DESIGN OF AN 8 BIT GENERAL PURPOSE MICROCONTROLLER

A Thesis Submitted to the
Graduate School of Natural and Applied Science of
Dokuz Eylül University
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical and Electronics Engineering

by
Nalan ERDAŞ

January, 1998
İZMİR

M.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Mustafa Gündüzalp
(Advisor)

Dr. Yavuz Şenol
(Committee Member)

Dr. Haldun Sarnel
(Committee Member)

Approved by the
Graduate School of Natural and Applied Sciences

Prof. Dr. Cahit Helvacı
Director

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Assoc.Prof.Dr. Mustafa Gündüzalp for his valuable guidance, to my friends for their understanding and helps, to the researchers of TUBITAK Marmara Araştırma Merkezi-Gebze and TUBITAK Electronics Research Institute-Ankara for their guidance and to my family for their support.

ABSTRACT

This thesis includes the design of an eight bit general purpose microcontroller and the conversion of the logical design into Very Large Scale Integrated (VLSI) Circuit layout. The design style is the top-down approach. After the general structure and ability of the microcontroller is described, the system blocks and the behaviour of these blocks are defined. Secondly the logical design is implemented in gate level and simulated by logic simulator. Finally, all of the subunits are converted into 1.6μ double metal CMOS VLSI layout and these units are simulated with analog and digital switch level simulator, individually and as a complete microprocessor.

ÖZET

Bu tez çalışmasında 8 bitlik genel amaçlı bir mikrodenetleyicinin Çok Büyük Ölçekli Tümleşik Devre Teknolojisiyle tasarlanması gerçekleştirilmiştir. Kullanılan tasarım tarzı tümden gelimdir. Tasarım sırasında, önce mikrodenetleyiciyi oluşturan temel bloklar ve bunların davranışları belirlenmiş, blokların mantıksal tasarımı yapılmıştır. Daha sonra, tasarlanan mantıksal devrelerin benzeşim çalışmaları yapılmıştır. Tüm birimlerin VLSI serimlerinin 1.6μ CMOS çift metal teknolojisi ile hazırlanıp simüle edilmesinin ardından altbirimler mikrodenetleyiciyi oluşturmak üzere serim planında yerleştirilmiş ve simüle edilmiştir.

CONTENTS

Chapter Three

OCEAN-THE SEA OF GATES DESIGN SYSTEM

Chapter Four

DESIGN OF MICROCONTROLLER

Chapter Five

CONCLUSIONS

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1. Introduction

The integrated circuit technology is the one of the most important fields of the electronics. Because, it minimizes the area of the circuit, parasitic effects and the cost. Furthermore, these devices are more reliable than the circuits that are set up of the discrete components. Therefore, the interest in this subject is increasing day by day.

In this thesis, Very Large Scale Integrated Circuit Design of a general purpose 8-bit microcontroller is made. The technology used for the design is 1.6μ, double metal CMOS technology. The selected technology is CMOS, because it has many advantages when compared with the other technologies. The most important advantages are the low power dissipation and high integration density. Furthermore, the symetrical structure of the CMOS makes the design easier.

The design work was performed on a PC which use Linux as operating system. Linux is a Unix like operating system that can work on personal computers. The Linux provides the computer to show a performance which is near the performance of a real workstation. Thus, the design and simulation programs run more effectively than the other PC-based programs. Furthermore, X-window based graphical interface provides high visual quality.

The design style of the integrated circuit design is sea-of-gates design style which includes some basic architectures such as transistors. Therefore, the design time that will be consumed for the design of the basic structures is minimized. However, limits that causes by the basic image are present. For example, the interconnection paths or

the feature sizes of the transistors can not be changed. Despite of its disadvantages, the sea-of-gates style is an effective way of design. The sea-of-gate design system Ocean is used for the design and simulation. It has various tools for layout design, simulation and extraction of the designed circuit with graphical user interface.

2. Features of The Microcontroller

The general features of the designed microcontroller will be explained briefly in this section.

All the operations of the microcontroller are implemented by means of internal data bus which carries 8 bits of binary information. This provides fast implemenation of the instructions.

The microcontroller can address 64 kilobytes of address space by its internal program counter and 16 bit address bus. Additionally, control signals for external program memory are produced by the control logic of the microcontroller.

Data send and receive operations between the microcontroller and external world are performed through the 8-bit bidirectional data bus. Input- output structure consists of seperate data latches for input and output operations and bidirectional port structure.

The microcontroller includes two 8-bit registers, 16 bytes of RAM, an 8-bit shift register and an arithmetic and logic unit for data operations.

The microcontroller can run 31 instructions which can be divided into six subgroups. These are Transfer instructions, Arithmetic and Logic Instructions, Rotate and Shift Instructions, Branch Instructions, Input/Output Instructions, Control Instructions.

While the process time of the longest instruction is three machine cycles, the shortest instruction is executed in one machine cycle.

The instructions of the microcontroller are performed using three addressing modes. These are Immediate Addressing Mode, Direct Addressing Mode and Register Addressing Mode.

3. Design Methodology

A microcontroller is a sequential logic device and all sequential devices can be described by their state tables. However, the preparation of the state table for huge devices, such as microcontroller, is too complex. Therefore, the system is partitioned into modules and these modules are connected to each other with common data and control lines. The operation unit of the modules is register and the convention that is used to define operations is known as register transfer method. The data stored in registers is in binary form and all the operations are performed on these binary numbers by means of control unit. The operation that is performed on data stored in registers is called microoperation. All the instructions implemented by the microcontroller are defined in a sequence of microoperations.

While designing system, first, the instruction set of the microcontroller is prepared. Then the behaviours of the blocks that will perform these operations and the microoperations are defined. Then, each block is logically designed and simulated. After the layout of the logic blocks are prepared using the layout editor of Ocean Design System, the circuits are extracted back from the layout and the simulations of these blocks are performed until acceptable results are obtained. Finally, the blocks of the microcontroller are placed in the floorplan and it is simulated as a complete system.

CHAPTER TWO

VLSI

1. Introduction

In this chapter, the general concept of Very Large Integrated Circuit (VLSI) Technology is discussed. First, history of the integrated circuits (IC) is given briefly. Then, the terminology of VLSI is explained, the technology that is used to manufacture an integrated circuit and production steps are described. Finally, the computer aids in VLSI design is discussed.

2. History of The Integrated Circuit Technology

In the early 1930s, predecessor of field effect transistors theoretically developed by Lilienfeld and Heil. Three researchers - Brattin, Bardeen and Schockley- invented bipolar junction transistor (BJT) in 1947 and 1948. Then, various BJTs were produced and applied in a wide range of instrumentation systems [4].

In the late 1958s Jack Kilby, an engineer in Texas Instruments, introduced the first integrated circuit. Robert Noyce of Fairchild also reported an integrated circuit next year. The age of microelectronics started with these events. There have been four generations of ICs: Small Scale Integrated Circuits (SSI), Medium Scale Integrated Circuits (MSI), Large Scale Integrated Circuits (LSI) and Very Large Scale Integrated Circuits (VLSI). Now, the fifth generation of ICs has been developed, Ultra Large Scale Integrated Circuits (ULSI). In Figure 2.1, years, technology, number of transistors and typical products are demonstrated [16].

