PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 1/00	A1	(11) International Publication Number: WO 92/21083
		(43) International Publication Date: 26 November 1992 (26.11.92)

(21) International Application Number: PCT/US92/04132

(22) International Filing Date: 15 May 1992 (15.05.92)

(30) Priority data:
702,016 17 May 1991 (17.05.91) US

(71) Applicant: THESEUS RESEARCH INC. [US/US]; 1916 Southeast Franklin Avenue, Minneapolis, MN 55414 (US).

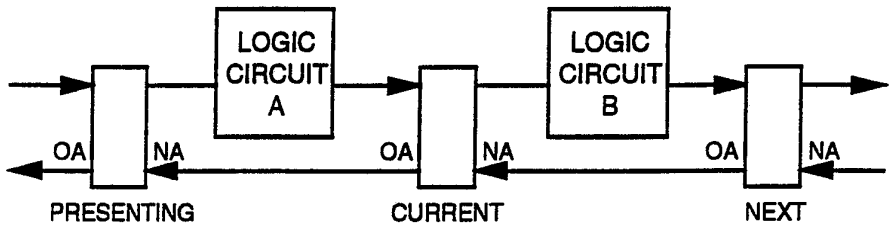
(72) Inventors: FANT, Karl, M. ; 1916 Southeast Franklin Avenue, Minneapolis, MN 55414 (US). BRANDT, Scott, A. ; 600 Second Street, Northeast, Hopkins, MN 55343 (US).

(74) Agent: SKINNER, Joel, D.; 332 Minnesota Street, 3100 First National Bank Building, St. Paul, MN 55101 (US).

(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BF (OAPI patent), BG, BJ (OAPI patent), BR, CA, CF (OAPI patent), CG (OAPI patent), CH, CH (European patent), CI (OAPI patent), CM (OAPI patent), CS, DE, DE (European patent), DK, DK (European patent), ES, ES (European patent), FI, FR (European patent), GA (OAPI patent), GB, GB (European patent), GN (OAPI patent), GR (European patent), HU, IT (European patent), JP, KP, KR, LK, LU, LU (European patent), MC (European patent), MG, ML (OAPI patent), MN, MR (OAPI patent), MW, NL, NL (European patent), NO, PL, RO, RU, SD, SE, SE (European patent), SN (OAPI patent), TD (OAPI patent), TG (OAPI patent).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: NULL CONVENTION SPEED INDEPENDENT LOGIC



(57) Abstract

An information processing system comprising at least one information processing unit (A, B). The information processing unit has at least one information processing member which resolves allowed values. Allowed values include at least one data value and at least one non-data value. At least one non-data value is a null value. The system further comprises a plurality of information transmission elements (OA, NA) for transmitting values to and from the information processing unit (A) and the information processing member.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**FI**	Finland	**ML**	Mali
AU	Australia	**FR**	France	**MN**	Mongolia
BB	Barbados	**GA**	Gabon	**MR**	Mauritania
BE	Belgium	**GB**	United Kingdom	**MW**	Malawi
BF	Burkina Faso	**GN**	Guinea	**NL**	Netherlands
BG	Bulgaria	**GR**	Greece	**NO**	Norway
BJ	Benin	**HU**	Hungary	**PL**	Poland
BR	Brazil	**IE**	Ireland	**RO**	Romania
CA	Canada	**IT**	Italy	**RU**	Russian Federation
CF	Central African Republic	**JP**	Japan	**SD**	Sudan
CG	Congo	**KP**	Democratic People's Republic of Korea	**SE**	Sweden
CH	Switzerland			**SN**	Senegal
CI	Côte d'Ivoire	**KR**	Republic of Korea	**SU**	Soviet Union
CM	Cameroon	**LI**	Liechtenstein	**TD**	Chad
CS	Czechoslovakia	**LK**	Sri Lanka	**TG**	Togo
DE	Germany	**LU**	Luxembourg	**US**	United States of America
DK	Denmark	**MC**	Monaco		
ES	Spain	**MG**	Madagascar		

NULL CONVENTION SPEED INDEPENDENT LOGIC

This invention relates to information processing systems and methods for manipulating and resolving data, and particularly to computer systems and methods. The invention is particularly useful for building circuits, and for building processors utilizing a plurality of such circuits.

Traditional electronic logic circuits are composed of continuously acting logic elements which are continuously asserting a potentially legitimate result. When new input data is presented to a circuit, the result asserted by the circuit might change several times before settling to the correct result. In general it is not possible to determine, in terms of the circuit output itself, when the correct result is asserted by the circuit.

The determination of completion has generally been accomplished by a representation external to the circuit, most often a system clock. Other types of external representations have also been used, such as a delay line associated with each circuit. Such external systems are provided to indicate when the output of the circuit is VALID. They do so by allowing sufficient time for the logic circuit to settle to a correct result before declaring the result values VALID.

External timing of the logic circuit requires that the control and logic representations be carefully engineered to coordinate their timing characteristics. Because they must be coordinated or synchronized, such systems are typically referred to as synchronous systems. In the case of a system clock, the circuit has to be carefully designed so that it is assured of settling to a correct result within one clock period. Similarly, a delay line must be long enough to accommodate the timing of the circuit and it must also be guaranteed that it's delayed control signal will be stable for a long enough time. Such synchronization considerations place significant complications on system design.

The existing technology of speed independent or Muller circuits does not postulate a NULL value integrated into the primitive transform elements. It relies instead on Boolean logic gates carefully arranged to provide the whole circuit with a specific resolution behavior. This cannot, however, eliminate all possible hazards due to circuit element delays. Transmission elements can introduce delays that could cause incorrect function of the circuit. The existing technology is not logically complete in that physical timing characteristics of the circuit element still have to be considered in any circuit design.

Speed independent circuits put the burden of completion integrity on the configuration of the circuit itself and cannot achieve a purely logical expression of the circuit. The NULL convention puts the burden of completion integrity on the primitive transform elements. This allows a purely logical expression of a circuit quite independent of the physical timing behavior of

all circuit elements.

A known concept using a non-data representation as a control means is the spacer code of dual rail encoding associated with speed independent circuits. Dual rail encoding, however, is an interface signaling protocol between circuits and is not a concept associated with the internal organization of the circuits themselves.

Despite the need for a system or environment in the art which enables autonomously acting and coordinated logic circuits to implement independently acting and locally controlled process representations without the need for external control representations, and which overcomes the limitations and problems of the prior art, none insofar as is known has been proposed or developed.

Accordingly, it is an object of the present invention to provide an information processing system and method for manipulating and resolving data. Another object of the invention is to provide a system and method which is useful for constructing information processing units and members such as circuits and gates and for constructing configurable processors utilizing a plurality of such circuits and gates.

Yet another object of this invention is to provide a system and method which enables processors having autonomously acting and coordinated logic circuits to implement independently acting locally controlled process representations without the need for external control representations. A further object of the invention is to provide a system and method as described above which utilizes the representation of control as a value with respect to the logic circuits and gates themselves.

Still another object of the invention is to provide a system and method of utilizing a null convention in logic and processor design and function.

SUMMARY OF THE INVENTION

The present invention provides an information processing system for manipulating and resolving data, which has at least one information processing unit for resolving combinations of data and non-data values, for example a logic circuit. The information processing unit or units each have at least one information processing member, for example a logic gate, also for resolving data. The unit further has a plurality of information transmission elements, for example conductors, each element transmitting the data to and from the one or more members or units.

The information transmission elements may be electrical, optical or any other transmission means known in the art. Data manipulated and resolved by the system and the system components described above consist of values which, for example, may represent physical states on or in the elements, members and units. Such physical states represent voltage, optical energy or any other medium which may be utilized to convey information pertaining to velocity, temperature, or angular position, for example. Importantly, the system and its components also transmit and resolve non-data values.

Each information processing member and unit has one or more information transmission

elements connected to itself for both input or presentation of values, and for output or assertion of result values. And, although elements are capable of transmitting only one value at a time, members and units are capable of resolving combinations of values (information) which are presented either individually over time via a single element, or simultaneously via multiple elements.

Allowed values are those which are resolvable by information processing members and units, and consist of at least one data value and at least one non-data value, at least one non-data value being a null value. The set of data values includes a single value or two values, for example, such as is used in traditional binary logic. In addition to the null value, one or more intermediate values, which are distinct from the null value and the data values, may be included in the set or group of allowed values.

Accordingly, in one embodiment of the information processing system of the present invention there are two allowed values, the first allowed values being a data value and the second allowed value being a null value. In another embodiment, there are three allowed values, the first allowed value being a data value, the second allowed value being a data value, and the third allowed value being a null value. In still another embodiment, there are three allowed values, the first allowed value being a data value, the second allowed value being an intermediate value, and the third allowed value being a null value. In a final embodiment of the present invention, there are four allowed values, the first allowed value being a data value, the second allowed value being a data value, the third allowed value being an intermediate value, and the fourth allowed value being a null value.

Each information processing unit maps from combinations of values presented to it to combinations of values to be asserted by it. To achieve play-through, the information processing members resolve values by asserting a value for each combination of values presented to it, such that (1) for valid combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values, and (2) for invalid combinations of presented values the asserted value is a null value.

To achieve hysteresis, the information processing members resolve values by asserting a value for each combination of values presented such that (1) for valid combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values which remains asserted until the combination of presented values becomes all-null, and (2) for all-null combinations of presented values the asserted value is a null value which remains asserted until the combination of presented values becomes valid.

With respect to intermediate value resolution, information processing members resolve values by asserting a value for each combination of values presented such that (1) for valid combinations of presented values the asserted value is a data value dependent on the particular combination of presented values, (2) for combinations of values including intermediate values the asserted value is an intermediate value and (3) for all-null combinations of presented values the asserted value is a null value.

The information processing unit cycles through resolution and non-resolution states to allow determination of the unit's (1) completion of a data resolution and (2) readiness to perform another data resolution. A resolution state occurs when the unit is presented with a valid combination of values and is asserting a valid combination of values. A non-resolution state occurs when the unit is presented with an all-null combination of values and is asserting an all-null combination of values.

