Symbolic Layout for Rapid Full-Custom Prototyping of High-Speed Telecommunications Chips

David G. Boyer
Robert R. Cordell
Bellcore
331 Newman Springs Road
Red Bank, N.J. 07701

ABSTRACT
Symbolic layout systems for VLSI chips are gaining popularity because they allow designers to create full custom layouts without concern for design rule details, allowing them to function at a higher level of abstraction. High performance, quality layout, technology insensitivity and high designer productivity thus result. The MULGA symbolic design system has been in use at Bellcore for design of research VLSI chips for the last five years. This system and Bellcore's experience with symbolic design of high performance VLSI will be described. Bellcore's research on a new symbolic design system, DASL, will be discussed. Symbolic layout is also a powerful technology for use in cell-based systems and silicon compilers, and its use in these areas will also be covered.

1. INTRODUCTION
Over the last five years the MULGA symbolic design system has been used in research at Bellcore to design high speed CMOS chips to demonstrate the technology for future broadband networks and services. The use of the symbolic approach has allowed designers to create full custom circuits to achieve high performance while attaining the high design productivity needed in a research environment. Chips designed with MULGA at Bellcore have reached speeds of over 240 Mhz/sec.

The use of symbolic design allows a person to design a full custom chip without worrying about process design rules. The same circuit can then be "compiled" for a variety of processes. Five different processes are currently supported in the design system, and new sets of CMOS design rules are easily installed. A variety of processes can thus be tried and compared with little extra work. Cost/performance tradeoffs can be evaluated quickly and easily. A switching chip was mapped completely (CIF tape in hand) to a new process in three days3. The ability to recompile a chip for a new process is especially useful for companies without a captive fabrication line, making it easy to go to a new vendor when necessary. This ability also allows a company to keep its product in the most up-to-date technology available, recompiling old designs into more advanced design rules as appropriate (Figure 1).

Finally, the ease of recompiling into different technologies facilitates a great amount of cell and module reuse, further enhancing designer productivity.

Another advantage of the system is that its ease of use allows a single designer to design quickly an entire chip. This leads to high productivity because the design is not passed from person to person. A 15K transistor custom chip was designed (from architecture to layout) in two months by an experienced designer.

The paper is organized as follows: symbolic layout methodology will be described and details of the MULGA symbolic design system will be discussed. Bellcore’s design methodology and experience with several chips will also be covered. Bellcore’s new DASL symbolic design system will next be described. The paper will conclude with a discussion of the application of symbolic layout in cell-based systems and silicon compilers.
2. SYMBOLIC LAYOUT AND COMPACTION

Symbolic layout systems with compaction are used to create full custom designs in a process-independent fashion. Symbolic contacts, wires, and transistors are used to represent the different circuit elements of the target technology. A designer uses symbolic circuit elements rather than mask geometries to create his or her design. A compaction program is responsible for creating the mask level description of a circuit. It spaces the circuit elements according to the design rules of a target process which are typically kept in a technology file. The compactor can compact a symbolic cell for a new fabrication process simply by using a new technology file. Consequently, a designer does not usually have to redesign a circuit to take advantage of a new fabrication process. Tracking an evolutionary process change can be done in as little as three days.

Symbolic design proceeds in a top-down, bottom-up manner. The designer first defines the system (chip) at the functional level, defining the major functional blocks to be used. The blocks will be broken down into parts according to their sub-functions to a level where the circuit design and the actual layout of the "leaf cells" is done. A leaf cell contains primitive circuit elements (transistors, pins, wires, and contacts) and usually implements a basic logic function like a NAND gate, a flip-flop, a full-adder, etc. (Figure 1). The leaf cells are then combined to create the composite cells (cells containing other cells). Composite cells, or "modules", are then combined until the top level of the chip design is reached. Composite cells are assembled with two techniques, either by abutment or with routing. A pitchmatcher is used in a symbolic layout system to automatically align the ports of the cells to be connected via abutment (Figure 2).

Symbolic layout not only greatly increases designer productivity, but also the productivity of CAD programmers is increased. Because explicit circuit elements are being used, programs that implement functions such as compaction and extraction are simplified.