Year	1947	1950	1961	1966	1971	1980	1990	2000
Technology	*Invention of the transistor*	*Discrete components*	*SSI*	*MSI*	*LSI*	*VLSI*	*ULSI**	*GSI†*
Approximate numbers of transistors per chip in commercial products	1	1	10	100–1000	1000–20,000	20,000–1,000,000	1,000,000–10,000,000	>10,000,000
Typical products	—	Junction Transistor and diode	Planar devices Logic gates Flip-flops	Counters Multiplexers Adders	8 bit micro-processors ROM RAM	16 and 32 bit micro-processors Sophisticated peripherals GHM Dram	Special processors, Virtual reality machines, smart sensors	

* Ultra large-scale integration
† Giant-scale integration

Figure 2.1- Microelectronics Evolution

In Figure 2.2, a simplified three dimensional view of FET is demonstrated. The FET is composed of a conductive gate region. The gate is separated from the surface of the substrate by a very thin insulating layer. Drain and source regions are formed on either side of the gate by diffusion. The minimum feature size is the minimum permissible value of L and W. For example in a 5μ process, the minimum value of L and W would be 5μ. On the other side, the vertical dimensions of the FET are much smaller than the lateral dimensions (x and y). For example, the thin insulating layer under the gate in a typical 5μ process is about 1000Å thick [4]. ($1Å = 10^{-10}$ m, $1\mu = 10^{-6}$ m)

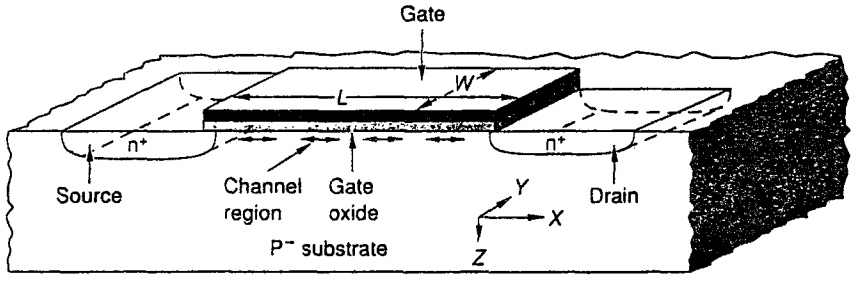


Figure 2.2-Three Dimensional View of FET

3. IC Design Technologies:

In Figure 2.3, types of major processes used in IC fabrication are shown. Process types are mainly divided into two groups, active and inert substrate processes. The physically large integrated circuits typically utilize the active substrates. Some specialized and/or low volume circuits use inert substrates. They are also used in most hybrid ICs.

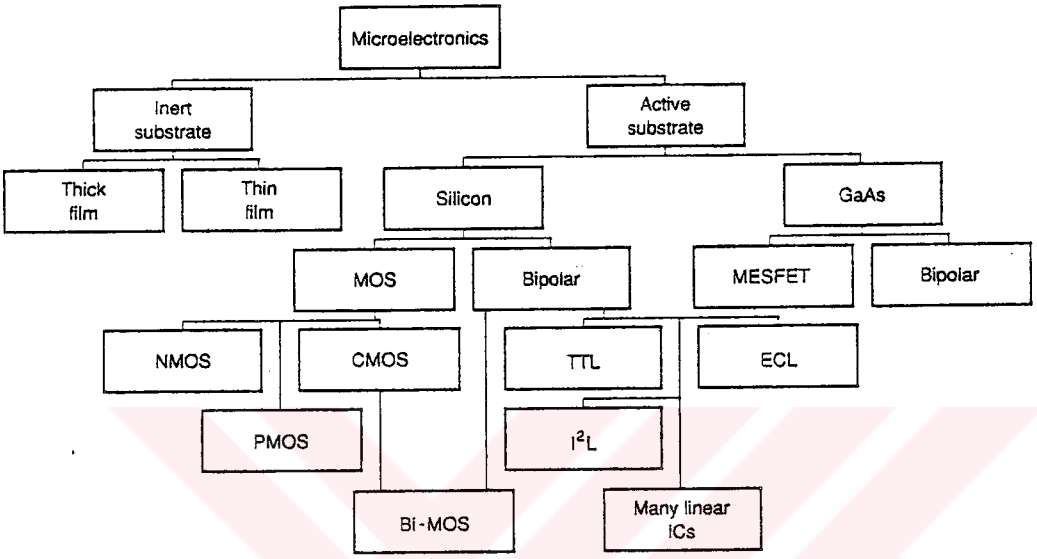


Figure 2.3- The Microelectronics Field

Two types of processes, thin and thick film processes, use inert substrates. These processes produce good resistors with quite good temperature characteristics. This is difficult to implement with active substrate processes. The active substrate processes generally use silicon or gallium arsenide as substrate. Two primary separate types of silicon processes are present. While MOS process uses the metal oxide field effect transistor (MOSFET) as the active device, bipolar process uses the BJT as an active device.

The advantages of the BJT technology are the capability to operate at high frequencies and large transconductance. However, the power dissipation is rather high and device density is not high as MOS process. TTL logic family, Emitter Coupled Logic (ECL), I^2L are fabricated in the bipolar process. Many linear ICs also are fabricated in the bipolar process.

The subgroups of MOS technology are NMOS, PMOS and CMOS. The basic devices in NMOS process is n-channel FETs, whereas the basic elements in PMOS are p-channel FETs. Although the PMOS process was used in some of the earlier MOS design, today it is rarely used. Because the mobility of the carriers in p-type material is poorer than as in n-type material. So, the electrical characteristics of PMOS devices are not attractive. In Complementary MOS (CMOS) process, n-channel and p-channel FETs are used simultaneously. As a result, the CMOS devices are more flexible than others. CMOS devices consume very low static power in digital applications. In analog applications, the circuit complexity is reduced relative to PMOS and NMOS devices. However, fabrication cost and silicon area required to basic functions increase. Generally the CMOS process is preferred to NMOS in most designs. Applications of CMOS design include memories, interfacing elements, microprocessors, basic logic functions, linear applications, mixed linear and digital applications.

Recently, there has been an effort toward combining both bipolar and MOS devices in a single process. This more complex and expensive process is termed Bi-MOS [4].

4. IC Design Process

The aim of the IC designer is to design an IC that implements a specific function with minimum labor and physical resources in a short time frame. Furthermore, a high production yield , a simple process and the a small die area is desired. In order to realize this goals, design methodologies were developed for VLSI circuits [4].

Two approaches to IC design are important: Bottom-up and Top-down In the bottom-up approach, the designer begins with the transistors and designs subcircuits of increasing complexity. Then, the designer interconnects these subcircuits to realize the specified function. In the top-down approach, the designer divides the system into groups and subgroups of simpler tasks [3]. This design can be implemented by three ways:

1. Full custom design: In this case, the designer makes all design work, that is, he designs all the circuitry, interconnections and communication paths, in the chosen technology.

2. Semi-custom design: In semi-custom design style, a standard cell library is used. The designer chooses specially designed circuits and subsystems, places in floor plan, and interconnects them.

3. Gate-array and Sea-of-Gates design : The standard logic elements are presented to the designer in these design styles. In this case, an image that includes basic elements such as transistors in a certain configuration is produced and the designer designs the interconnection/communication paths. Then the interconnection layer is manufactured [9].

