An interactive system for VLSI chip physical design

by W. H. Elder
P. P. Zenewicz
R. R. Alvarodiaz

The Federal Systems Division has developed a structured design methodology and a companion chip physical design system that has been used to build seven large VLSI chips (ranging in size from 7K to 36K logic primitives). Using the MVISA system, a logic designer has complete control and responsibility for the total chip design. Our experience has been that when this highly interactive software and methodology is used, chip physical design requires less than two weeks. This is a significant savings in design time; but more importantly the designer can allocate more schedule for logic design and simulation. This paper describes how FSD's unique interactive physical design system has improved productivity of VLSI design.

Introduction

An important advancement in design automation software has been the development of automatic wiring programs. To increase the percentage of automatically completed wires, a good placement of the circuits being connected became necessary. Therefore, automatic placement programs were created. These first programs were very expensive to run because they used large amounts of CPU time. In 1970, typical CPU times for placement and wiring ranged from one to two hours on an IBM System/370 Model 168-3 class system [1].

The cost/performance ratio of computing was also very different during the early 1970s. This favored using large batch programs designed to complete as much work as possible in one costly job. At the same time, interactive processing was becoming popular for the productivity gains it gave to many applications, despite terminal and computing costs. However, Design Automation (DA) applications were not appropriate interactive applications.

A concurrent development (or perhaps an effect) was the use of design services. These services were provided by groups of data processing people trained to run DA programs, especially the physical design programs. As a result, part of the design schedule and budget was allocated for this expense.

Initially, the design services group took design engineer inputs and interfaced these data to the DA programs. This group's main purpose was to convert logic to a physical design. Solutions were evaluated for performance by the design engineer, who made any logic changes required, and then the process started over. This interaction between different organizations required duplicate design training, often created interface problems, and sometimes caused schedule delays. The required knowledge and the cost and complexities of running most DA programs (especially the physical design programs) prohibited their use by many people. There was considerable effort in manual placement of wires required to complete the physical design. This group was also trained in this aspect.

During the mid-1970s, hardware computing costs decreased, performance increased, and interactive DA applications became feasible. Logic capture, also called logic

entry, was one of the first phases of design implemented as an interactive application. In this application, the logic designer interfaced directly with a DA program. Before this the logic designer transferred source data (e.g., a logic diagram) to a design services group to translate to machine-readable format. Now the designer entered the logic diagram data directly.

IBM's Federal Systems Division developed an interactive logic entry program which became widely used throughout IBM. The program's acceptance by the engineering community and the productivity gains set the stage for other interactive DA applications. From the concepts developed in this original program, a complete interactive DA system for wiring cards and boards was developed. This base was extended to a chip design system capable of placing and wiring over 37 000 gates of logic. The goals of the DA system were as follows:

- Maximum usefulness to an engineering community with minimal data processing skills and start-up training.
- Complete control by the logic designer over all aspects of chip design.
- 99 percent automatic physical design; 100 percent checked and guaranteed error-free.
- Total elapsed time less than one hour for placement, wiring, and performance analysis.
- Audit trail of all actions executed.

Algorithm selection was key in meeting many of these goals. Typical algorithms for automatic placement, wiring, and performance analysis are CPU-intensive. It is difficult to provide effective interactive applications for CPU-intensive functions. The system structure, algorithms, and data structures were selected and implemented with interactive response time in mind. This paper discusses the features and key algorithms of an interactive DA system that achieved this and the other stated goals.

Design automation system overview

Since 1981, FSD has used a design system known as Manassas VLSI Interactive System for Automation, abbreviated herein as MVISA, for physical design for most of its chips. MVISA is a graphically aided interactive design automation system. Its operational functions and features reduce the development schedule and improve engineering productivity. This is accomplished because the logic designer controls the physical design and makes any trade-offs and logic design adjustments required by the physical layout. This approach also eliminates the schedule and cost impacts of involving a group of physical design specialists.

MVISA provides a set of subsystems for logic entry, predesign checks, placement, wiring, logical/physical checking, mask data generation, performance analysis, and audit. The system operates under an MVS/TSO or VM/CMS operating system with a minimum region of six million bytes. The subsystems are integrated via a system monitor with their respective design data and technology rules. The monitor provides facilities for selecting and managing design and technology rules data located in external files. It also controls the interactive execution of the MVISA subsystems. **Figure 1** illustrates the organization of the MVISA system.

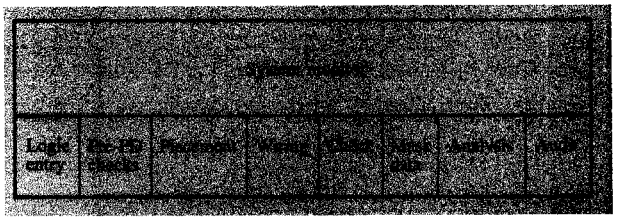

Figure 1

Organization of system functions in MVISA.

Any IBM 3270-type display terminal can be used to operate and control the system. Full-screen menus describe available options and minimize data entry. Program function keys and optional lightpen input functions further enhance the ease of operation. This approach allows even infrequent users to use MVISA with minimum training (one week or less). On-line HELP functions are also available. The IBM 3277 graphic attachment (RPQ 7H0284) provides an interactive vector-graphic display capability.

Within the MVISA system is an integrated data structure that combines logical and physical data with technology rules data. For execution efficiency, these data, common to all subsystems, reside in main memory and are key to providing satisfactory interactive response time to the user. However, to prevent excessive memory requirements, dynamic memory space allocation and management techniques are also employed.

MVISA uses a linked list data structure for the logical and physical design data. Important factors in the design of the data structure were compactness of data, access speed to commonly used data through linked lists, hierarchical structure for quick access, flexibility to add new data fields without changing the base structure, and completeness of data content to accommodate all design system functions. Many of these factors influence interactive response time or the suitability for foreground execution.

The MVISA design system applies to masterslice (gate array) and Master Image design methodologies. An introduction to FSD's Master Image chip approach and an overview of the physical design methodology are given before the detailed discussion of the DA system.

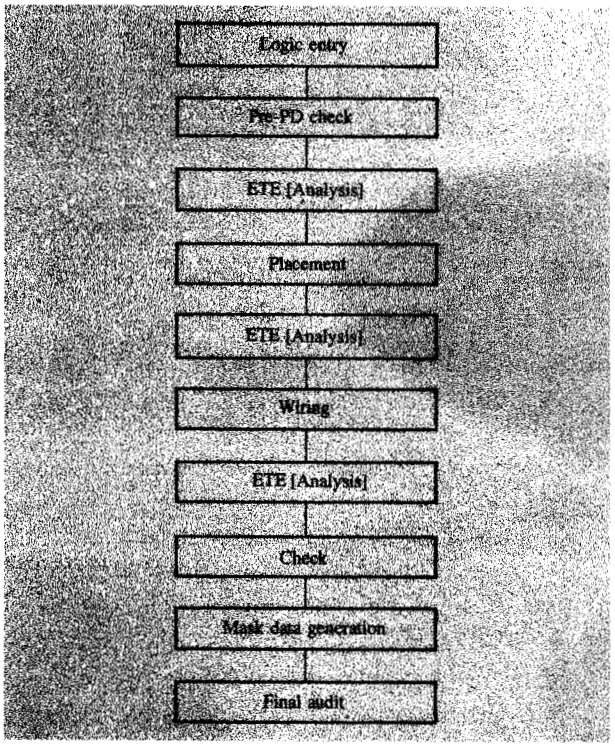

Figure 2

Master Image physical design methodology.