There are two commonly employed compaction techniques today - constraint graph compaction and virtual grid compaction. The constraint graph approach is the most popular technique used today. In the constraint graph approach the designer typically places circuit element symbols on a fine grained physical grid. A constraint graph then is created to represent the circuit. The nodes of the graph represent the elements of the circuit and the edges represent the spacing needed between the different elements of the circuit. Placement of the circuit elements is done by giving the elements locations that are based on the longest path through the graph to the element. The compactor brings circuit elements as close together as design rule spacing requirements will allow. Constraint graph compactors are one-dimensional compactors and require at least one X compaction pass and one Y compaction pass. They are typically used as one of the tools available to a designer and are not typically set up to handle abutted hierarchical designs. Pitchmatching is done explicitly by the designer. For more information on constraint graph compaction see reference 3.

Virtual grid symbolic layout is a structured approach to symbolic design. The relative placement of the circuit elements is determined by their location on the input grid. This grid serves only to delineate the relative topology of the circuit elements, and does not correspond to a physical coordinate system. A virtual grid line's location is governed by the largest location required by an element on that grid line. A virtual grid compactor assigns physical locations to the grid lines, observing the worst case spacing constraint experienced by the grid line.

The virtual grid concept extends naturally to a hierarchical design. Abutment connection points are identified explicitly by the designer by placing a "pin" at the desired boundary component. Symbolic cells are then placed on the virtual grid. Abutment points are identified between two cells when they share a common edge and when they both have elements that
contain the same layer at the same grid point. Cells are permitted to have coincident edges (their edges can share the same grid line), but they are not permitted to overlap. A symbolic module is then pitchmatched. The hierarchical symbolic layout is used by the pitchmatcher to identify the abutment points which are to be pitchmatched.

3. THE MULGA DESIGN SYSTEM

3.1 BELLCORE Design Cycle Overview

Symbolic design is typically done in a hierarchical, iterative fashion as depicted in Figure 3. A top-down, bottom-up method is commonly employed. In the majority of cases a high-level schematic is created on a schematic capture system such as a Mentor Graphics workstation. An example of the schematic for a portion of a chip can be seen in Figure 4. During this phase of the design different architectures are explored and the building blocks used in the schematic are decomposed down to a gate/leaf cell level. Logic simulations are done to verify functionality and to obtain estimates of performance.

A structural hierarchy is defined that is typically used to define the building blocks that will then be designed with the symbolic layout system. In other words, the schematic from the workstation serves as the structural floorplan for the chip and the chip is laid-out/assembled bottom-up.

The design then continues on the MULGA workstation where the symbolic layout is done. The chip is assembled bottom up with the leaf cells being first designed. Modules/leaf cells that are tightly coupled to each other (a large amount of local communication) are typically connected by abutting the cells together. Modules that are loosely connected or geometrically less compatible are routed together.

4. Mentor schematic of an 8x9 multiplier;

The decision to connect cells via abutment or routing is often a matter of judgment and style. Large modules with many different types of cells that are connected with abutment may result in the cells being stretched by pitchmatching to an unacceptable degree. At the other extreme, routing all leaf cells together results in wasted area because of the necessarily large routing area needed.

After the core of the chip is assembled the I/O frame for the chip is "automatically" generated and the core is placed in the frame and routed to it. During the bottom-up design phase, verification of the chip's functionality is done with several internal simulators and with the RINK' design verification system. Design rule checking is done with a commercial DRC package.

The design methodology presented has been successfully used to design over 40 chips that worked as designed the first time. Performance of the chips has usually met or exceeded expectations and shows that CMOS is a viable technology for the digital network of the future.

3.2 MULGA Virtual Grid Compaction

The bottom up creation/assembly of a chip is performed with the MULGA suite of design tools. The leaf cells are first designed and compacted. The leaf cells are laid out symbolically on a virtual grid. A virtual grid is used in MULGA to establish the relative placement of circuit elements and does not correspond to a physical grid. In the virtual grid approach the compactor gives locations to the virtual grid lines, not to the circuit elements themselves.
5a. Symbolic circuit with a poly wire lying between an N-diffusion device and a P-diffusion device.

5b. If no backtracking is done, the N-diffusion device will not "see" the P-diffusion device and will be placed as close to the poly wire as design rules permit. This introduces a design rule violation between the N-diffusion device and the P-diffusion device.

