Fast Structured Design of VLSI Circuits[1]

J. Gu and K. F. Smith

UUCS-TR-88-024

VLSI Group
Department of Computer Science
University of Utah
Salt Lake City, UT 84112
gu@cs.utah.edu

Abstract

We believe that a structured, user-friendly, cost-effective tool for rapid implementation of VLSI circuits which encourages students to participate directly in research projects are the key components in digital integrated circuit (IC) education. In this paper, we introduce our VLSI education activities, with the emphasis on the presentation of Path Programmable Logic (PPL) design methodology, in addition to a short description of a representative student project. Students using PPL are able to implement MOS or GaAs VLSI circuits with several thousands to over 100,000 transistors in a few weeks. They have designed and built numerous VLSI architectures and computer systems which play an influential role in various research areas. Our educational activities and the Utah Annual Student VLSI Design Contest supported by over a dozen leading American firms have attracted multiple university involvement in recent years.

Keywords: Integrated Circuit (IC), IC education, Path Programmable Logic (PPL), teaching experience, Very Large Scale Integration (VLSI).

[1]This work has been supported in part by DARPA under Contract No. DAAC-11-84-K-0017, in part by ACM/IEEE 1987-1988 Design Automation Award, and is currently supported in part by ACM/IEEE 1988-1989 Design Automation Award and a University of Utah Research Fellowship.

1 Introduction

Digital IC education is of widespread and growing interest throughout academic institutions, governmental agencies, and industrial organizations. Industry is concerned about digital IC education since economic advantages accrue to organizations that can consistently produce well-engineered hardware artifacts on time and within budget. Governmental agencies are interested in the role of digital IC education in maintaining a technological infrastructure that will provide for national security and ensure a competitive position in the international marketplace. Academia is motivated by a desire to serve the needs of modern society and by the intellectual challenges inherent in this emerging discipline.

Since research and development of VLSI architecture and computer hardware are labor intensive activities, better education and training for students are essential elements of better quality and increased productivity. Digital IC education must satisfy the current demand in digital VLSI engineering which are: (1) Better and higher quality hardware; (2) Flexible designs to permit easy modification and unforeseen usage with other hardware; (3) Shorter and more predictable development times; (4) Better prepared professionals, and (5) More designers.

During the last decade, there has been a steady increase in educational activities related to digital IC education. Our purpose here is not to discuss the general issues of these events. Rather, we will limit discussion of the following three questions that help drive the major theses of this paper.

1. In digital IC education, how can we train students to build real architectures in a relatively short period, say a few weeks within a single quarter?

2. How can we extend digital IC education activities to various research areas? e.g., is it possible to have students directly involved in research projects during a VLSI project class?

3. How can we allow large numbers of students to have hands-on design experience using very limited facilities?

In this paper, we present a user-friendly, cost-effective, digital IC design tool called Path Programmable Logic (PPL) [18,23,24]. PPL lends itself to rapid VLSI circuit and system design. Using PPL tools, students are able to implement MOS and GaAs IC circuits with 10,000 to over 100,000 transistors in a few weeks. Students have successfully designed, had fabricated and tested numerous Application Specific Integrated Circuits (ASICs), Reduced Instruction Set Computers (RISCs), microprocessors and multiprocessors.

Student involvement in advanced VLSI research projects has strongly influenced and contributed to many research programs at the University of Utah under NSF, DARPA, and NASA grants and contracts. The appendix gives a partial listing of VLSI research projects which cover a wide range of research areas, such as digital signal and digital image processing, computer and multiprocessor networks, digital communication and coding systems, computer arithmetic, fast database operation, computer resource management, artificial intelligence, computer graphics and computer vision, parallel and distributed computer simulation, and many other applications.

In Section 2, we introduce the PPL design methodology. An example of PPL design is given. We are interested in the psychological and technical background of students using PPL for digital IC education. In Section 3, we describe our IC curricula with the emphasis on the organization of the student VLSI projects. To illustrate the "real world" experience students obtained during their

project class, in Section 4, we present an example of a small student VLSI project designed using PPL tools. Finally, in Section 5, we discuss the impact of our VLSI course using PPL as a tool in digital IC education.

2 The PPL Design Methodology

We have long realized the fact that to facilitate digital IC design, a tool which supports different levels of design procedures, ranging from composite layout, through schematic and logic design, to symbolic manipulation is necessary; to be useful for rapid VLSI design, we must have such a tool which completely operates on a symbolic layout level and avoids the lower level details.

Research by the VLSI Group at University of Utah during the last several years has resulted in a symbolic, structured, and process-independent IC design methodology, known as PPL. PPL has been used as an educational tool in many VLSI design courses. PPL is user-friendly and cost-effective and it lends itself to rapid digital IC design.

2.1 The Principles of PPL Methodology

PPL is a cellular, digital IC design technique similar in structure to a PLA (Programmable Logic Array). In PPL, however, the AND and OR planes of the PLA are merged into a single plane. The AND condition of the input signals are formed on the rows of the PPL and the OR conditions are formed on the columns. Cells in addition to those needed to perform the sum of products realization of a function are also provided and may be inserted into the grid at *arbitrary locations*. These include for example flip-flops, inverters, loads, row and column connections, and pass transistors. Consequently both combinational and sequential circuits can be easily designed.

Using this methodology, design of a circuit is performed by placing cells which can be represented by logical symbols on a grid representing the integrated circuit (see Figure 4). When the grid is completely populated, it is both the logic representation and topological layout of the circuit. The cells have predefined schematic and composite representations. They are custom designed to optimize performance and size for the chosen integrated circuit process.

2.2 An Example

The design of PPL circuits is best understood by examing a simple binary counter (or divide by 4) circuit. The counter states transitions are given in Figure 1.

This counter circuit can be implemented using PLA design, as shown in Figure 2. This circuit consists of three structures: the AND plane, the OR plane, and the state variable memory (memory plane). The inputs to the AND plane enter on column wires and the outputs exit on row wires. The inputs to the OR plane enter on row wires and the outputs exit on column wires. It is assumed that the column signals will be true when low, i.e., $1 = low$. Each of these three parts is compact in itself, but they cannot be merged into a single, coherent structure.

If we now fold the AND and OR planes onto one plane, a better structure which is illustrated in Figure 3 can be obtained. Note that the output columns have been interspersed with the input columns. The folded planes can now be viewed as a single grid, with each location containing either

Figure 1: The Binary Counter Sequence

Figure 2: The PLA Implementation of a Binary Counter

AND elements, *OR* elements, or memory elements as shown by the shaded boxes. This concept was extended to allow arbitrary circuits elements to be placed in any grid location of the folded plane.

Bearing this structural difference in mind, let's now look at how PPL design differs from conventional circuit design. In a conventional IC circuit design, we generally first make a logic design by drawing a circuit schematic, then simulate the design (this processes usually repeats itself many times to refine the original design), finally we layout the composite patterns on a graphical VLSI work station.

In PPL design, we do not have to experience the tedious procedures in conventional custom design. Instead, the counter sequence in Figure 1 can immediately be mapped into PPL symbolic layout as shown in Figure 4. Note that we have combined rows $R2$ and $R4$ into a single row because $F2$ is always set whenever $F2$ and *Enable* are both 1, i.e., $F2$ toggles regradless of the state of $F1$.

Figure 3: The PPL Design of a Binary Counter

Note that in PPL design, two distinct operations are performed on the layout of Figure 3. First, a symbol is assigned to each shaded box. Second, we made all of the shaded boxes an integral multiple of a unit cell. In this case, the **1**, **0**, **R**, and **S** cells are unit cells. The flip-flop cell **F** is composed of 2 × 3 or 6 total unit cells. This flip-flop would now be considered to be a "macro" cell.

In this binary counter example, six PPL cells, i.e., **1**, **0**, **S**, **R**, **I**, and **F**, are used. For simplicity, we only show the PPL symbol. The PPL symbol, the NMOS schematic and the NMOS composite layout of two representative PPL cells, i.e., the PPL cell "**1**" and cell "**R**", are shown in Figure 5. The logic synthesis including the PPL layout for this binary counter is governed by some simple **if-then** rules which can be derived directly from the counter sequence in Figure 1. For example, to place PPL cells on the third row (i.e., R3) in Figure 4) of the PPL layout plane, the two simple If-Then rules which are shown in Figure 6 are applied. The counter circuit schematic (using NMOS for this example) described by PPL symbolic layout is illustrated in Figure 7.

2.3 The Psychological and Technical View of PPL

PPL cells are vital to the performance of the VLSI circuits and systems designed using the PPL methodology. At the lowest composite layout level, they must meet circuit schematic design, various electrical specifications, and multiple topological connection requirements. At the higher symbolic logic design level, the cell partition scheme for PPL symbolic cell representation must support a general, rule-based, hierarchical, and process-independent IC design methodology. To keep our attention from implementation details, the following discussion concentrates on the impacts of using PPL for digital IC education. For details of NMOS, CMOS and GaAs PPL cell implementations see [9,10,11,8,21].

Among many unique features of PPL IC design techniques, the following five aspects are of

Figure 4: PPL Symbolic Layout for a Binary Counter

Figure 5: Symbol, Schematic, and Composite Layout of Two PPL Cells, **1** and **S**

Rule 1: *If* $Q_1 = \mathbf{0}$ and $Q_2 = 1$, *Then* Cell **S** sets Q_1 to *1*.
Rule 2: *If* $Q_2 = \mathbf{1}$ and $Q_1 = 0$, *Then* Cell **R** reset Q_2 to *0*.

Figure 6: PPL Design Rules for Binary Counter (from State 01 to State 10)

Figure 7: Binary Counter Circuit Schematic (NMOS)

significant educational importance. We will now take a close overview of their psychological and technical background.

1. LOGIC SYNTHESIS USING PPL.

As shown in the Binary Counter example given in Section 2.2, circuit synthesis including layout using PPL can be done, in most cases, based on a straightforward and *natural understanding* of the given circuit functional specifications. From a psychological point of view, the concepts of PPL are easily understood by students.

Referring to the above counter design, from the counter sequence in Figure 1, a set of PPL design rules for the Binary Counter can be derived. Two of these PPL rules are shown in Figure 6 which represent the counter state transition from counter state $Q_1\ Q_2 = 01$ to the next counter state $Q_1\ Q_2 = 10$. The PPL symbolic VLSI counter circuit layout can be directly implemented in terms of these rules, as shown in the second row in Figure 4.

It is interesting to note that circuit logic synthesis using PPL is a *functionally conceptualized* mapping rather than a *schematically conceptualized* design. For example, cell **0**, or cell **1**, senses column logic states and invokes a new row logic state, which forms an AND term in the PPL circuit and presents a PPL rule in an **If** statement (like a *noun* which represents an object in a rule statement). Similarly, cell **S** (i.e., *set*), or cell **R** (i.e., *reset*), sets a column logic state according to the current row logic state, which produces an OR term in a PPL circuit and generates a rule in a **Then** statement (like a *verb* which represents an action in a rule statement).

2. UNIVERSAL AND FLEXIBLE IC BUILDING BLOCKS.

The philosophy of a scheme for partitioning cells in PPL [11] can be summarized as: (1) partitioning IC circuit cells into the *simplest and smallest blocks* so that any general IC circuit and system can be built hierarchically; (2) maintaining a set of necessary functional macro cells so that the redundant efforts for building those frequently used circuits can be avoided. The simple PPL

cells can be as small as a single transistor or one metal layer connection. The functional macros in PPL may be as large as a flip-flop or a full adder. Indeed, any circuit or system block could be used in this scheme as a PPL macro, e.g., a RISC system could be a macro in the PPL cell data base.

Based on the generality and flexibility of the PPL design methodology, students are capable of designing and implementing any digital IC with a high degree of freedom. They can use simple PPL cells to build any desired circuit module which forms part of a larger architecture. This primitive and easily acceptable way of learning gives students full freedom to extend their capability and creativity in digital circuit and system design.