Master Image chip approach

FSD's need to design low-volume complex chips on tight schedules led to the Master Image chip design approach. This approach was chosen over a custom design alternative (which would have given greater chip performance) because design and schedule costs could not be justified for low volumes. Gate array and masterslice approaches were discarded because of lower achievable function at nearly the same development cost. The FSD Master Image is similar to the PHILO [2] approach developed at IBM laboratories in Rochester, Minnesota and Burlington, Vermont. This approach uses a predesigned image with a fixed power bus distribution and preallocated areas for circuits and wiring. Combined with the image is a circuit and macro library which includes a full range of macro function and complexity. Higher-function macros achieve the high gates per mm^2 associated with custom design and corresponding improved performance. **Table 1** lists the circuits and macros available in the 2.0-μm Master Image library. These library macros were chosen to meet the signal processing types of applications most common in FSD systems.

Table 1 Circuit and macro functions in 2.0-μm Master Image library.

Function	*Cells*
1–4-W NOR	1
5–9-W NOR	2
Tri-state OCD	1
Receiver	1
Transceiver	1
2-1 Mux	1
4-1 Mux	2
Exclusive-OR	2
LSSD SRL	2
SRL clock driver	5
Clock driver	3
4-bit ALU	32
Carry save adder	6
9-bit Parity	10
4-bit Up/down counter	10
4-bit Mag. comparator	10
16 × 16 Multiplier	630
16 × 54 RAM	228
32 × 32 RAM	228
64 × 32 RAM	304
64 × 18 RAM	192
256 × 16 ROS	200
128 × 16 ROS	120

Image description

The Master Image chip image, shown in **Figure 2,** is 7.4 × 7.4 mm in size and has a total of 168 C4 pads, 116 of which can be used for signal I/Os. Voltage and ground power buses are supplied on second-level metal to the I/O circuits which reside under the C4 pads around the chip periphery. Power supplied to the internal cells is isolated from the I/O circuit's power on the chip to decouple noise through the bus distribution system. Power to the internal cell structure is supplied through three second-level vertical buses at the center and sides of the cell array. Voltage and ground first-level metal fingers run horizontally through each row of cells. The power busing was designed to handle a maximum of 2 mW per cell location, considering both current densities and voltage drop effects on circuit design. Each circuit and macro adheres to this power budget and can be placed anywhere on the image.

The 7.4-mm chip contains a total of 2432 cells arranged in 15 double rows and two single rows of 76 cells. A cell is an area that has five logic service terminals (LSTs). Between each LST is a free wiring track. The basic cell can contain up to a four-input NOR. Table 1 lists the cell counts for the available functions in the Master Image 2.0-μm nMOS technology. An 8.0-mm chip size with 2952 cells and 178 signal I/Os is also available. For the 1.25-μm nMOS technology, an 8.0-mm chip size with 8468 cells and 179 signal I/Os is available.

Global wiring for the chip is performed using first- and second-level metal. First-level metal is used for circuit

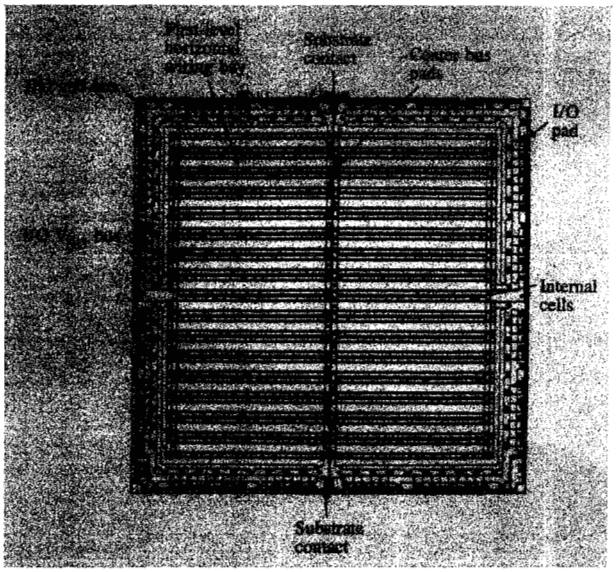

Figure 3

Chip image for 7.4-mm chip for 2.0-μm nMOS master image.

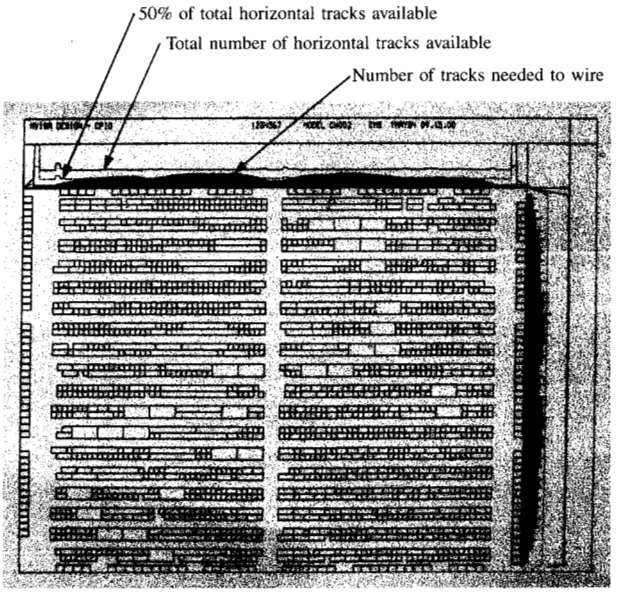

Figure 4

Wirability histogram showing demand vs. availability after placement.

implementation, thereby blocking the cell rows. Therefore, first-level global wiring runs horizontally in the wiring bays between the cell rows. Second-level metal is used for vertical wiring in the areas not blocked.

Master Image physical design methodology

This section briefly describes the general flow using MVISA to design a Master Image chip as illustrated in **Figure 3.** A more detailed discussion of the complete methodology can be found in [3].

Using the LOGIC ENTRY subsystem, the user begins the MVISA design by entering the logic description into the design database. The LOGIC ENTRY subsystem allows the interactive creation and editing of logic diagrams. Once the logic has been entered, checks are made to verify correct usage of blocks on the logic diagrams. Missing or incorrectly named pins are detected. This check is run after any logic change. The next step is the predesign [PRE-PD] checks. Entry into PRE-PD checks causes the internal database to be initialized with the physical design description and checks design integrity.