5c. For the correct placement of the N-diffusion device, grid lines preceding the grid line that the poly wire is on must be checked. The necessary spacing between the N-diffusion and P-diffusion device is then observed.

Virtual grid compaction begins by first examining spacings in one direction only (we will use the X direction as the first compaction pass). Each X grid line is compared with parallel neighboring X grid lines for spacing requirements. A spacing is required with a neighboring X grid if elements on two neighboring X grids have the same Y coordinate value. The final spacing of an X grid will be the greatest spacing it has encountered with respect to their parallel neighbors. In order to accommodate spacings that exert their influence over a number of symbolic grid lines, backtracking must be done (Figure 5). The grid line to be placed is compared to previously placed columns until the distance between the current column and the prior column exceeds some worst-case process value. This is possible because virtual grid compactors do not allow grid lines to change their ordering. When the spacing between the present grid line and some previously placed grid line exceeds the worst-case process value it is guaranteed that none of the grid lines that have not been checked (they have a smaller location then the "previously placed grid line") will have an influence over the grid line under question.

During Y compaction (the second compaction pass) corner spacings are accommodated. An element being placed during Y compaction is spaced against diagonal neighbors within the worst-case process window as well as against its adjacent neighbors. Elements on grid lines are not only spaced against elements with the same X grid value but also with nearby elements with different X values.

3.2.1 Fence Compaction Fence compaction eliminates the need to do backtracking in a virtual grid compaction and is linear in run time. The virtual grid limits the potential locations of the symbolic elements. This permits the use of a "picket fence" data structure for each grid line. The picket fence can be thought of as a bin for each grid that is used to collect and retain information about previously placed elements.

A "picket fence" data structure records the last placement of each layer for the given process, and thereby keeps track of the necessary parallel neighbor information. There is a fence data structure for each Y grid line during X compaction. Each fence, in turn, has one "picket" for each layer of the process for which compaction is being done. Each picket contains a rectangle that represents the last placement of a rectangle of the corresponding layer along the fence structure's grid line. The contour of a given layer is captured by the pickets for that layer.

During the second compaction pass Y grid lines are placed in much the same way as grid lines were placed during X compaction. In Figure 6b the contour of the aluminum layer (the contents of the aluminum pickets contributed by all the fence data structures) is depicted for the Y compaction of the circuit found in Figure 6a. The updated fence that results after the grid line is placed can be seen in Figure 6c. Diagonal constraints are handled during this compaction pass and the
6b. Y virtual grid 4 is being placed with respect to the aluminum fence. The fence outline corresponds to the already placed aluminum layers.

6c. After Y virtual grid 4 is placed the fence pickets are updated to reflect the placement of this grid line. The aluminum fence is shown after the updating is done.

actual Euclidean distances for the corners being spaced are used. Diagonal constraints are handled by spacing the elements at the point under question with the pickets of the grid lines that fall within a design rule window of the point.

3.3 MULGA Pitchmatching

The virtual grid concept extends naturally to a pitchmatching environment. Abutment points are identified between two cells when they share a common virtual grid and when one or both have symbolic elements on the common boundary. Cells are permitted to have coincident edges, but they are not permitted to overlap. The virtual grid allows easy identification of the pitchmatching points in a module and identifies the corresponding connection points in neighboring cells that need to be aligned. The pitchmatcher’s function is to align the ports of the cells that are to be abutted. This is accomplished by stretching the cell whose ports have the smaller spacing to align with the span of the cell with the larger port spacing.

Pitchmatching is hierarchical. Modules can be created by pitchmatching abutted leaf cells, but larger modules can also be created by pitchmatching abutted modules together. Such hierarchical capability is a major factor responsible for the power of the system.

3.4 MULGA Routing

Modules can also be connected by routing. A river routing utility interconnects neighboring cells. For each mask module, a rectangular bounding box is created automatically; this is used at the next higher level of the hierarchy. Manual routing of cells is supported and has been used on some designs. A commercial block place and route package is being integrated with the system.