3. ONE STEP, MULTIPLE LEVEL CIRCUIT DESIGN.

PPL has the unique advantage among CAD tools of giving a *simultaneous high-level view* of the symbolic, the logic, the schematic, the composite layout and the interconnect of a VLSI circuit in a single step. This provides not only the technical background to train students who become familiar with all levels of the IC circuit design, but also frees them from many time-consuming, labor-intensive tasks when constructing a real IC architecture. For example, they do not have to be stuck on the low level details of composite layout, the extraction of circuit schematics, and doing design rule checks (DRC), etc.

4. PROCESS INDEPENDENCE.

Students have been trained in digital IC design based on several different IC processes (e.g., NMOS, CMOS, and GaAs). It is our intention to let students do the design based on the functional requirements and then select the technology to fit the applications.

A clever feature of PPL design is its *process*, or *technology independence*. For example, by replacing generic PPL cells in the technology database, a PPL chip designed using an NMOS technology can be easily transferred into chips which will be fabricated using CMOS and/or GaAs technologies, or vise versa. The higher level PPL symbolic representation for NMOS, CMOS, and GaAs technologies could remain the same. The original study of PPL technology independence in circuit design and an earlier experiment can be found in [7,6] which were reported by Gu and Smith in [11]. Jacobs also discussed PPL technology independence and many details in [13]. Further developments in this direction are described in [6,22,12].

5. USER-FRIENDLY INTERACTION.

User-friendly interaction of the PPL system is a direct consequence of the above four features. As discussed in Section 2.4, Section 5, and in [11], an order or more of magnitude reduction in design time (as compared to other semi-custom IC design techniques) for integrated circuits can be achieved using the PPL design methodology.

The psychological and technical background overview and our experience (discussed below) indicate that PPL is a friendly IC design tool for students for rapid prototyping of digital VLSI circuits. Meanwhile it also promises dense circuitry. References [11,2] give some examples of different circuits built in industrial companies using PPL (and/or SLA) and compares full-custom design to PPL design for various performance and sizing figures.

2.4 PPL Design Tools

The PPL design system, or PPL design tools, is a concrete embodiment of PPL IC design methodology. There are primarily two software packages in the PPL system, a structured logic editor *Tiler* [15] and a unit delay simulator *Simppl* [14], which are used alternatively in the design process.

The PPL tools have the following features:

- All user interaction is achieved through a conventional CRT terminal connected to a mainframe or personal computer. This eliminates the bottleneck created when costly graphics systems are required or a few special VLSI design work stations are available.
- The circuit is represented as an array of the PPL symbols. In this representation, the locations within the array correspond to actual physical locations in the fabricated ICs. Thus, the editor gives the user simultaneous perception of both the circuit's logic function and its topological appearance.
- *Tiler* user interface is similar to a high-level, screen-oriented text editor such as *Emacs*, or *Jove*. *Tiler* allows cursor movement and scrolling in any of four directions and supports many helpful editing commands, including replication of, deletion of, and relocation of a single PPL cell or a large collection of PPL cells. *Tiler* gives the user a full screen view into the complete PPL design, allowing easy movement through the entire IC circuit design.
- The use of symbols within the editor makes it possible to do many high-level operations, involving symbol manipulation and pattern recognition, that are not feasible in conventional IC design systems. These operations include the enforcement of topologically and electrically required placement restrictions (illegal adjacencies of cells, and illegal placement locations – stipulated by shared wires and cell mirroring), and the automatic placement of connections between physically adjacent cells.

Tiler is a special interactive editor used for designing VLSI circuits using the PPL methodology. PPL tiles are inserted into a rectangular grid by typing characters that represent the tiles. When *Tiler* is first started, the user is prompted for the technology to be used for designing a PPL circuit. When a technology name is given by the user, a technology file is loaded into *Tiler*.

Simppl is a switch level simulator designed for simulating circuits built using the PPL methodology. In addition to using a switch-level model, *Simppl* makes use of multiple logic-value (six for MOS) to more accurately model the operation of soft and hard nodes, bussing, wired logic, and dynamic logic. There are both soft and hard values for the true and false logic levels. Since the complete circuit schematics have been automatically defined when PPL symbols were populated on the entire plane, simulation by *Simppl* is straightforward without requiring the extraction of circuit schematics for the simulator.

3 VLSI Course at University of Utah

We now take a look at two major areas of concern in digital IC education, i.e., how to teach IC design, and how to conduct a VLSI project course. The VLSI course taught at the University of

Utah runs for three consecutive quarters. In Section 3.1, we introduce our IC background education during the first quarter and the third quarter. Most materials taught were practiced by students are based on a variety of CAD tools which are available at other universities and industry companies. In Section 3.2, we describe students' VLSI projects which are developed based mainly on PPL methodology. All projects were conducted and accomplished during the second quarter. The last section shows some of the projects submitted to the Utah Annual Student VLSI Design Contest.

3.1 IC Background Education

During the first quarter of the series, students are taught the fundamentals of transistor operations, transistor sizing for DC operation, stick diagram representations, MOS (including NMOS and CMOS) processes details, MOS design rules, composite layout, and transient analysis techniques including capacitance and resistance calculations. In this quarter, each student is required to produce a composite of a fully custom design which contains approximately 300 $\sim$ 800 transistors, using either UC Berkeley VLSI design tools [20,17], or VTI (VLSI Technology Inc.) design tools [25]. This is a circuit of the student's own choosing. We required that students make detailed logic design, block diagrams, schematics with transistor sizes, stick diagrams, composite layout, design rule checks (DRC), and timing and DC analyses. Final check out includes successful on-line circuit simulation and a project report. The circuits designed during this first quarter are normally sent for fabrication to MOSIS (DARPA sponsored IC fabrication facility), so that students can test them during the third quarter.

Student projects in the second quarter are discussed in Section 3.2 below.

The third quarter is designed for students who are seriously considering a career in which the integrated circuit plays a major role in VLSI design, etc. Topics covered range from device modelling, second order modelling effects, and various kinds of simulation techniques, to real IC chip testing, and GaAs IC circuit design. In the first quarter and the third quarter, a number of text books [16,19,26] and a large number of handouts, are used.

3.2 VLSI Project Course

The second quarter is a project-oriented course. The purpose of this project is to give students some "real world" experience with the design and implementation of VLSI systems and computer architectures. A typical instance of a VLSI project during this single quarter involves the following three stages of activities.

PHASE 1: PROJECT IDENTIFICATION.

During this period, students are taught the basic knowledge of PPL design methodology and PPL design tools. They are first required to form a number of IC design groups each containing 1 $\sim$ 3 people for small projects. For larger projects, up to 5 people are allowed to form a group. Students in each group submit a research proposal for some intended project topic. Since most students taking this class are doing graduate research programs or come from industrial companies, their project topics are generally chosen from their thesis research topic or from industrial projects. For a few students who lack ideas, a list of options is provided. Each project proposal is critically judged in terms of its technical significance and engineering feasibility. During the project identification

period, there are overlapped interactions among students, instructor and TAs, industry experts, and student advisors.

PHASE 2: RESEARCH AND DEVELOPMENT.

This is an intensive R & D period during which students design and implement their proposed VLSI project. After learning the PPL design methodology and the PPL design tools, students in a project group usually work on their own portion of circuits divided within a project group. Each student maintains close interactions with the project director (i.e., instructor, TAs, or student advisors). Both regular and drop-in meetings with project directors are encouraged. A regular project review session is arranged every two weeks, so that students of each individual project group can present their current project progress to all directors. Depending on the project situations, we usually assign students some relevant papers to read, set up some tutorial discussion for some groups, and monitor regularly the student project progress.

PHASE 3: PROJECT EVALUATION.

At the end of the second quarter, each group project is evaluated in terms of three criteria: Check-off of circuit simulation and layout, group performance during the final oral project presentation, and final written report. The final oral project presentation and final report are examined in terms of 5 categories with a total number of over 30 criteria, each marked with the scale from 1 (poor) to 10 (excellent). An example of the student projects and a partial listing of student VLSI projects designed during the past several years can be found in Section 4 and the Appendix.

3.3 Utah Annual Student VLSI Design Contest

To extend our VLSI design activity for both education and research purposes, to stimulate student's interest and enthusiasm in VLSI design, and to attract public attention in VLSI education, we have organized an annual *Utah Annual Student VLSI Design Contest* since last several years. This is an ongoing contest and has been supported by AT&T, Evans and Sutherland, Inc., Hewlett-Packard, Unisys, National Semiconductor, VLSI Technology Inc., ECAD, Ford Scientific Lab. and others with a total of over a dozen leading American firms who are interested in VLSI engineering. There are usually 10 to 15 awards that are granted to student project groups every year. Typical awards include HP personal computers and calculators, digital IC equipment, cash ($5,000, $3,000, $2,000, $1,000, etc.), and tuitions. Each award is accompanied with an award certification. Students (not limited to the VLSI course) who are involved in VLSI research are eligible to submit their projects in a well-documented technical report. Experts from industry carefully judge and rank each award recipient. Usually there have been many people, including governmental and university VIPs, industry experts, faculty members and students, joining in this event. The original contest was held at the University of Utah. In recent years, Utah State University, Brigham Young University, and University of Michigan at Ann Arbor have participated in the competition.

4 An Example of the Student Projects

A small fraction of representative student VLSI projects which have been accomplished during the course, research projects, and thesis programs in recent years are listed in Appendix. The project scale ranges from several thousand transistors to over 100,000 transistors. According to

the statistics collected over the past several years, the average design time to complete the project is a total of about 200 person hours. About 20 to 40 hours are spent on learning PPL design methodology and getting familiar with the PPL design tools. Half of the remaining time (i.e., about 80 to 90 hours) is used to study, define, and design the project to be implemented. The rest of the time (i.e., another 80 to 90 hours) is for PPL implementation of the VLSI circuits. Most of the circuits which were fabricated using NMOS, CMOS, or GaAs[2] technologies were later tested and found to be functionally correct. A large number of the projects were reported in previous collections of the Utah Annual Student VLSI Design Contest, University of Utah Computer Science Department technical reports, and various publications.

In order to illustrate the student experience obtained during the VLSI project course, we have included below a brief description of a typical student project, i.e., a Virtual Shuffle Dynamic RAM Controller (VSDRC) circuit design. The complete system design, learning PPL design method and tools, and the VLSI chip layout (containing about 10,000 transistors in a 3 μ NMOS technology) was accomplished by Wilhelm Burger, a first year graduate student, in approximately 220 hours within a single quarter. In the following, a brief summary from his submitted project report is given, for details see [3][3].

4.1 General Description

A Virtual Shuffle Dynamic RAM Controller (VSDRC) is a microcomputer peripheral device. VSDRC performs two fundamental operations, i.e., *bit-reversal* and *perfect shuffle*, as shown in Figures 8 and 9. VSDRC can provide necessary signals to address, refresh, and drive 256K dynamic RAMs. Additionally, VSDRC can also perform a variety of address permutations under program control, which make it a useful component in numerous signal and image processing applications (e.g., FFT, matrix transposition, computing Kronecker product, and multidimensional FFT).

Concept and Ideas

The basic idea of the *virtual shuffle* is simple: since the address translation rules for the required class of memory permutations can be expressed as the *rearrangement of address bits*, it is feasible to obtain the same result by merely *rearranging address lines* between the CPU and the memory *without actually moving* any elements in memory. The only requirement is a mechanism in the address path to the memory (may be part of the entire memory) that is capable of *rewiring* address lines *under program control* in such a way that, once the address lines are rewired, the *CPU sees a shuffled memory.* In other words, the CPU finds each memory element in its "shuffled location," although in reality each element stays in its original memory cell (see Figure 10).

System Level View

If the *address switching mechanism* is made part of this memory itself, a *Virtual Shuffle Memory Unit (VSMU)* is created. Such a VSMU can be used in a board-level system (Figure 11). To interface the VSDRC to the bus system, only the minimum of external hardware (a clock generator,

[2]The GaAs chips were fabricated at Vitesse Semiconductor Corp., Rockwell International, and TriQuint Semiconductor.

[3]Courtesy of Wilhelm Burger.

decoder, etc.) is required, as shown Figure 12.