5. Yield

Typically some portion of the dies on a wafer do not meet the performance specifications. "The percentage of the dies that meet the performance specifications is termed the wafer yield" [4]. In general, while die area increases, the yield decreases. Therefore, die area plays a major role in economics of IC design.

The specific types of circuits, the design methodology followed by the designer, the layout itself and the physical fabrication process defines the yield. There are some factors that affects yield. These are dust particles, crystal defects, alignment errors, breakage and human handling errors and parameter drifts. Dust particles on wafer will cause local defects. If these dust particles are large they cause problems such as failure of transistor, breaks in an interconnection, or shorting of two adjacent devices, levels or interconnections. Therefore, all the operations are done in vacuum. Operators wear special clothes and the outer media should also be disinfected. These types of defects are generally randomly distributed over the surface of the wafer and from wafer to wafer. Prior to wafer fabrication crystal defects may be present and that can cause local circuit defects such as failure of transistors. Parameter drifts will cause transistors to have characteristics different from the desired. "Failures due to parameter variations are termed soft faults. Failures that cause complete transistor failure or interconnection errors are named hard faults." [4]. A hard fault generally causes a part of the circuit or all of the chip not to function , while circuits with soft faults may be functional, but fail to meet specifications. Faults cause to reject of a die, so it increases the effective cost of an IC. Furthermore, they are often difficult and expensive to detect the faults. So, the testing problem should be addressed at the design stage. ICs often contain additional circuitry which is used in testing the circuit itself.

6. IC Production Process

In order to produce an integrated circuit several production steps are followed. Here are the major process steps that are used in Bipolar and MOS technology.

6.1 Crystal Preparation

In the most bipolar and MOS integrated circuits a single crystal of silicon that is lightly doped with n-type or p-type impurities is utilized as substrate. These crystals are sliced from large right-circular cylinders of crystalline to have lengths up to 2m and diameters vary from 1 to several inches. The typical thickness of slices are 250μ to 400μ [4].

6.2 Photolithography

Photolithography is used to pattern the layers of an IC. A viscous liquid, named photoresist is applied in a thin, uniform layer to the entire surface of a wafer following cleaning. The layer is about 1μ thick. While the photoresist is putting on, the wafer is spun at high speed. After application, the photoresist is hardened by baking. As its name implies, photoresist is a photosensitive chemical. So, it can be changed by exposure to ultraviolet light. Thus, photoresist is used with a mask to define certain areas of the wafers. IC masks are high-contrast (black on clear) Photolithographic positives or negatives. The masks are typically made of glass covered with a thin film of opaque metal. After the exposure, the resist is selectively removed from unwanted areas. Then, another baking step is used to harden the photoresist [3].

6.3 Deposition

During the process of integrated circuits, films of various materials must be applied to the wafer. These films can be thin (200Å or less), but may be as thick as 20μ for thick film circuits. Insulators, resistive films, conductive films, dielectric, n-type and p-type semiconductor materials and dopants are created by deposition method. The deposition techniques are physical vapor deposition, chemical vapor deposition (CVD) and screen printing for the thick films. The physical and chemical vapor deposition is nonselective and are placed uniformly over the entire wafer [4].

Evaporation and sputtering are the two types of physical vapor deposition: In the evaporation, the material that is to be deposited is evaporated by controlling the temperature and pressure of the host material environment. A film is formed when the material condenses. Sputtering refers the bombardment of the host material with high energy ions. A disk from the host material is used as a cathode of an high-voltage, low-pressure discharge system. Anode is the wafer. When a high voltage of 1,000 to 20,000V is applied to the system the molecules of the host material dislodge. These molecules reattach themselves to the surface of the wafer.

6.4 Etching

The selectively removal of the unwanted material from the surface of the substrate is called as etching. Photoresist and masks are used for this purpose. Generally, several etching processes are applied to a single IC. The chemicals, called etchant, react with the unprotected layers of the wafer while not affecting the protected areas. The wet and dry etching are used in manufacturing. The wet etch (or chemical etch) use liquid etching agents, which are applied to substrate surface. These etchants etch horizontally as well as vertically into the surface of the substrate. This horizontal etching causes undercutting of the patterned areas.

Dry etching is also called plasma or ion etching. It uses a stream of ions and electrons to blast material away. It is done by creating ions in a glow discharge and directing the ions to the target. Ions will be typically penetrate about 800Å of oxide or photoresist [4].

6.5 Diffusion

Diffusion in IC processing is controlled forced migration of impurities into the substrate or adjacent material. The subsequent diffusions generally cause some additional migration, but at normal temperatures it takes tens of years for the additional movement to become significant.

There are several methods for introducing impurities. A solid deposition layer or a gaseous layer above the surface can be used as a source of impurities. Impurities can also be accelerated to selectively bombard the substrate so that they actually become lodged inside the substrate very near the surface in ion implantation. Ion implantation is very accurate however it causes significant crystal damage near the surface.

6.6 Conductors and Resistors

In the integrated circuit production process for interconnection of the components conductors are required. Aluminum or other metals are often used for this purpose. These metals are typically deposited in a thickness of typically about 6000-8000Å, patterned and etched to leave interconnects where desired. Metal films are particularly useful for interconnects.

For small current flow, nonmetallic films are used for conductors and interconnects. Polysilicon is one of the most popular nonmetallic conductors. Polysilicon is composed of a large number of nonaligned, randomly oriented, small silicon crystal. Polysilicon is a good conductor when heavily doped and a good resistor when lightly doped. Polysilicon is often used as gate in MOSFETs and as an electrode for capacitors.

6.7- Oxidation

The growth of an oxide layer on the surface of the wafer is called oxidation. Since the substrate or surface material is typically silicon, the oxidation produces silicon dioxide. The SiO_2 layer serves as a very good insulator.

Another method of the obtaining SiO_2 layer is to apply chemical vapor depositon. This technique is used extensively when the SiO_2 layer must cover something other than silicon.

6.8- Packaging and Testing

The last steps in IC process are the packaging and testing. Generally, the product is tested by using test plugs in order to see if it is acceptable.

Nowadays, the test patterns are began to place into scribe lines, the grid lines where cuts will be made to separate dies. A wafer prober is used to make mechanical contact with the test plugs so that electrical measurement can be made. If the process parameters are within the tolerance, the individual dies are automatically probed and electrically tested. Defective dies are marked with the ink and later discarded. Later the dies are separated. Following, separation, the individual dies are attached or bonded to a carrier of the IC package. Wire bonds are subsequently made from the pins of the package to the appropriate locations on the

die. The bonding wires are typically of either gold or aluminum. After the wire bonds are complete, the packages are formed or closed and final electrical tests are performed [3].

7. Basic MOS Processes

MOS process is one of the most favorite integrated circuit manufacturing processes. It involves NMOS, PMOS and CMOS technology. The following sections will introduce basic MOS processes.