The information processing unit can perform a specific data resolution function by asserting a predetermined combination of values for each combination of presented values. The unit can further store values by asserting a combination of values equal to a previously presented combination of values. The unit can select values by asserting a combination of values which are a subset of a first combination of presented values relative to a second combination of presented values, and can also distribute data by asserting a combination of values equal to a first combination of presented values as a subset of it's combination of asserted values relative to a second combination of presented values.

The information processing system further comprises bounding means for asynchronously coordinating value presentation to the information processing units. Each bounding means comprises a null-valid detector for determining a unit's (1) completion of a data resolution and (2) readiness to perform another data resolution; means for storing value combinations, and means for communicating with other bounding means.

The null-valid detector is preferably an information processing unit which asserts a null value when it's combination of presented values is all-null and continues asserting a null value until it's combination of presented values becomes valid, whereupon it asserts a data value and continues asserting a data value until it's combination of asserted values becomes all-null. The means for storing is preferably a unit which asserts a combination of values equal to a previously presented combination of values. The communication means comprises:

(a) means for informing all existing preceding bounding means that a first null-valid detector has detected a valid combination of presented values, the valid combination of presented values has been stored in the storing means, and that all existing preceding bounding means can now assert an all-null combination of values;

(b) means for informing all existing preceding bounding means that the first null-valid detector has detected an all-null combination of presented values and all existing preceding bounding means can now assert a valid combination of values;

(c) means for detecting that all existing succeeding bounding means have detected and stored a valid combination of values resulting from the valid combination of values stored in the storing means and asserted by the bounding means, whereupon an all-null combination of values can be asserted by the bounding means; and

(d) means for detecting that all existing succeeding bounding means have detected an all-null combination of values asserted by the bounding means, whereupon a valid combination of values can be asserted by the bounding means.

A generally configurable information processing system is constructed of an information processing unit for resolving combinations of values, a unit for storing combinations of values, a unit for configuring value presentation relationships relative to a second combination of values, and bounding means for asynchronously coordinating value presentation between units.

The configuring units, for example a distributor or selector, configure value presentation relationships among at least one resolving unit and at least one storing unit relative to a combination of directive values, for example program instructions, asserted by a first bounding means and presented as the second combination of values of the configuring unit.

A resolution configuration exists during the presentation of a valid combination of directive values to the configuring unit resulting in the presentation of a valid combination of values to the storing unit. A non-resolution configuration exists during the presentation of an all-null combination of directive values to the configuring unit resulting in the presentation of an all-null combination of values to the storing unit.

Data resolution is accomplished by a progression of alternating resolution configurations and non-resolution configurations relative to a progression of combinations of directive values.

These and other benefits of this invention will become clear from the following description by reference to the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows the general structure of NULL convention logic circuits bounded by boundary elements,

Fig. 2 shows the Boolean truth tables with the NULL value added,

Fig. 3 shows an acyclic combinational network of association relationships,

Fig. 4 shows a NULL convention combining element with feedback specification,

Fig. 5 shows the Boolean truth tables with NULL and INTERMEDIATE values added,

Fig. 6 shows a Boolean logic half adder expression,

Fig. 7 shows a NULL convention threshold logic expression of the half adder process,

Fig. 8 shows a fault monitoring NULL convention threshold logic expression,

Fig. 9 shows a NULL convention threshold element with feedback,

Fig. 10 shows an INTERMEDIATE encoded threshold 2 combining element,

Fig. 11 shows an INTERMEDIATE encoded threshold 2 element with 3 inputs,

Fig. 12 shows a half-adder expressed with INTERMEDIATE encoded elements,

Fig. 13 shows a bondary element - combinational expression unit,

Fig. 14 shows the boundary element internal structure,

Fig. 15 shows interacting boundary elements,

Fig. 16 illustrates a pipeline of boundary elements,

Fig. 17 illustrates a fan-in configuration of boundary elements,

Fig. 18 illustrates a fan-out configuration of boundary elements, and

Fig. 19 illustrates a generally configurable expression composed from NULL convention

expressions and boundary elements.

DESCRIPTION OF THE PREFERRED EMBODIMENT

I. OVERVIEW

The present invention provides a new information processing system or representational environment which eliminates the need for synchronization means such as clocks, delay lines, or peripheral transition signaling means. The system itself provides a NULL (N) value which represents control. The NULL value is utilized as a flag of invalidity such that when it is asserted, data on a particular input is not valid. If, on the other hand, a nonNULL or data value is asserted, then the input is valid. The NULL value is added to the traditional binary values of TRUE (1) and FALSE (0) to yield a three-value logic system. Alternatively, the NULL value can be added to a single data value to yield a two-value logic system. In this case a dual-rail encoding can be used to obtain two data values necessary for conventional logic representations.

Since the representation of control is added as an extra value to the logic of an information processing circuit or unit itself, instead of in the form of an independent external representation, the circuit is made autonomously determinable and the engineering coordination and synchronization complications associated with two independent representations are thus avoided. The circuits and information processing gates or members provided thereby autonomously express their own completion of data resolution, thus enabling them to operate completely asynchronously in relation to each other.

An essential characteristic of such asynchronous transmission control is that events are controlled locally and are no longer constrained to occur at specific times or during discrete intervals. Accordingly, circuit design is made more simple and straightforward. Moreover, this can be accomplished using standard binary logic.

As shown for example in the table below, the addition of a NULL value to an existing logic circuit does not alter the established logic of the representation. For example, standard two-value logic gates are replaced by corresponding three-value logic gates. The NULL value indicates meaninglessness within the representation.

	N	0	1
N	N	N	N
0	N	0	0
1	N	0	1

AND

	N	0	1
N	N	N	N
0	N	0	1
1	N	1	1

OR

N	N
0	1
1	1

NOT

The NULL value is added to the existing truth tables so that if either input to a gate is NULL, the output will be NULL. An input to a gate is VALID only when both inputs are nonNULL. An INVALID input is one with at least one NULL value. As soon as both inputs are nonNULL, the

input is valid and a nonNULL (DATA) value will be asserted. No gate will assert a VALID output unless it's input is VALID. Each gate exhibits a distinct resolution event (the output transition from NULL to nonNULL) whenever a VALID input is presented to it. Each gate indicates the completion of it's own logical transformation process.

A practical example of the use of the NULL value convention is an AND logic gate implemented with a transistor circuit and having the three voltage values of +5, 0, and -5. If 0 is assigned to represent the NULL value, the -5 and +5 are VALID data values. When 0 is asserted to either input of the AND gate, the output will be forced to 0. If non-0 values (-5 or +5) are asserted on both inputs to the gate, the output is the logical AND of the two values the non-0 voltages represent. If 0 is never asserted (except perhaps in transition), then the gate acts just like a standard logic gate.

As previously discussed, the NULL value convention can also support single-data-value representations. Referring to the table below, this form of representation only has two values, NULL (N) and DATA (D). Two-value NULL convention logic is related to threshold logic. Since there is only one nonNULL value, data must be represented by quantities of DATA values. In the truth tables, D is DATA and N is NULL. For the threshold 1 table a single DATA value will set the output to DATA. For the threshold 2 table both inputs must be DATA for the output to be DATA.

	N	A
N	N	A
A	A	A

Threshold 1

	N	A
N	N	N
A	N	A

Threshold 2

In addition to utilizing the NULL value at the gate level of a circuit, the NULL value may also be utilized at the circuit level to indicate the proper transfer of data between circuits. The representation element that manages the presentation of data between two NULL convention logic circuits is called a boundary element. As shown in FIGURE 1, null convention logic circuits bounded by such elements A and B behave as independently proceedable units that accept data, process it, and then pass it on to other circuits or units. The input data, in addition to expressing values, is either VALID or all-NULL, thus incorporating the control information necessary for determining it's representational validity

The assertion and presentation of value combinations involves a handshake protocol convention between two circuits. In the protocol of two control variables each with two values, the input data itself serves as one of the control variables. The input data in addition to expressing the values of the data is either VALID or all-NULL. These two states of VALID and all-NULL in the data itself provide two values for one of the handshake variables. So the data stream itself becomes one of the handshake variables.

An input is received and stored in a current boundary element and stably asserted until the

next boundary element can receive it. Then the current boundary element is freed up to receive another input. Data flows through a representation as packets from boundary element to boundary element. As discussed with respect to FIGURE 1, between boundary elements there can be any NULL convention logic circuit that will perform some transform on the data. A NULL convention logic circuit bounded by boundary elements will cycle through completely NULL states and completely VALID states so that the NULL convention criteria that allows determination of the completion of the resolution of a NULL convention logic circuit is satisfied.

Boundary elements can be associated in various structures, for example a simple pipeline structure. The pipeline is completely autonomous. A VALID input presented to the first element in the pipeline will begin a sequence of interactions that will propagate the data from element to element all the way through the pipeline. As each element sees a VALID input, it will receive it and assert it to the next element in the pipeline. Several data packets can be simultaneously propagating through the NULL convention pipeline just like any other pipeline. The propagation rate of the pipeline is determined by the longest transmission delay between two boundary elements. Boundary elements can also be associated in fan-in, fan-out, and circular configurations.

A VALID input at a circuit representing a desired process utilizing associated boundary elements will trigger the progression of events that is the represented process. As the events proceed, the circuit resets itself to be triggered again. As completed processing results arrive at the output of the circuit, their completion is signaled by the assertion at the output of the values comprising the completed processing results themselves.

Boundary elements are preferably used to partition a representation into discrete independently proceedable units that may be complex internally, but that have relatively simple interfaces between them. This allows many representational elements to be simultaneously operating, increasing the throughput of the circuit.

The system of the present invention can process data with NULL values via its information processing members (logic gates) that resolve combinations of data values. In contrast, the prior art technology can merely transmit an indeterminant value along with data values over transmission elements (wires, for example). As a result of this advantage the system is speed independent at every level in that the information processing units (circuits) and members (gates) of this system report their own completion.

The NULL convention logic system provides a representational environment in which autonomously acting and coordinated NULL convention logic circuits can implement independently acting and locally controlled process representations. External global control representations are not needed. The system of this invention is applicable to digital computers, telephone switching systems, and in a variety of control systems, particularly those amenable to asynchronous processing and control.

In the description below, the introduction of the NULL value into logic expression is discussed in terms of two data value expressions and in terms of one data value expressions.