4.2 Functional Description

Internal Structure

The VSDRC consists of the following functional blocks (see Figure 13).

- *Crossbar Switch*: This 18 x 18 array of switches routes input to output addresses to perform the virtual shuffle by address manipulation. Each switch element has its own single bit memory cell in place, whose content determines whether the switch is open or closed. The Crossbar Switch is programmed by setting or resetting these memory cells.
- *Register Files A and B*: Register File A holds the current routing configuration of the Crossbar Switch, e.g., it stores for each input line (one of 18 crossbar rows) the corresponding output address line (one of 18 crossbar columns). Whenever the Crossbar Switch is *reprogrammed*, the previous configuration is read from Register File A, modified and stored simultaneously in the Crossbar Switch and in Register File B. Subsequently the new configuration in Register File B is copied over to Register File A before another programming sequence can be started.
- *Permutation Controller* It generates the addresses and control signals required to modify the routing configuration of the Crossbar Switch, by using Register File A as the source for the original configuration and storing the new configuration in Register File B. There are two registers in Permutation Controller for N (range of operation) and K (a Perfect shuffle parameter), which are loaded under program control.
- *Program Controller*: Coordinates internal programming.
- *Refresh Timer* Generates periodic refresh requests at a rate of approximately 128 every 2 milliseconds. These internal refresh requests have priority over memory requests from the CPU, refresh requests generated during an active CPU memory access period are kept "pending" and executed as soon as the CPU cycle has terminated.
- *Refresh Address Counter*: Supplies 9 bit row addresses for the periodic memory refresh operations.
- *Cycle Controller*: Generates the RAM control signals and drives the internal multiplexers and the asynchronous XACK-logic.

VSDRC Programming

Shuffle commands are issued to the VSDRC by writing to specific memory locations in a reserved address space (selected by the external address decoder) by polling input *PGM* low (active). The lower six address bits (A0, ..., A5) are interpreted as the program instruction as shown in Figures 14, 15, 16, and 17. Four different kinds of instructions are provided:

- *Soft Reset:* has the same effect as a *Hard Reset* (pulling both *PCS* and *PGM* low) with the exception that the refresh counter is NOT reset, thus providing undisturbed refresh

operation. The crossbar switch is set to its original configuration and all internal registers are set to default values. This mode of operation is intended primarily for *testing* purposes (see below), since the controller can be set to a "known" state under program control.

- *Load N:* the internal N-Register can be set to values in the range 1, ..., 17, setting the *range of operation* for the Bit Reversal and the Perfect Shuffle, i.e., how many bits of the input address will be affected by these operations.
- *Bit Reversal:* starts the internal programming sequence to reconfigure the address switch for the Bit Reversal. Only the lower N+1 bits of the address range are affected, internal routing for the higher address bits remains unchanged.
- *Perfect Shuffle:* loads the shuffle-parameter K into the corresponding register and starts the internal programming sequence for the Perfect Shuffle (K). Again only the lower N+1 bits of the address range are affected, internal routing for the higher address bits remains unchanged.

Detailed processor interface (read, write, and handshaking design), two phase clock generation, basic timing sequence, and dynamic RAM interface are contained in [3].

4.3 Implementation Issues

Functional Blocks and Floorplan

The VSDRC circuit was implemented in a 3.0 μ NMOS technology using PPL design tools. It fits in a 40-pin dual-in-line package, with three pins used for Vdd, Ground, and Substrate. Logic simulation on the complete circuit has been performed as well as an approximate timing analysis to ensure correct operation at the projected minimum clock frequency of 4.0 MHz.

Implementation Details

The complete PPL symbolic VLSI layout including the bounding pad edges of the circuit is shown in Figure 18. The internal structure described previously can be identified easily.

4.4 Testing

Since all pins of the 40-pin package are required for circuit operation and a larger package size would be an overkill, *no dedicated test pins* could be provided. Thus any information about the internal operation must be derived from the states of input and output pins. Three efficient *testing procedures* based on current built-in circuitry were designed and are shown below:

Simple Go/Nogo Test

Without any memory or programming request from outside, the VSDRC will perform periodic refresh operations on the external dynamic RAM. Thus a simple *initial Go/Nogo test* can be applied by resetting the circuit and observing the generated refresh address and memory drive signals without any stimulus from outside (except for the reset and clock signals). This provides at least partial cues about the proper operation of the following circuits components:

- Refresh Address Counter,
- Address Multiplexers,
- Refresh Timer, and
- Cycle Controller.

A logic analyzer or a test system with sufficient resolution in time (Textronix DAS) is required. This test will also give immediate information about the speed that can be achieved from the circuit.

Address Path Test

Again no external memory requests are applied to the VSDRC. When the VSDRC does not perform a memory refresh operation, the lower nine input address bits are fed through (inverted) the crossbar switch and the multiplexers. By generating suitable 1/0-patterns for the input Addresses, information about the following components can be obtained:

- Address Multiplexers,
- Lower half of the Crossbar Switch,
- "Reset" - sequence of the Program Controller,
- Register File A, and
- Internal write-address and read-address Busses.

Shuffle Operation Test

Since the VSDRC is designed as a *microcomputer peripheral device*, it lends itself to be tested in such an environment after the initial basic tests (above) have been completed. A simple testbed is shown in Figure 19. The CPU performs memory-read and programming operations on the VSDRC and reads back the output addresses generated by the circuit. This test will cover all those circuit components not affected by earlier tests, including

- Permutation Controller,
- Program Controller,
- Register Files A and B, and
- Crossbar Switch.

Reference [3] shows detailed implementations, such as clock distribution, timing balance between the gated pulses and the final non-inverted clock signals, tradeoffs for size and control complexity in the Crossbar Switch design. It also shows the implementation of a switching cell, optimization speed using PPL tools by alternating PPL layout, splitting Crossbar Switch into two part to avoid longer transmission lines, and so on.

5 The Impact of the VLSI Course and Projects

We believe that the VLSI course and student projects based on the use of PPL and other VLSI tools available from universities and industry make several contributions to the education of the students.

1. Demystifying digital VLSI design and enhancing student's confidence and independence in VLSI architecture design.

Students are always very concerned about designing a complete VLSI system with thousands or tens of thousands of transistors. This represents an overwhelming task for those students who have very little background in digital IC design. However, students have generally been successful after a few hours of instruction on PPL design methodology to complete this task, so long as they are provided with some basic design examples. They gradually build small pieces and assembly them into a large VLSI system. After the completion of the project, they all feel that it is not mysterious to design a complete, large VLSI system. It has been strongly indicated that students' confidence and independence in digital VLSI design has been greatly enhanced.

2. Students have been thoroughly experienced in different levels of VLSI design based on different IC technologies.

During the entire course of the VLSI project, students have been trained in major IC design phases. Students are allowed to build full-custom PPL cells by themselves in order to optimize the overall PPL layout, or directly use available PPL cell sets and go directly to high level symbolic architecture design. There are currently two NMOS technologies, four CMOS technologies, and two GaAs technologies for the PPL cell sets that are available [9,10,11,8]. By simply typing technology specifications at the beginning of the PPL editor, i.e., *Tiler*, students can experience IC circuit design based on different semiconductor technologies.

3. Fast prototyping of VLSI circuits is one of the key issues to increase student's creativity and project quality.

The PPL tools allow students to implement VLSI circuits with design times that are an order of magnitude faster than other design tools. A group of four students with good academic standing can implement a *given* VLSI design (with over 70,000 transistors) in less than a weeks time (about 30 to 50 person-hours). Therefore, during a single quarter, students in some groups design and implement several VLSI chips for different purposes, or several different VLSI architectures to perform the same task. This feature of fast prototyping of VLSI systems allows students to make their own assessment, comparison, and performance evaluation for different competing VLSI circuits and systems, which greatly raises the quality of our IC educational activities. On the contrary, when using conventional IC design tools, it may take many man-years of efforts to accomplish the VLSI chip layout of a complex IC circuit. In that situation, it is impossible for students to make a performance judgement for different architectures while being absorbed in tedious handcraft jobs.

4. Students realize many higher level system issues in VLSI system design.

Different from student projects based on separate digital components in which system functionality is highly limited and isolated, digital IC design at the VLSI level tends to be more dependent on its functional context and the technical environment where the hardware is designed. As described in the VSDRC project, student started to think about problems like: how to interface with the outside world, how to design the higher level control and micro programming strategies, what

are good ways for further IC testing after fabrication, and many other real world issues with clock distribution, and engineering tradeoffs. It is important that VLSI projects should be more than just another hardware project. The project is most effective as a technical tool when it makes students aware of these higher system level issues.

5. Students have been trained during the entire course of the research project including project management.

Each and every student is involved in a certain portion of the research project during the VLSI project course. They initiate their project and actively explore and identify research topics, write project proposals, prepare oral presentations, develop and implement the IC circuits and systems, solve practical technical problems, write technical reports, and cooperate with their co-workers. The VLSI project course has been a real world engineering practice and rehearsal for students.

6. The VLSI projects provide a solid background for many research projects.

According to the statistics collected during the last several years, many students have used PPL tools and methods they learned during the VLSI class in their research projects towards the completion of their MS and PhD programs and have made fundamental progress. For examples, many students have built RISC and advanced computer architectures in various research projects. Many new research results have been found during the development and implementation of their VLSI architectures [5,1,4]. Since student projects are involved in university or industry research (such as image processing, multiprocessor architecture, discrete event simulation, etc.), they have also learned many relevant advanced research topics.

7. Train students to meet the current demands of digital VLSI design.

We observed five current criteria in digital VLSI design and three questions in Section 1. We believe our curricula which are based on the PPL system and the other use of standard design tools can satisfy these challenges.

6 Conclusions

It is generally believed that the research frontier is moving more rapidly than the education frontier in the area of digital VLSI circuits and systems design. We believe, by using the PPL design methodology and a variety of other CAD tools and encouraging students to participate directly in research projects, not only the education frontier can move as fast as the research frontier, it also has a strong impact to push the research frontier forward.

7 Acknowledgements

To AT&T, ECAD, Inc., Evans and Sutherland, Inc., Ford Scientific Research Lab., Hewlett-Packard, Mentor Graphics, National Semiconductor, Seattle Silicon, Shell Oil Development, Unisys, VLSI Technology, Inc., and those for their financial support to Utah Annual Student VLSI Design Contest and awards. Without Wei Wang's support, this manuscript would have been impossible. We also thank Steve Jacobs for writing the major sections of the PPL tools and Wilhelm Burger for his VSDRC design.

Appendix

A Partial Listing of Recent Years VLSI Projects[4]

• Digital Signal Processing and Digital Communication:

A Self-Timed Double-Ended Queue (83), by K. Stevens. NMOS, PPL design.

Digital Pulse Width Mode Digital to Analog Converter (1983), by D. Reading. NMOS, PPL design, 36 pins.

Universal Digital Filter (1986), by C. Yang, W. Hsu, and W. Chung.

An Encoder for Vector Quantization (1986), by J. Bradley.

Huffman Decoder Chip (1986), by L. Copp and D. Frederiksen. 3.0 μ CMOS, PPL design, 4,940 transistors, 38 pins.

A Single Chip Viterbi Decoder Architecture (1986), by D. Morrell. 3.0 μ CMOS, PPL design.

Speed Tracking Morse Code Decoder (1986), by J. Barrus. 3.0 μ CMOS, PPL design, 6,418 transistors, 26 pins.

The SREC Error Detection and Correction Circuit (1986), by S. Folsom. CMOS, PPL design, 8,922 transistors.

Morse Code Generator Chip (1986), by B. Harmer. NMOS, PPL design, 6,524 transistors, 37 pins.

Gallium Arsenide Programmable Communications Interface Chip for High Speed Data Compression (1987), by B. Moss. GaAs, PPL design, 11,306 devices (7,657 FETs and 3,649 Diodes).

VLSI Implementation of A Maximum-Likelihood Decoder for the Golay (24, 12) Code (1987), by A. Abbaszadeh. 3.0 μ NMOS, PPL design, 40,000 transistors.