7.1 NMOS Process

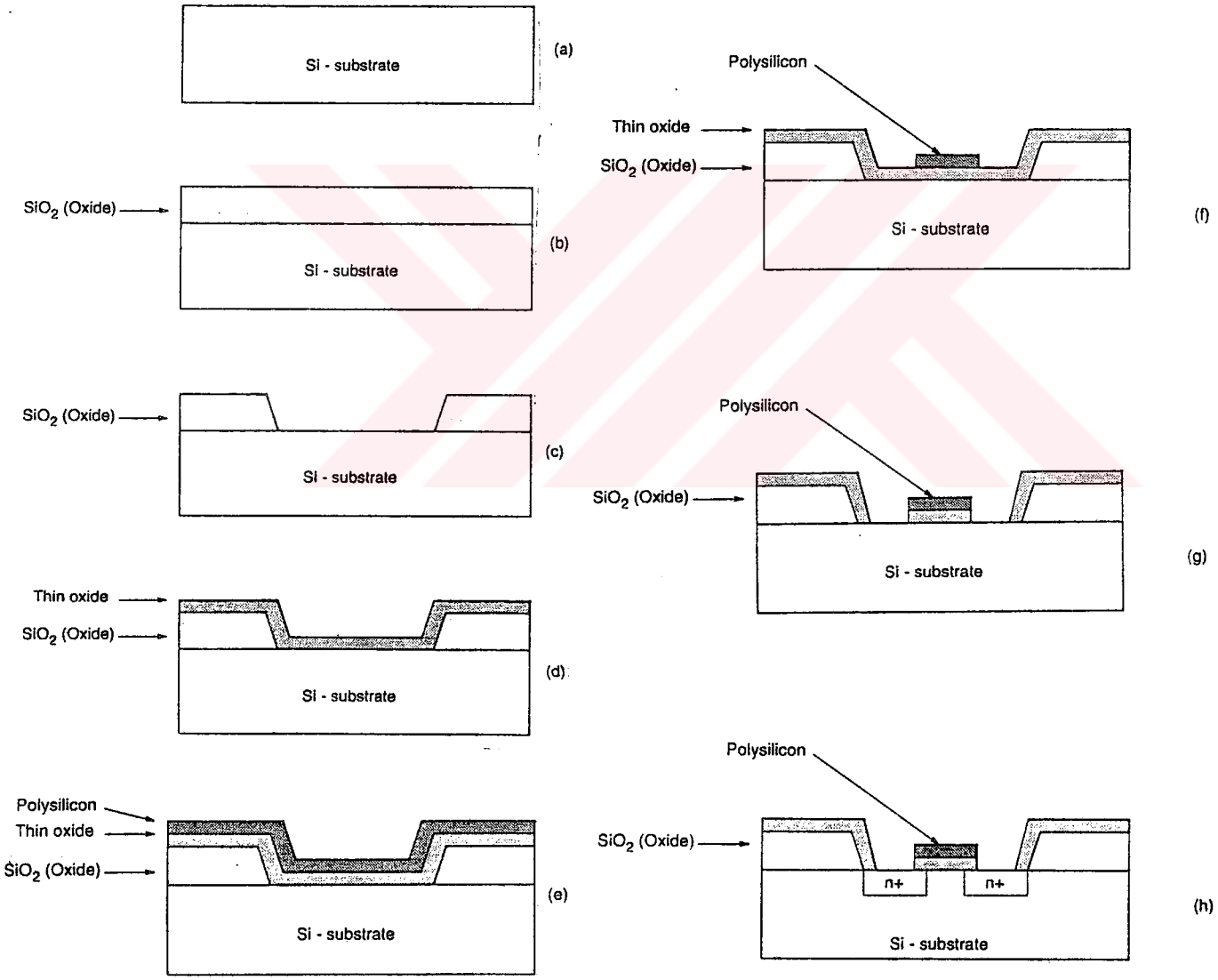
In this section a generic double-polysilicon self-aligned silicon gate NMOS process will be discussed.

N-channel enhancement and depletion MOSFETs, capacitors, resistors can be manufactured in NMOS process. The starting point is a polished p-type silicon disc of 500μ thick for all components. Firstly, entire wafer is oxidized. The layer is typically 1μ thick. Then, photoresist (Si_3N_4) is applied and moat or n^+ diffusion mask that defines where n implants are desired is used to pattern the surface. After exposure, the photoresist is removed from the certain areas, so is the underlying SiO_2. A heavier selective implant is required in regions that are to serve as the channels of the depletion transistor. To achieve this, a second layer of photoresist is applied to the entire wafer. The second mask, termed implant mask, is used to pattern photoresist, so that only channel regions are unprotected. Another n-type implant is used to make the exposed regions n-type. After stripping the photoresist, a thin layer of SiO_2 , typically 0.1μ is deposited. This layer often called as gate oxide. Then, a thick polysilicon layer is formed in the gate region by chemical vapor deposition. If it is desired, resistors and the first plates of capacitors are produced in this step by forming polysilicon layer. Again, a SiO_2 layer that serves as dielectric for capacitors is deposited. A second polysilicon layer should be applied if the capacitors are included in the circuit [4].

The next step is to apply n+ diffusion to the entire surface in order to form drain and source region. Since the gate region is masked by the polysilicon and the gate oxide, it does not require an additional mask and for this reason this process is termed self-aligned. Later, another insulating layer and the photoresist layer are applied over the entire surface. In this step the mask that defines the contact holes for drain, source and gate regions is used. The

whole chip surface is metallized. This metal layer is then masked and etched to form required interconnection paths [3].

Large, square metal areas are called bonding pads and are used to connect the contacts with the IC package. The entire surface is finally covered with a passivation layer, often called glass or p-glass. This provides the long-term stability of the IC by minimizing atmospheric conditions. The passivation layer is typically 10,000Å thick. Since this layer is an insulating layer, it is necessary to again pattern it and make openings above the metal pads to attach to bonding pads. The process steps are demonstrated in Figure 2.4 [9].



