbolic data transformations as well as symbolic control expression. The first form of completeness is that, both data transformation and control must be expressed in the most primitive inherently enforced **mutually exclusive value assertion domain**. In this domain, only one value in a set can be asserted at a time. Every expression environment has a primitive mutually exclusive value assertion domain. For Boolean logic, this domain consists of True and False.

The True and False values of traditional Boolean logic expresses only data meanings. The first step needed to make Boolean logic symbolically complete is to add the control value NULL (not a valid data) to the domain. Figure 1 shows the NULL value added to the traditional Boolean truth tables and specifies the behavior of these preliminary NULL Convention Logic gates.

	T	F	N
T	T	F	N
F	F	F	N
N	N	N	N

AND

	T	F	N
T	T	T	N
F	T	F	N
N	N	N	N

OR

T	F
F	T
N	N

NOT

Fig. 1. Boolean truth tables with NULL value added

We now have a **three value logic**. Each wire can assert one of three values (T, F or N) and each gate resolves three input values. True and False are data values and NULL is not a data value. The **completeness of input criteria** is enforced for data; a gate asserts data value only when a complete set of data values is present at its input.

2.1: The data resolution wavefront

The completeness of input criteria is the key to delay insensitive logic circuits. The completeness of input criteria for each gate scales up for combinational circuits as a whole. Consider Figure 2.

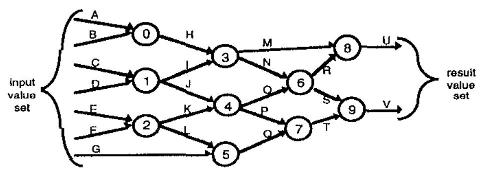

Fig. 2. Example combinational circuit

Assume that the circuit is in an all NULL state. That is, all signals are NULL. If A changes to data, gate 0 will continue asserting NULL. Gate 6, for instance, does not assert data until both N and O are data at the gate. It doesn't matter how long it takes or in what order the data values on N and O arrive at thr gate. The transition of gate 6 from NULL to data asserts the validity of that result data value and asserts the completion of the gate's resolution entirely symbolically. Each gate is a **synchronization node** managing an orderly wavefront of correct result values.

This behavior scales up for the circuit as a whole. The circuit as a whole will not assert a complete set of result data values until a complete set of input data values has been presented to the circuit and has propagated through the circuit (resolution of the input values is complete). Thus, **the circuit as a whole enforces the completeness of input criteria for data.**

The completion of the result can be detected by the output's transition from all NULL to all data. **The circuit indicates its own completion of resolution, autonomously and purely symbolically.**

2.2: The NULL wavefront

To express the completeness of input criteria for data, the circuits must begin in an all-NULL state. After each data set resolution the circuit must be returned to this NULL state. Thus, the circuit cycles between the all NULL state and the data resolution state. When the circuit enters the all NULL state, it is ready for a new set of input data. Figure 1 shows that the completeness of input criteria is not enforced for NULL values in relation to data values. If one input value is NULL then the result value will become NULL and all output values of the circuit can become NULL while there are still data values at the input or internal to the circuit.

Thus, the **second step** needed to make Boolean logic symbolically complete is for the gates to enforce the completeness of input criteria for NULL in relation to data (as well as for data in relation to NULL). This can be accomplished by adding a fourth value (Intermediate) to the primitive mutually exclusive value assertion domain, or by adding a feedback variable to each gate.

2.3: The Intermediate value solution

Adding the Intermediate value to the primitive mutually exclusive value assertion domain of the logic provides a solution that is purely delay insensitive and purely symbolically complete, as shown in Figure 3.

	T	F	I	N
T	T	F	I	I
F	F	F	I	I
I	I	I	I	I
N	I	I	I	N

AND

	T	F	I	N
T	T	T	I	I
F	T	F	I	I
I	I	I	I	I
N	I	I	I	N

OR

T	F
F	T
I	I
N	N

NOT

Fig. 3. Intermediate value truth tables

We now have a **four value logic**. Now, a gate will only assert data when both inputs are data, and will only assert NULL when both inputs are NULL. Thus, the completeness of input criteria for both data and NULL is enforced. The result values transition from all NULL through Intermediate values to all data, and then from all data through Intermediate values to all NULL. Again, the behavior of each gate scales up for the circuit as a whole.