• Computer Arithmetic and Algorithms:

Design of A Clipper (1983), by L. Putnam.

Implementation of A Systolic Array for Polynomial GCD (1984), by J. Samantarai.

A Fast 32-Bit Parallel Adder Using a Carry Generate / Propagate, and Lookahead Design (1984), by C. Anderson.

A Fast, 8 x 8 Multiplier (1984), by S. Parandoosh.

A PPL Implementation of The Radix-16 Digit Slice (1985), by G. Carroll, M. Gaspar, and R. Neff. 4.0 μ NMOS, PPL design, 51 pins.

Floating Point Machine (1986), by M. Haycock.

Parallel Sorting Chip (1987), by C. Tsue, J. Tsai and J. Gu. 3.0 μ NMOS, PPL design, 28 pins.

Linear Interpolator Chip (1987), by L. Grayston.

A 3-D Linear Interpolator Design (1988), by S. Lee.

A Log N Time Complexity 24 Bit Fixed Point Multiplier (1988), L. Lai.

• Microprocessor and RISC computers:

[4]This partial listing are come from some of the Utah Annual Student VLSI Design Contest Reports and some currently ongoing research projects. Some data are drawn from available project reports.

8-bit AM2909: Microprogram Sequencer (1985), by A. Abbaszadeh and R. Naimimohasses. NMOS, PPL design, 40 pins.

A Four Bit Microprocessor Slice (1985), by P. Jackson and M. Whitaker.

PRISC: A PPL-Based Reduced Instruction Set Computer (1986), by R. Ginosar and A. Harsat. 4.0 μ NMOS, PPL design, 4 MHz.

A Simple CPU (1988), by J. Aalberg, A. Eatchel, J. Cook, and A. Walbeck. 2.0 μ CMOS, PPL design, 7,000 transistors, 24 pins.

- **Computer and Multiprocessor Networks:**

Inter-Process Element Transceiver (1986), by C. Wang and Y. Kan.

RediLink: An Architecture for the RediFlow (1986), by J. Slater and J. Nusbaum. 3.0 μ NMOS, PPL design, 84 pins.

Design and Implementation of Chip ATQ (1986), by L. Skedinger.

A VLSI Implementation of the Synchronized Packet Interconnection Network (1987), by N. Michell. 3.0 μ scalable CMOS, PPL design.

A Low Power CMOS Local Area Network Controller (1988), by J. Gimbel. 2.0 μ CMOS, VTI design, 7,152 transistors, 87 pins.

- **CPU and Computer Resource Management Units:**

The Soft Controller: A Self-Timed Microsequencer for Distributed Parallel Architectures (1984), by K. Stevens.

UART (1984), by D. Rumbellow.

Dynamic RAM Controller (1985), by N. Michell and C. Shrivastava. NMOS, PPL design.

A Virtual Shuffle Dynamic RAM Controller (1986), by W. Burger. 3.0 μ NMOS, PPL design, 9,520 transistors, 40 pins.

68010 Memory Management Unit (1986), by J. Crossland. NMOS, PPL design, 20,736 transistors, 56 pins.

Content Addressable Memory (1986), by D. Kumar, T. Kiang, and M. Hwang.

A High-Bit Density Content Addressable Memory Cell (1987), by D. Kumar.

- **Computer Security:**

Data Encryption and Decryption Device (1984), by L. Letham and S. Rueckert.

- **Architectures for Artificial Intelligence:**

DRA1 Chip Implementation (1986), by D. Ku. 3.0 μ NMOS, PPL design, 39,502 transistors, 64 pins.

Parallel DRA2 Architecture (1986), by W. Wang and J. Gu. 3.0 μ NMOS, PPL design, 6,382 transistors, 35 pins.

- **Fast Database Machines:**

Improved Character Machine Chip (1985), by J. Schimpf. CMOS, PPL design, 40 pins.

- **Computer Graphics:**

A VLSI Processor for Raster Graphics (1983), by D. Anderson and J. Kennedy.

A 2D Addressable Area Filling RAM (1983), by E. Brunvand.

Incremental Line Generator (1986), by L. Weber and C. Johnson. CMOS, PPL design, 8,038 transistors.

VLSI PIXBLT and Compositor Chips (1986), by H. Bennion, B. Turner, and C. Wilcox. NMOS, PPL design, two chips, 4,024 + 3,998 transistors.

• **Image Processing and Computer Vision:**

VIDEO Image Controller (1984), by M. Bradakis.

Real-time CMOS Image Processor (1986), by B. Hutchings. CMOS, PPL design, 9,440 transistors, 63 pins.

A Laser VIDEO Display Control Unit (1986), by C. Benham.

VLSI Object Identification Chip (1987), by J. Lin and C. Lin. 3.0 μ NMOS, PPL design, 44 pins.

VLSI for CAGD (1987), by N. Yu and J. Lee.

A Real-Time Image Digitization Controller (1988), by R. D. Taylor. 2.0 μ CMOS, PPL design, 8,000 transistors, 50 pins.

• **Digital Audio Processing :**

Digital Chip Set for RECOG Speaker Recognition (1983), by G. Sellani, R. Kent, and K. Yoon. NMOS, PPL design, 28 pins.

Programmable AUDIO Tone Generator Controller (1987), by T. Mack and J. Beutler.

Sound Dosimeter ASIC (1988), by B. Walker, B. C. Drew, G. C. Brower, G. Bage, and T. White. 2.0 μ CMOS, PPL design, 18,000 transistors,

• **Parallel and Distributed Computer Simulation Engine:**

A Written-Bit Array for Distributed Time Warp Simulation Engine (1987), by W. Kuo and J. Gu.

Translation Lookaside Buffer for the Rollback Chip (1988), by N. Mani and S. Mantha. 2.0 μ CMOS, PPL design, 20,664 transistors, 111 pins.

• **Computer I/O Devives:**

A Low Cost Digital Logic Analyzer Display Controller (1984), by D. Green. NMOS, PPL design, 28 pins.

A VLSI Controller for Interfacing A TekTronix 4025 Computer Display Terminal with A VerSatec 1200 Printer/Ploter (1986), by R. Omid.

Hardware Print Spooler Chip (1987), by B. Duncan.

• **Hardware for Design Automation:**

A Smart Parallel Processing Automatic Router (1987), by T. Briscoe and R. Pennington. 2.0 μ CMOS, VTI design, 21,000 transistors, 20 pins.

• **VLSI Testbeds:**

IC Testing System using A CMOS Controller Chip (1986), by E. Hatch and T. Zohner.

Continuity Testing System Controller Chip (1987), by K. Stout, S. Taft, and T. Tateyama. 3.0

μ NMOS, PPL design, 5,516 transistors, 40 pins.

A VLSI Tester Based up on a Single Custom Chip (1988), by J. Aldous, G. Bearnson, M. Shore, R. Baseri, and R. Bangerter. 2.0 μ CMOS, PPL design, 28 pins.

- **Bioengineering Equipment:**

Never Simulator Controller (1983), by D. Harris.

Dynamic Memory Support for The Ultrasonic "MiniScanner" (1986), R. Kutcipal and R. Subramaniam.

Electron Microscope to Computer Interface (1986), by M. Jones.

- **Nuclear Engineering:**

Geiger-Mueller Counter Chip (1987), by R. Durtchi and J. Cope. CMOS, VTI design, 41,879 transistors, 37 pins.

- **Military Applications:**

Programmable Satellite Antenna Controller (1986), by G. Borstadt, J. Wood, and D. Richmond.

Infrared Target Simulator (1986), by J. Smith and K. Barton.

- **Games, Entertainment, and Consumer Devices:**

BMX Starter GTE Controller (1984), by C. Toone.

A One-Touch Telephone System (1984), by D. Chung.

TVC-15 Darkroom Time Management Chip (1985), by J. Pennock and B. Baxter.

Bicycle Performance Monitor (1986), by G. Brown and J. Flanagan. 3.0 μ CMOS, PPL design, 9,590 transistors.

Recordable Snowmobile Tachometer (1986), by J. Smith and K. Smith. 3.0 μ NMOS, PPL design, 11,000 transistors, 35 pins.

Moisture Gauge – Data Collection Platform Interface (1986), by R. Self, M. Pejoumand, and M. Akhavan.

The TELCON Chip (1987), by B. Madsen. 3.0 μ NMOS, PPL design, 34,152 transistors, 61 pins.

Television Time Display Controller (1987), by R. Branson and R. Mecier.

Data Logger Chip to Expand the Home Computer into a Data Acquisition and Analysis System (1988), by L. R. Barnum.

- **Frequency Generator and Counters:**

A Universal Frequency Generator (1985), by G. Kenner.

Presettable Frequency Counter (1986), by E. Reihlen. NMOS, PPL design, 7,950 transistors, 40 pins.

Design of an 8-Digit High Speed CMOS Frequency Counter (1987), by M. Jensen.

A VLSI Programmable Frequency Counter (1987), by M. Durlam.

Figure 8: Example for a Bit Reversal Operation on a Memory of Size = 8

Figure 9: Example for a Perfect Shuffle Permutation on a Memory of Size = 8

Figure 10: The Concept of the Virtual Shuffle

Figure 11: Virtual Shuffle Memory Board Level System

Figure 12: Interfacing the VSDRC

Figure 13: VSDRC Block Diagram

Figure 14: Programming a Bit Reversal

Figure 15: Programming a Perfect Shuffle

Figure 16: VSDRC Instruction Table

Figure 17: Programming Sequence Timing

Figure 18: The VSDRC PPL Program of VLSI Composite Layout

Figure 19: Testing the VSDRC in a Microcomputer Environment

Biography of K. Smith and J. Gu

K. F. Smith has served as the acting chairman of the Department of Computer Science at the University of Utah and is presently a Professor of Computer Science and Professor of Electrical Engineering at the University of Utah. He received his PhD from the University of Utah in 1982. He had over 20 years of experience in the integrated circuit industry prior to coming to the University. His area of specialization is in the development of CAD tools for the design of structured logic integrated circuits. He is the author of over 41 technical articles and 16 patents. Kent and Jun have been involved in many MOS and GaAs VLSI architecture design, implementation, and fabrication projects and several MOS and GaAs PPL system development projects.

J. Gu (S'86) received the B. S. degree in electrical engineering from the University of Science and Technology of China (USTC), Hofei, P. R. China, in 1982. Since September 1985, he has been a graduate student of Computer Science at the University of Utah, Salt Lake City, where he is currently a Ph.D candidate and a research assitant professor.

He worked as a faculty member, from 1982 to 1983, at the Department of Electrical Engineering, USTC. He has been twice awarded the ASC Fellowships and twice awarded the University of Utah Research Fellowships. He was a recipient of the ACM/IEEE academic scholarship award and is presently a recipient of the special ACM/IEEE academic scholarship award. Since 1987 he has been a Principal Investigator on some NSF supported projects. His current research interests includes artificial intelligence and robot vision, computer architecture, fault-tolerant computing, intelligent silicon compilation, and MOS and GaAs PPL systems for ultra fast VLSI architectures.

Mr. Gu is a student member of the Association for Computing Machinery and the International Neural Network Society, and a member of Sigma Xi.

References

[1] A. D. Abbaszadeh and C. K. Rushforth. VLSI Implementation of a Maximum-Likelihood Decoder for the Golay (24,12) Code. *IEEE Journal on Selected Areas in Communications*, 6(3):558–565, Apr. 1988.

[2] Storage/logic arrays finally get practical. *Electronics*, 29–33, Jan. 1986.

[3] W. Burger. A Virtual Shuffle Dynamic RAM Controller. May 1986. The 4th Utah Annual Student VLSI Design Contests.

[4] R. M. Fujimoto, J. J. Tsai, and G. Copalakrishnan. Design and Performance of Special Purpose Hardware For Time Warp. In *The 15th Annual International Symposium on Computer Architecture*, pages 401–408, Jun. 1988.

[5] R. Ginosar and A. Harsat. PRISC: A PPL-Based Reduced Instruction Set Computer. In *Proceedings of 15th IEEE Isreal Conference*, Apr. 1987.