II. NULL CONVENTION LOGIC

1. Introduction

Among its component elements, traditional Boolean logic circuit exhibits time dependent interaction relationships as well as symbolic-value-dependent interaction relationships. The symbolic-value-dependent interaction relationships depend on the truth tables of the logic gates. The time-dependent interaction relationships depend on the propagation delay times of the component elements. These two aspects of expression are independent of each other in that time interaction relationships are irrelevant to the specification of symbolic interaction relationships and the particular symbolic interactions specified are irrelevant to the specification of the time relationships. Yet these two aspects of expression must be carefully and explicitly coordinated for a circuit to correctly resolve its symbolic interactions.

The symbolic interactions are the purpose of the circuit, while the time interaction relationships are intrusions of inconvenient reality. Furthermore, time interaction relationships such as hazards, races, clock skew, and increased capacitance with component aging are a major source of difficulty in designing and manufacturing electronic logic circuits. It would be convenient if circuit expression were dependent only on symbolic interaction relationships, and if a circuit could express the completion of its resolution and the validity of its result values independent of any propagation delays among its component elements. Circuits could be expressed solely in terms of symbolic interaction relationships, and their behavior would be determined solely by their symbolic interaction relationships.

Attempts to eliminate time dependencies are appropriately referred to as speed-independent circuits. They are also referred to as asynchronous circuits or Muller circuits, after D. E. Muller, who pioneered the pursuit in the late 1950s. These attempts are always expressed within the traditional context of binary data and Boolean logic operations and focus on symbolically deterministic transmission of values between circuits rather than on symbolic determination of the circuits themselves. The circuits themselves are viewed as standard Boolean combinational logic circuits and still rely on a time-dependent relationship to assert the validity of their result values. Typically, a delay line associated with the circuit and carefully coordinated with the propagation delay time of the circuit, locally asserts the validity of the circuit's result values, much as the traditional system clock asserts global result value validity. While the pursuit of speed-independent circuits has produced many interesting results, it has not delivered a complete solution. Speed-independent circuits still rely on a few time-dependent assumptions such as the above mentioned delay lines associated with combinational circuits and the insignificant difference in transmission delay of local transmission lines (equipotential regions).

The solution to symbolically determined circuit behavior is to be found in a logic different from traditional Boolean logic. Time relationship difficulties arise with traditional Boolean logic circuits, essentially because each element of the circuit (combining elements {logic gates} and

transmission lines) resolves and expresses insufficient meaning. Two data values can be expressed symbolically by a single element, but the same element cannot express when the data is valid and when it is invalid. A single control element (clock) can symbolically express the times of validity and invalidity but cannot express what is valid or invalid. These two partial expressions must be combined to express the complete meaning of, "valid data," and the combination of the two symbolic expressions, asserted by two independent elements, necessarily has a time relationship independent of their symbolic interaction relationship.

There are, in fact, two time relationships between the two expressions. One might be called a primary time relationship and the other a secondary time relationship. The primary time relationship is purely symbolic and determines when the data is actually valid. The secondary time relationship depends solely on the propagation times of the elements and determines whether the expression of data validity (a clock value, for instance) coincides with an actually valid expression of data (the result values of a circuit, for instance). For traditional logic circuits to behave correctly, the secondary time relationship between these two expressions must be carefully coordinated. The expression of data validity must arrive strictly after the data is actually valid and the valid data must be stably asserted for the duration of the expression of data validity. This will be called a critical secondary time relationship because the two symbolic expressions have an absolute and necessary time synchronization relationship. A non critical secondary time relationship is one that is propagation-delay-dependent but does not have an absolute synchronization relationship.

For relationships between expressions in a circuit to be purely symbolic, critical secondary time relationships have to be eliminated. The only way to eliminate the secondary time relationship is to combine the two partial meanings into a single complete meaning expressed in a single mutually exclusive assertion domain by a single component element. Each primitive element of a digital electronic logic circuit, such as a wire or logic gate, is a mutually exclusive assertion domain. Each of the meanings above (data and control) was expressed in a separate mutually exclusive assertion domain, each asserted by a single circuit element. A mutually exclusive assertion domain can express two or more values (meanings), but can only express one value at a time. Each digit position of the Hindu-Arabic number system, for example, is a mutually exclusive assertion domain. Each position can assert only one of the ten possible values at time.

Combining the two separate meanings of "data" and "validity of data" into the single expression of a mutually exclusive assertion domain results in a unique family of logic with its own primitive expression of meaning and primitive operations to combine those meanings. The new logic will first be introduced by enhancing the traditional Boolean logic. It will then be shown that the new logic can be implemented effectively with a single data value, resulting in a form of discrete threshold logic. Finally, it will be shown how logic expressions can be combined to express symbolically determined systems.

2. Data validity assertion and the NULL value

The assertion of data validity will be accomplished by adding a third value, NULL, to the traditional two-value mutually exclusive assertion domain of Boolean logic. Each transmission line, for instance, can now assert one of three values: TRUE (T), FALSE (F), or NULL (N). The values T and F will be collectively referred to as DATA values. The NULL value asserts meaninglessness in relation to data. It means that this mutually exclusive assertion domain is not asserting meaningful data. If a DATA value is asserted, the DATA value is valid. A transition from NULL to a DATA value marks the beginning of validity. A transition from a DATA value to NULL marks the end of validity.

This use of a NULL value is well understood within the discipline of speed-independent circuits as a component of a transmission protocol between circuits. Its use is embodied in the dual-rail encoding convention shown in Table I, where the three values are encoded on two wires with two values. The two wires form a mutually exclusive assertion domain maintained by convention.

11 = illegal state
10 = TRUE
01 = FALSE
00 = NULL

Table I. Dual rail encoding.

This encoding, however, is viewed only as a signaling protocol between circuits. The circuits themselves are still viewed as traditional Boolean logic circuits.

Circuit behavior is the essence of the issue, of course, and the behavior of the circuit depends on how the primitive combining elements behave in resolving combinations of presented value sets. Resolution of combinations of value sets with two DATA values and one NULL value requires three-value logic.

The NULL value is added to the traditional specifications of the combining elements of Boolean logic primitive combining elements such that if any input value is NULL, a NULL result value is asserted. A DATA result value is asserted only when all of the input values are DATA. FIGURE 2 shows the NULL value added to the traditional binary truth tables to specify the behavior of NULL convention combining elements.

As can be seen from the truth tables of FIGURE 2, if either input is a NULL value, the combining element will assert a NULL result value, and it will not assert a DATA result value until both input values are DATA. This will be called the *completeness of input criteria.* A DATA result value is not asserted unless a completely formed input set of DATA values is presented to the

combining element. Furthermore, the asserted result DATA value is the correct result value for the presented input DATA value set.

The combining element begins with all NULL values presented to the input and asserting a NULL result value. When a complete input DATA set is presented, the resolution of that data set is completed when the asserted result value of the combining element changes from NULL to a correct result DATA value. The combining element simultaneously asserts the correct result DATA value, asserts the validity of the result DATA value and establishes the completion of its resolution. No other element and no secondary time relationship is associated with the assertion of these three facts.

3. Combinations of combining elements

This behavior scales up through combinations of combining elements, as shown in FIGURE 3, to endow a large combinational expression of interconnected elements with the same behavior as a single element.

Since the input of the result elements of the combinational expression is dependent on the input of the combinational expression itself the DATA input to the result elements cannot be complete and they will not assert a complete result DATA value set until the input DATA value set of the combinational expression is complete. If one input value remains NULL, at least one result value of the combinational expression will remain NULL.

Assume the combinational expression starts in an all-NULL state and a complete input DATA set is presented and maintained on the input. As each combining element sees a complete input DATA set, it will assert a correct DATA result value. Result DATA values will propagate through the combinational expression in an orderly wavefront of correct result DATA values until a complete result DATA set is asserted for the combinational expression. No invalid or spurious values will be asserted anywhere. If any asserted value inside the expression remains NULL, at least one result value will remain NULL. The combinational expression asserts the correct result values for the expressed process, the validity of those values and the completion of its resolution when all its result values become DATA.

No matter how long it takes DATA values to propagate to the input of a combining element, the combining element does not switch its result value until all of the input data values have arrived. When the result value does switch, it switches directly from the NULL value to the correct result values for the presented input DATA set. The combining element does not assert any spurious result values due to partially formed input DATA sets. To put it somewhat differently, the completeness of input criteria makes each combining element a synchronization node.

The addition of the NULL value did not change the transform specifications for the DATA values. The NULL convention combining elements can replace the standard binary logic gates of a standard combinational logic circuit one for one, and the circuit will provide the identical logic function as before. It will simply resolve in a more orderly manner and assert its own completion,

which it was not capable of doing with the standard binary logic combining elements.

If a combinational expression asserts completion of resolution when its asserted result values have changed from all NULL to all DATA, it follows that an expression is ready to receive a new presented input value set when it has returned to an all-NULL state and all of its asserted result values are NULL. Since a combinational expression is solely responsive to its presented input values, the input DATA values must be unpresented and all NULL input values presented. The NULL values will propagate through the expression, setting all DATA values to NULL until all the result values are NULL.

While the specifications for the combining elements discussed so far provide for an orderly wavefront of transition from all NULL to all DATA, they do not provide for an orderly wavefront of transition from all DATA to all NULL.

4. Data invalidity assertion

The difficulty arises because the completeness of input criteria is not enforced for input NULL value sets in relation to DATA values, as it is for input DATA value sets in relation to NULL values. As can be seen from the truth tables of FIGURE 2, if only one input value becomes NULL, the result value becomes NULL. NULL result values can race through the combinational expression and set all the result values of the expression to NULL, while internal values and even input values may still be DATA values. The expression asserts its readiness to receive a new input DATA set while DATA values are still lurking in the expression, and the new input DATA set may get mixed up with old DATA values from the previous DATA set resolution cycle.

It must be guaranteed that the entire expression is NULL when all of its result values become NULL. There are two solutions to this problem, both of which introduce a hysteresis behavior to the combining element that enforces the completeness of input criteria for both the DATA values and the NULL values. One solution is to add a feedback loop around the combining element, making it a state machine, and involves a noncritical secondary time relationship. The other solution is to add another value to the mutually exclusive assertion domain and is purely symbolic with no secondary time relationship whatsoever.