We now have a circuit that is a complete expression. It can signal when it is ready to accept new data and when it has completed a data resolution. An intermediate value logic circuit is a symbolically complete process expression and is purely symbolically determined. It is a theoretically complete and general solution to delay insensitive circuit synthesis.

The addition of the NULL value or the Intermediate value did not change the transform specifications for the data values. Intermediate value NULL Convention Logic™ gates can replace the gates of a standard Boolean logic combinational circuit one for one, and the circuit will provide the identical logic function as before.

2.4: The feedback solution

The feedback solution makes each gate a state machine with hysteresis of its result value. Figure 4 shows the truth table for the feedback gate.

	R	N									F									T								
INPUT	A	N			T			F			N			T			F			N			T			F		
	B	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F	N	T	F
RESULT	R	N	N	N	N	T	F	N	F	F	N	F	F	F	F	F	F	F	F	N	T	T	T	T	T	T	T	T

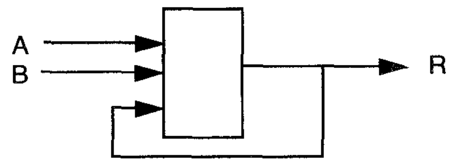

Fig. 4. The feedback gate and its truth table

R is the result variable fed back to the input. When the gate is asserting NULL it is in the NULL (N) state and will continue asserting N until both input values become data (T,F). Likewise, when the gate is in a data state (R = F or T) it will continue asserting a data value until both input values become NULL (N). **The feedback gate enforces the completeness of input criteria for both data in relation to NULL and for NULL in relation to data.** As before, the behavior of each gate scales up for the circuit as a whole, and the circuit as a whole enforces the completeness of input criteria for both data in relation to NULL and for NULL in relation to data.

While the intermediate value solution is fully delay insensitive, the feedback solution has a non-critical timing relationship involved. The feedback path around each gate must stabilize faster than successive wavefronts of transition that pass through the circuit as a whole. This relationship is easily met. Thus, the feedback solution is effectively delay insensitive.

3: Two-Value NULL Convention Logic™

We have shown how to make Boolean logic symbolically complete as a four value logic system and effectively symbolically complete as a three value logic system. The next step is to show how the NULL convention can be applied to binary systems.

If the logic is limited to two values in its primitive mutually exclusive value assertion domain (wire) and one value must express NULL, then True and False must be expressed with two wires (one wire asserting DATA means TRUE, and the other wire asserting DATA means FALSE). With no inherent means to prevent any two wires from expressing their DATA values simultaneously there is no longer an inherently enforced mutually exclusive value assertion domain for data meanings as there was when two data meanings on a single wire were by physical necessity mutually exclusive.

However, each wire is still an inherent mutually exclusive value assertion domain that asserts either DATA or NULL. To reestablish mutually exclusive assertion of data meanings, a convention is maintained that only one wire in a group of wires may assert its DATA value at a time. This groups is called a **mutually exclusive assertion group**.

A mutually exclusive assertion group can be any size. A group of ten wires can directly express decimal numbers with each wire expressing a digit value. The two wire mutually exclusive assertion group is identical to dual rail encoding, which is traditionally used as a transmission protocol between speed independent circuits. The mutually exclusive assertion group in NULL Convention Logic is a much more general concept than dual rail transmission encoding. It is not just a transmission protocol but is an inherent part of the logic itself.

With the data values split up another delay sensitivity issue arises but this sensitivity can be limited to a single wire delay in relation to successive wavefronts and can be considered noncritical in the same sense that the feedback path around the gate can be considered noncritical.

3.1: The single data value logic gate

The NULL value expresses the abscence of data and therefore cannot contribulte to data resolution. Since each input to a gate can express only one DATA value, the only distinguishing property when combinig DATA values at a gate is how many DATA values are presented. Therefore, NULL Convention Logic gates are naturally implemented as **discrete threshold gates**. An input data set is complete if the number of signals asserting DATA is at least the threshold of the gate. Figure 5 shows a 5 input, threshold 3 gate. If three or more inputs are DATA, then the gate asserts DATA. Otherwise, it asserts NULL.

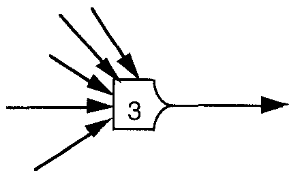

Fig. 5. Threshold gate

Figure 6 shows a NULL Convention Logic half adder circuit.