In addition, net capacitance estimates for each net are obtained to identify loading conditions. This estimate adds driven circuit pin capacitance to an estimated wiring capacitance to produce a result that is a function of fan-out. Thus, "critical" net loading conditions can be flagged and corrected early in the design cycle.

Failure at any point in the verification process requires the user to re-enter the LOGIC ENTRY subsystem, correct the logic, and rerun the checks.

An Early Timing Estimator (ETE) is available in the ANALYSIS subsystem. Estimated capacitances are used to calculate individual block delays. Worst-case path delays are accumulated from primary inputs to registers, from registers to registers, and from registers to primary outputs. The user can then anticipate the "critical" paths before starting the chip physical design. However, since physical design can be accomplished in less than one hour and provides accurate capacitance data, this option is not always used.

Both automatic and manual options aid in chip circuit placement. Large macros (those occupying more than 32 cells) must be preplaced manually on the image. During manual placement, using the Tektronics 618 display and joystick, the user evaluates the wiring flow and places the macros in "natural" positions. The remaining circuits are placed automatically by the program.

Wirability analysis checks the placement quality and displays the results on a Tektronics 618 display. This analysis gives a histogram of wiring demand versus wiring channel availability and is used to identify chip areas with potential wiring problems. **Figure 4** is an example of the wiring demand histogram.

After placement, The Early Timing Estimator (ETE) in the ANALYSIS subsystem can again be used to calculate

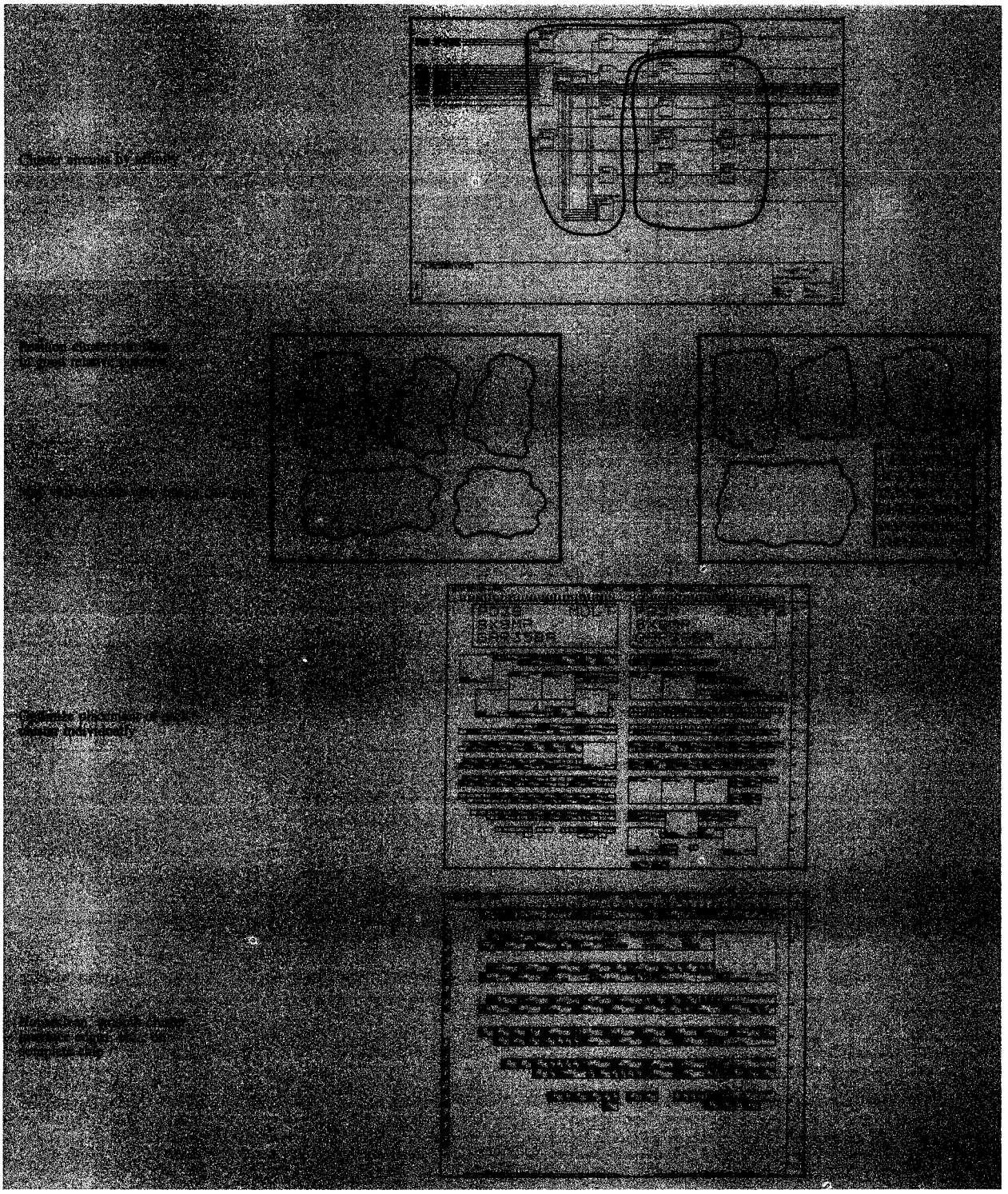

Figure 5

Steps in automatic placement process.

delays using the placement locations and estimated wire lengths. If there are no placement changes required, automatic wiring is performed.

The WIRE subsystem has three available modes: automatic channel packer, automatic line probe, and manual. The bulk of the wiring is done using the channel-packing algorithm. This algorithm normally results in a completion rate above 99 percent for a 2.0-μm chip. The line probe algorithm completes any remaining connections. Manual wiring using the Tektronics 618 display and joystick is used primarily for rerouting wiring required by logic changes.

After wiring, ETE is always used to evaluate performance using actual wiring lengths to get actual wiring capacitance and delay calculations.

The completed design is checked for wiring correctness and ground rule violations. Additional checks, using the CHECK subsystem, are power and capacitance limits. Total chip power dissipation and net capacitances are checked against specified maximums.

The final physical design step is generation of mask graphics data. These data are in an IBM standard format, Graphics Language/One (GL/I).

The AUDIT subsystem generates a complete audit trail of all executed options and the results of all checks made. This report is reviewed by manufacturing prior to fabricating the design.

MVISA design system

This section provides additional details on the techniques used for automatic placement, wiring, and timing analysis.

Automatic placement

Figure 5 depicts an overview of the placement process, which includes clustering, zoning, initial placement, and improvement.

- *Clustering phase*

During clustering, individual circuits are grouped to form new placement entities. In the first clustering pass, the clusters are individual circuits. As repetitive passes are performed, the clusters may become groups of circuits. The grouping algorithm uses the pairwise attraction force between current clusters, the external pulls on the clusters, and their relative sizes. The strength of the attractive forces is a function of the number of nets connecting the clusters being considered. Multiple passes are performed, possibly producing new clusters and new strengths. The number of passes is a user-controlled parameter. Limits are placed on the resulting cluster size. The clustering phase ends when no more clusters are formed or the preset number of passes is reached. This process is similar to that of Feuer et al. [4] and Lallier and Jackson [5].