5. Invalidity assertion with element feedback

The result value is fed back to the input of the combining element making it a three input element. An example combining element specification for this solution is shown in FIGURE 4.

If the result R is NULL, it remains NULL until all of the input values are DATA; then the result R asserts a result DATA value that is a correct transform result for the presented input DATA set. The specific DATA transform is specified by the result values inside the dark box. The example shows an AND transform. Once the result asserts a DATA value, it will not return to asserting a NULL value until both input values are NULL. This combining element specification enforces the input completeness criteria for both DATA and NULL value sets.

This element behavior also scales up for combinational expressions of interconnected combining elements. It should be clear at this point that it is the completeness of input criteria that scales up the delay independent behavior of each combining element to delay independent behavior of an entire combinational expression. If a completeness of input criteria is enforced for each combining element, the completeness of input criteria is enforced for the entire combinational expression and an orderly wavefront of correct result values ensures that no spurious result values will be asserted by the combinational expression.

When starting from an all-NULL state, a combinational expression will assert the completion of resolution of a presented input DATA set when all of the result values have changed from NULL to DATA. After a resolution, and in a completely DATA state, the expression is reset to a completely NULL state by presenting all NULL values to its input. The NULL values will propagate in an orderly wavefront through the DATA values without leaving any lingering DATA values behind. When all the result values are NULL, the combinational expression is completely NULL and is ready to accept a new input DATA value set.

The feedback solution has a secondary time relationship in that a definite amount of time is required for the feedback path to propagate and stabilize the state of each combining element after the result value has already been asserted by that element. It is possible for the next wavefront of change to arrive before the combining element state has stabilized; however, this is a noncritical time relationship because the stabilization period must simply be much shorter than the period between wavefronts of value change. Since the feedback propagation path is much shorter than the propagation path associated with successive wavefronts, this should be easily achievable.

The second solution has no secondary time relationships whatsoever and is purely symbolically determined.

6. Invalidity assertion with the INTERMEDIATE value

The second solution to the assertion of data invalidity is to add another value, called the INTERMEDIATE value, to the mutually exclusive assertion domain. The combining element specifications are defined such that when the presented input is all DATA, a correct result DATA value is asserted; when the presented input is all NULL, a NULL result value is asserted; and for all other cases when input is any mixture of DATA, NULL or INTERMEDIATE, an INTERMEDIATE result value is asserted. The transform specifications for this solution are shown in FIGURE 5.

The actual DATA transform is specified inside the dark box. Notice that the NOT combining element does not have an INTERMEDIATE value. Because it has only one input, the completion of input criteria cannot be violated for either DATA or NULL input. It has no intermediate or incomplete input states. For the same reason, the NOT combining element also does not require the feedback path associated with the previous solution.

The completeness of input criteria is enforced for both DATA and NULL values by all the combining element specifications. A DATA result value is only asserted when all the presented

input values are DATA. A NULL result value is only asserted when all the presented input values are NULL. For all cases of partially formed input the asserted result value is INTERMEDIATE. The combining element asserts the completion of a resolution of a complete presented input DATA set when its result value is DATA. It asserts its readiness to accept new input DATA when its result value is NULL.

Again, notice that these combining elements can be substituted one for one for traditional logic gates in a traditional combinational logic circuit without affecting the logical functionality of the original circuit. Also, because the combining elements enforce the completeness of input criteria for both DATA and NULL, their behavior scales up for combinational expressions of combining elements.

NULL convention logic with the INTERMEDIATE value is purely symbolically determined. Its symbol resolution behavior is not affected in any way by the propagation delay of any element in an expression.

7. Single DATA value logic

It is generally accepted that at least two DATA values are required to express information and that binary logic is a minimal and primitive form of expression. There must, indeed, be at least two values in any primitive mutually exclusive assertion domain to provide a minimum discrimination of meaning, but the meaning of these two values need not both be DATA. One of the two values can be a DATA value and the other can be the NULL value.

Since a single DATA value cannot differentiate DATA meaning, this role must be assumed by the primitive mutually exclusive assertion domains themselves. Each mutually exclusive assertion domain must assert a single distinct data meaning.

With binary logic, a single primitive mutually exclusive assertion domain with two DATA values could assert two mutually exclusive DATA meanings. With a single DATA value, these two mutually exclusive DATA meanings must be asserted by two primitive mutually exclusive assertion domains, only one of which asserts its DATA value at a time. The DATA value indicates that a primitive mutually exclusive assertion domain is asserting its meaning; the NULL value indicates that a primitive mutually exclusive assertion domain is not asserting its meaning. Only one of the two primitive mutually exclusive assertion domains can assert DATA at a time. Both primitive mutually exclusive assertion domains can assert NULL at the same time.

The two primitive mutually exclusive assertion domains form a single mutually exclusive assertion domain that is established by convention instead of by physical necessity, as is the case with voltages on wires. Groups of primitive mutually exclusive assertion domains form what might be called secondary mutually exclusive assertion domains and will be called mutually exclusive assertion groups.

The example group with two primitive mutually exclusive assertion domains is identical to dual-rail encoding, but while dual-rail encoding is viewed as a transmission protocol encoding three values on two lines with two DATA values each, the mutually exclusive assertion group is a

different expression strategy that forms a unique integrated logic system. As many primitive mutually exclusive assertion domains as desired can be included in a mutually exclusive assertion group. Decimal representation, for instance, could be directly expressed with mutually exclusive assertion groups of ten primitive mutually exclusive assertion domains each.

8. Single DATA value combining elements

If there is only one DATA value, and all primitive mutually exclusive assertion domains assert the same DATA value, the differentiation capability for combinational interaction is not the same as with Boolean logic. The primitive mutually exclusive assertion domains presenting input values to a combining element can only assert DATA or NULL. NULL indicates meaninglessness, so it cannot enter into the DATA interaction consideration. Only the DATA values can be significant, and the only property that can be discriminated from combined DATA values is quantity. A combining element can only determine whether a particular quantity of DATA values is present or not; whether a quantity threshold has been reached or not. Therefore, the logic of single DATA value expression must be discrete threshold logic.

The quickest way to grasp single DATA value expression is to compare it with a familiar binary logic expression. A half-adder logic circuit is shown in FIGURE 6. Transmission lines (primitive mutually exclusive assertion domains) A and B of the logic circuit each have two DATA values and express four meanings. Transmission lines C and S, each of which also has two DATA values, express four result meanings.

FIGURE 7 illustrates a single DATA value expression of the half-adder. The number inside each combining element indicates its threshold, or the number of DATA values required for it to assert a result DATA value.

The four input meanings and four result meanings, each expressed with two transmission lines and two DATA values in the binary logic circuit, are now expressed with four transmission lines, each with one DATA value. The formerly physically enforced primitive mutually exclusive assertion domains of A, B, C, and S are now expressed as mutually exclusive assertion groups enforced by convention.

If the A and B input groups assert only one DATA value each, only one of the threshold 2 elements will assert a DATA result value. The enabled threshold 2 element then enables its proper result DATA values through the threshold 1 elements. Only one threshold 2 element will assert a DATA result value, and that value will enable only one DATA value in each result mutually exclusive assertion group.

If the convention, that only one DATA value can be asserted by the mutually exclusive assertion group at a time, is enforced with the presentation of the input DATA values to the expression, the expression will maintain the convention with its asserted result values, and the convention will be maintained at the input of any expression to which each result mutually exclusive assertion group is presented. Any size structure of associated expressions similarly formed will internally maintain the convention of asserting only one DATA value per group. The

convention need only be explicitly enforced at the input interfaces to the structure itself.

Enforcement of the convention can be simply monitored at selected internal places of the expression. The circuit of FIGURE 7 can be monitored with a four input threshold 3 element, as shown in FIGURE 8.

If three or four DATA values are ever asserted at once, the convention has been violated and there is probably a failure somewhere in the structure. The other illegal situation is for both values of one group to be all DATA while the other group remains all NULL. In this case, the expression will be frozen and will fail to resolve, which is also an easily detected and traceable condition. If one group remains all NULL, the circuit will also freeze and will not assert a result DATA value set. The expression either works correctly, freezes, or generates a fault signal.

9. Data invalidity again

The above example will resolve the question of DATA validity but not the question of DATA invalidity. Starting in an all-NULL state, it will not assert a DATA result until a complete input DATA set is presented (one DATA in each group), and the asserted result will be the correct resolution result for the presented DATA set. However, when the input begins returning to NULL, the asserted result can become NULL before all the input values are NULL. This is identical to the first example where the NULL value was added to Boolean logic and, the solutions are identical. The choices are to add feedback to the combining elements or to use an INTERMEDIATE value.

10. Feedback for threshold combining elements

The feedback solution for threshold combining elements that provides the required hysteresis behavior is particularly simple. One just connects the result to the input with a weight of one less than the threshold.

FIGURE 9 shows a threshold 4 combining element with the result value fed back to the input with a weight of 3. If the element is asserting NULL, it requires four input DATA values before it will assert a result DATA value, whereupon the feedback raises the total presented input data values by three. As long as one real input value is DATA, the combining element will remain above its threshold and continue asserting its result DATA value. Only when the last real DATA value has become NULL and the real input is completely NULL will the combining element switch from asserting a DATA result value to asserting a NULL result value. The feedback values become NULL and the combining element will then continue to assert NULL until at least four real input values are DATA. Assuming that four DATA values is a complete input DATA set for the combining element, the combining element enforces completeness of input criteria for both DATA values and NULL values.

A threshold 1 combining element similarly to the NOT element above does not require the feedback connection because its completeness of input criteria cannot be violated for either

DATA or NULL.

If feedback is added to the threshold 2 combining elements in the half-adder example above, its result DATA values will be asserted until the entire input is NULL. Because the completeness of input criteria is enforced for both DATA and NULL, the behavior scales up for combinational expression just like the other examples.

Again, the feedback solution includes a noncritical secondary time relationship between the feedback propagation time and the wavefront propagation time.