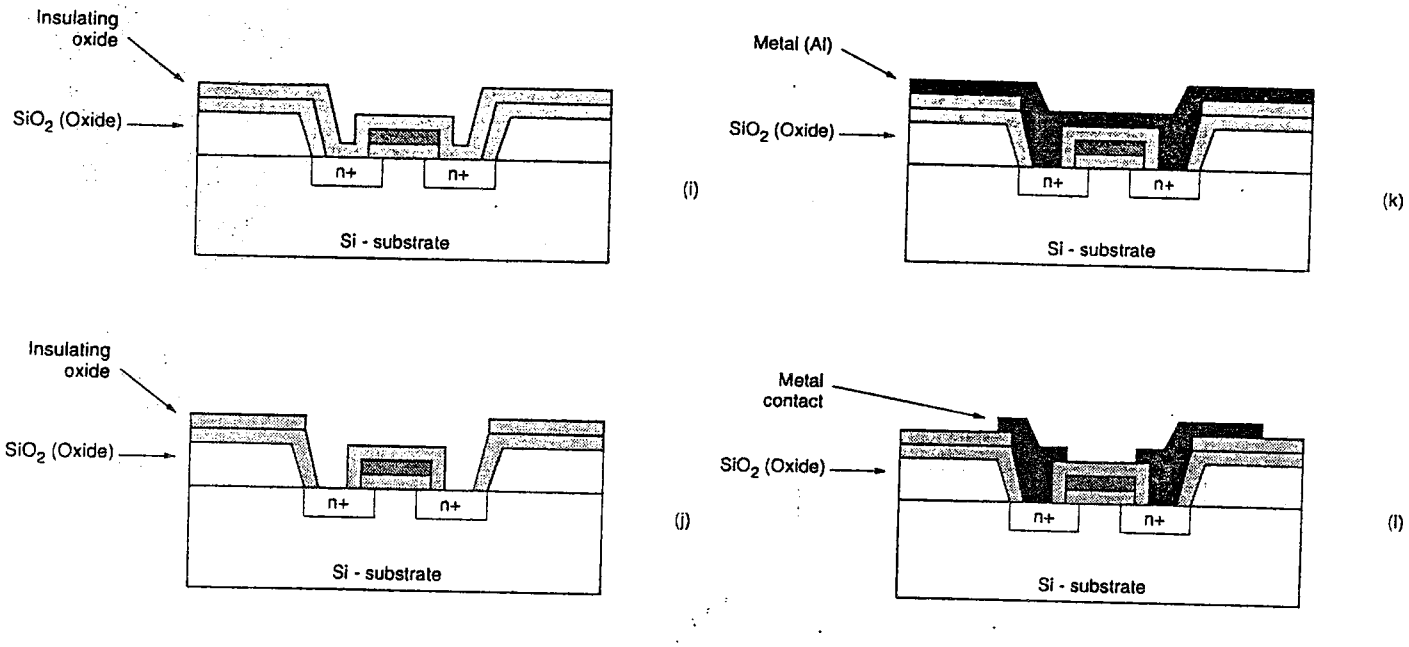


Figure 2.4 - NMOS Process Steps

7.2 CMOS Process:

The devices that are available in this process are n-channel MOSFETs, p-channel MOSFETs, capacitors, resistors, diodes, npn bipolar transistors, pnp bipolar transistors. There are a number of approaches to CMOS fabrication, including the p-well, the n-well, the twin tub. The p-well process is widely used in practice and the n-well is also popular.

7.2.1 The p-well Process

In the p-well process, the substrate is n-type where p type devices can be formed in. A deep p well is diffused in the substrate in order to form n type devices. The process steps are similar to NMOS process steps.

7.2.2. The n-well Process

In the n-well process, the substrate is p type and a deep n-well is created. Although p-well process is widely used, n-well fabrication is also gaining acceptance. N-well CMOS circuits are also superior to p-well because of the lower substrate bias effects on threshold voltage and lower parasitic capacitance.

7.2.3. The Twin-tub Process

In this process, both n-well and p-well is created in an high resistivity n-type material. Thus, the characteristics of both n and p type devices can be controlled effectively. But, the area required to implement an element is larger than the p- and n-well processes.

For all of these processes, the masks are prepared by the designer. The size, shape and spacing of the components should be carefully determined. Because these values affect the performance of the circuit. The designer should optimize the design requirements such as minimum size and maximum capability, with constraints.

8. Computer Aided Design Tools

Design automation and verification are important for fabricating integrated circuits of today. When circuits consisted of only a few of transistors and gates, layout and checking of circuits by hand were reasonable. As circuit complexity increased to thousands of transistors, manual tools were no longer sufficient for design. Thus, computer based design aids became important. Computer aided design tools consist of simulators, layout editors, design rule checkers etc.

Integrated circuit layout methods include both synthesis of control logic and handcrafting of critical building blocks that will be repeated. These layout pieces are entered into a computer to allow a mechanized help with replicating, checking and plotting the complete layout. Design layouts may be entered via tools that converts graphic layout information to computer-readable form or may be entered directly as text files. The tools that are used for design and verification of the integrated circuits will be considered below.

8.1 Geometrical Specification Languages

Geometrical specification languages contain primitive structures such as wires and boxes to specify geometrical shapes and layout levels. Then, the entered textual layout is converted to mask information for the fabrication of an IC [4].

8.2 Graphics Editors

A layout can be entered into a computer through an interactive CRT graphics editor. A corresponding data file is kept in computer memory to describe the displayed geometries. This data file can be converted into geometrical specification file or can be stored for further use.

The memory and disk representations of layout geometries of graphical editors are typically in binary form and every one of them is different from another. For this reason, common interchange formats are defined. In the university environment, the geometrical specification language CIF (Caltech Intermediate Form) is a common interchange format among universities and between universities and MOSIS fabrication houses. The name MOSIS stands for MOS implementation system and it is the name of a program sponsored by the U.S. Department of Defense Advanced Research Projects Agency. Designs from a large number of different institutions are combined on a multiproject chip format in MOSIS. Many fabrication houses use these common design rules. Thus, the fabrication cost is reduced relative to the cost for an independently implemented IC. In industry, EDIF (Electronic Design Interchange Format) was defined as the interchange format. Most industrial CAD tools provide conversions between their internal format and EDIF [4]. In addition, most industrial CAD tools convert their internal format to a special binary for mask generation [3].

8.3 Design Rule Checks

As described earlier, ICs have several layers. Because of lithographic and processing constraints, a minimum resolution exists for the structures that can be fabricated on silicon and requiring higher resolution may lead to nonfunctional circuits. Furthermore, geometrical relationships among layers are exist and violation of these relationships may cause failures. As a result, a set of guide lines called design rules is specified [4]. Design rules are generally well-documented specifications listing minimum widths of features, minimum spacing allowable between adjacent features, overlap requirements, and other measurements that are compatible with the given process. Some examples of design rules are demonstrated in Figure 2.5 [9].