3.5 I/O Frame Assembly

The I/O frame for the chip is created with a graphical tool that allows the user to create the I/O frame in an interactive fashion. The process starts with the user specifying the die size for the chip and the number of pad slots desired. A frame is then created and displayed that has empty slots for all the pads. Slots can be left empty and guard rings are generated

7. Example of an I/O frame and its "rat's nest" routing to the outline of the core of the chip. Symbols are used for each I/O cell that reflect the type of I/O cell that is being used and the size of the cell.
automatically. The user selects the desired I/O cell and an icon that reflects the actual size of the cell is placed in the appropriate slot. The user also selects the pin on the core of the chip to which the pad is to be connected. Note, that the "icon" is stored in a process-independent fashion so that the I/O frame description can be used later for a different process. The relative locations of the cells are specified, but their absolute locations and sizes are determined by the target process being used for the design.

Once the frame is created, "rat's nest route" wires are used to connect the terminals on the core of the chip to the appropriate pads in the frame (Figure 7). A router is then run that first river routes all power busses in the preferred layer. A channel route is next done between the core of the chip and the I/O frame. This has proven to be an effective tool for assembling the completed chip. Not all our designers use this tool, however, because of its somewhat restrictive nature. Corner pads are not supported by the tool, so that on pad-limited designs some designers elect to do the frame assembly and routing manually in order to reduce the size of the die.

4. Simulation and Verification

The cells and the modules of the design are simulated with a variety of simulators during the course of the design. Leaf cells are usually simulated with the SPICE circuit simulator. Modules are usually simulated with the EMU timing simulator and chip level simulation is done with the SOISIM switch-level simulator. There are a number of verification tools available to designers including the VTI circuit extractor used in conjunction with some internally developed pattern matching software. The RINK layout verification system has also been used to perform layout extraction and netlist comparison coupled with a good graphical interface. A number of critical design errors have been found with RINK, including shorted power busses in a handcrafted design.

5. Bellcore Experience

The MULGA system has been in use at Bellcore for almost five years, and has been employed in the design of the majority of Bellcore's CMOS VLSI research prototype chips. Most of these chips were designed exclusively in the MULGA environment. On some other designs MULGA was used to design the majority of the chip while other parts of the chip were designed with a polygon editor in order to create analog circuits (not well supported in MULGA) or to create circuits with higher density or higher performance. The MULGA system has been especially well-suited to the research environment at Bellcore where there is a strong emphasis on high speed circuits. As mentioned earlier, the research environment requires tools that can enable a designer to turn around designs quickly so that research results are available on a timely basis and so that there is little reluctance to try new ideas.

Although most Bellcore chips done thus far are not highly complex by today's standards (they are typically 10,000 to 50,000 transistors), most are pushing the technology employed to very high speeds; this gives rise to the need for full-custom transistor-level layout. The need for high speed follows from the research on high speed network elements (eg. crosspoint switches) for the future broadband lightwave networks and services (eg. digital television) for which those networks will provide transport. It is desirable, for example, to be able to employ low-cost, high density CMOS technology for all circuits that have to operate at rates up to 155 Mb/sec.

Over forty chips have been done so far, ranging from test chips to complex signal processing chips approaching 100,000 transistors. The system has been very effective in providing quality layout and high productivity. This software has enabled small teams of designers to do entire designs, from algorithms to logic design to layout to test. In several cases, important chips have been done by a single designer in a few months.

Virtually all the chips worked as designed on first silicon. The system also facilitates fast learning by new users. In fact, a number of the chips were designed by undergraduate six-month co-op students without prior VLSI design experience.

Table One summarizes the function, speed and designer productivity of some of the chips that have been designed at Bellcore with the MULGA symbolic layout system. Chips designed at Bellcore have employed processes ranging from 3.0 microns to 1.2 microns, but most designs thus far have been in 2.0 microns. All the designs shown in Table 1 were done in 2.0 micron technology.

Conversations with industry experts indicate that a density between 1.5 and 2 transistors per square mil in 2 micron technology is very good for hand-packed custom layouts of chips that are mostly random logic. Corresponding standard cell designs will generally be less than 0.5 transistors per square mil. Most of the designs in Table 1 fall somewhere in between. In many cases the density within key modules is significantly higher (important for speed), but less care was taken in global layout efficiency because minimizing chip area is not as important to research as short design time.

Typical industry productivity for hand-packed layout is on the order of 10 to 20 transistors per day for chips that are mostly random logic. Table 1 shows that significantly higher productivity has been achieved, in many cases by relatively inexperienced designers.