11. INTERMEDIATE value for threshold combining elements

Like the above examples, the INTERMEDIATE value is added to the specification of the combining elements such that when the presented input is all DATA, a result DATA value is asserted; when the presented input is all NULL, a NULL result value is asserted; and for all other cases when input is any mixture of DATA, NULL, or INTERMEDIATE, an INTERMEDIATE result value is asserted.

The INTERMEDIATE value solution can be implemented with two real values by encoding value states with multiple elements similar to dual-rail encoding. Table II shows the encoding with the INTERMEDIATE value.

00 -> NULL
10 -> INTERMEDIATE
01 -> INTERMEDIATE
11 -> DATA

Table II. INTERMEDIATE value encoding

FIGURE 10 shows a single threshold 2 combining element for the encoded values. The single encoded threshold 2 combining element is composed of two simple threshold combining elements. The threshold 1 combining element indicates that there is at least one INTERMEDIATE value presented to the input. The threshold 4 combining element indicates that two DATA values are presented to the input. The encoded result value will only assert a DATA code when two DATA inputs are presented. It will only assert a NULL code when two NULL inputs are presented at the input. For all other input configurations, an INTERMEDIATE code will be asserted.

The encoded combining element enforces the completion of input criteria for both DATA and NULL. FIGURE 11 shows a three-input threshold 2 encoded combining element.

The half-adder example of FIGURE 7 implemented with INTERMEDIATE value combining elements would look like FIGURE 12. The i preceding the threshold value indicates an INTERMEDIATE value combining element.

This may seem a rather expensive solution, but again the INTERMEDIATE value solutions

are purely symbolically determined. Their symbol resolution behavior is not affected in any way by the propagation delay of any element in an expression, including the transmission lines.

12. Combinational expression composition with the boundary element

To function properly, NULL convention logic expressions rely on a specific protocol of input presentation. Complete and stably presented DATA input value sets must alternate with all-NULL input value sets. The presentation of an all-DATA value set must be maintained until the combinational expression asserts resolution completion by all of its result values becoming DATA. Then an all-NULL value set can be presented to the input and must be maintained until the expression asserts its readiness to accept a new input DATA set by all of its result values becoming NULL. This protocol must be observed when any combinational expression is interfacing with any other combinational expression.

This input presentation protocol can be managed by a standard interfacing agent called a boundary element. The boundary element can be viewed as associated with the input of each combinational expression. It receives value sets from other expressions and manages the cycling and integrity of input presentation to the expression. A boundary element-combinational expression pair as shown in FIGURE 13 is the unit of combination for systems of associated combinational expressions.

The boundary element must: (1) monitor the input values, (2)determine when they are all DATA, (3) store the input DATA set and stably present it to the combinational expression, (4) determine when resolution is complete, (5) present all-NULL input values to the combinational expression to reset the entire expression to NULL, (6) recognize when the complete expression is reset to NULL, and (7) wait for a new input DATA set. All of this can be achieved with a memory, the ability to recognize all-NULL and all-DATA input value sets, and a familiar two-signal (request-acknowledge) handshake protocol between boundary elements.

The memory is necessary to store the input DATA set so that it can be stably presented to the combinational expression independently of the expressions that originally presented the DATA values. A presenting combinational expression need only present its result DATA values until the input DATA set has been stored in the boundary element memory.

An all-DATA input value set and an all-NULL input value set are the two discrete boundaries of interexpression value set communication. An all-DATA input value set means that an input DATA set is fully formed and can be resolved. An all-NULL input value set must occur before another input DATA set can begin forming.

The unique aspect of the NULL convention boundary element is that the presentation value set itself is the request handshake signal from the presenter to the receiver, with the signal's logical states being all-NULL values and all-DATA values. A single acknowledge handshake signal communicates from the receiver to the presenter. The following conversation summarizes the exchange in the context of the NULL convention. The presenter is in bold text and the receiver is in plain text.

I am presenting a DATA set to you.
(all-DATA value set presented to boundary element)

I have received your DATA set
(DATA value asserted on acknowledge signal)

I understand you have the DATA set
(all-NULL value set presented to boundary element)

Thank you for the DATA set
(NULL value asserted on acknowledge signal)

A boundary element must consist of a memory element, a NULL-DATA detection element, and a protocol resolution element. A boundary element operates via a cooperative interplay of these three expression elements and might look something like FIGURE 14.

The NULL-DATA detect expression monitors the presented input value set and determines when it is all DATA and when it is all NULL. The protocol expression manages the interactions with other boundary elements and also manages the presentation cycles to the combinational expression. The memory buffers the presented input DATA set.

The protocol will be discussed in terms of presenting current and next boundary elements, as shown in FIGURE 15. The presenting boundary element is the previous boundary element presenting an input DATA set to the current boundary element. The next boundary element is the succeeding boundary element to which the current boundary element asserts its result values.

The current boundary element starts in a completely NULL state with an all-NULL presented input value set and asserting all-NULL values to its combinational expression. It is awaiting the presentation of an input DATA set. When an input DATA set is presented by the presenting boundary element, the NULL-DATA detect will recognize this fact and assert a DATA value to the protocol expression. The protocol expression will then perform a receive state sequence, which first stores the presented DATA set into the memory and also presents the memory contents to the combinational expression by enabling it through the NULL gate. It then asserts a DATA value on the acknowledge signal to the presenting boundary element to indicate that the presented DATA set has been received.

The presenting boundary element can then unassert the DATA set and present an all-NULL value set to the current boundary element. The NULL-DATA detect of the current boundary element detects the all-NULL value set and asserts a NULL value to the protocol expression to indicate this fact. The protocol expression then asserts a NULL value on the acknowledge signal to indicate to the presenting boundary element that the receive sequence is

completed. The presenting boundary element may then present a new DATA set to the current boundary element at any time, but the DATA set will not be received and stored until the memory of the current boundary element is free. The memory must stably assert the stored DATA set to the combinational expression until resolution is completed by the combinational expression and the result DATA set has been received by the next boundary element.

The deliver state sequence began when the DATA set was presented from the memory to the combinational expression. The DATA value wavefront will propagate through the combinational expression, and eventually all the result values will be DATA that will be directly presented to the next boundary element as its presented input data set. The next boundary element will recognize the complete DATA set and begin a receive state sequence. It will store the DATA set and assert DATA on the acknowledge signal to the current boundary element. Upon receiving the DATA value on the acknowledge signal, the current boundary element will force the presented output of the memory to all NULL. The memory is now free to store a new DATA set from the presenting boundary element.

The NULL values will propagate through the combinational expression, and eventually all result values will be NULL and an all-NULL input value set will be presented to the next boundary element. At this point, the next boundary element will assert a NULL value on the acknowledge signal to the current boundary element, indicating the completion of the transaction.

If a new DATA set has been stored in the memory of the current boundary element it can now be enabled through the NULL gate and presented to the combinational expression. If no DATA set has been presented, the current boundary element will await the presentation of a DATA set.

The boundary element ensures that the values presented to the combinational expression associated with the boundary element will cycle through all-NULL values alternating with all-DATA values. The value presentation protocol required for the NULL convention expressions to behave correctly is established by the boundary element.

13. Boundary element association structures

Boundary elements can be associated in a variety of structures. The simplest structure is a pipeline, as shown in FIGURE 16.

The pipeline is completely autonomous. When a DATA set is presented to the first boundary element, the pipeline will begin a sequence of transactions that will propagate the DATA set from element to element completely through the pipeline. As each boundary element sees an input DATA set, it will receive it and assert it to the next boundary element in the pipeline. Several DATA sets can be propagating simultaneously, just like any other pipeline. The propagation rate of the pipeline is determined by the longest propagation delay between two boundary elements.

Boundary elements can be associated in a fan-in configuration that builds an input DATA set from several presented DATA sets.

In FIGURE 17, three boundary elements are combining their asserted DATA sets to form the input DATA set for another boundary element. The receiving boundary element will not recognize the input DATA set until all three asserted DATA sets are all DATA. It does not matter when these asserted DATA sets became all DATA; each one will be stably asserted until it is acknowledged and its asserting boundary element completes a deliver state sequence. The DATA value on the acknowledge signal will not occur until the receiving boundary element sees an all-DATA input value set. When an all-DATA input value set is presented, the receiving boundary element will assert DATA on the acknowledge signal, whereupon each presenting boundary element asserts all-NULL values. When the input value set for the receiving boundary element is all NULL, it will assert NULL on the acknowledge signal to the presenting boundary elements, completing the deliver state sequence for each of the presenting boundary elements.

As shown in FIGURE 18, a fan-out configuration in which one presenting boundary element delivers its DATA set to several other boundary elements requires an acknowledge collector to ensure that all the receiving boundary elements have received the presented DATA set before setting it to NULL.

The acknowledge collector is like any other combinational expression that maintains completeness of input criteria for both DATA and NULL values. When all its input is DATA, it will assert a DATA value, and when all its input is NULL, it will assert a NULL value. The acknowledge signal seen by the presenting boundary element is the acknowledge consensus of all of the receiving boundary elements.

Complex structures of boundary elements can be formed by interassociating these basic structures. An expression of associated boundary elements is like a chain reaction poised to be triggered. A valid input DATA set will trigger the progression of events that is the expressed process. As the events proceed, the expression resets itself to be triggered again by an input DATA set. The expression is complete in itself. No external or global driving influence such as a clock is needed. Only presented input DATA is needed. There is nothing special or magic about the expression; it is just a specific associational structure of expressional conventions that are themselves structures of primitive expressional elements.

14. Generally configurable process expression

A generally configurable process expression can be constructed from NULL convention logic expressions associated via boundary elements as shown in FIGURE 19. Three specific logic expressions are required; the selector expression, the distributor expression and the memory expression.

The selector expression receives a combination of values which form the name of the source of input DATA values to the expression. When the data presented by the selected source is all-DATA the selected data values are asserted as the result values of the selector expression.

The distributor expression receives a combination of DATA values and a combination of

values which form a destination name. When all presented values are non-NULL the distributor expression asserts non-NULL result values at the named destination.

The memory expression will maintain the last presented DATA values and continuously present them as result values. A half boundary element is associated with the memory expression such that the memory expression will perform a receive state sequence and assert an acknowledge value. A deliver state sequence is not needed since the memory is continually presenting the stored values and the selector expression selects one of the asserted memory values.