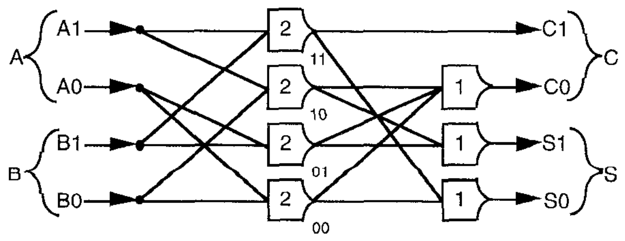

Fig. 6. NULL Convention half adder circuit

A Boolean half adder expresses 4 possible input data values with two data values (0, 1) on two wires (A, B). The NULL Convention half adder circuit expresses the same 4 input data values with one data value (DATA) on four wires (A0, A1, B0, B1). Each binary data value A, B, C and S is expressed by a mutually exclusive assertion group of two wires (e.g. A0, A1 for A). A complete input data set for the circuit is two DATA values, one from each group (A and B). The completeness of input criteria for each threshold 2 gate is 2 DATA values. It can be seen that for any threshold 2 gate to assert a data result value there must be one DATA value asserted in each input group and that only one threshold 2 gate in a column will assert a result data value. **The gates, as well as the circuit as a whole, enforce the completeness of input criteria for data**. When the result values transition from NULL to a complete result data set, which in this case is one DATA value for each result group (C and S), then the asserted result values are a correct resolution of a complete input data set. The completeness of input criteria for each gate scales up for the circuit as a whole and as before, **the completeness of resolution can be determined by simply monitoring the result values.**

It will be noticed that if the mutually exclusive assertion convention is enforced for the input of the circuit, then the circuit itself maintains the convention at its output. A system of such circuits communicating among themselves will maintain the convention among themselves. The convention only has to be explicitly enforced at the input to the system.

3.2: The NULL wavefront again

A single data value NULL Convention Logic circuit still has to be returned to an all NULL state before accepting a new data set. As in the Boolean logic examples, the completeness of input criteria for NULL must be enforced. As before, there is a feedback solution and an Intermediate value solution.

3.3: The Feedback Solution

The feedback solution is simple, straightforward and inexpensive. To provide the necessary hysteresis behavior, the result value is fed back with a weight of one less than the threshold as shown in Figure 7. The rounded gate symbol represents a gate with the hysteresis behavior.

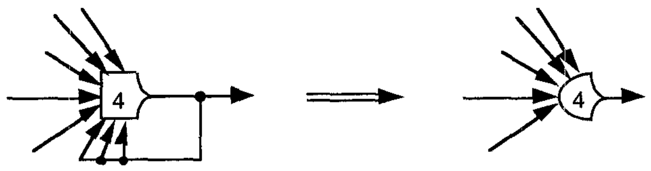

Fig. 7. Threshold gate with weighted feedback of threshold -1

Beginning in a NULL state with all signals NULL, the gate will not assert a DATA result value until its input data set is complete, which in this case is four DATAs. With the feedback, the gate will not return to NULL until all of its input values are NULL. This behavior is shown in Figure 8.

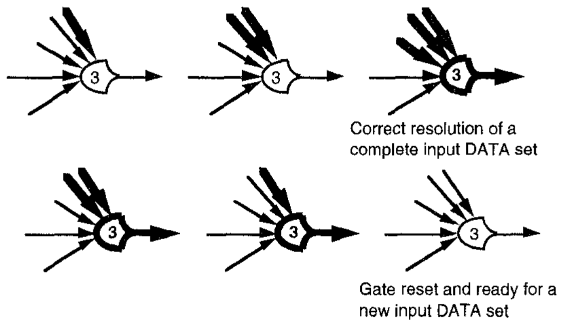
3
3
3
Correct resolution of a complete input DATA set
3
3
3
Gate reset and ready for a new input DATA set

Fig. 8. Hysteresis behavior of NULL Convention Logic™ gate

The hysteresis gate now enforces the completeness of input criteria for both data in relation to NULL and for NULL in relation to DATA. A threshold 1 gate does not require a feedback connection because its completeness of input criteria is inherently enforced for both DATA and NULL.

3.4: The Intermediate Value Solution

Since we are limited to two values in the most primitive mutually exclusive value assertion domain we must encode Intermediate on top of the mutually exclusive assertion group encoding. DATA, Intermediate and NULL are expressed using two wires with the following encoding.