Most of the chips in Table 1 perform switching or synchronization functions for digital transmission systems or digital signal processing functions for digital television systems. The asynchronous 16 X 16 crosspoint switch operates at over 250 Mb/sec and includes ECL-compatible inputs and outputs. It was originally done in a 3.0 micron process (140 Mb/sec), and that is the version for which the design time is shown. A later version was migrated from a 2.4 micron process to a 2.0 micron process in three days.

The 16 X 16 Discrete Cosine Transform chip was designed by a small team of three researchers in 18 person months. The device includes novel algorithms and architecture, and customized circuits and layout. The Bellcore DCT chip is the
first to put a 16 X 16 DCT on a single chip. A key to success here is that symbolic layout helped individuals to operate over the full span of the design hierarchy.

As technology progresses many of these designs will be migrated with minimal effort to obtain higher performance (Figure 8). The ability to do such migrations, made possible by the technology insensitivity of symbolic layout, is important to research because one or more generations of technology advancement can be expected before the results of the research find their way into real applications. Such technology insensitivity is also valuable in evaluating new processes and in seeing how well a design will work in a new process.

6. A NEW SYMBOLIC DESIGN SYSTEM - DASL

Bellcore is currently doing research on a new symbolic design system called DASL. This system provides a highly interactive environment for virtual grid symbolic design. A unique data structure ties together the symbolic and mask representations of a circuit. The DASL graphics editor utilizes a dual window approach; one for the symbolic and one for the mask. An integrated design environment results which gives the designer better feedback and control over the design process. The new data structure is arranged topologically and supports routines which need quick access to nearest neighbor information such as those for compaction and extraction. The use of this data structure has permitted the development of a split-grid compactor that has run times as fast as a virtual grid compactor while producing results that are comparable to those of a constraint graph compactor. It takes a couple of seconds to extract, compact and display an average leaf cell (30 transistor circuit). A highly interactive design system results.

The DASL design system is the outgrowth of our experience with the MULGA virtual grid symbolic design system. A number of goals were set for DASL that were a result of our experience with MULGA. For example, it was desired to provide a direct link between the symbolic and mask worlds. In MULGA the mask data produced by the compactor is totally separate from the symbolic representation. The mask data is just a list of rectangles with no element information retained. One benefit of tying together the mask with the symbolic is the ability to identify a node in the symbolic world and see its corresponding mask. Another goal was to provide a compactor that gives locations to groups of circuit elements, rather than to virtual grid lines. Virtual grids impose an arbitrary constraint on a symbolic design by giving locations to grid lines, not to circuit elements. The third goal for DASL was to provide a truly interactive system leading to greater designer productivity. This has been achieved by using the unique data structure mentioned above that speeds the compaction and extraction processes. A final goal was to have the DASL graphics editor be window based and available on a variety of workstations. The DASL graphics editor (XD) uses the X11 windowing system and currently runs on DEC, Sun, and Apollo workstations. Other workstations and PC's

8. Cross point chip for two different processes. The die sizes for the two versions of the chip are the same.
will be used when their versions of X11 are running. When XD is run a pair of windows is allocated for the editing session. The symbolic cell is displayed in one window, and the compaction/pitchmatching result (the mask data) is displayed in the other.

6.1 Split-Grid Fence Compaction

The first compaction pass (X in this description) proceeds in a similar way to virtual grid compaction. The only difference is that grids are split apart by identifying compaction groups (Figure 9a) and these groups are placed rather than the grids. A compaction group is a collection of elements that fall on the same grid line and need to be placed together. For example the source, gate, and drain of a horizontal device and the contacts and wires connected to the device that fall on this same horizontal grid line all need to be given the same Y location. A group is compacted by determining the spacing necessary for each element in the group. Elements are spaced with respect to elements in the neighboring X groups (the fence) sharing the same Y virtual grid coordinate.

After the first compaction pass each of the X groups is given a picket fence data structure. In virtual grid compaction each grid line was given a picket data structure. In split grid compaction the compaction groups are given the pickets. Another difference in the two approaches is that the compaction groups can change their relative ordering. A group with a high group number might receive a location that is smaller than a group with a lower group number. Therefore,

9a. Y groups on Y virtual grid 4 (Y groups 2 and 3) and the aluminum fence to be spaced against.