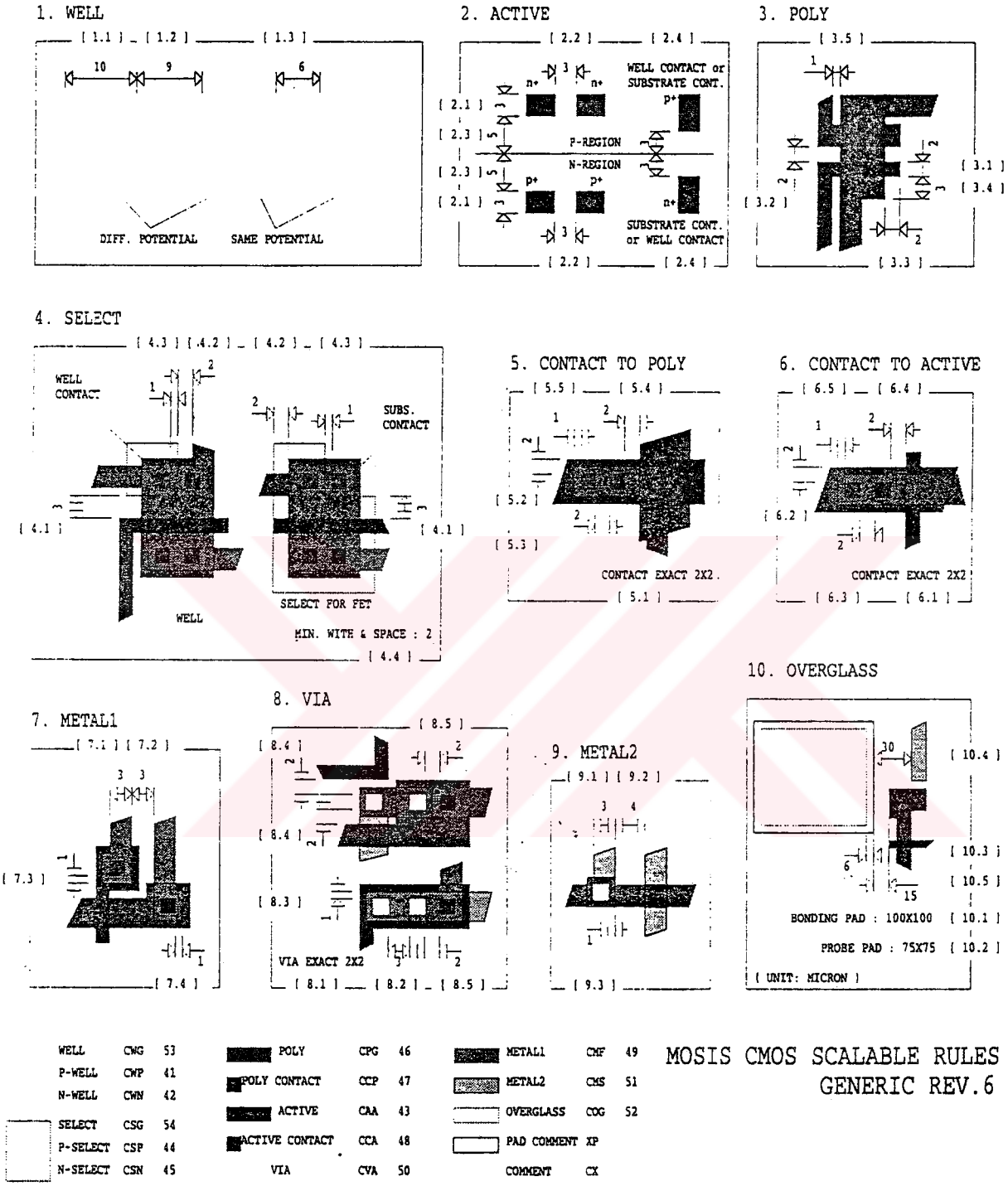


Figure 2.5-MOSIS Design Rules (Sample Page)

8.4 Circuit Extraction

An integrated circuit is designed and then its actual layout is prepared by using computer aided design tools. The circuit is characterized by machine readable specification files for the mask making steps. Before manufacturing, this layout should be compared with the original circuit, but it is too difficult for a designer to determine whether this information accurately describes the original circuit. Fortunately, computerized methods exist to extracting the information from geometrical specification file. This process is called circuit extraction. A circuit extracting program expands the geometrical specification file to layer to layer description of geometries and their placements, and a netlist, a set of statements that specifies the elements of a circuit, is obtained. The extracted circuit can be compared with the original circuit specified by the designer. This comparison is called a layout versus schematic (LVS) design verification step.

8.5- Digital Circuit Simulation

Circuits whose external operation is fully digital may require accurate circuit simulation to model critical signal delay paths. A common used circuit simulator is Spice circuit simulator. Because of the large number of transistors in digital circuit such as microprocessors, it is not feasible to perform circuit simulation for the entire circuit. It is too time-consuming. So, verification of the operation for the large circuit should be accomplished by other means. Many times a simulation at the logic level or switch level can provide sufficient information for the digital circuit's functionality. Sometimes even logic simulation programs are too slow to model entire processor's behavior. In these cases special hardware simulators are required.

In the logic level simulation, the operation of a circuit block is described in terms of its behavior such as AND, OR or NOT. More complex digital blocks such as full adders and multiplexers are each described by their corresponding behavior. The circuit inputs are specified as binary values at discrete time intervals. Logic simulator outputs are provided as binary values as well. Pure logic simulation does not model time delays. In order to model time delays and to find the critical paths, timing analysis methods are used. In hardware simulation approach, special-purpose hardware executes many simulation steps in parallel. That means, the hardware consists of many identical processing units. Thus, the simulation becomes faster.

CHAPTER THREE

OCEAN : THE SEA-OF-GATES DESIGN SYSTEM

3.1- The Overview of The System

Ocean is a comprehensive chip design package which was developed at Delft University of Technology, the Netherlands. It includes tools for the synthesis and verification of semi-custom sea-of-gates and gate-array chips. The structure of the design system and tools are demonstrated in Figure 3.1.

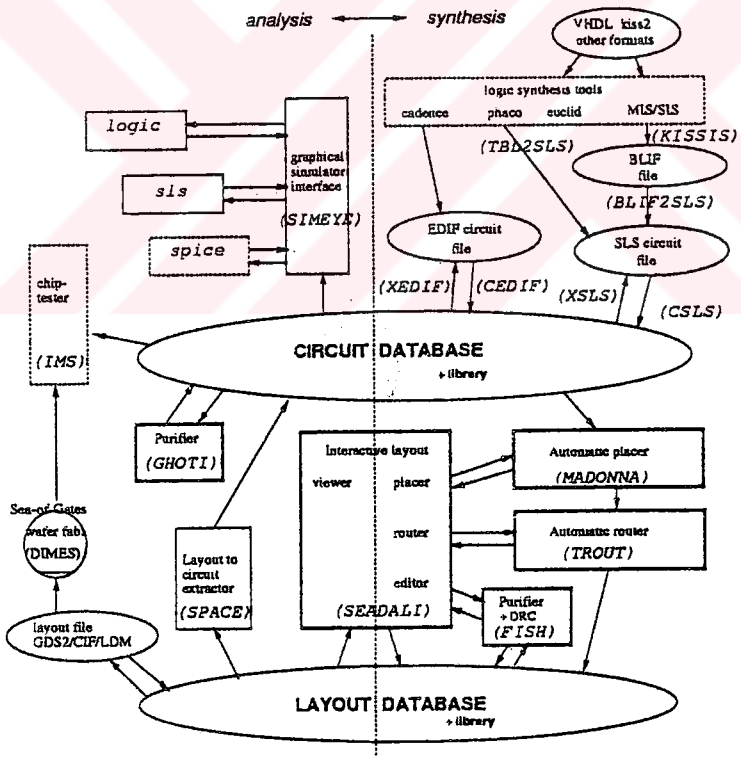


Figure 3.1-Overview of The OCEAN Design System

In contrast to other sea-of-gates and gate-array systems, Ocean has a hierarchical (full-custom-like) layout style on sea-of-gates. Modules of any shape and size can be used and created. Therefore, complexity can be handled effectively.

Ocean allows to run logic synthesis tools such as SIS and the circuits can be entered in various format. The textual circuit entry can be made in SLS-network format. Additionally, Edif circuit interface is present. Designed circuits can be interactively simulated at logic level, switch-level and Spice.

Another feature of Ocean is automatical place and route option. Both the placer and the router make efficiently use of the unused space in the sub-modules. Any number of interconnect layers can be handled. Many special provisions for semi-custom layout (e.g. substrate contacts, power rails connection) are supported. Unused transistors can be automatically converted into capacitances for power decoupling.

Also, place and route operation can be performed manually. Interactive layout editor, Seadali provides general interface for automatic and manual layout generation. Any existing layout cell can be viewed or modified, an empty layout cell can be created, instances can be placed automatically or manually, correctness of the layout against the design rules and circuit description can be verified.