[6] J. Gu. Intelligent Silicon Compilation. March 12, 1985. Research Proposal.

[7] J. Gu. PEditor: A Rule-Based Prototype PPL Editor. 1985.

[8] J. Gu and K. F. Smith. *Design and Implementation of 1.0 μ GaAs PPL System for Ultra Fast VLSI Architectures.* Technical Report - Internal Circulation Only, Dept. of Computer Science, Univ. of Utah, Jul. 1987.

[9] J. Gu and K. F. Smith. *Design and Implementation of Optimal PPL Cell Set using 2.0 μ Metal Gate CMOS Process.* Technical Report UUCS-TR-85-117 - Internal Circulation Only, Dept. of Computer Science, Univ. of Utah, Jun. 1985.

[10] J. Gu and K. F. Smith. *Design and Implementation of Optimal PPL Cell Sets using 3.0 μ, 2.0 μ, and 3.0/1.2 μ Scalable CMOS Processes.* Technical Report - Internal Circulation Only, Dept. of Computer Science, Univ. of Utah, Sept. 1985.

[11] J. Gu and K. F. Smith. KD2: An Intelligent Circuit Module Generator. In *Proceedings of 1986 IEEE International Conference on Computer Design: VLSI in Computers*, pages 470–475, Oct. 1986.

[12] J. Gu and K. F. Smith. *Structured, Technology Independent VLSI Design.* Technical Report UUCS-TR-89-008, Dept. of Computer Science, Univ. of Utah, to appear 1989.

[13] S. R. Jacobs. Automatic Technology Translation of Very Large Scale Integrated Circuits. Sept. 1986. Thesis Proposal.

[14] S. R. Jacobs and K. F. Smith. *SIMPPL User's Guide.* Technical Report UUCS-TR-86-004, Rev.2, Dept. of Computer Science, Univ. of Utah, Nov. 1987.

[15] S. R. Jacobs and K. F. Smith. *TILER User's Guide.* Technical Report UUCS-TR-86-006, Rev.2, Dept. of Computer Science, Univ. of Utah, Nov. 1987.

[16] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, 1980.

[17] A. R. Newton, A. Sangiovanni-Vincentelli, and et al. *OCT Tools Distribution 1.0.* Electronics Research Laboratory, EECS Department, UC Berkeley, Mar. 1987.

[18] S. S. Patil and T. A. Welch. A Programmable Logic Approach for VLSI. *IEEE Trans. on Computers*, C-28(9):594–601, Sept. 1979.

[19] D. A. Pucknell and K. Eshraghian. *Basic VLSI Design: Systems and Circuits, 2ed.* Prentice-Hall, Englewood Cliffs, 1988.

[20] W. S. Scott, R. N. Mayo, G. Hamachi, and J. K. Ousterhout. *The 1986 VLSI Design Tools.* Computer Division, EECS Department, UC Berkeley, 1986.

[21] K. F. Smith. Implementation of SLA's in NMOS Technology. In *Proceedings of the VLSI 81 International Conference*, Aug. 1981.

[22] K. F. Smith, T. Carter, J. Gu, S. R. Jacobs, and R. Ginosar. *PPL: A Technology Independent VLSI Design Tool for NMOS, CMOS and GaAs.* Technical Report, Dept. of Computer Science, Univ. of Utah, 1989, to appear.

[23] K. F. Smith, T. M. Carter, and C. E. Hunt. Structured Logic Design of Integrated Circuits Using the Storage/Logic Array (SLA). *IEEE Trans. on Electron Devices*, Ed-29(4):765–776, Apr. 1982.

[24] K. F. Smith, B. E. Nelson, T. M. Carter, and A. B. Hayes. Computer-Aided Design of Integrated Circuits Using Path-Programmable Logic. In *Proceedings of 1983 IEEE Electro Professional Program Session Record*, IEEE Computer Society Press, April 1983.

[25] VTI. *VTI Design Tools.* VLSI Technology Inc., 1985.

[26] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design - A Systems Perspective.* Addison-Wesley, Reading, 1985.

State Diagram

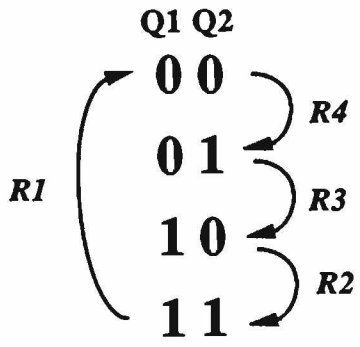

Fig. 1

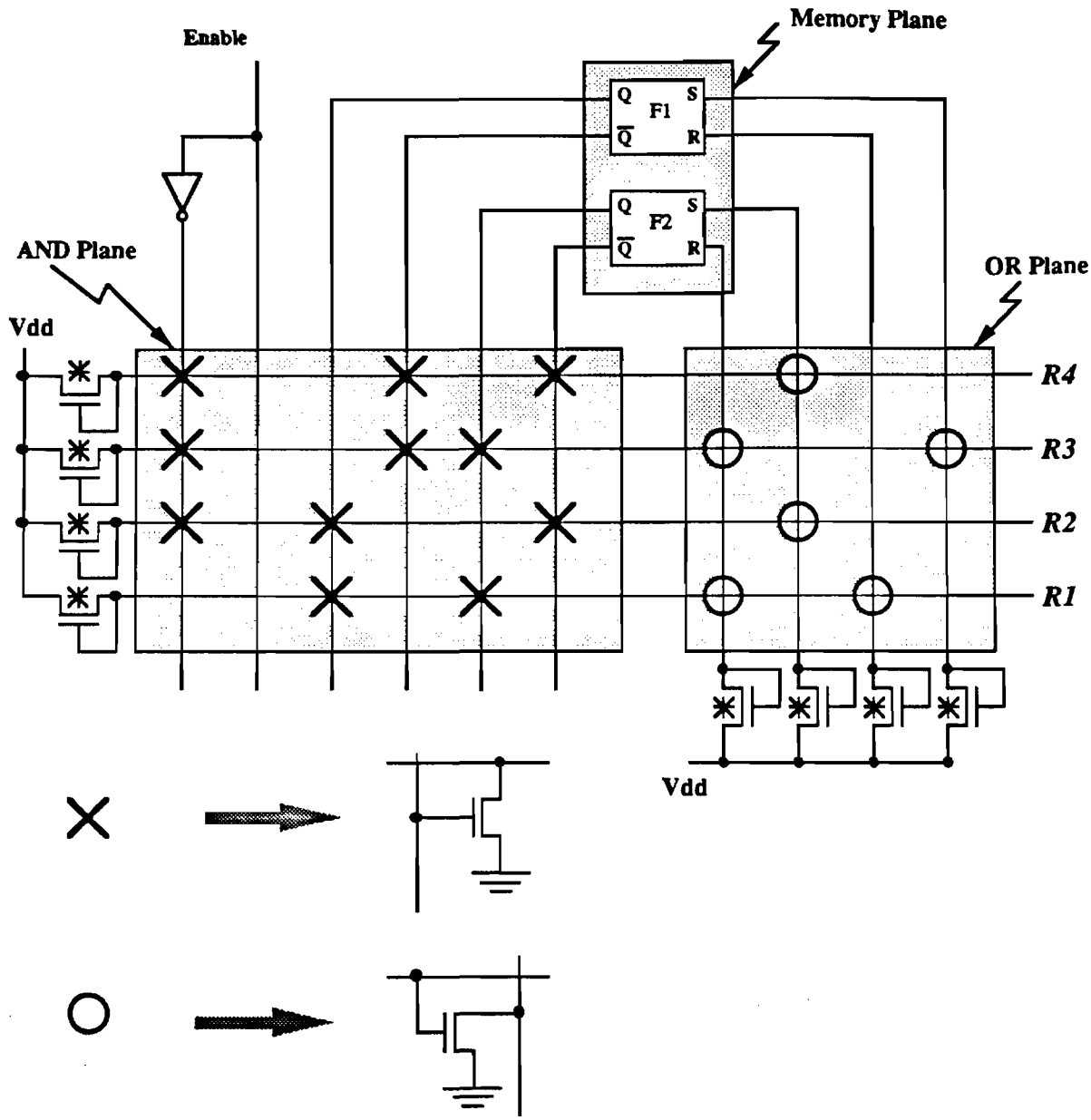
Enable
Memory Plane
Q
S
F1
Q
R
Q
S
F2
Q
R
AND Plane
OR Plane
Vdd
R4
R3
R2
R1
Vdd