The clusters formed are used as objects for the zoning phase of placement. Clustering tends to alleviate the local optimum problem, since closely coupled circuits are moved together. This approach uses much less computer time because the number of objects is drastically reduced. This reduction is essential to maintain a reasonable interactive response time.

- *Zoning phase*

The zoning step establishes cut lines, imaginary lines dividing the chip into sections called zones. The first cut line is vertical and divides the chip into two equal zones. The second is horizontal and divides these two zones into four, and so on. As each cut line is introduced, zones become smaller. The first two cut lines are illustrated in **Figure 6.** V1 is the first vertical cut line, a chip bisector; H1 is the first horizontal cut line, a chip quadrasector.

Clusters move across cut lines attempting to minimize the number of nets crossing the line. Minimizing wiring congestion across a cut line also minimizes total wire length for the design. A cluster assigned to a particular zone must remain in that zone or a zone derived from its original zone.

The zoning process repeats until the designer-specified number of cut lines is made. Each cluster is assigned to a chip area. The size of this area varies according to the cluster size and number of cut lines. For example, when two cut lines are used [quadrasection], the area of each of the four zones would equal one-quarter of the total chip area. The zoning process is similar to that of Corrigan [6] and Breuer [7]; the major difference is that clusters are used, rather than individual circuits. Lallier and Jackson [5] used clusters, but with a different interchange technique.

- *Initial placement phase*

After zoning, clusters are dissolved into individual circuits and placed into their assigned zones. Cluster placement uses a form of constructive initial placement similar to that of Hanan and Kurtzberg [8]. This algorithm uses the half perimeter length of the net's enclosing rectangles as a minimization parameter. **Figure 7** shows an example of the half perimeter for a three-terminal net. The half perimeter is equal to L1 + L2. Both the placed and unplaced circuits are used in the calculations as all circuit zone areas are known. The result of using all circuits is better than the classic constructive initial placement technique described in [8].

- *Improvement phase*

After the clusters are broken and placed, an improvement phase is performed. Circuits or groups of circuits are moved to improve wirability by reducing local wiring congestion. The net half perimeters of the moved circuits are recalculated; only good moves are kept. Because it is assumed that the previous steps have derived a somewhat "good" placement, the circuit movement is restricted to cells within the zone assigned during the zoning process. The improvement step repeats until either no improvement is

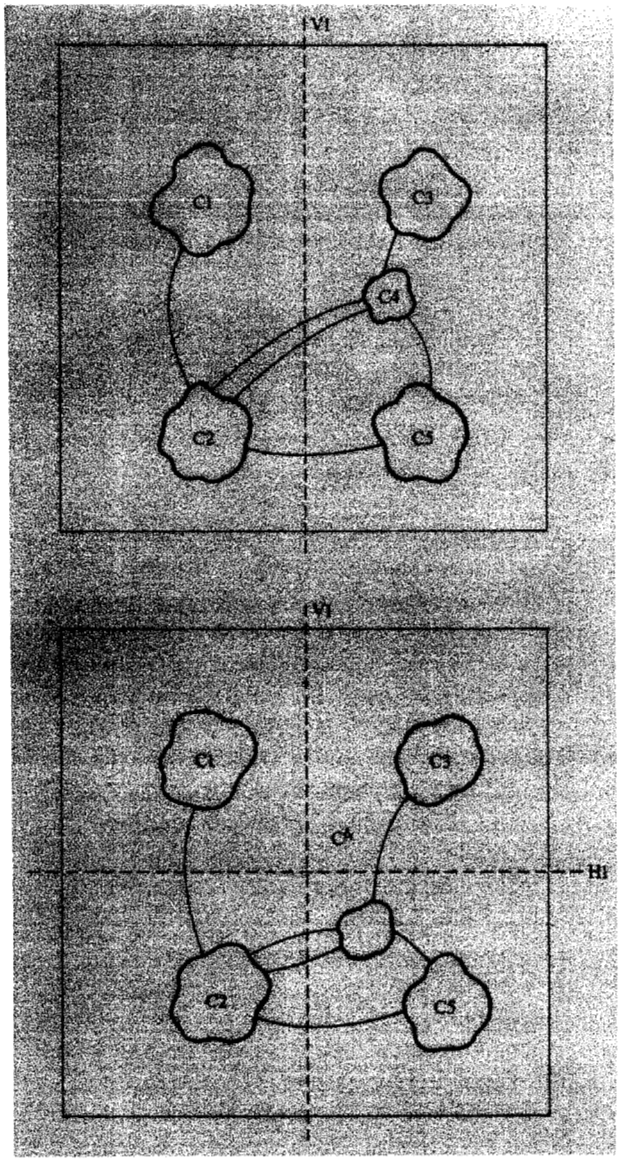

Figure 6

Circuit clusters are moved to minimize wires crossing cut lines.

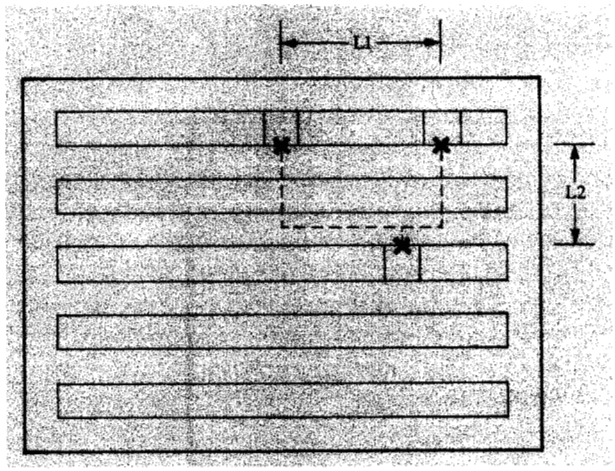

Figure 7

Net half perimeter is minimum length to wire net.

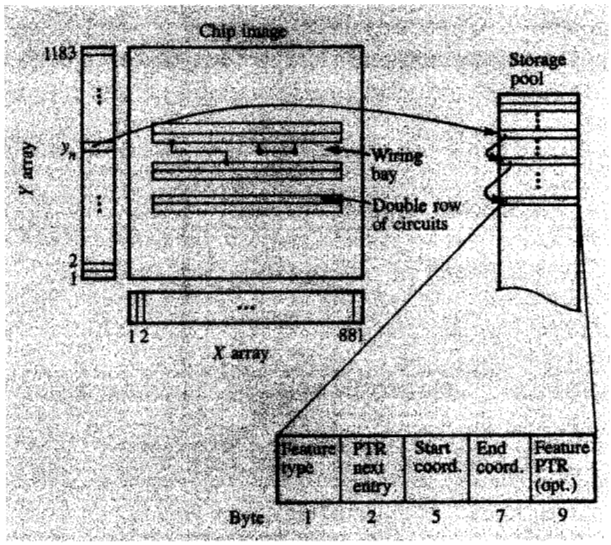

Figure 8

Data structure scheme for modeling chip image.

obtained, or the designer-specified number of passes is reached.