The directive memory holds configuration directives that consist of a group of names to be presented to the selector expressions and the distributor expressions. Each directive specifies the selection of two values from two memory expressions, the presentation of those values to one application expression, the delivery of the result value to a specified memory expression and the name of the location of the next directive to proceed. When the delivery of the result to memory is completed the memory asserts an acknowledge that indicates that the directive process is complete and that another directive can be serviced.

The cycle of directive servicing is controlled by two and a half boundary elements the half element being the one associated with each memory expression. The current directive boundary element maintains and presents the value combinations of the directive to the selector and distributor expressions until the directive resolution is completed. The next directive name boundary element maintains the memory location of the next directive until the next directive is received into the current directive boundary element. The boundary elements are associated in a loop that will remain actively cycling through consecutive directives as long as there is a valid next directive name. The boundary elements will cycle the selector and distributor expressions through all-DATA and all-NULL states as each directive is resolved. When the next directive name is NULL the process will cease.

Given a properly formed set of directives any desired structure of association relationships among the data in the memory and the available application expressions can be expressed. Once the expression is set up and the first directive name is inserted into the next directive name boundary element the expression will begin cycling through directives and resolve the expression quite autonomously without benefit of any time reference or of any other form of external control.

15. Summary

NULL convention logic in its various forms provides a technique for expressing circuits that are purely (with INTERMEDIATE value) or effectively (with feedback) symbolically determined. This means that the circuits can be designed with purely symbolic considerations. The design need not consider propagation delays, and a timing analysis is not needed. All failure modes due to timing problems are eliminated. Since the circuit behavior is purely symbolic, its behavior can be easily monitored for faulty operation.

The NULL convention is a general expression technique. It can be added to existing logic or it can be used in its minimal two-value form. The two-value form of NULL convention logic preserves all of the advantages of Boolean logic for electronic implementation (two values, simple gates) while providing locally autonomous speed-independent behavior.

The boundary element provides a means of assembling NULL convention circuits into large systems while preserving the advantages of the circuits for the system as a whole. A structure of boundary elements is an autonomously behaving structure of locally independent cooperating parts. It is a true distributed expression. No central or global control or monitoring is required for the structure to perform correctly. Also, like a single circuit, the structure is purely symbolically determined.

An information processing system for use in manipulating and resolving data is constructed incorporating the NULL value. The system comprises one or more information processing units resolving combinations of values, such as a logic circuit. Each information processing circuit in turn comprises one or more information processing members, such as logic gates, for resolving combinations of allowed values, such members being communicatively connected via information transmission elements, for example electrical conductors, which transmit value (physical states) between members. Importantly the value combinations (information) comprise at least one data value and the NULL value. As previously discussed, multiple data values may be utilized. Additionally, the system may utilize additional non-data values, other than the NULL value such as the INTERMEDIATE value.

The value combinations resolved via the information processing members, and transmitted via the information transmission elements may comprise at least one value, either data or non-data. Such value combinations include 1) the set including standard binary data values and the NULL value, 2) the set including only one data value and the NULL value, 3) the set including one data value, the NULL value, and the INTERMEDIATE value.

In an information processing unit which comprises multiple information processing members, the information processing unit maps from combinations of values presented to it to combinations of values it asserts. The information processing unit may perform a particular data resolution by asserting a specific combination of values for each combination of presented values.

Each information processing member resolves value combinations by asserting a values for each combination of values presented to it, such that (1) for VALID combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values, and (2) for INVALID combinations of resented values the asserted value is a NULL value. Alternatively, member resolution may be accomplished by asserting a value such that (1) for VALID combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values which remains asserted until the combination of presented values becomes all NULL, and (2) for all NULL combinations of presented values the asserted value is a NULL value which remains NULL until the combination

of presented values becomes VALID, thus achieving hysteresis. Finally, with respect to systems utilizing intermediate values, the information processing member alternatively resolves value combinations by asserting a value such that (1) for VALID combinations of presented values the asserted value is a data value dependent on the particular combination of presented values, (2) for combinations of presented values that include data values and non-data values the asserted value is an intermediate value, and (3) for all NULL combinations of presented values the asserted value is a NULL value.

The information processing unit cycles through resolution and non-resolution states to allow determination by other information processing units of the information processing unit's (1) completion of a data resolution and (2) readiness to perform another data resolution. A resolution state occurring when the information processing unit is presented with a valid combination of values and is asserting a valid combination of values. A non-resolution state occurring when the information processing unit is presented with an all NULL combination of values and is asserting an all NULL combination of values.

THAT WHICH IS CLAIMED:

1. An information processing system comprising at least one information processing unit, the information processing unit having at least one information processing member, the at least one information processing member resolving allowed values, allowed values including at least one data value and at least one non-data value, at least one non-data value being a null value, and the system further comprising a plurality of information transmission elements for transmitting values to and from the at least one information processing unit and the at least one information processing member.

2. The information processing system of Claim 1, wherein the information processing unit comprises a plurality of information processing members which are interconnected by the information transmission elements, the information processing unit mapping from combinations of values presented to the information processing unit to combinations of values asserted by the information processing unit.

3. The information processing system of Claim 1, wherein the information processing member resolves values by asserting a value for each combination of values presented to the information processing member, such that (1) for valid combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values, and (2) for invalid combinations of presented values the asserted value is a null value.

4. The information processing system of Claim 1, wherein the information processing member resolves values by asserting a value for each combination of values presented to the information processing member, such that (1) for valid combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values which remains asserted until the combination of presented values becomes all-null, and (2) for all-null combinations of presented values the asserted value is a null value which remains asserted until the combination of presented values becomes valid.

5. The information processing system of Claim 1, wherein there is at least one second non-data allowed value which is distinct from the at least one null value, the at least one second non-data value being an intermediate value, and wherein the information processing member resolves values by asserting a value for each combination of values presented to the information processing member, such that (1) for valid combinations of presented values the asserted value is a data value dependent upon the particular combination of presented values, (2) for combinations of presented values which include intermediate values the asserted value is an intermediate value, and (3) for all-null combinations of presented values the asserted value is a

null value.

6. The information processing system of Claim 2, wherein the information processing unit cycles through resolution and non-resolution states to allow determination of the information processing unit's (1) completion of a data resolution and (2) readiness to perform another data resolution, a resolution state occurring when the information processing unit is presented with a valid combination of values and is asserting a valid combination of values, and a non-resolution state occurring when the information processing unit is presented with an all-null combination of values and is asserting an all-null combination of values.

7. The information processing system of Claim 6, wherein the information processing unit performs a data resolution function by asserting a predetermined combination of values for each combination of presented values.

8. The information processing system of Claim 6, wherein the information processing unit stores values by asserting a combination of values equal to a previously presented combination of values.

9. The information processing system of Claim 6, wherein the information processing unit selects values by asserting a combination of values which are a subset of a first combination of presented values relative to a second combination of presented values.

10. The information processing system of Claim 6, wherein the information processing unit distributes values by asserting a combination of values equal to a first combination of presented values as a subset of its combination of asserted values relative to a second combination of presented values.

11. The information processing system of Claim 6, wherein the system comprises a plurality of information processing units, and wherein the system further comprises a plurality of means for bounding the information processing units by asynchronously coordinating value presentation among the information processing units, the bounding means comprising:

(a) at least one null-valid detector for determining an information processing unit's (1) completion of a data resolution and (2) readiness to perform a data resolution;

(b) means for storing at least one combination of values; and

(c) means for communicating with other bounding means.

12. The information processing system of Claim 1, wherein the system is utilized in a logic system, and wherein the information processing unit is a logic circuit, the information processing member is a logic gate, the information transmission elements are discreet

conductors, and the values are physical states of the conductors, at least one state being assigned to represent the null value, the remaining states being assigned to represent data values.

13. An information processing system comprising:

(a) at least one information processing unit for resolving combinations of values;

(b) at least one information processing unit for storing combinations of values;

(c) at least one information processing unit for configuring value presentation relationships by asserting a combination of values whose relationship to a first combination of values presented to the information processing unit is determined by a second combination of values presented to the information processing unit;

(d) at least one bounding means for asynchronously coordinating value presentation; and

(e) each information processing unit having at least one information processing member for resolving allowed values, the information processing unit further having a plurality of information transmission elements for transmitting values to and from the information processing member, allowed values including at least one data value and at least one non-data value, at least one non-data value being null value.

14. An asynchronous information processing system for manipulating and resolving data, comprising:

(a) a plurality of information processing units for resolving of allowed values, the information processing units having a plurality of information processing members, the information processing members being for resolving allowed values allowed values including at least one data value and at least one non-data value, at least one non-data value being a null value;

(b) a plurality of information transmission elements interconnecting information processing members, each information transmission element transmitting allowed values from and to the information processing members, and from and to the information processing units;

(c) each information processing unit mapping from combination of values presented to the information processing unit to combinations of values asserted by the information processing unit; and

(d) each information processing unit cycling through resolution and non-resolution states to allow determination of the information processing unit's (1) completion of a data resolution and (2) readiness to perform another data resolution, a resolution state occurring when the information processing unit is presented with a valid combination of values and is asserting a valid combination of values, and a non-resolution state occurring when the information processing unit is presented with

an all-null combination of values and is asserting an all-null combination of values.