Fig. 2

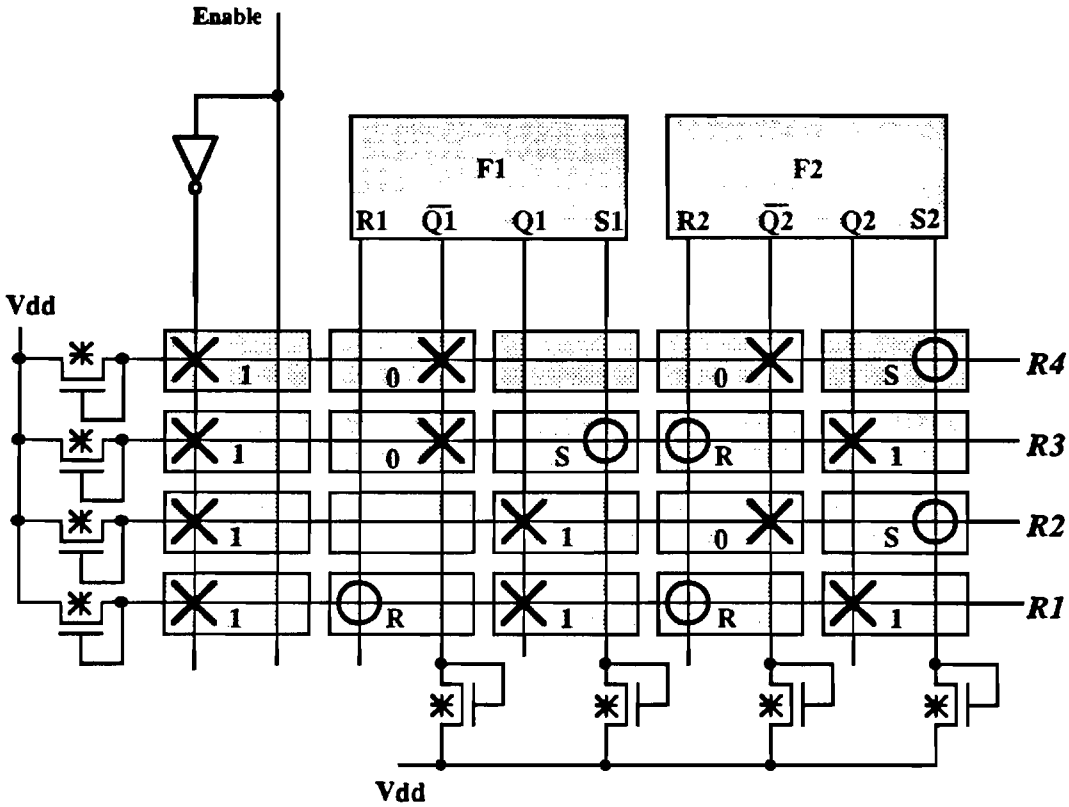

Fig. 3

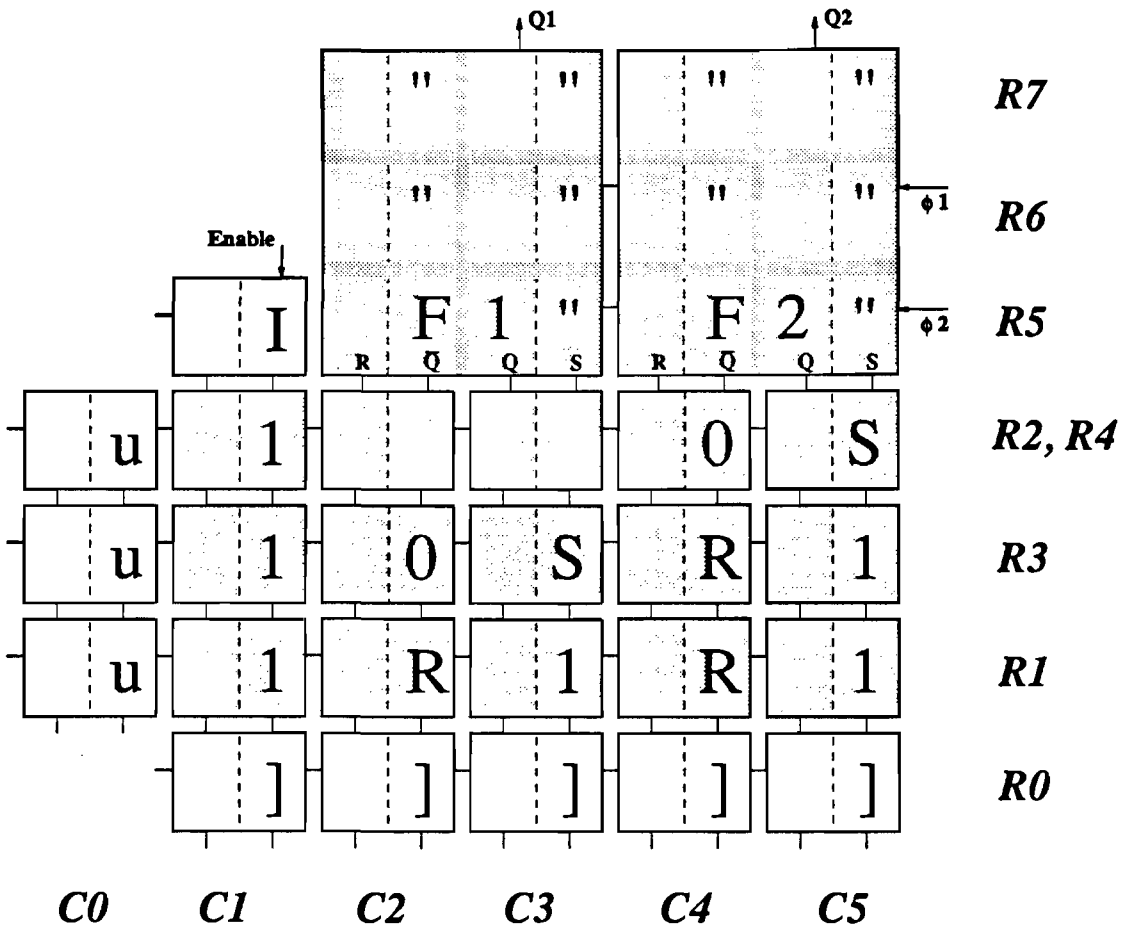
Q1
Q2
Enable
I
F 1
F 2
R
Q̄
Q
S
u
1
0
S
R
]
φ1
φ2
R7
R6
R5
R2, R4
R3
R1
R0
C0
C1
C2
C3
C4
C5

Fig. 4

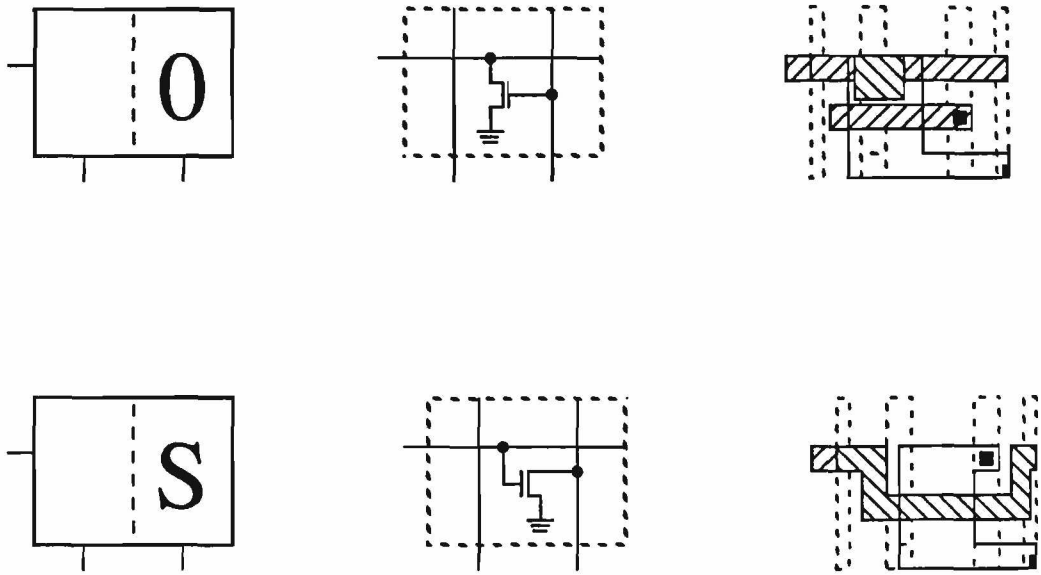

Fig. 5

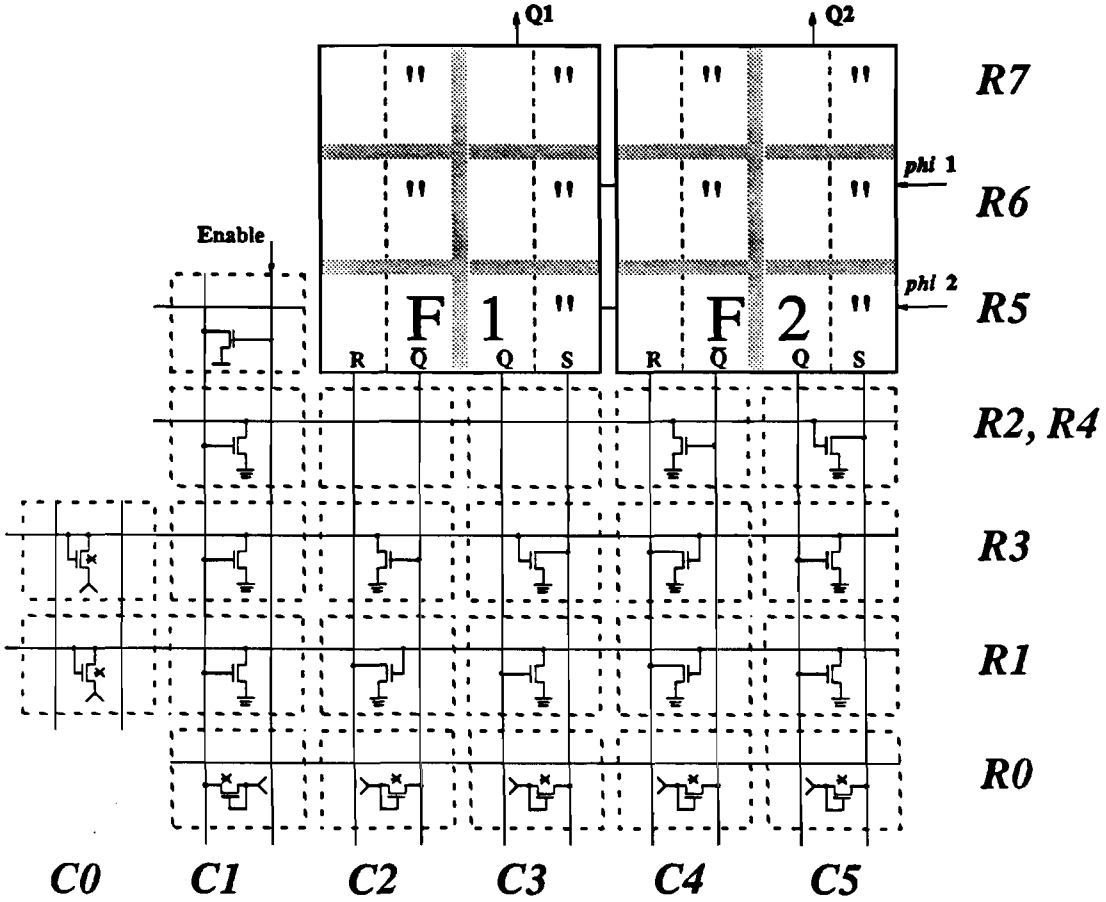

Fig. 7

(Fig. 6 in page 6)

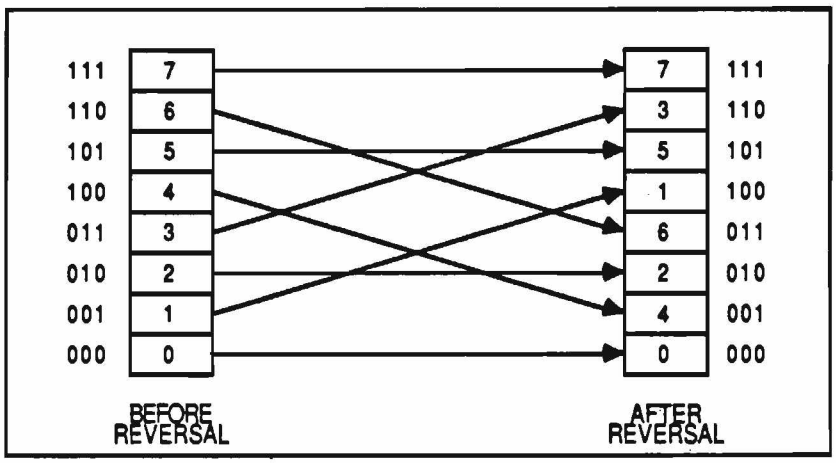
111
110
101
100
011
010
001
000
7
6
5
4
3
2
1
0
7
3
5
1
6
2
4
0
111
110
101
100
011
010
001
000
BEFORE REVERSAL
AFTER REVERSAL

Fig. 8

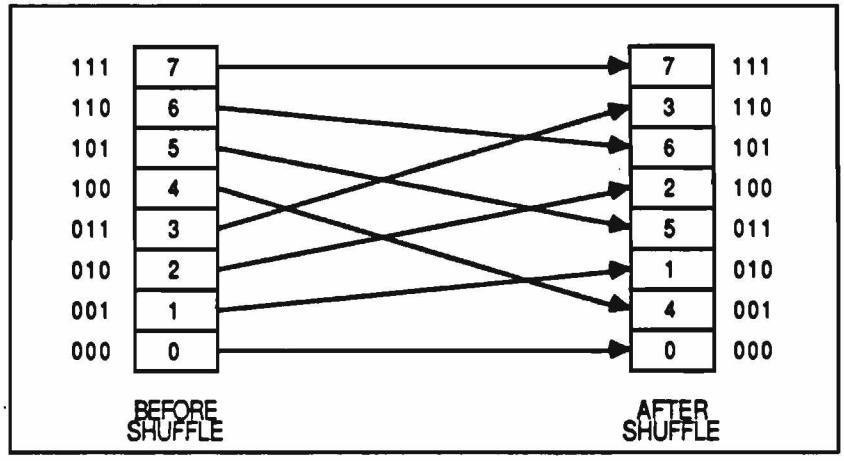
111
110
101
100
011
010
001
000
7
6
5
4
3
2
1
0
BEFORE SHUFFLE
7
3
6
2
5
1
4
0
AFTER SHUFFLE
111
110
101
100
011
010
001
000