The connectivity and correctness of the layout design can be verified with Ocean tools. Any short-circuits and unconnects are indicated in the layout. Design rule checking and layout purification of manual layouts is performed. Objects are snapped to the grid. Stacked vias are indicated.

The circuit can be extracted back from the layout with the extractor tool, which is capable of extracting accurate parasitics.

Interactive simulation of the original circuit and the extracted circuit on three levels of accuracy is possible with simulation tool. The Ocean can run on HP, Sun and 386/486 PC machines.

3.2- Why Sea-of-Gates Design Style

The Ocean design system is targeted to produce sea-of-gates layout. This means that they deal with a prefabricated transistor pattern on the chip. To implement a circuit, these transistors are interconnected with metal wires. The advantages of using sea-of-gates design style are as follow:

The fabrication time is minimized. Because the chips are prefabricated (the transistors are already on the master image), the silicon foundry only processes the masks related to metal wires.

The design time is minimized. The time involved in designing a cell layout is reduced as compared to full-custom. Because the transistors are preplaced on the image.

The chip cost is minimized. The layout design starts with a prefabricated master image. This is a semi-manufactured article that can be produced in large quantities.

Full-custom properties on Semi-Custom chip is provided. Efficient implementation of structured logic (RAM, PLA etc.). In contrast to the conventional gate-array, a sea-of-gates does not have pre-defined routing channels. This enables a much more compact and clean implementation of structured circuits such as processors.

3.3- Primary Programs in OCEAN

The primary programs in Ocean Design System are listed below:

- SEADALI : Interactive layout editor, general interface for automatic and manual layout generation.
- MADONNA : Automatic placer.
- TROUT : Automatic router, connectivity verifier.
- FISH : Layout purifier for sea-of-gates, design rule checker.
- SPACE : Layout extractor.
- SLS : Logic level and switch-level simulator.

- SIMEYE : Interactive simulator interface
- GHOTI : Circuit purifier for spice.
- CSLS, XSLS,

  CEDIF, XEDIF : Write or read netlist formats.

3.4 - Master Images and Process Technologies

The OCEAN system comes with technology files and libraries for three images:

•Fishbone : A gate-isolation image in 1.6μ CMOS process with two layers of metal (Fig 3.2)

•Octagon : A symmetrical and rotatable image in 0.8μ CMOS process with 3 layers of metal (Fig 3.3).

• Gatearray : Old fashioned row-based gate-array in 4μ single-layer CMOS process (Fig 3.4).