9b. Y groups 2 and 3 after placement. Note that Y group 2 was "moved" off of Y virtual grid 4, effectively splitting the grid.

10. The results of compacting the ICCD87 afa benchmark cell. Wire minimization was done during the DASL compaction. The DASL results are 16% smaller than the MULGA results.
the fence is sorted in the order of the physical placement of the groups. To place a Y group, a given element in the group is checked against the corresponding group fences that fall within a design rule window of the location of its X group fence (Figure 9). Splitting the grids results in better compaction (leaf cells areas average 10% smaller). Elements can be stretched more than desired, however, because they will be placed as close to the left edge of the cell (for X compaction) as design rules permit. Because of this, "wire minimization" is important in split grid compaction (as well as in constraint graph compaction). In DASL, the split grid compactor "pulls back" the elements that have been unnecessarily stretched. The result is that not only has the cell size been reduced when compared to MULGA, but the circuit elements are not stretched unnecessarily, leading to better circuit performance. The result of a DASL compaction of the afa cell from the ICCD87 compaction benchmarking session can be found in Figure 10.

6.2 DASL Pitchmatching

Currently under development is a split-grid pitchmatching scheme. A prototype pitchmatcher is currently working, and it is expected to be completed by the end of 1988. This scheme is an outgrowth of techniques employed by Symbolics. The Symbolics scheme is a pitchmatching technique for virtual grid lines; the DASL scheme extends this scheme to a split grid approach.

The first step in this approach is to compact a cell with all the shared boundary components. Cells that abut with a cell have their edge components deposited on the appropriate location in the cell. The cell is then compacted with these new components. Elements that do not fall on the cell edge are kept a half design rule from the edge to insure proper spacing with elements in the neighboring cell that do not fall on the edge. A constraint graph is built during the compaction with boundary components. The graph has the minimal spacing required between all groups. Cell abstraction is also done at this time; connection information is extracted from the cell so that it can be treated as a black box in the hierarchy. Connected points are members of the same parent group and are placed together. In other words, a cell's group, which is a member of a parent group, is pitchmatched when it's location needs to be increased so it will align with the neighboring port.

Once the connection group has been located the constraint graphs for each affected cell are used to update the locations of the cell's groups to reflect the pitchmatching of one of its ports. When a group in a connected cell needs to be pitchmatched, only groups that have spacing requirements with the pitchmatched group are moved. The added space is rippled through the cell. This approach retains the advantage of split grid compaction. Pitchmatching of the mul4X4 cell from the ICCD87 compaction benchmark session produced area results that are 14% better than those produced with the MULGA pitchmatcher. Note that each distinct environment that a cell finds itself in results in a different mask set.

7. SYMBOLIC LAYOUT IN CELL-BASED SYSTEMS AND SILICON COMPILERS

Although Bellcore's use thus far of symbolic layout has been primarily for full-custom styles of layout where speed has been a dominant objective, the advantages of symbolic layout clearly extend to higher-level design methodologies as well. Specifically, symbolic layout systems are well-suited as the back-end layout generation vehicle for standard cell systems and silicon compilers.

Symbolic standard cell systems store their cell libraries in symbolic form and compact cells into mask layouts only when they are used in a design. The technology insensitivity of symbolic layout preserves the library investment for many different present and future processes. At the same time, the ease of cell creation afforded by symbolic layout reduces the initial effort required to develop a library. These two factors in turn can permit the design of a larger, richer library that may include more complex cells. It also becomes practical for individual designers to add special custom cells of their own to the library for certain functions where, for example, they need extra speed.

Symbolic layout is important in at least two areas of silicon compilation. One of the most important elements of any silicon compiler is the suite of layout generators for regular structures such as ROMs, RAMs, PLAs, multipliers and datapaths. Symbolic layout makes the design of such generators straightforward because a library of symbolic cells can be easily assembled into the required array by a recipe implemented in a simple high-level procedural language. A key advantage is that compaction and pitchmatching of the cells in the array is handled automatically by the symbolic layout system. In non-symbolic approaches to generators, hand crafted cells have to be carefully designed to perfectly match all other cells with which they will be abutted when they are arrayed in many different legal combinations. The advantages that technology insensitivity brings to symbolic standard cells are, of course, equally important to generators employed in silicon compilers. Furthermore, due to the process-independent nature of symbolic design, the generators can be mapped "for free" to a new process.

