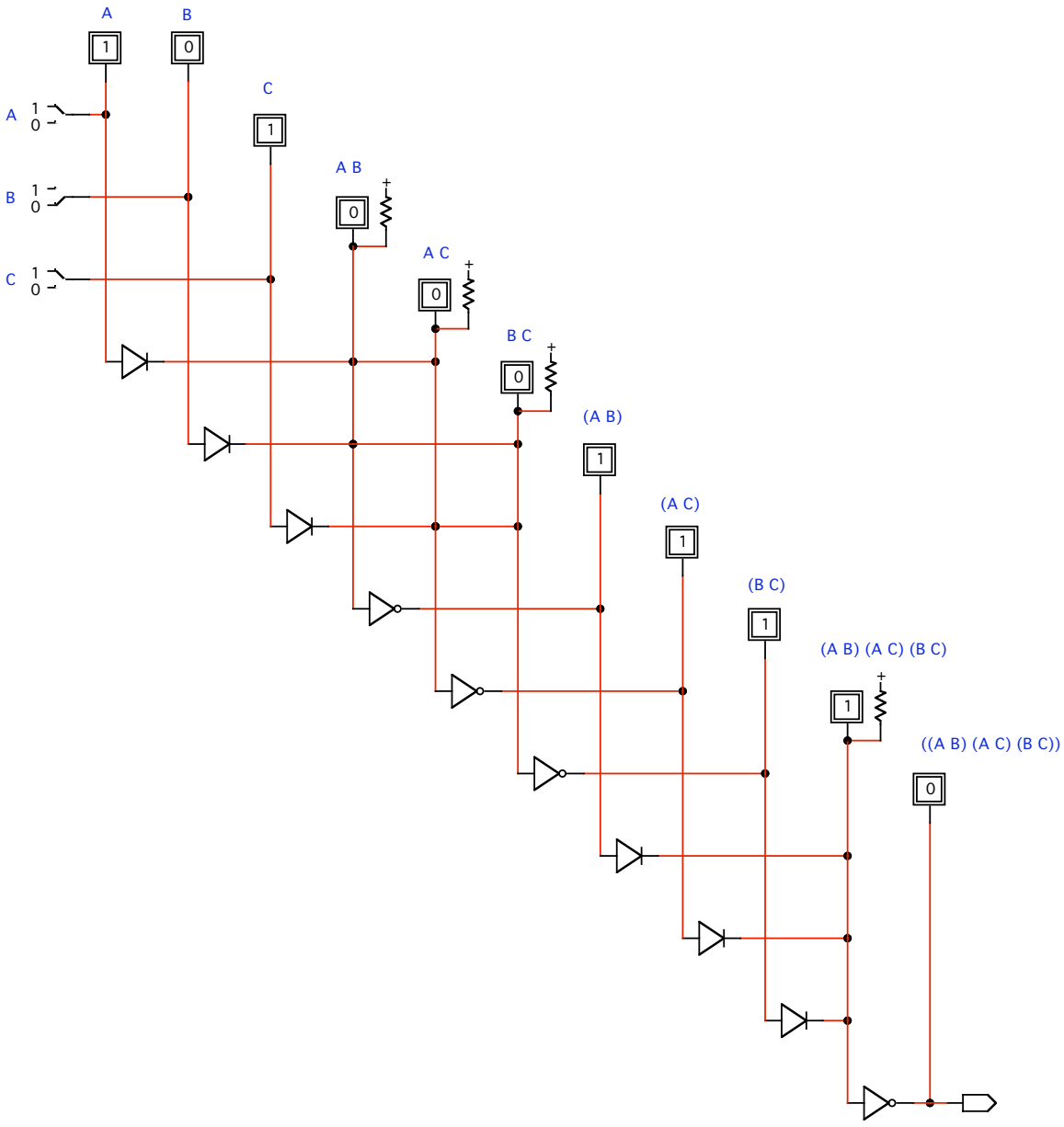
COMESH HARDWARE MODELS

William Bricken

June 2002

Distinction networks can be actualized as circuits with only inverters, using the technique of wiring connections together to achieve the semantics of logical OR. Rather than generalized NOR gates, incoming wires are simply joined, forming a *wire-or*. The dnode itself is then a simple inverter.