I1	I2	meaning
DATA	DATA	DATA
DATA	NULL	Intermediate
NULL	DATA	Intermediate
NULL	NULL	NULL

As shown in Figure 9, each input to a gate is two wires and a single intermediate value gate consists of several threshold gates without hysteresis. A True-False pair is now expressed with four wires.

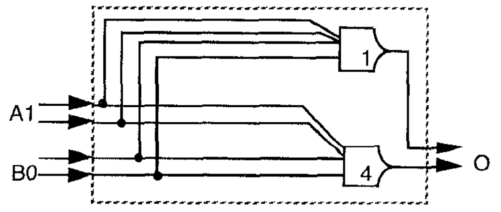

Fig. 9. Intermediate encoded 2 input threshold 2 gate

Starting with the intermediate value gate in an all NULL state, if one DATA value is applied to A1, the threshold 1 gate meets its threshold and will assert a DATA result value and the result value for the composite gate will assert DATA/NULL to express the encoded Intermediate value. The result value remains intermediate until all four input wires are DATA, then both outputs assert DATA. When one input wire returns to NULL the threshold 4 gate asserts NULL, and the asserted result is again DATA/NULL expressing the encoded Intermediate value. The Intermediate encoding is maintained until all the input wires become NULL, at which point both outputs assert NULL. Thus, the Intermediate value composite gate **enforces the completeness of input criteria for both data <u>and</u> NULL with playthrough threshold gates.**

Intermediate encoding is a theoretically complete and general solution, but is very expensive in terms of resources. The feedback solution has a non-critical time constraint, but is more practical and economical. Both solutions enforce the completeness of input criteria for both DATA and NULL, and these properties scale up for circuits as a whole, as shown in Figure 10 for a full-adder using the feedback solution.

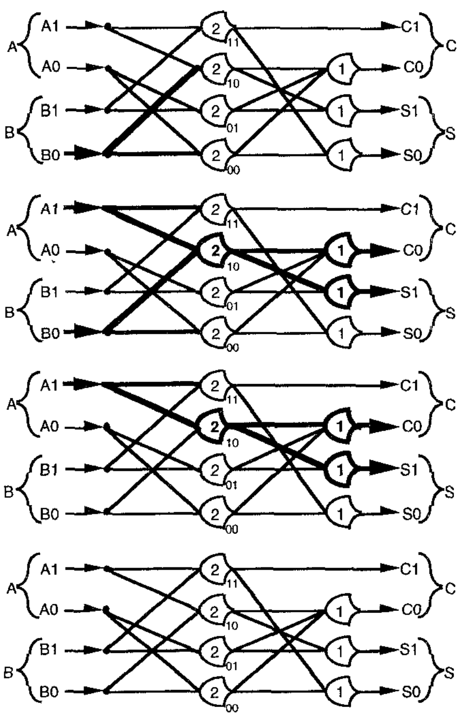

Fig. 10. NULL-DATA cycle for hysteresis gate circuit

Beginning with the circuit in an all NULL state, DATA is asserted on B0. The result values remain NULL. When A1 asserts DATA, a complete input data set is present, and C0 and S1 assert data. (One result value in each output group is asserted.) This constitutes a complete result data set. Thus, the circuit as a whole enforces the completeness of input criteria for DATA in relation to NULL.

When B0 returns to NULL, the outputs circuit continue asserting DATA. Only when

A1 also returns to NULL do the result values return to NULL. Thus, the circuit as a whole enforces the completeness of input criteria for NULL.

As with the three-value and four-value Boolean logic examples, the completion of resolution of a complete input data set and the readiness of the circuit to receive a new input data set to resolve can be determined by simply monitoring the result values. **The circuit is symbolically complete and manages its own interactions with its external environment.**

4: Interesting observations on single data value NULL Convention Logic™ circuits.

A single data value NULL Convention Logic™ circuit can be conveniently monitored for faults as shown in Figure 11.

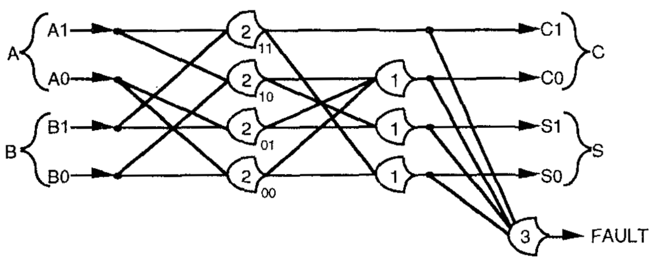

Fig. 11. Fault monitoring NULL Convention Logic circuit.