Automatic wiring

Automatic wiring uses several wiring techniques. The line-packer algorithm, or channel router, attempts to "pack" line segments in the least area in the bays. This very structured algorithm connects only logic service terminal (LST) end points and allows only two bend wires in the bay and no wrong-way wiring. These constraints assist the algorithm's efficiency. It runs in an interactive environment and has a completion rate over 99 percent.

The probe algorithm uses a limited amount of wrong-way wiring and LSTs, vias and existing wire segments in the net as targets to complete a connection. In practice, the line probe algorithm is used for those connections not completed

by the line packer. This combination has proven to be very effective for FSD's Master Image chips. A 7.4-mm chip with a 80–90 percent cell utilization can be wired in 2–3 minutes of IBM 3033 CPU time. This helps meet both response time and schedule goals. Manual wiring, also available, is used for wiring changes or to prewire nets before automatic wiring. Additional performance statistics are discussed in a later section.

• *Chip image modeling*

Both wiring algorithms require a description of the chip's physical layout to place connections without violating any technology ground rules. The method of describing the chip image to the design automation programs is influenced by chip size and the allowed wire periodicities. For processing efficiency, it is desirable to have an in-storage image representation which contains data describing blockages, vias, LSTs, wires, etc. A typical method is an array with data entries for each possible chip grid location. This method usually requires at least one byte of storage for each location. For the 7.4-mm 2.0-μm Master Image chip, this translates to an 881 × 1183-element array requiring 1.04 million bytes of storage for each wiring plane. The increasing storage requirements of larger chip sizes and more advanced technologies make this technique impractical. Keeping the execution storage requirements consistent with interactive execution necessitates an alternate approach for an image data structure.

Because much of the data stored on adjacent grid locations is the same, e.g., blockage-covered areas, wires spanning multiple grids, and much unused area, only the actual features are needed in the data structure. These features are stored using a threaded list technique; each feature requires ten bytes of storage. These features are stored in a common pool. The data for each entry include a pointer to the next entry, the feature type, starting coordinate, ending coordinate, and an optional pointer into the design system data structure. **Figure 8** shows how data are accessed by an array for each axis of the chip. The entry in the Y array for a given Y coordinate points to the head of the list for all plane 1 data at that Y coordinate. For processing efficiency, entries are sorted according to their starting coordinates.

Two sets of lists are kept for each plane. The primary list contains blockages, vias, LSTs, and wires in the primary direction. The secondary list contains wires in the wrong-way direction. This technique, effective in reducing storage requirements, provides a 10:1 reduction compared with the byte/grid method. This storage reduction is necessary to contain the 1.25-μm chip image and still provide interactive execution of the wiring algorithms.

• *Line-pack algorithm*

This algorithm is similar to many channel routing programs [9–11], since a global phase preprocesses the connections.

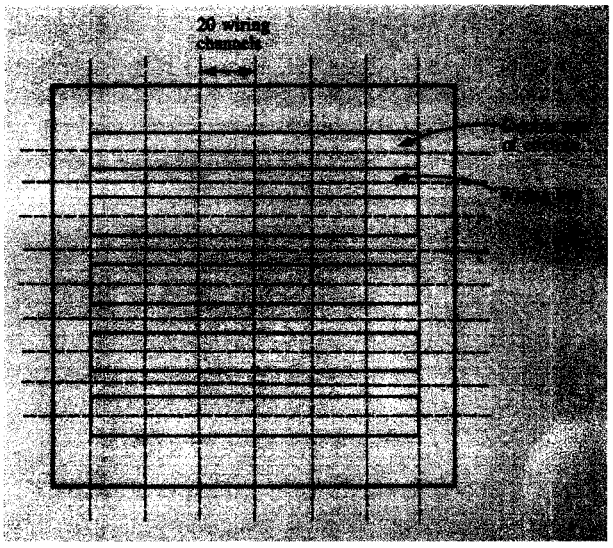

Figure 9

Global cut lines create global rectangles for global routing.

The global phase determines a coarse route for each two-point wiring connection. The chip image is subdivided into global rectangles using vertical cut lines approximately 20 wiring channels apart and horizontal cut lines through the middle of each wiring bay and the center of each double row of circuits. This is shown in **Figure 9.** This operation creates large rectangles on the chip image with sides of approximately 20 wiring channels each.

The global phase attempts to find a path through the rectangles for each wiring connection. The image description is first examined along each cut line for blockages, resulting in the number of wiring channels available on each global rectangle's edge. A "directed maze-runner technique" is used to seek a path between connections. A cost measure algorithm calculates the least costly direction for leaving any rectangle. Cost measures include remaining wire channels available, direction change, and X/Y weighting. The cost goal is to minimize congestion, thereby increasing the probability that all wire connections will be made during the detailed wiring phase.

Ordering effects known to influence the global wiring phase [11] are reduced by routing the wire connection list four times. During each routing, the connections are made using the least costly path. On the first pass, wires compete only against the connections made during this pass. On each succeeding pass, each wire's previous route is first subtracted out of the total routing; then a new least costly route for the connection is determined. This connection is thus routed in the presence of the total chip wiring, which is continuously

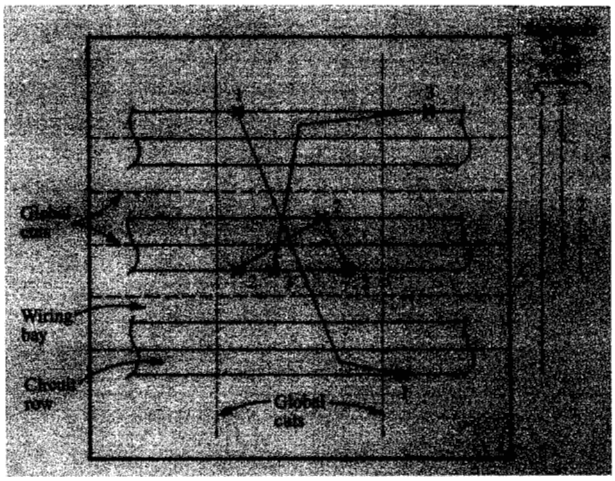

Figure 10

Creation of vertical segments to be wired from global paths.