Computational mesh (Comesh) is an array-based technique for the construction of both combinational and sequential semiconductor circuitry. The specialized array is a triangular matrix of bit storage sites with inverters on the diagonal.

The mesh is an array of row and column wires which can be connected at selected intersection points (cross-points). The functionality of the target circuit is encoded in the configuration of connected cross-points. Essentially, each connected cross-point corresponds to a physical wire in a conventional ASIC. Row connections are fanout of a signal; column connections are wire-ORs of a collection of signals. Diagonal inverters manage the flow of computation, by inverting each column they create a network of NOR logic. In combination, NOR logic constructed from wire-OR columns and diagonal inversions fully implements all combinational and sequential logic networks.

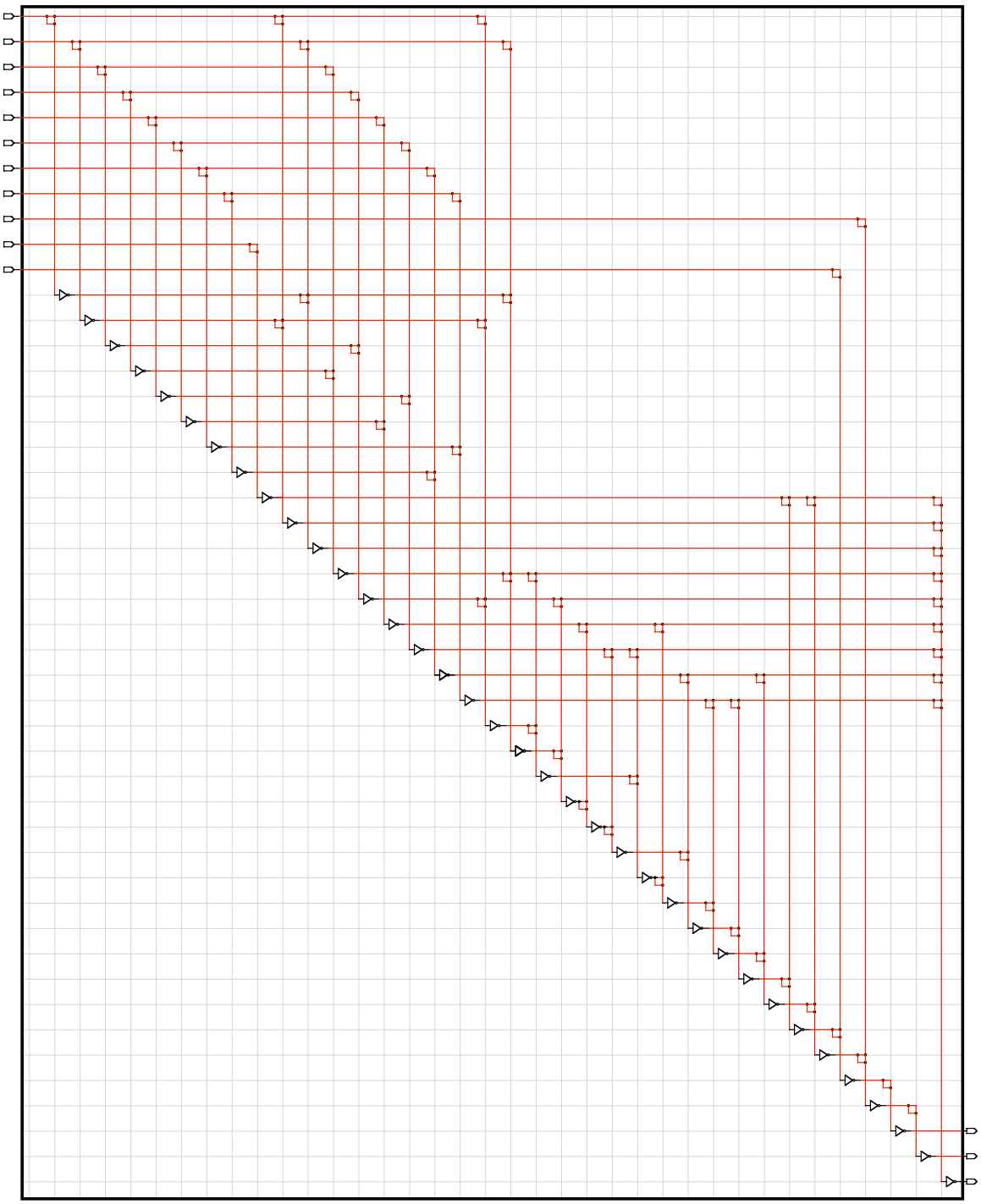
Comesh is not composed of logic gates in the conventional sense. Each column and inverter does implement the functionality of a NOR gate with an arbitrary number of inputs. Each row does implement the fanout of that abstract NOR gate. Input is defined by designated input rows; output is defined by designated output rows. However, Comesh cross-point configurations are explicitly structures of Iconic Logic. Rows are boundary objects, either atoms or containers. Connections in a column indicate which boundary objects share a common space. A column then is *a bounded space*. Connections in a row indicate which spaces contain the row boundary object. The diagonal inverter specifically represents the act of crossing a boundary, moving computational focus from the column space to the row container of that space.