Fig. 9

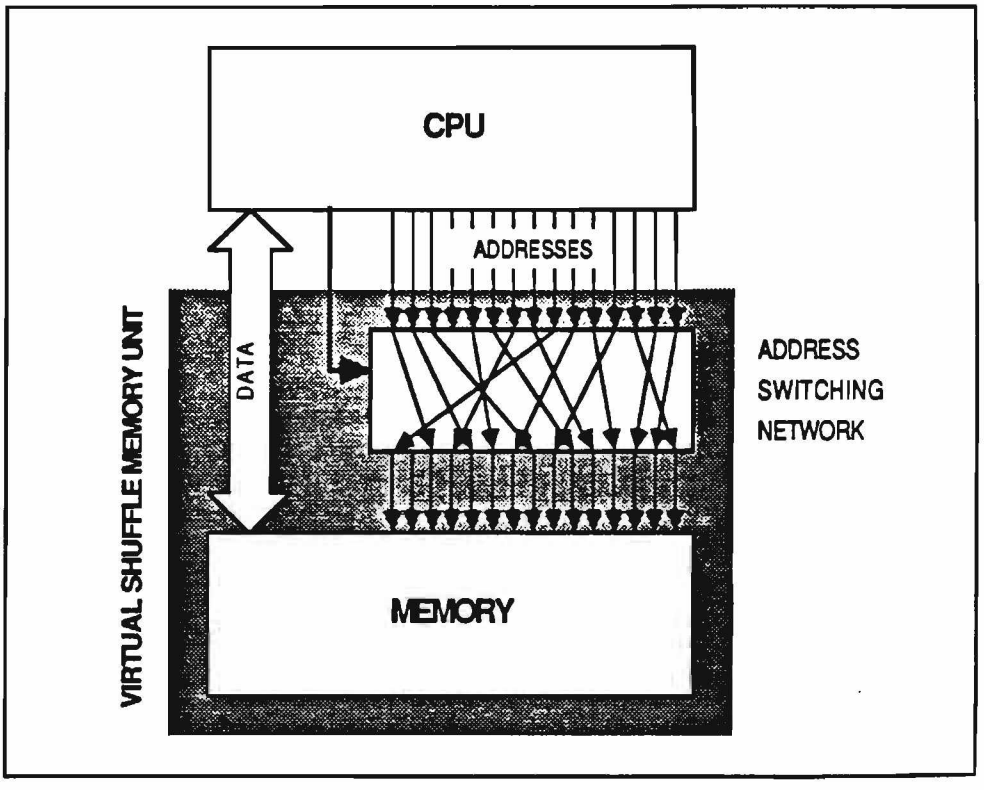
CPU
ADDRESSES
DATA
VIRTUAL SHUFFLE MEMORY UNIT
ADDRESS
SWITCHING
NETWORK
MEMORY

Fig. 10

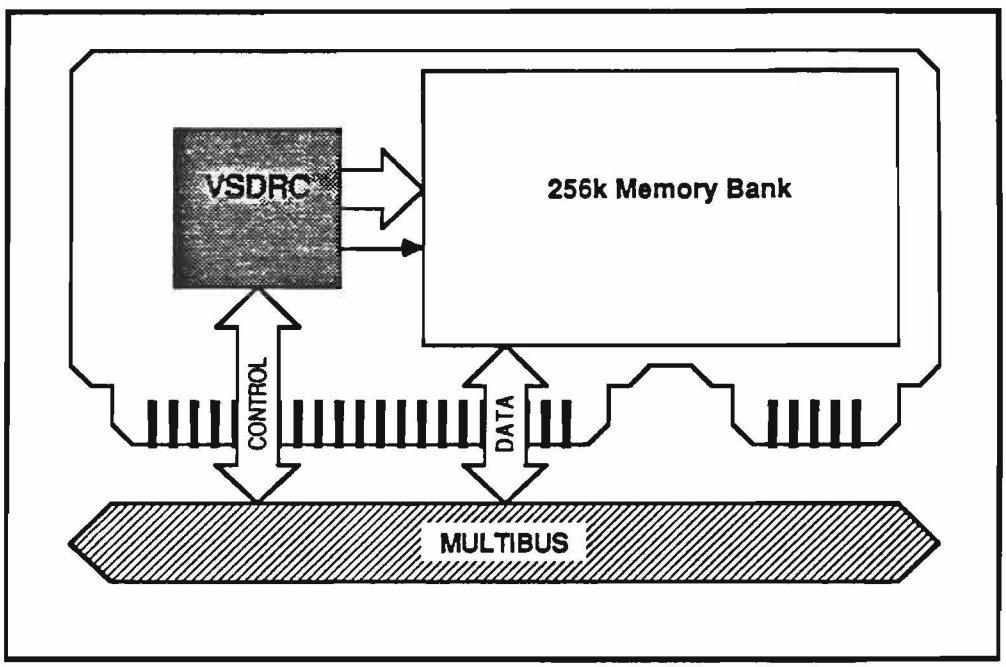

Fig. 11

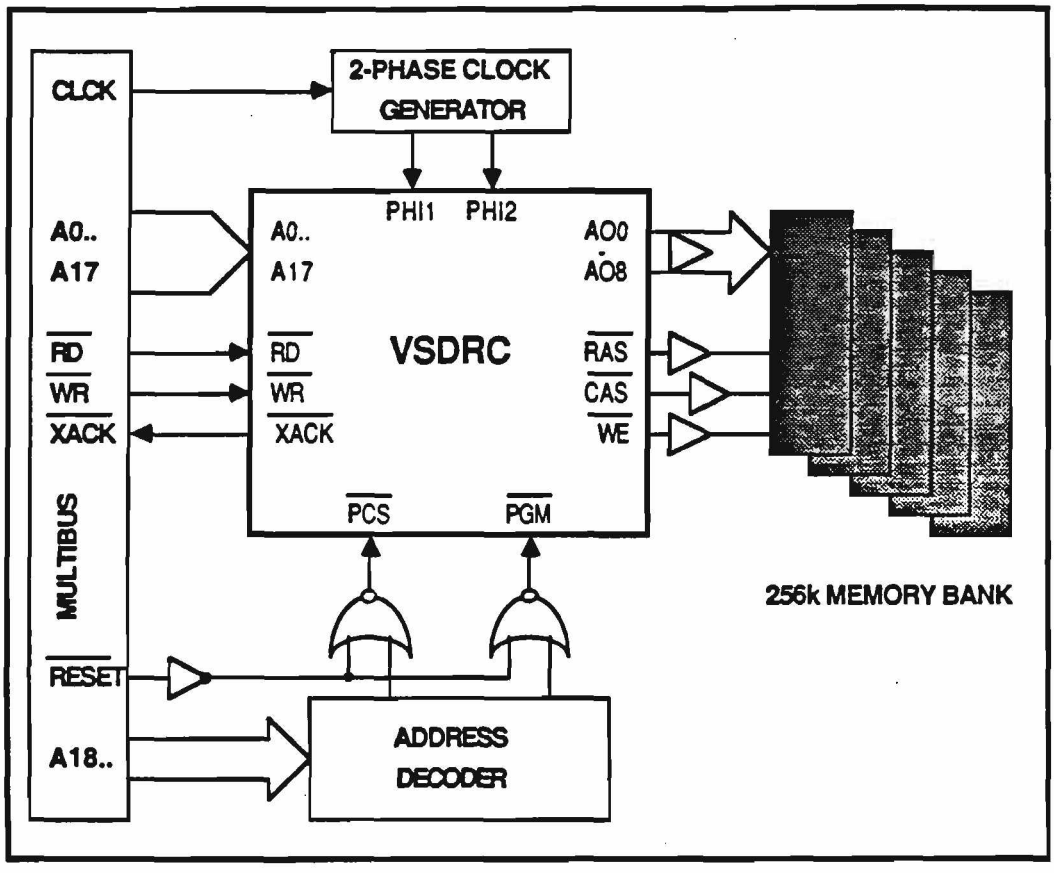
CLCK
2-PHASE CLOCK GENERATOR
PHI1
PHI2
A0..
A17
A0..
A17
AO0
AO8
RD
WR
XACK
RD
WR
XACK
VSDRC
RAS
CAS
WE
PCS
PGM
MULTIBUS
256k MEMORY BANK
RESET
A18..
ADDRESS DECODER

Fig. 12

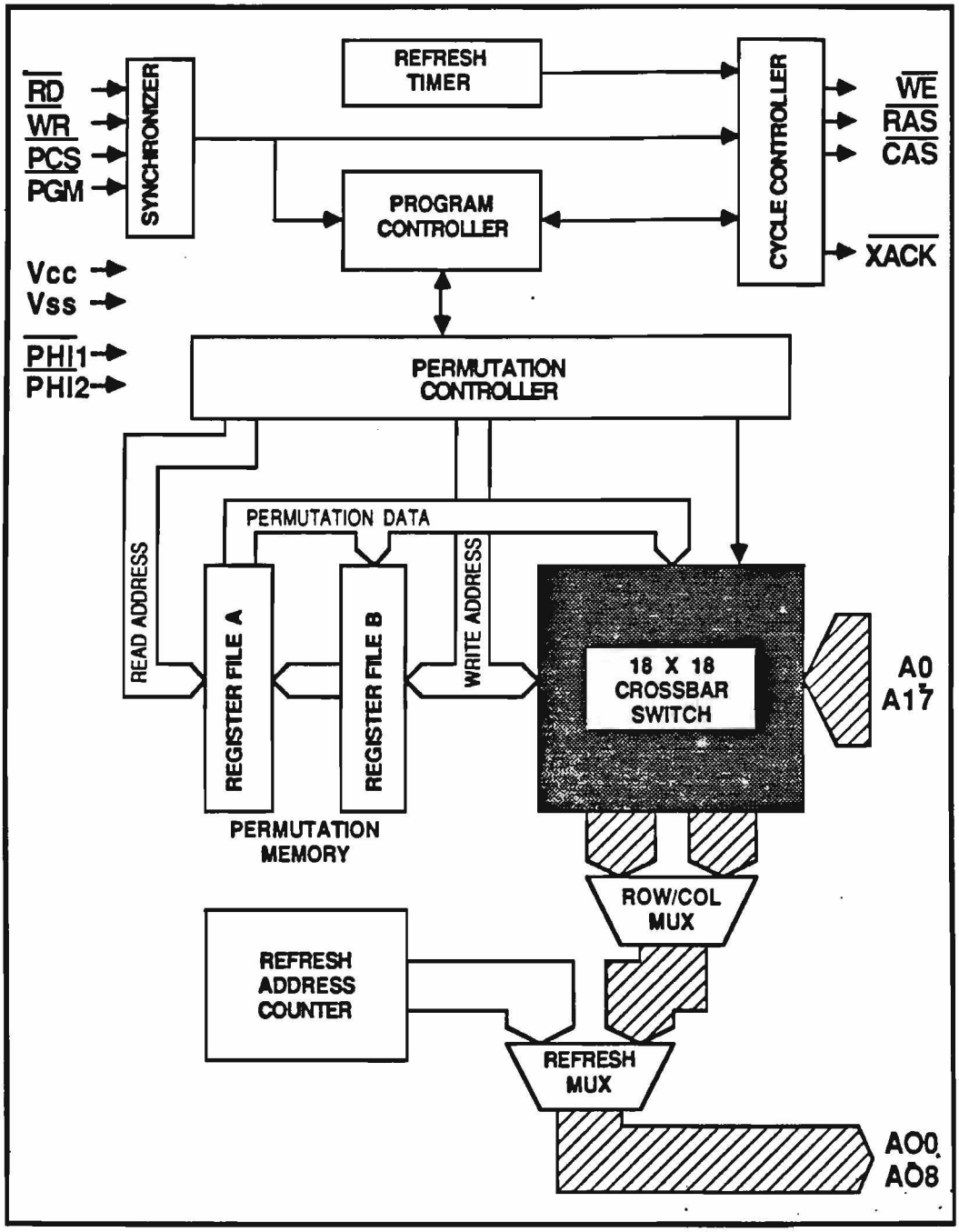
RD
WR
PCS
PGM
SYNCHRONIZER
REFRESH TIMER
CYCLE CONTROLLER
WE
RAS
CAS
XACK
PROGRAM CONTROLLER
Vcc
Vss
PHI1
PHI2
PERMUTATION CONTROLLER
PERMUTATION DATA
READ ADDRESS
WRITE ADDRESS
REGISTER FILE A
REGISTER FILE B
18 X 18 CROSSBAR SWITCH
A0
A17
PERMUTATION MEMORY
ROW/COL MUX
REFRESH ADDRESS COUNTER
REFRESH MUX
AO0
AO8