If exactly one DATA value is asserted in each output group, then the resolution of a complete input data set is correct. If three DATA result values simultaneously assert, then an error occurred, and the threshold three gate will signal an error. If the circuit only asserts one DATA result value the circuit will fail to announce completion of resolution, which can be detected with a watchdog timer. **A NULL Convention Logic circuit will always either:**

1. **Assert a correct result**
2. **Fail to complete resolution**
3. **Assert an explicit error signal**

A fault will be detected as soon as it causes an actual resolution error. This holds for all single point faults. Complementary faults are required to produce a valid encoding that may not be detected.

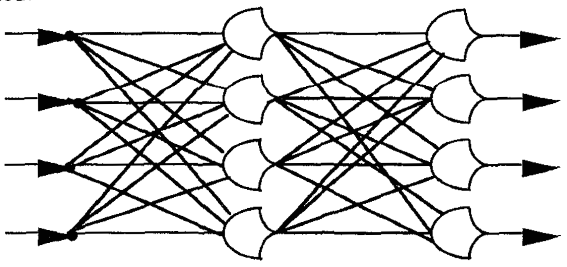

Fig. 12. Similarity to neural nets

Another observation is that single data value NULL Convention Logic circuits are similar to neural nets as shown in Figure 12.

The single data value NULL Convention Logic circuit performs the same discrete symbolic processing as thae Boolean circuit, while at the same time being a more complete and autonomous expression of the process than the Boolean logic circuit and furthermore, resmbles a neural net expression. The fully connected neural net of Figure 12, with an input layer, a hidden layer and an output layer, can be configured to express identically the NULL Convention Logic circuit by setting the thresholds of each node and setting the weights of each connection appropriately to zero or one. The NULL Convention Logic circuit might be viewed as a pretrained neural net.

5. The Asynchronous Register

To build a system out of NULL Convention combinational logic circuits we must be able to synchronize and register the wavefronts among the component combinational circuits. This can be accomplished purely in terms of NULL Convention Logic. Registration can be accomplished with a rank of NCL gates, and synchronization can be accomplished with a single NCL gate as shown with the pipeline in Figure 13.