15. An asynchronous, configurable information processing system for manipulating and resolving data, comprising:

(a) at least one information processing unit for resolving combinations of values, said resolving unit asserting a specific combination of values for each combination of values presented to the information processing unit;

(b) at least one information processing unit for storing combinations of values, said storing unit asserting a combination of values equal to a previously presented combination of values;

(c) at least one information processing unit for configuring value presentation relationships between resolving and storing information processing units by asserting a combination of values whose relationship to a combination of values presented to the information processing unit is determined by a combination of directive values presented to the information processing unit;

(d) bounding means for asynchronously coordinating value presentation between information processing units;

(e) a plurality of information transmission elements for transmitting values from and to the information processing units and information processing members; and

(f) each information processing unit having a plurality of information processing members for resolving allowed values, allowed values including at least one data value and at least one non-data value, at least one non-data value being a null value;

(g) whereby, the configuring information processing unit configures value presentation relationships relative to a combination of directive values asserted by the bounding means and presented to the configuring information processing unit and, whereby a data resolution is accomplished by the presentation of alternating sequences of valid directive values and all null directive values to the at least one configuring information processing unit

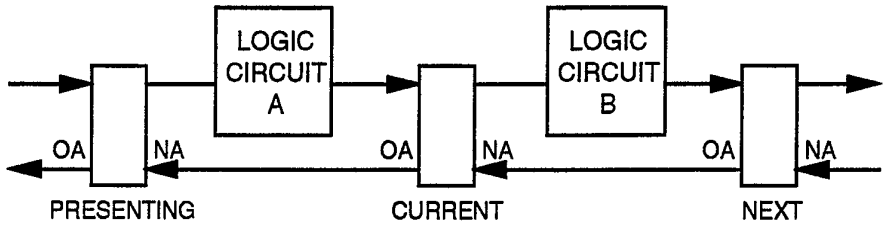


FIG. 1

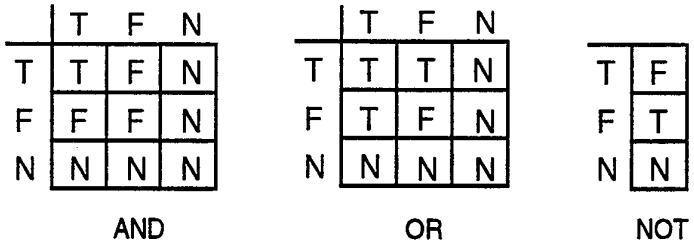


FIG. 2

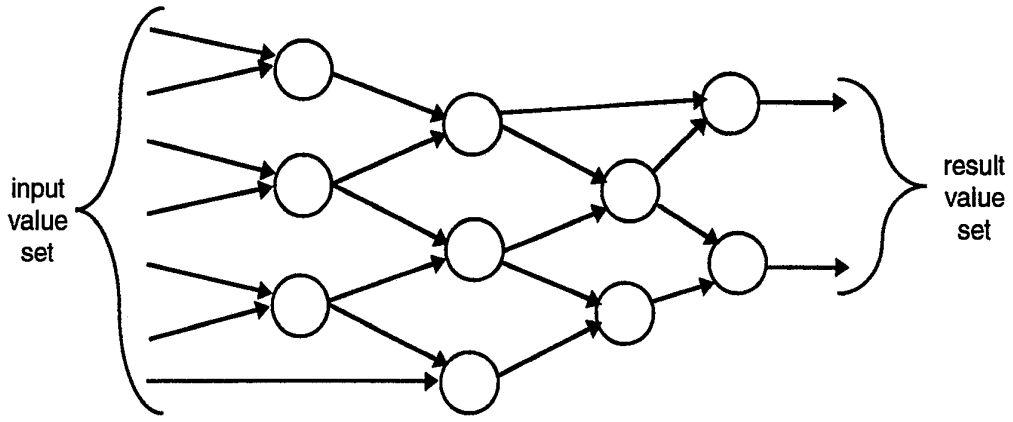


FIG. 3

	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