Fig. 13

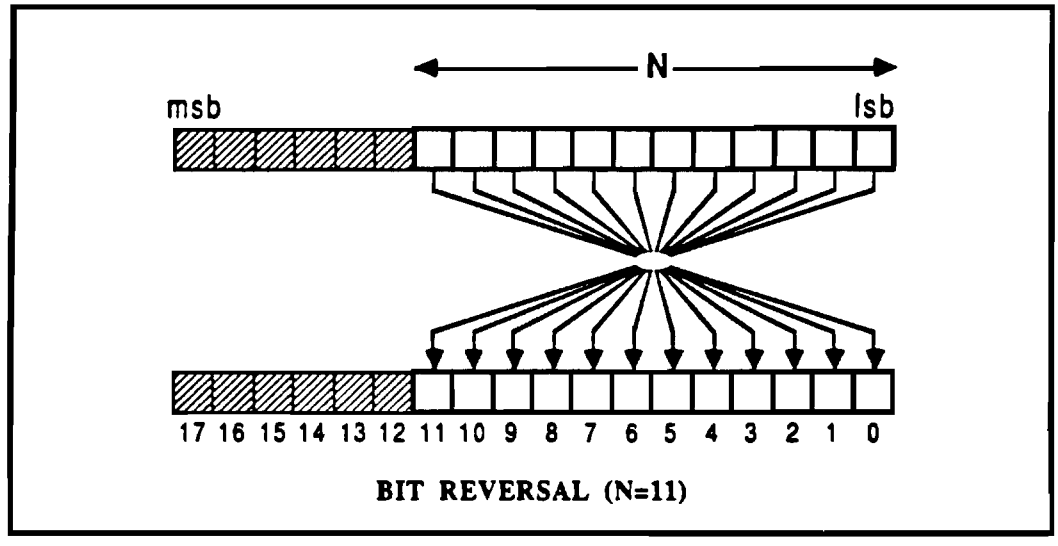

Fig. 14

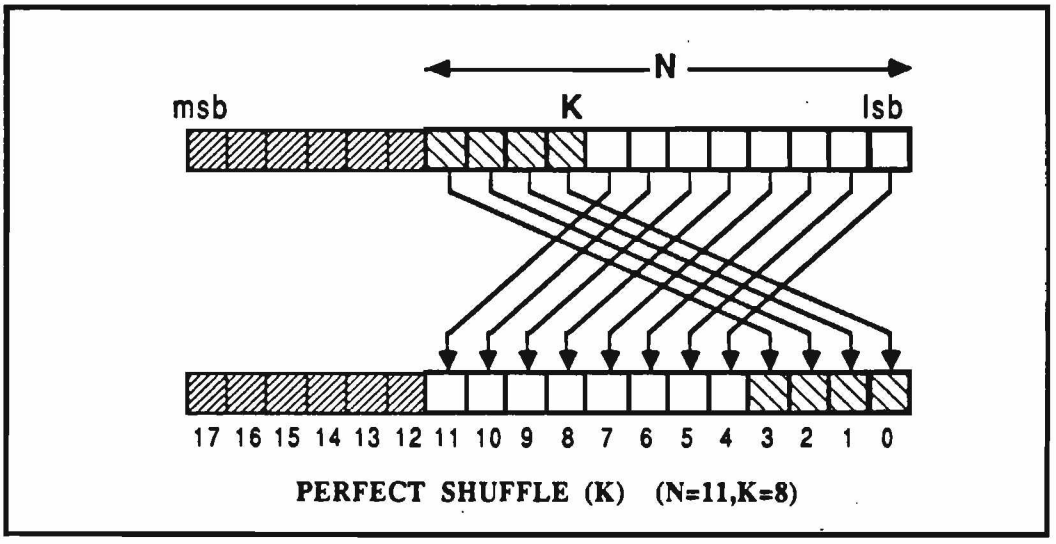
N
msb
K
lsb
17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
PERFECT SHUFFLE (K) (N=11,K=8)

Fig. 15

A5	A4	A3	A2	A1	A0	FUNCTION	
0	0	0	0	0	0	Bit Reversal	
0	k4	k3	k2	k1	k0	Perfect Shuffle (K)	(K=1..N)
1	0	0	0	0	0	Soft Reset	
1	n4	n3	n2	n1	n0	Load N	(N=1..17)

Fig. 16

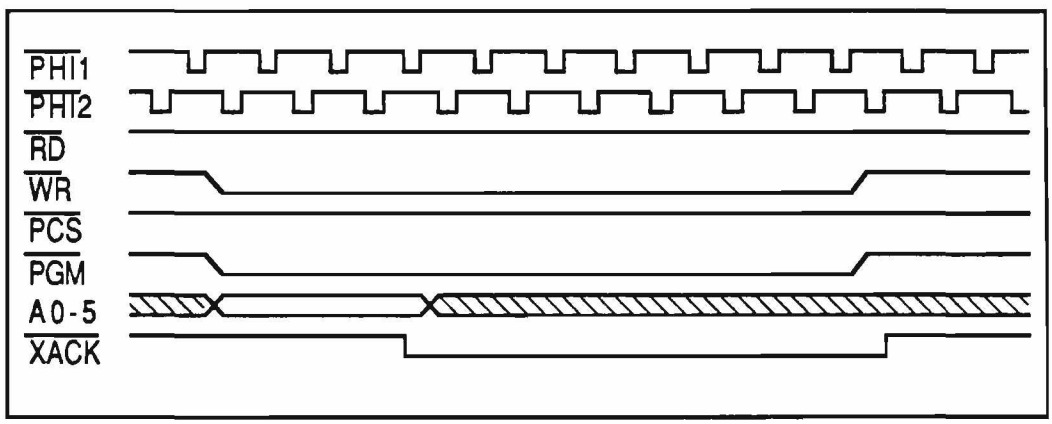

Fig. 17

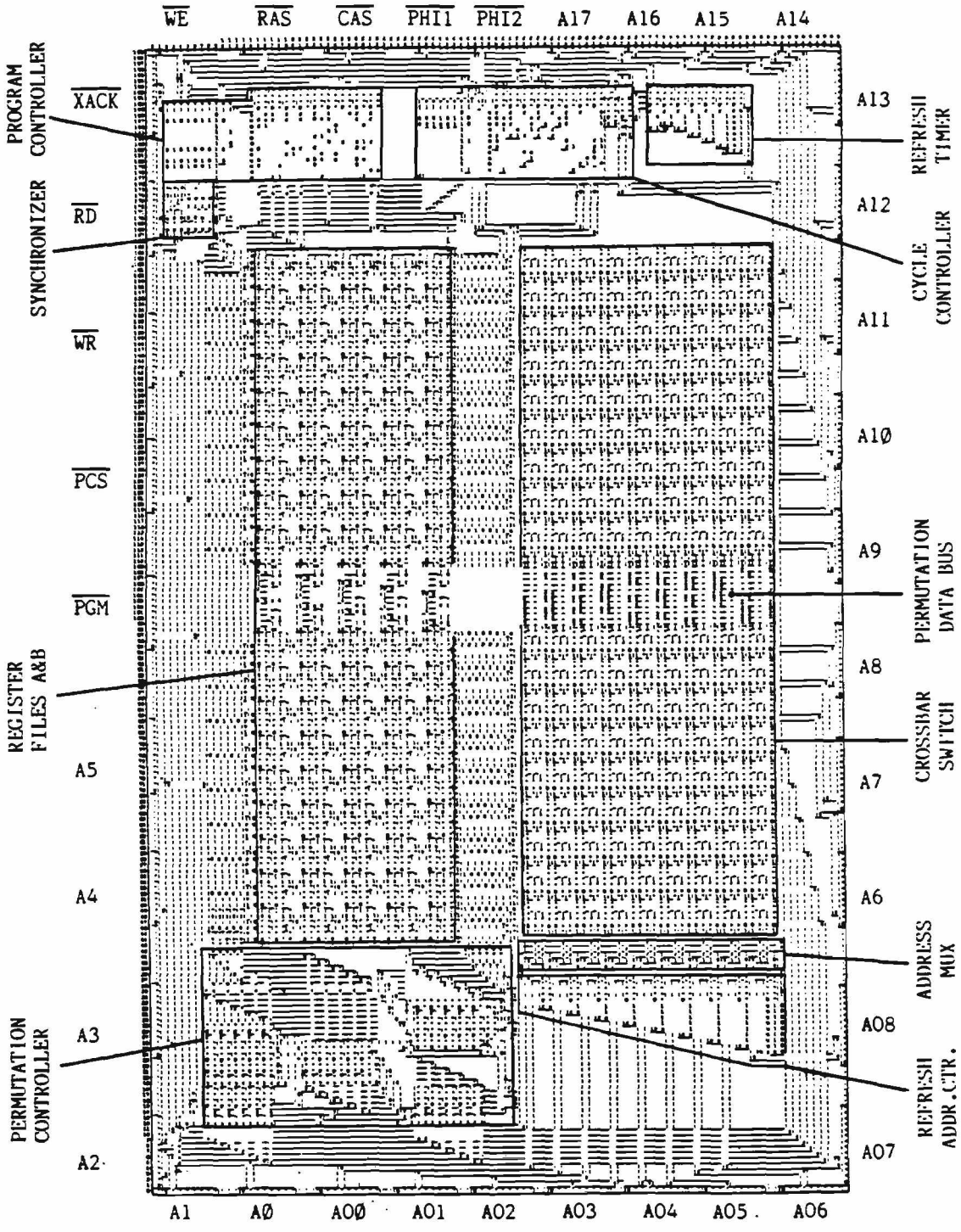
WE
RAS
CAS
PHI1
PHI2
A17
A16
A15
A14
PROGRAM CONTROLLER
XACK
SYNCHRONIZER
RD
WR
PCS
PGM
REGISTER FILES A&B
A5
A4
PERMUTATION CONTROLLER
A3
A2
A13
A12
A11
A10
A9
A8
A7
A6
A08
A07
REFRESH TIMER
CYCLE CONTROLLER
PERMUTATION DATA BUS
CROSSBAR SWITCH
ADDRESS MUX
REFRESH ADDR.CTR.
A1
A0
A00
A01
A02
A03
A04
A05
A06

Fig. 18

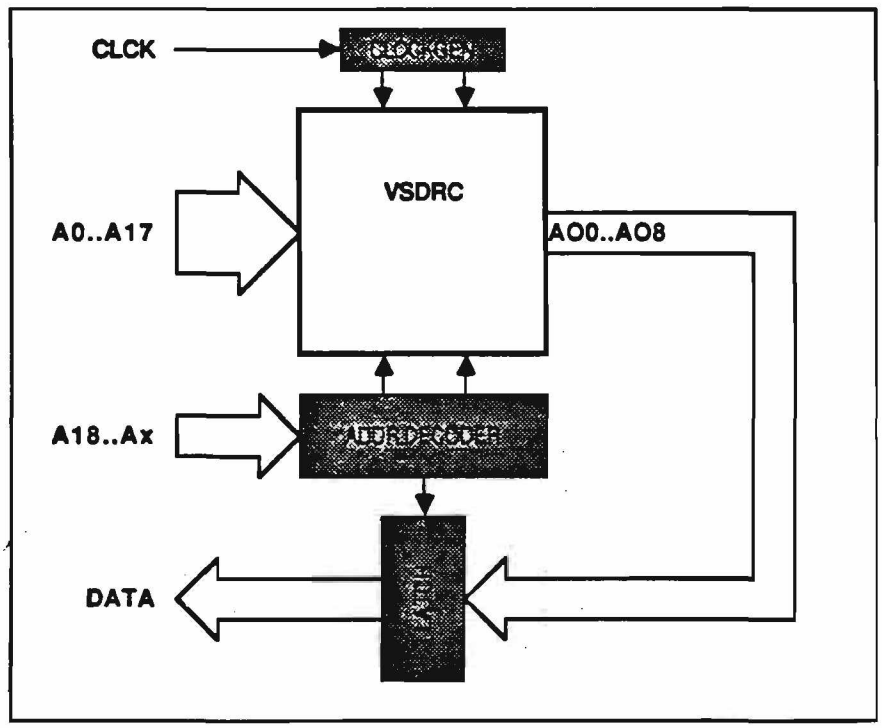

Fig.19