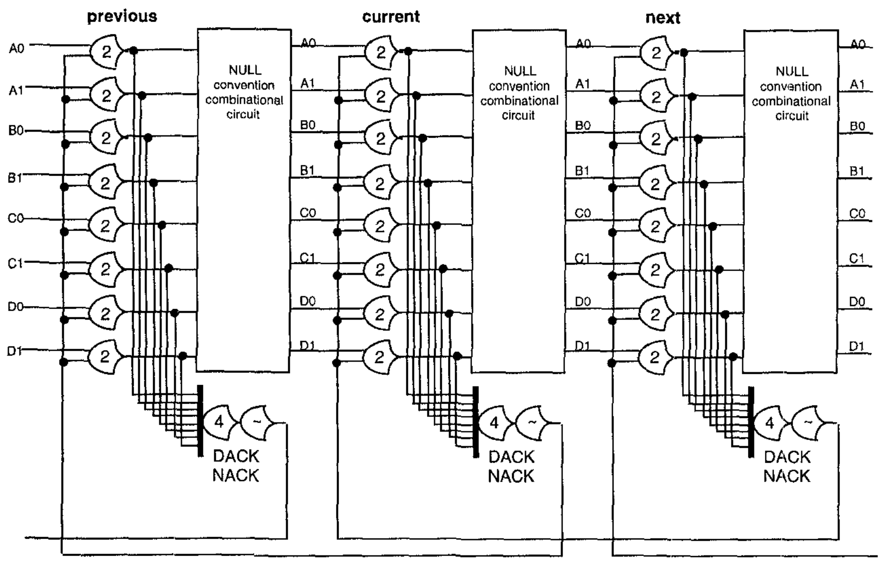

Fig. 13. NULL Convention Logic™ asynchronous pipeline

The rank of threshold 2 gates between each combinational circuit is the register and the threshold 4 gate below each combinational circuit is the synchronizer which we will call the watcher. There are four mutually exclusive groups of two presented to each register so four DATA values is a complete DATA input set. The watcher will recognize the complete DATA set of four values and assert a DATA value which is inverted by the ~ gate to NULL. This means that a complete DATA wavefront has been received from the previous circuit and a request for the next NULL wavefront is sent to the previous circuit. When a complete NULL wavefront is received, the watcher will assert NULL, which the ~ gate will invert to DATA. This means that a complete NULL wavefront has been received from the previous circuit and a request for the next DATA wavefront is sent to the previous circuit. These request lines are presented to the register gates. When a request line is DATA, a DATA wavefront will be allowed to play through the register gates and that wavefront will be stored as long as the request line remains DATA. The same is true for the NULL request and the NULL wavefront. The behavior of the registers is discussed in detail elsewhere. Here we will just point out that the expression of the communication among the combinational circuits and the combinational circuits themselves is purely in terms of NCL. There are no extra-logical considerations in the expression.

6: Properties of NULL Convention Logic™ Circuits

NULL Convention Logic conveniently and straightforwardly delivers all of the traditionally expected benefits of asynchronous circuit design with considerably less design cost and risk.

Lower design cost and risk. The clock is eliminated with all its attendant design complexities and risks including clock skew. The system can be designed in parts and then directly composed. There are no global coordination issues as with synchronous systems.

Lower power consumption. Only portions of the system that are doing useful work consume power. There is no spurious switching of transistors. The NULL state is an inherent and automatic power idle state. The clock driving power is eliminated. NCL systems operate entirely in terms of synchronized wave fronts of level transitions in one direction. There are no pulses or edge triggering involved in the circuit behavior. The asynchronous nature of NULL convention Logic circuits distributes the demand for power.

Average case performance. NCL circuits signal their completion of resolution and their readiness to accept new input data. The processing delay of some circuits is dependent on the particular input data set and NCL circuits take advantage of this.

Convenient technology migration. Because the logic is inherently delay insensitive it is insensitive to the behavior properties of the physical implementation. Null Convention Logic circuits are insensitive to changes in implementation technology, to changes in scale, and to propagation delay changes due to aging or manufacturing variations eliminating portability and evolvability issues.

Automatic adaptation to physical properties. The delays of the various circuit elements change differentially with the change in physical parameters such as voltage, temperature, age, manufacturing variations and different implementation environments. Since the circuits are delay insensitive, they will continue to operate correctly over a large range of variation of these physical parameters.

Reliability. All failure modes due to timing problems (race, hazards, skew, etc.) are eliminated. NCL provides advantages for design cost and risk, reliability of circuit performance and evolvability beyond the barriers of complexity and feature size facing clocked Boolean logic.

Speed of operation. NULL Convention Logic circuits are speed competitive even though they require two propagation cycles per unit of processing. They will operate at the full rate the logic and material allow also taking advantage of average case propagation behavior. There are no margins added onto worst case propagation delays as with clocked circuits.

Two value NULL convention logic preserves all the advantages of traditional Boolean logic (two values, simple gates, straightforward synthesis) while, additionally, providing self determined, locally autonomous, self synchronizing, delay insensitive, and fault detecting behavior.

7: Progress to date in proof of concept circuits

Five NCL chips with 12 circuits have been designed and fabricated to date. These circuits range from simple shift registers to adders and 4-bit multipliers. All circuits have been designed with no analysis of circuit component delays. All circuits have been tested and operated correctly on the first fabrication pass with speeds ranging from 75 MHz for the slowest multiplier to 370 MHz for the fastest shift registers. The circuits have been tested to operate correctly over continuously changing power supply voltage ranging from 0.7 volts to 9.0 volts and over wide temperature ranges.

6. References

1. C. L. Seitz, "System Timing." in Introduction to VLSI Systems, ed. by Carver Mead and Lynn Conway (Reading, Mass., Addison-Wesley, 1980), pp. 242-262.
2. Stephen H. Unger, Asynchronous Sequential Switching Circuits (New York, Wiley-Interscience, 1969)
3. 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, January 1992, pp. 2-10.
4. Ivan E. Sutherland, "Micropipelines", Communications of the ACM , Vol. 32, No. 6, June 1989, pp. 720-738.
5. J. A. Brzozowski and C-J. H. Seger, Asynchronous Circuits, (New York, Springer Verlag, 1994)
6. M. Kishinevsky, A Kondratyev, A. Taubin and V. Varshavsky, Concurrent Hardware (The theory and practice of self timed design), (New York, Wiley, 1994)
7. Karl M. Fant and Scott A. Brandt, Null Convention Logic System, US patent 5,305,463 April 19,1994.