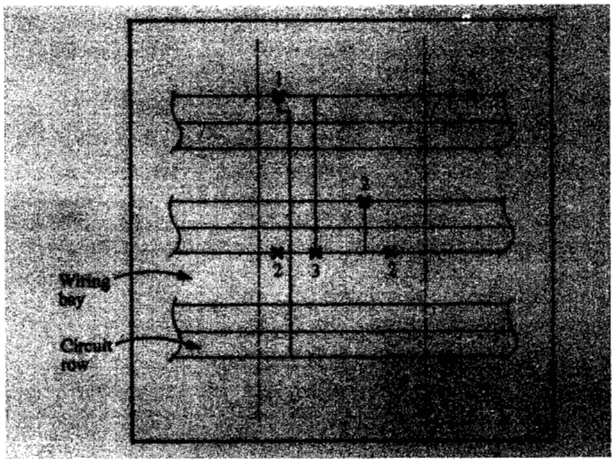

Figure 11

Possible final vertical wiring for previous example.

changing. This iterative process redistributes the wire routing to minimize congestion. Additional controls are available to limit the maximum number of wires that can cross any global rectangle's edge. Experimental results on actual chips, varying the wiring order and the number of global passes, have shown only minor variations in total wire length.

The line-pack algorithm uses the global paths as a roadmap for creating the detailed placement of each wire segment. The algorithm begins by creating all inter-bay vertical wiring. This transforms all vertical communication to the boundaries of each wiring bay so that the wiring problem is reduced to the classical one of routing a channel with LSTs on each edge. Each wiring bay is then wired, making horizontal track assignments and creating additional vertical segments to connect the LSTs. The wiring is completed by routing the connections in the two vertical wiring bays that exist on the left and right sides of the chip that are used for the I/O wiring. The same algorithm is used for these bays.

The vertical inter-bay wiring is processed in sections, one at a time. These sections are defined by the vertical cut lines used during the global wiring phase. From the global paths, a list of vertical line segments that cross circuit rows is created. These segments are sorted by location of their X global rectangles. A simple example is shown in **Figure 10.** The exact Y coordinates of the segment ends can be assigned at this time. First, the longest segments are assigned, attempting to make a direct connection to an LST at either end. If a direct connection cannot be made, an assignment in the adjacent channel, either left or right, is attempted. If this is unsuccessful, a search is made across the full width of the global rectangle to find a location for the wire segment. Each segment is processed only once. **Figure 11** illustrates a possible vertical wiring for the example used in Fig. 10.

After the inter-bay wiring is completed, each wiring bay has its original LSTs and the points created by the vertical wiring step, called pseudo-LSTs, on its edges. Net identification data is tied to each of these LSTs and pseudo-LSTs. There are many restrictions and technology conditions to be accounted for in the wiring of the bay. For example, technology ground rules often put restrictions on the placement of vias relative to each other or to LSTs, e.g., do not allow vertically adjacent vias. These conditions are controlled through a set of external rules datasets for each technology.

The first step in the bay wiring determines the set of horizontal wire segments required. The algorithm treats each two-point connection in the bay separately, as in Deutsch's dogleg channel router [12]. The bay is scanned from left to right, matching net numbers. Each potential horizontal segment found in this process is matched against the global routings to verify its inclusion in the bay.

Once the set of horizontal segments is identified, a search is made to discover cyclic constraints or "constraint loops." Attempts are made to break these loops by examining the channels to either side of each segment's end points. An open channel in any one of these spots allows a one-grid-length wire to be placed at the LST, thus moving the LST location for the bay wiring operation and breaking the loop. Before the move is made, the potential location is checked to verify that a new loop is not being created. A simple example is given in **Figure 12.**

Each horizontal segment in the bay is assigned a type code depending on the connection to be made on each end.

Figure 13 shows all possible connection types. This scheme allows all segments to be grouped by type and reduces the number of segments that have to be tested when choosing a segment for a potential track assignment. An array can be set up for the top and bottom bay edges which contains at most two entries per X coordinate for testing whether a constraint has been satisfied to allow assignment of a segment.

The actual wire segment assignment proceeds from the left edge of the bay [10]. Segments are sorted by their lower X coordinate. The first segment which satisfies all constraints and technology conditions is chosen for placement in the first track at the top of the bay. The search continues for the next lowest X-coordinate segment that can fit in the partially filled track and so on until the track is filled or no more candidates are available. Track filling alternates between the top and bottom of the bay. The vertical wire segments created during the bay wiring are merged with the segments formed during the vertical wiring step, reducing the total number of segments required.

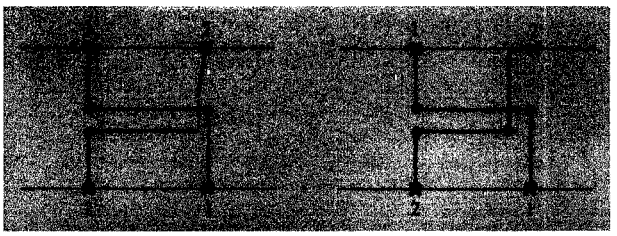

Figure 12

Breaking a constraint loop through the use of an open channel next to LST.

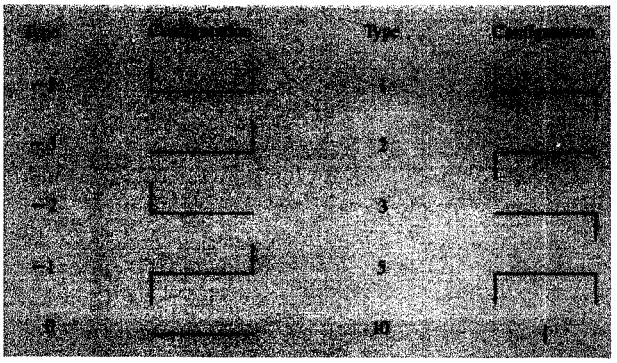

Figure 13

Possible configuration types for two-point connections used by line-pack algorithm in wiring bay.

Timing verification

Timing verification is a critical analysis task in the design of any chip. In particular for the nMOS process, circuit delays are highly dependent on circuit power and chip wiring capacitances. Since the chip wiring loading is not known until chip physical design is complete, it is very important to be able to couple a timing analysis/verification tool tightly into the physical design loop. Previous methods of performance analysis [13, 14], although very effective, were noninteractive and costly in computer resources, and required a long learning curve.

The Master Image design approach uses complex macros and the timing analysis program, the Early Timing Estimator (ETE), to take full advantage of this macro concept. Each macro is characterized as an entity in a technology rules dataset so expansion or decomposition into primitives is not required. Checking, such as data setup time at a latch, is easily done by testing signal arrival relationships at the boundary of a storage element.

The ETE attempts to provide the designer with quick, inexpensive timing information throughout the design process. It allows the designer to change the logic design and/or physical design, completing the design cycle in a shorter time and with a higher degree of confidence in the chip's performance.

The ETE is a subfunction of the ANALYSIS subsystem, and like all MVISA functions, operates interactively via menus. The ETE uses the MVISA data structure for the logic description and for the physical design data required for wiring capacitance calculations. Since Master Image designs use a library of macros (varying from simple NOR circuits to large 3000-gate multipliers), the rules dataset gives the delay from the input to an output pin. Analyzing the macros without expansion increases the computer processing efficiency and the understandability of the analysis reports. Both of these improve the usability of the ETE in the interactive environment.