	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
N	N

NOT

FIG. 5

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

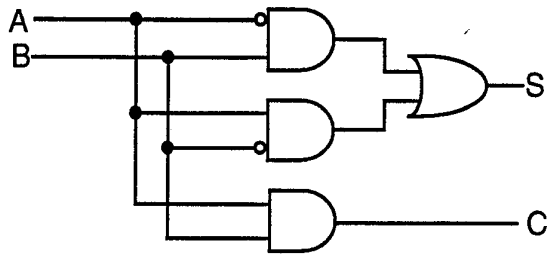


FIG. 6

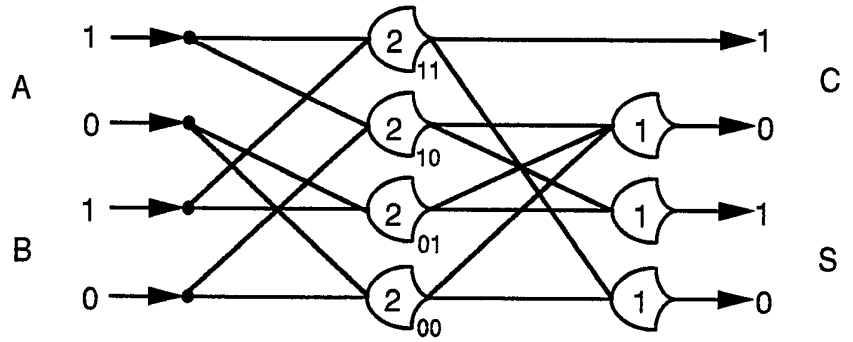


FIG. 7

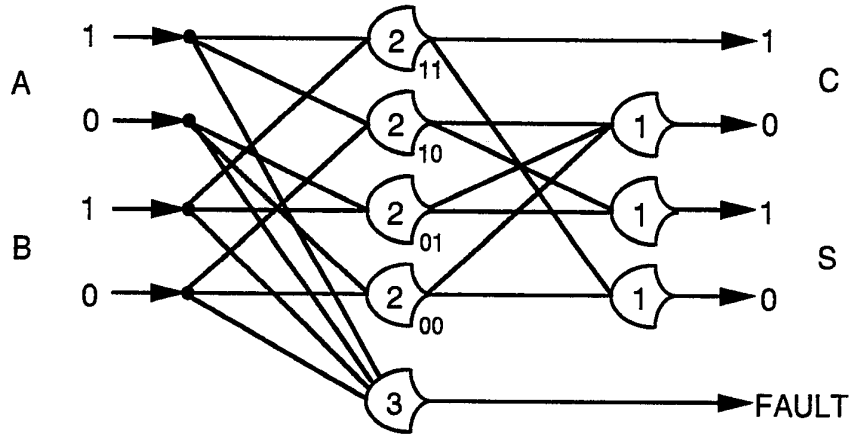


FIG. 8

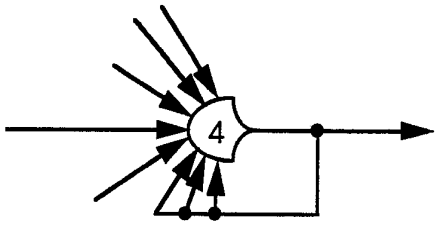


FIG. 9

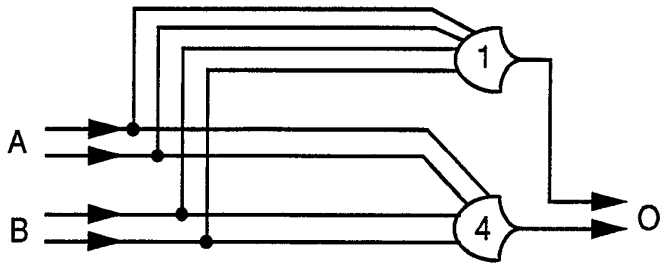


FIG. 10

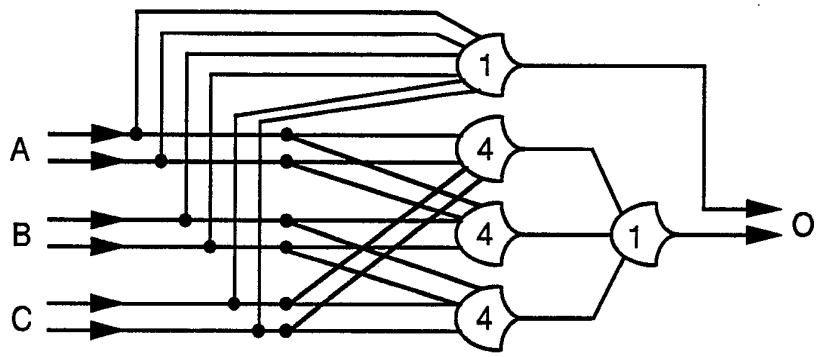


FIG. 11

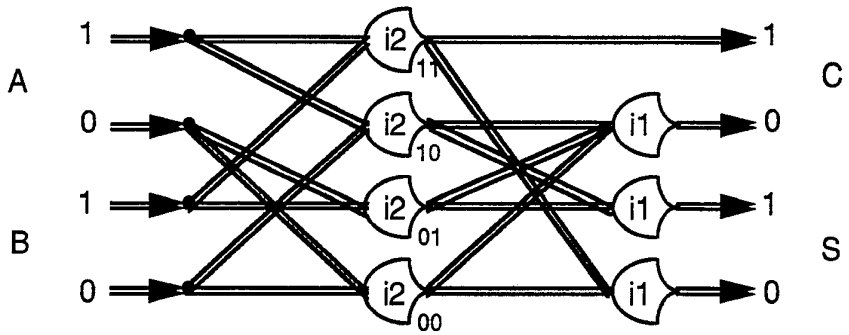


FIG. 12

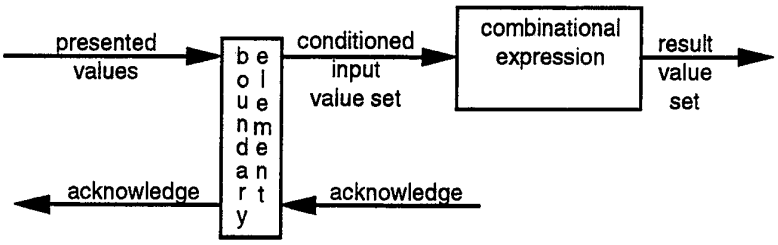


FIG. 13

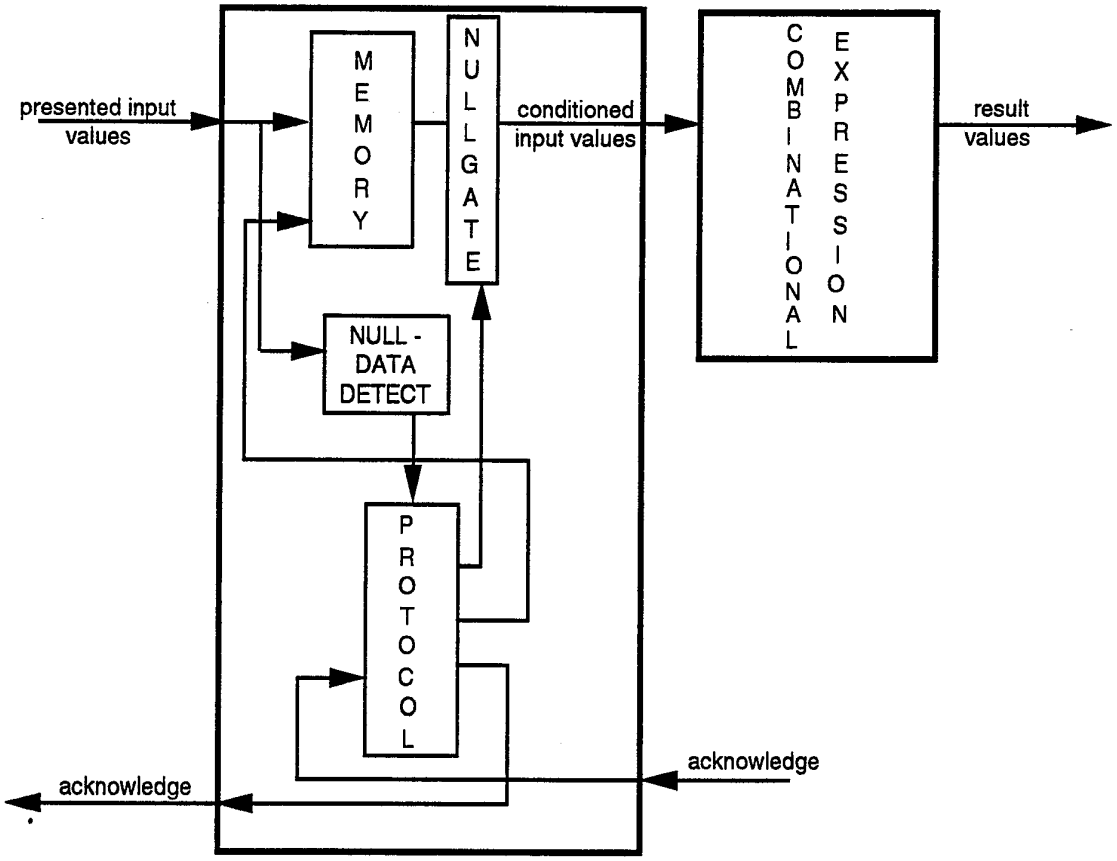


FIG. 14

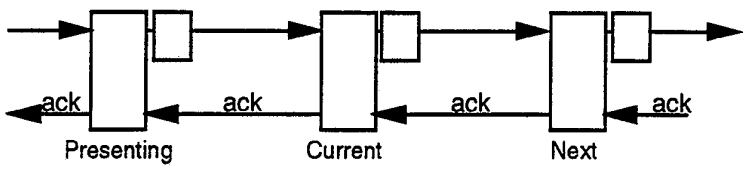


FIG. 15

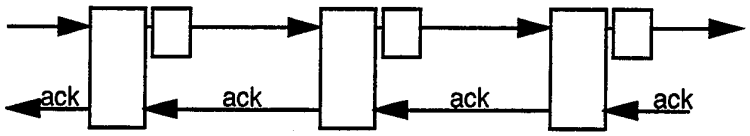


FIG. 16

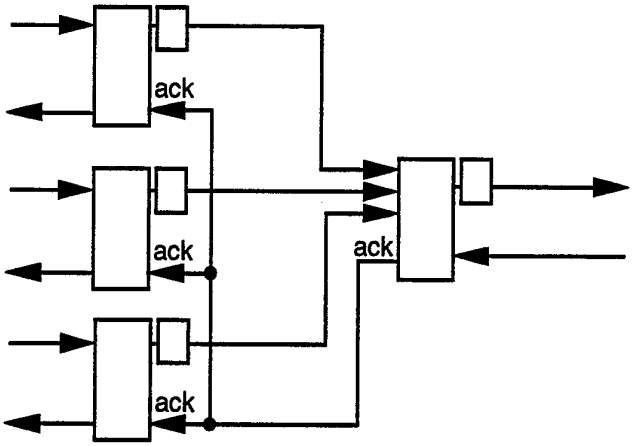


FIG. 17

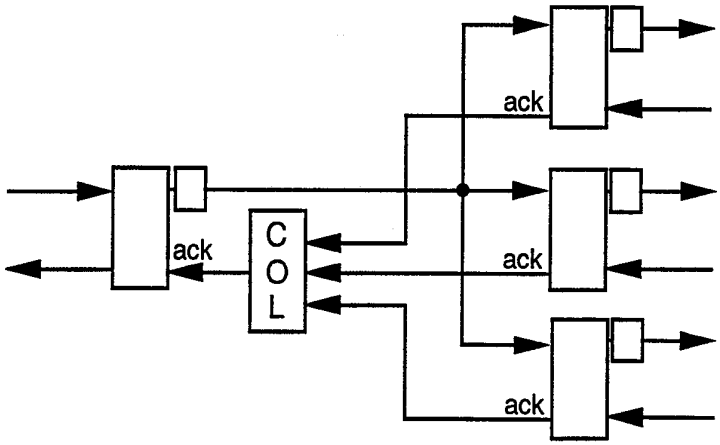


FIG. 18

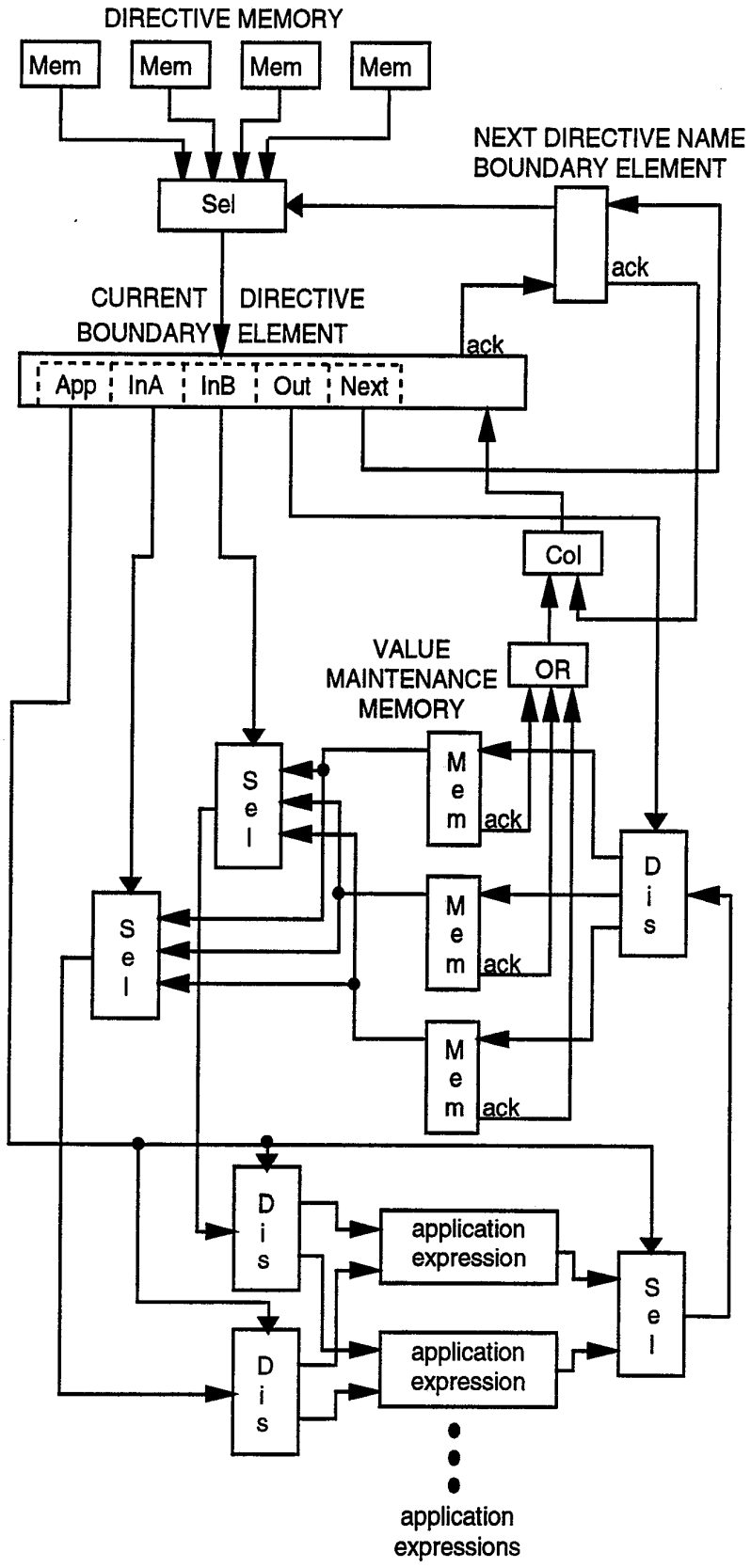

DIRECTIVE MEMORY
Mem
Mem
Mem
Mem
NEXT DIRECTIVE NAME
BOUNDARY ELEMENT
Sel
ack
CURRENT
BOUNDARY
DIRECTIVE
ELEMENT
ack
App
InA
InB
Out
Next
Col
VALUE
MAINTENANCE
MEMORY
OR
Sel
Mem
ack
Dis
Sel
Mem
ack
Mem
ack
Dis
application
expression
Sel
Dis
application
expression
application
expressions

FIG. 19

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US92/04132

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 1/00
US CL :395/800
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/800

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS, US Pat Database searched. The search terms are: Miller and 395/?; speed (3A) independent; three value logic; null value; null logic.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	N, IEEE, Transaction on Computer. Vol. C-29, No. 10, October 1980 Anthony S. Wojcik, Pg. 889-898. See at least Section II, III and IV.	1-15
A	U.S.,A, 4,845,633 Furtek 04 July 1989 See Abstract.	1-15

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search
21 JULY 1992

Date of mailing of the international search report
14 SEP 1992

Name and mailing address of the ISA/
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. NOT APPLICABLE

Authorized officer
DAVID Y. ENG
Telephone No. (703) 308-1635

Form PCT/ISA/210 (second sheet)(July 1992)★