Comesh "registers" are also rows which generate output signals; the difference between logic and registers is that register row outputs are reentered into the array as inputs (together with external input signals) at each internal clock cycle. Thus Comesh registers are not flip-flops, and do not behave as flip-flops, rather their behavior is equivalent to that of Comesh gates. This makes timing the Comesh exceedingly simple.

A Comesh replaces both conventional wiring and conventional timing with its own internal structures. All timing is Comesh pipelined, all wiring is stored in Comesh cross-points. Comesh provides multilevel logic functionality in a PLD-like device, without the exponential explosion of two-level AND-OR logic. Since timing is defined by the Comesh rather than by wiring connecting arbitrary sequences of gates, Comesh timing design is highly predictable. Since wiring is defined by the location of cross-point connections, place-and-route is equivalent to loading the mesh configuration. Performance is therefore independent of wiring complexity, and largely insensitive to the timing complexity in the original design.

A Comesh Block

The Diagram below illustrates a small circuit, a 4-bit magnitude comparator with an enable for each comparison type (<,=,>), as a Comesh circuit.



The gray underlying grid represents the unprogrammed Comesh. Each small square at a grid intersection is a programmed connection. Darker (red) lines are active wires connected by the programmed intersections.

Signals enter in the top-left corner and exit in the bottom-right corner. In the example, eleven inputs signals enter at top-right. These are the two 4-bit binary numbers being compared for magnitude, and the three enables, one for each output. Three signals exit in the bottom-left, these are the results of the magnitude comparison, one signal for each of *greater-than*, *equal*, and *less-than*. Active cross-points represent an OR function, the inverters which form the diagonal convert the OR into a NOR function. Columns with multiple active connections represent n-ary OR; rows with multiple connections represent fanout of the inverted column signal.

Characteristic of the array is that all signals which travel from the top-left to the bottom-right traverse exactly the same length of wire, equal to twice the width of the square array. The number of visits that any particular signal makes to the diagonal inverters represents the critical path of that signal. The greatest number of diagonal visits of any signal is the greatest critical path of the Comesh circuit.