Sequential and register-transfer logic designs operate under clock cycle constraints. Correct operation requires that all logic paths must meet some timing constraint: that is, data signals must arrive at storage elements according to some specification. The ETE determines the worst-case delay from any primary input (PI) or storage element to any succeeding primary output (PO) or storage element. Once the delays are known for all data and clock signals arriving at a storage element, tests can be carried out to verify that the proper relation exists between data and clock signals.

Figure 14 shows the general situation in register transfer logic. Here we have two Shift Register Latches (SRLs) and some connecting combinatorial logic. The output of SRL1 is triggered by the occurrence of CLKB′ and propagates through some combinatorial logic until it reaches the data

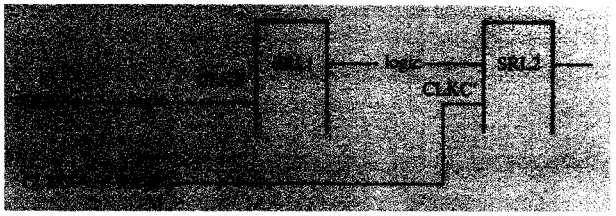

Figure 14

General register-transfer logic configuration.

input of SRL2. This data signal then is clocked into SRL2 by CLKC′. The clock signals CLKC′ and CLKB′ are derived from the clock signals CLKC and CLKB supplied at the chip I/Os.

The clock signals reaaching the SRLs are modified by the combinatorial logic used in the clock distribution network. This distribution logic can generally change the clock pulses depending on the number and types of circuits used. Different loading conditions in the distribution logic cause a timing skew in the clocks reaching each SRL. These distribution effects in the clock networking must be accounted for in determining the proper operation of each SRL.

To guarantee the reliable operation of SRL2, the data signal, whether it is a logic "0" or "1," must be available and stable for an amount of time before CLKC′. This time, the data setup time, is determined by the SRL circuit designer and is included in the ETE rules dataset. Additional data coded into the rule specify the minimum clock pulse width and clock separation required for the SRL operation. For this discussion we have only mentioned SRLs; the same type of data and tests are required for any macro containing clocked storage elements.

• *Logic leveling*

Our purpose so far is to determine and test the relation of the data to the clock signal at storage element inputs. To ensure the test, the worst-case arrival times for both logic polarities are needed. These worst-case arrival times are computed by ETE in a single pass through the logic blocks or circuits in the design. To bring about this single pass analysis requires a preprocessing step called "leveling" which sets up the order of computation for the logic circuits. To provide maximum processing efficiency, each block delay is computed only once, and at the same time, the cumulative path delay at its output is determined. This means that all inputs required for each block's computation must be available. These inputs are calculated assuming that all chip inputs are at level 0. The level of any block is taken to be one more than the highest level source of any of its inputs. Thus, the delay computations can proceed by processing all level 1 blocks, level 2 blocks, etc.

The ETE does this preprocessing leveling step starting with primary inputs to the logic being analyzed. The levels increase until a storage element or primary output is reached. The algorithm used for leveling assumes that the logic does not contain any functional feedback loops. However, detection of a feedback loop forces the program to end with an error message.

Results

To date, there have been seven 2.0-μm chips and one 1.25-μm chip designed using the MVISA design system. After a single design pass, all manufactured chips were functionally correct. The typical design cycle, from logic entry to release of manufacturing mask data, is four to six months. The schedule is highly dependent on chip complexity and the designer's experience, with most of this time spent for logic simulation. Each chip's physical design was completed in less than two weeks.

Typical CPU times and design complexities for some of these chips are shown in **Table 2.** In Table 2 a gate is defined as a three-input NOR or equivalent logic.

The first designs limited cell utilization to 80 percent. This was very pessimistic. Table 2 shows that Part Number A at 88 percent utilization met the goal of over 99 percent automatic wiring with the channel-routing algorithm. This table also shows that the most challenging designs were the 1.25-μm design (Part Number E) and the 8-mm 2-μm design (Part Number D).

The 1.25-μm design uses very large custom macros (some over 600 cells) which block second-level metal. Table 2 reflects this in the custom macros percent value of 55. Because of this second-level blockage, all vertical wiring is routed around these macros. However, the width of these custom macros leaves few cells between the macro and the power buses to route vertical wiring. But, since the macros contain a large amount of prewired logic, the number of nets (1991 in Table 2) and therefore the number of connections required during chip design (about 5000), is low when compared to the number of logic gates on the chip (over 36 000). The channel-routing algorithm achieves about 98 percent automatic wiring on this design.

The 8-mm chip uses the same 2-μm circuit library as the 7.4-mm chip. Since many of the macros span cells and wiring bays, the number of wiring channels between the cell rows is the same for both images. The number of channels chosen for the 7.4-mm image was 27. This bay size was set using the models of Heller et al. [15] for predicting wiring space requirements and realizing that the use of macros reduces the LST density per gate on the chip. This compares with 26 wiring channels used in the PHILO master image chip design system [2], which is a 6.2-mm

Table 2 MVISA performance and chip statistics.

Part number	A	B	C	D	E
nMOS tech (μm)	2.0	2.0	2.0	2.0	1.25
Chip size (mm)	7.4 × 7.4	7.4 × 7.4	7.4 × 7.4	8.0 × 8.0	8.0 × 8.0
Cells on chip	2432	2432	2432	2952	8468
Cells used (%)	88	68	80	93	79
Custom macros (%)	17	14	17	0	55
Number gates	8276	7638	8928	11772	36272
Gates/mm^2	197	182	214	233	735
Global nets	1016	997	1176	2271	1991
Auto wired[1] (%)	99.8	99.7	99.9	98.5	98.4
CPU placement[2]	6.5	6.0	6.5	9.0	10.0
CPU wiring[2]	1.25	1.15	1.32	3.0	3.5

[1] Channel-pack algorithm.
[2] IBM 3033 min.

chip image. Using the same size wiring bay for the larger chip increases the contention for the wiring bay area.

The wiring also becomes more difficult when a design uses no custom macros (circuits greater than 32 cells). This is true of Part Number D, which is also an 8-mm chip and uses over 93 percent of the cells. This chip contains more nets (2271) and therefore requires more connections (5287) to be made during chip wiring than previous 2-μm designs. The channel router is successful on 98.5 percent of the connections, slightly lower than the 99 percent goal.

The current challenge is an 8-mm design which occupies 99 percent of the cell area. The chip was successfully wired (only one unrouted connection) using a combination of the algorithms described and placement algorithms under development. These new algorithms have not been regression-tested on other chip designs to verify their overall effectiveness.

Summary

The proven ease of using the design system and the success of the chip designs have overcome any initial resistance in the design community and have brought acceptance of the design philosophy. A large measure of this success is due to the economical CPU time usage achieved by the physical design algorithms described. These algorithms have allowed for interactive design of VLSI complexity chips. This real-time design environment is achieved using an integrated database, an innovative image-handling technique, macros, and a pragmatic approach to physical design.