The second area in silicon compilation where symbolic layout is important is that of automated cell creation. Parameterized cells typically have a predefined layout structure, but some of their characteristics, such as their strength or the number of inputs to a gate, can be selected for a particular application. More complex functional parameterization permits the selection of combinations of features, such as synchronous reset and three-state outputs on a flip-flop. Symbolic layout makes such parameterization easier because geometric details are handled automatically, freeing the parameterizing software to deal only with topology and symbolic device sizing. Synthesized cells, with unique layout that may originate from a transistor- or gate-level netlist, also benefit in the same way from a symbolic layout back end.
8. CONCLUSION

Symbolic layout has been in use at Bellcore and a few other companies for some time, and its broader use in industry is emerging. The symbolic layout methodology has demonstrated high performance, productivity and first-silicon success. New symbolic systems, such as DASL, will provide still higher productivity and quality of layout while extending the applicability of symbolic layout to other technologies, like GaAs, BiCMOS and bipolar. Symbolic layout will also be employed as a key back-end layout element in many higher level tools such as silicon compilers. Most significantly, its technology insensitivity will preserve the considerable investment in layouts, from cells to chips.

9. ACKNOWLEDGEMENTS

Deborah Rappaport prepared this paper in her usual expert fashion. We would also like to acknowledge the rest of the DASL group - Phil Benjamin, Steve Okun, Anna Reidy, and Deborah Rappaport.

References


TABLE 1

EXPERIENCE

SOME EXAMPLES

<table>
<thead>
<tr>
<th></th>
<th>SPEED (MHz)</th>
<th>TRANSISTORS (K)</th>
<th>DENSITY (TR/MIL2)</th>
<th>DESIGN TIME (PERSON-MONTHS)</th>
<th>PRODUCTIVITY (TR/DAY)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 X 16 Crosspoint</td>
<td>200</td>
<td>7</td>
<td>.62</td>
<td>6</td>
<td>60</td>
</tr>
<tr>
<td>16 X 16 DCT</td>
<td>15</td>
<td>75</td>
<td>1.0</td>
<td>18</td>
<td>210</td>
</tr>
<tr>
<td>Digital Phase Aligner 1</td>
<td>45</td>
<td>15</td>
<td>.72</td>
<td>2</td>
<td>380</td>
</tr>
<tr>
<td>Digital Phase Aligner 2</td>
<td>150</td>
<td>22</td>
<td>.85</td>
<td>3</td>
<td>370</td>
</tr>
<tr>
<td>4 - Tap Digital Filter</td>
<td>20</td>
<td>18</td>
<td>.69</td>
<td>8</td>
<td>110</td>
</tr>
<tr>
<td>8 - Tap Digital Filter</td>
<td>30</td>
<td>32</td>
<td>.68</td>
<td>8</td>
<td>200</td>
</tr>
<tr>
<td>Video Line Store</td>
<td>40</td>
<td>90</td>
<td>1.68</td>
<td>6</td>
<td>750</td>
</tr>
<tr>
<td>Frame</td>
<td>100</td>
<td>6</td>
<td>.34</td>
<td>6</td>
<td>50</td>
</tr>
<tr>
<td>Batch - A</td>
<td>60</td>
<td>20</td>
<td>.79</td>
<td>6</td>
<td>170</td>
</tr>
<tr>
<td>Batch - B</td>
<td>60</td>
<td>10</td>
<td>.17</td>
<td>3</td>
<td>170</td>
</tr>
<tr>
<td>Banyon</td>
<td>60</td>
<td>12</td>
<td>.69</td>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>Regenerative Phase Aligner</td>
<td>100</td>
<td>7</td>
<td>.51</td>
<td>4</td>
<td>90</td>
</tr>
<tr>
<td>Transmit TSI</td>
<td>8</td>
<td>7</td>
<td>.55</td>
<td>5</td>
<td>70</td>
</tr>
<tr>
<td>Receive TSI</td>
<td>8</td>
<td>7</td>
<td>.57</td>
<td>5</td>
<td>50</td>
</tr>
</tbody>
</table>