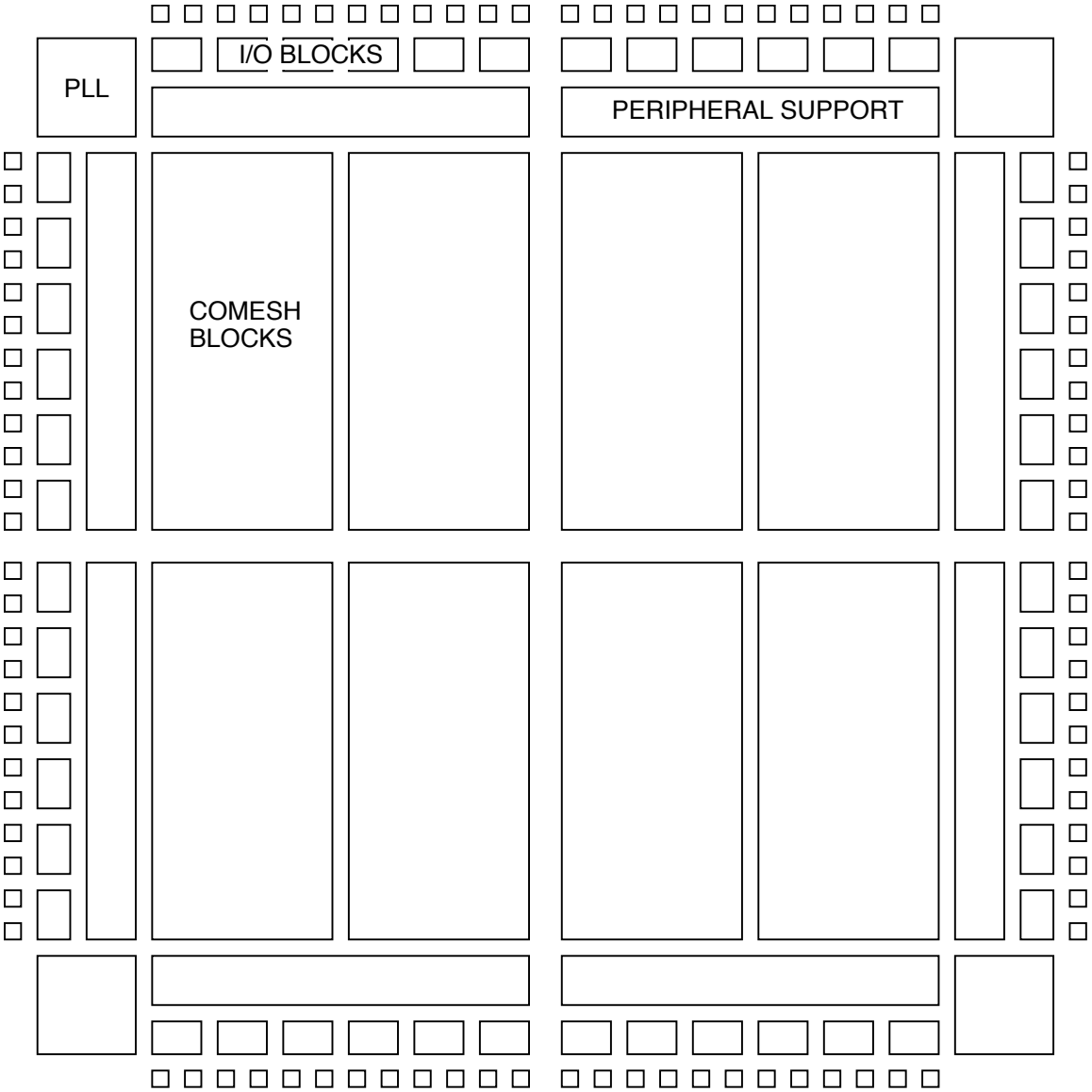
The configuration of the active cross-point intersections is dictated by a pun distinction network. Distinction networks specify a collection of signals associated with a label. This data structure is an adjacency list which is interpreted as a collection of containment relations. The Comesh configuration is the adjacency matrix for the same distinction network.

Each row identification label is the same as each column identification label. Thus a cell (i, i) represents the containment of a container by itself, an impossible structure. Rows represent containers, while connections on a row identify the containers which contain the row label. Columns represent containers; connections in a column identify particular rows which that container contains.

Comesh Chip Architecture

Comesh blocks such as the one illustrated above are limited in physical size due to pragmatic electrical and physical constraints. Thus a hardware implementation architecture of appreciable size must be composed of several Comesh blocks. The figure which follows illustrates an abstract chip layout which incorporates several Comesh blocks.

In the figure, large Comesh blocks are placed in the context of other supporting circuitry such as clocking regimes (PLL), input/output facilities and protocol management circuitry (I/O BLOCKS), and various other non-Comesh peripheral support circuitry such as configuration management, security, and testing (PERIPHERAL SUPPORT).


PLL
I/O BLOCKS
PERIPHERAL SUPPORT
COMESH
BLOCKS