The database is available to all subsystems without data transformations, and parameter-related items are easily retrieved through a linked list structure. The image data storage mechanism effectively partitions information for the wiring algorithms, since only a very small subset of the total data is required for each operation. Also, the use of macros not only increases circuit density per unit area, but further simplifies the placement problem by reducing the number of objects to be placed on an image. In addition, because there are fewer external LSTs in a macro-oriented design, there are fewer connections required during global chip wiring. Another factor that adds to the effectiveness of the design system is that since the designer can interact easily with the system, the placement and wiring algorithms' solution need not be "perfect"; an unplaced circuit or unrouted connection can quickly be added by a designer using the graphic aid. Also, multiple wiring algorithms are provided to take advantage of their individual strengths. The philosophy has been to provide a strong, flexible set of tools to support the designer's intelligence in the design process.

Future plans

Department of Defense application requirements and the semiconductor industry are rapidly moving into CMOS and submicron technologies. Lithography advances are allowing chip sizes to increase up to 14 mm. The MVISA design system must be adapted to meet these new advances. CMOS and submicron applications may require more wiring levels and thus new wiring algorithms. The increased data volumes that are presented by larger chip sizes and submicron technologies will require new techniques in image modeling and physical design processing. The incorporation of more hierarchical processing in the design methodology will also be required as gate counts approach or exceed 100K per chip.

Acknowledgments

The authors would like to acknowledge the contribution of the FSD Owego DA organization under the management of J. Sents, especially J. Cooper (who has since left IBM) and R. Vincent. They developed the baseline for this chip system with the OASIS card/board system. To this base, the Manassas DA development department added the function described. We would also like to extend our appreciation to

the VLSI design community and center of competence for the feedback they provide which helps MVISA to continue to meet the designer's need. And above all, we would like to thank John Isaac for his leadership in creating the MVISA system.

References

1. P. W. Case, M. Correia, W. Gianopulos, W. R. Heller, H. Ofek, T. C. Raymond, R. L. Simek, and C. B. Stieglitz, "Design Automation in IBM," *IBM J. Res. Develop.* **25,** No. 5, 631–646 (1981).
2. R. Donze, J. Sanders, M. Jenkins, and G. Sporzynski, "PHILO—A VLSI Design System," *Proceedings of the 19th Design Automation Conference,* 1982, pp. 163–169.
3. K. Ahdoot, R. Alvarodiaz, and L. Crawley, "IBM FSD VLSI Chip Design Methodology," *Proceedings of the 20th Design Automation Conference,* 1983, pp. 39–45.
4. M. Feuer, W. R. Heller, et al., "EMERALPS—Automatic Partitioning and Placement of Logic Circuits on Weinberger Images," *Proceedings of the IBM Design Automation Conference,* 1977, pp. 9–22.
5. K. W. Lallier and R. K. Jackson, "A New Circuit Placement Program for FET Chips," *Proceedings of the 16th Design Automation Conference,* 1979, pp. 109–113.
6. L. Corrigan, "A Placement Capability based on Partitioning," *Proceedings of the 16th Design Automation Conference,* 1979, pp. 406–413.
7. M. A. Breuer, "Min-Cut Placement," *J. Design Automation & Fault Tolerant Computing* **1,** No. 4, 343–382 (1977).
8. M. Hanan and J. M. Kurtzberg, "Placement Techniques," *Design Automation of Digital Systems, Theory and Techniques,* M. A. Breuer, Ed., Prentice-Hall, Inc., Englewood Cliffs, NJ, 1972, Vol. 1, Ch. 5, pp. 213–282.
9. K. A. Chen, M. Feuer, K. H. Khokhani, N. Nan, and S. Schmidt, "The Chip Layout Problem: An Automatic Wiring Procedure," *Proceedings of the 14th Design Automation Conference,* 1977, pp. 298–302.
10. A. Hashimoto and J. Stevens, "Wire Routing by Optimizing Channel Cell Within Large Apertures," *Proceedings of the 8th Design Automation Workshop,* 1971, pp. 155–169.
11. B. S. Ting and B. N. Tien, "Routing Techniques for Gate Array," *IEEE Trans. Computer-Aided Design* **CAD-2,** No. 4, 301–312 (1983).
12. D. N. Deutsch, "A Dogleg Channel Router," *Proceedings of the 13th Design Automation Conference,* 1976, pp. 425–433.
13. Robert B. Hitchcock, Sr., Gordon L. Smith, and David D. Cheng, "Timing Analysis of Computer Hardware," *IBM J. Res. Develop.* **26,** No. 1, 100–105 (1982).
14. R. B. Hitchcock, "Timing Verification and the Timing Analysis Program," *Proceedings of the 19th Design Automation Conference,* 1982, pp. 594–604.
15. W. R. Heller, W. F. Mikhail, and W. E. Donath, "Prediction of Wiring Space Requirements for LSI," *Proceedings of the 14th Design Automation Conference,* 1977, pp. 39–42.

Received January 11, 1984; revised May 3, 1984

Rita R. Alvarodiaz *IBM Federal Systems Division, 9500 Godwin Drive, Manassas, Virginia 22110.* Ms. Alvarodiaz is manager of the VLSI design automation project. She joined IBM in 1968 at the East Fishkill, New York, Components Division laboratory, where she worked on design automation software. In 1969 she transferred to the Manassas, Virginia, Components Division, where she designed a system for producing dynamic and static FET chips and subsequently was appointed manager. In 1980 she joined the VLSI systems group in her present capacity. Ms. Alvarodiaz received a B.A. in mathematics from Southern Connecticut State College, New Haven, Connecticut.

Walter H. Elder *IBM Federal Systems Division, 9500 Godwin Drive, Manassas, Virginia 22110.* Mr. Elder is manager of the VLSI Design Automation Development Department. He joined IBM in 1967 at the Gaithersburg, Maryland, Federal Systems Division laboratory, where he worked on signal processing projects and design of a FFT hardware attachment for the IBM System/360. In 1970 he transferred to the Manassas, Virginia, Components Division, where he contributed to the development of design system approaches for producing dynamic and static FET chips. In 1978 he transferred to the Federal Systems Division at Manassas, joining the VLSI systems group, where he developed the MVISA channel routing programs and performance analysis approach. He received an IBM Outstanding Technical Achievement Award in 1983 for his design of the Early Timing Estimator. Mr. Elder received a B.S. and M.S. in electrical engineering from the University of Maryland, College Park, in 1963 and 1967, respectively. He is a member of Eta Kappa Nu, Phi Kappa Phi, and Tau Beta Pi.

Peter P. Zenewicz *IBM Federal Systems Division, 9500 Godwin Drive, Manassas, Virginia 22110.* Mr. Zenewicz is an advisory programmer in the VLSI Design Automation Development Department. During his seventeen years with IBM, he has worked in all areas of computer-aided design, including placement, wiring, and checking. He began his career at the Federal Systems Division, Owego, New York, developing programs for the SPRINT and OASIS design automation systems. In 1979 he transferred to Manassas, Virginia, to continue his design automation development work in creating the MVISA system. Mr. Zenewicz received a B.A. in mathematics from Gannon University, Erie, Pennsylvania.