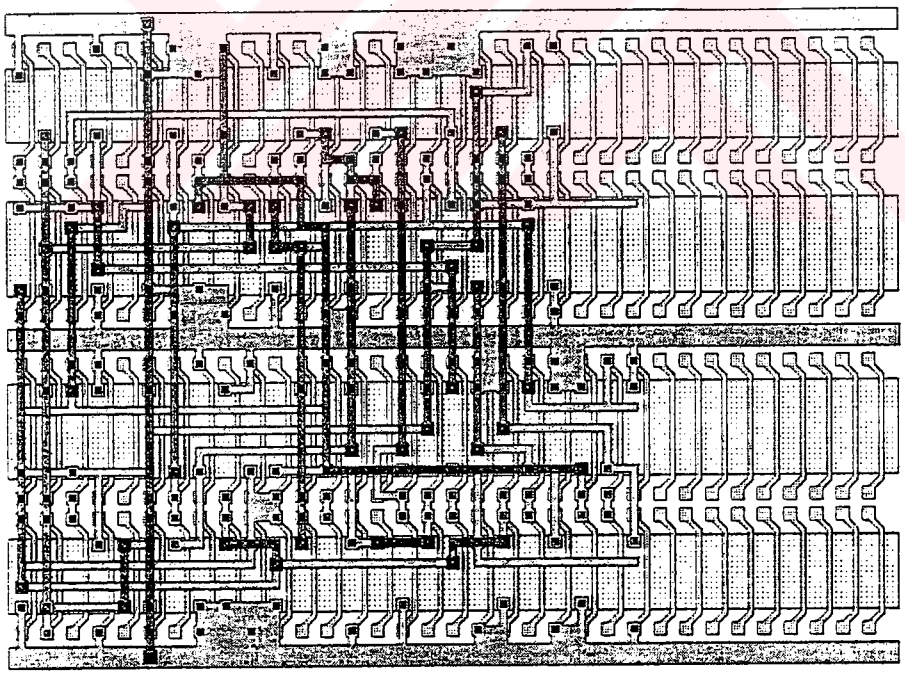


Figure 3.2- Fishbone Image

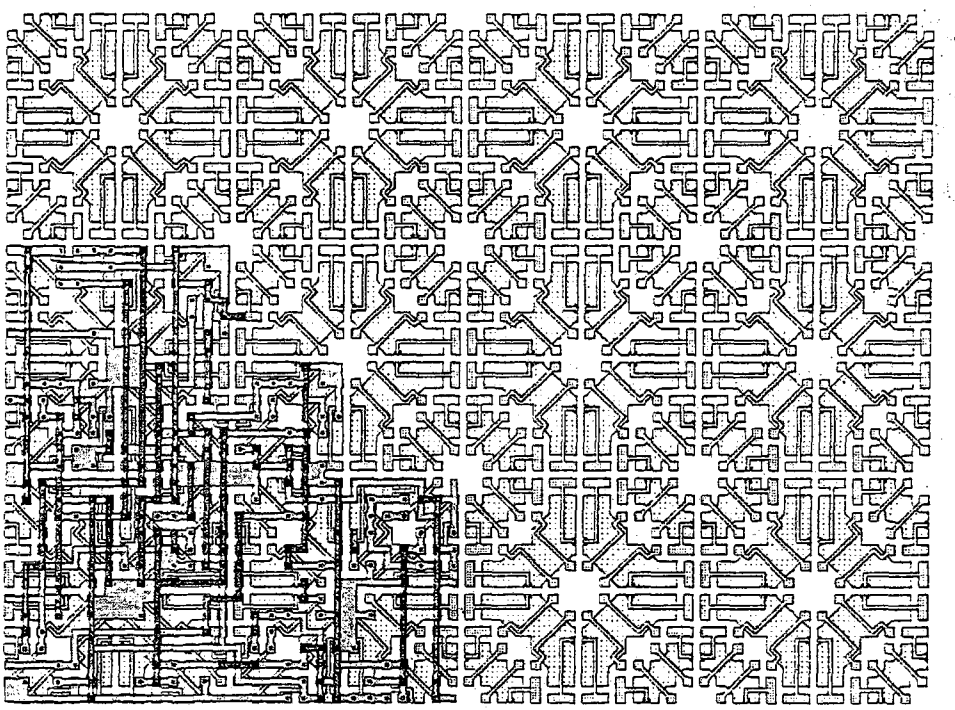


Figure 3.3- Octagon Image

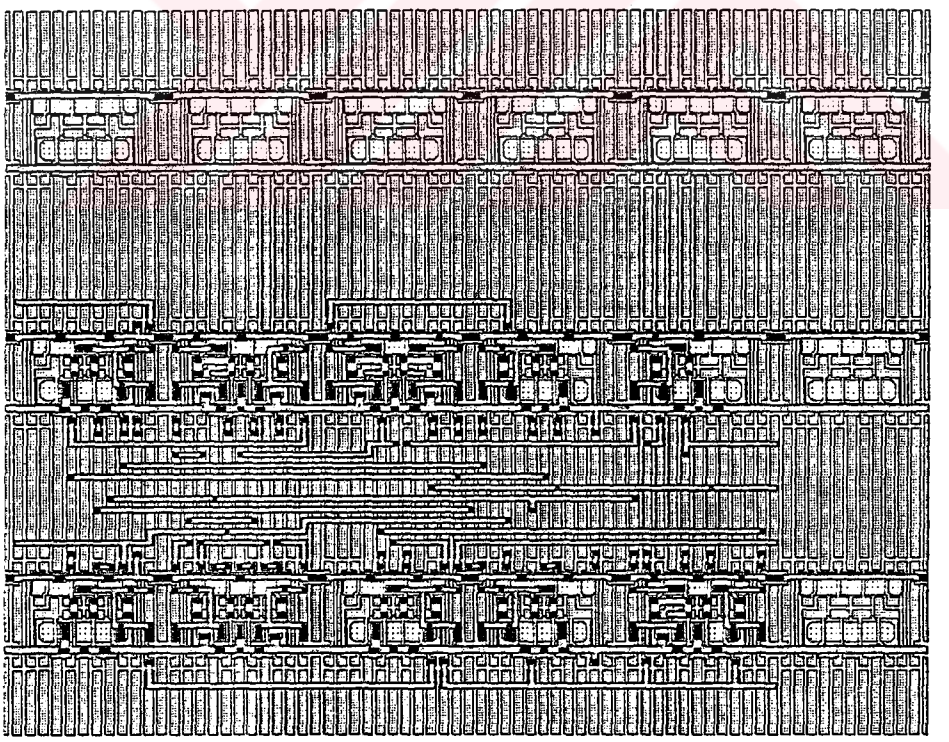


Figure 3.4- Gatearray Image

The layout of an entire 200,000 transistor chip in the 'fishbone' image is included with OCEAN. This chip has 144 pins on the area of approx. 10 x 10 mm. For smaller circuits the chip can be configured as a multi-project chip containing 2 or 4 separated designs.

3.5 - The Properties of The Fishbone Image

The overall properties of the Fishbone image will be explained briefly in order to have an insight of the design work. The Fishbone image is fabricated in the C3TU process, the Delft derivative of the Philips C3DM process.

The feature size is 1.6μ. Two layers of metals. These are; metal 1, the metal layer that is closest to the image (in), metal 2 that is the second metal layer (ins). Also, the contact layers are present; con, cop and cps, contacts from in to n-type diffusion, p-type diffusion or poly; cos, contact between in and ins.

The fishbone image consists of series of n-type and p-type transistors. The connections between transistors and contacts may only be placed certain grid points.

3.6 - Switch-Level Simulator

SLS is a switch-level simulator that can be used to simulate the logic and timing behavior of large digital circuits. It has accurate algorithms to predict the timing behavior of MOS circuits containing > 100,000 transistors. There is an X-window based user-interface to graphically edit the input signals and to inspect the simulation output signals.

The SLS simulator has three different simulation levels:

1. Purely logic simulation based on abstract transistor strengths: It computes logic states by only considering node states and transistor types.

2. Logic simulation based on exact transistor dimensions and node capacitances: This level uses resistance division and capacitance division algorithms to compute logic states. It finds correct logic states in much more situations than conventional switch-level simulators, e.g. when a resistance division occurs between a saturated transistor and a non-saturated transistor.

3. Logic and timing simulation based on transistor and node parameters: RC time constant evaluations are used to approximate real voltages by piecewise linear voltage waveforms. This not only provides delay times for the circuit, but is also delivers an accurate representation for transient effects like spikes and races.

Apart from electrical network elements like MOS transistors, resistors and capacitors, an SLS network may contain (i) gate primitives like inverters, nands, nors, etc. and (ii) user-defined function blocks like roms, shift registers, multipliers. The behavior of function blocks is described by the user in the C programming language: it is specified by the user how the values of the output terminals and the state variables are computed from the values of the input terminals and the state variables.

Two examples of the SLS files are shown below:

TRANS.sls

```
network trans(terminal sel, i, j, vss, vdd)
{  penh w=29.6u l=1.6u (sel, 1, vdd);
   penh w=29.6u l=1.6u (1, i, j);
   nenh w=23.2u l=1.6u (sel, 1, vss);
   nenh w=23.2u l=1.6u (sel, i, vss);
}
```

RAMTEST.sls

```
extern network dfr11 (terminal D, R, CK, Q, vss, vdd)
extern network trans (terminal sel, i, j, vss, vdd)
extern network buf40 (terminal A, Y, vss, vdd)
extern network ram2 (terminal Bit1, Bit2, Sel1, Sel2, vss, vdd)

network ramtest(terminal in, in1, r, ck, s0, s1, s2, out, s01, s11, s21, out1, vss,
vdd)
{
{inst1} dfr11(in, r, ck, g1, vss, vdd);
{inst2} trans(s1, g1, bo, vss, vdd);
{inst3} buf40(bo, g, vss, vdd);
{inst4} trans(s2, g, to, vss, vdd);
{inst5} dfr11(to, r, ck, out1, vss, vdd);
{inst6} buf40(out1, out, vss, vdd);
{inst7} ram2(g, h, s0, s01, vss, vdd);
{inst8} dfr11(in1, r, ck, g11, vss, vdd);
{inst9} trans(s11, g11, bo1, vss, vdd);
{inst10} buf40(bo1, h, vss, vdd);
{inst11} trans(s21, h, to1, vss, vdd);
{inst12} dfr11(to1, r, ck, outx, vss, vdd);
{inst13} buf40(outx1, outx, vss, vdd); }
```

Both the SLS results are displayed by the X-window based user-interface SIMEYE. Apart from normal commands like zooming in and out, SIMEYE has a special edit mode that allows the user to create and edit logic input signals. These input signals can be used for both SLS and SPICE simulations. For SLS, both a digital representation and a waveform representation can be displayed. Also, for SLS, abstract output values can be displayed like an integer value for a bus of output signals. A sample output file for an one bit adder cell is given in Figure 3.5

SLS
version: 3.0
SIMULATION RESULTS

time in 1e-08 sec	v s	v d	I 0	I 1	C 0	C 1	S
0.000	0	1	0	0	0	0	0
1.250	0	1	1	0	0	0	1
2.500	0	1	0	1	0	0	1
3.750	0	1	1	1	0	1	0
5.000	0	1	0	0	1	0	1
6.250	0	1	1	0	1	1	0
7.500	0	1	0	1	1	1	0
8.750	0	1	1	1	1	1	1
10.000	0	1	0	0	0	0	0
11.250	0	1	1	0	0	0	1
12.500	0	1	0	1	0	0	1
13.750	0	1	1	1	0	1	0
15.000	0	1	0	0	1	0	1
16.250	0	1	1	0	1	1	0
17.500	0	1	0	1	1	1	0
18.750	0	1	1	1	1	1	1
20.000	0	1	1	1	1	1	1

network : add nodes : 19

Figure 3.5- Example Output File of Switch Level Simulator

CHAPTER FOUR
DESIGN OF THE MICROCONTROLLER

1. Structure of The Microcontroller

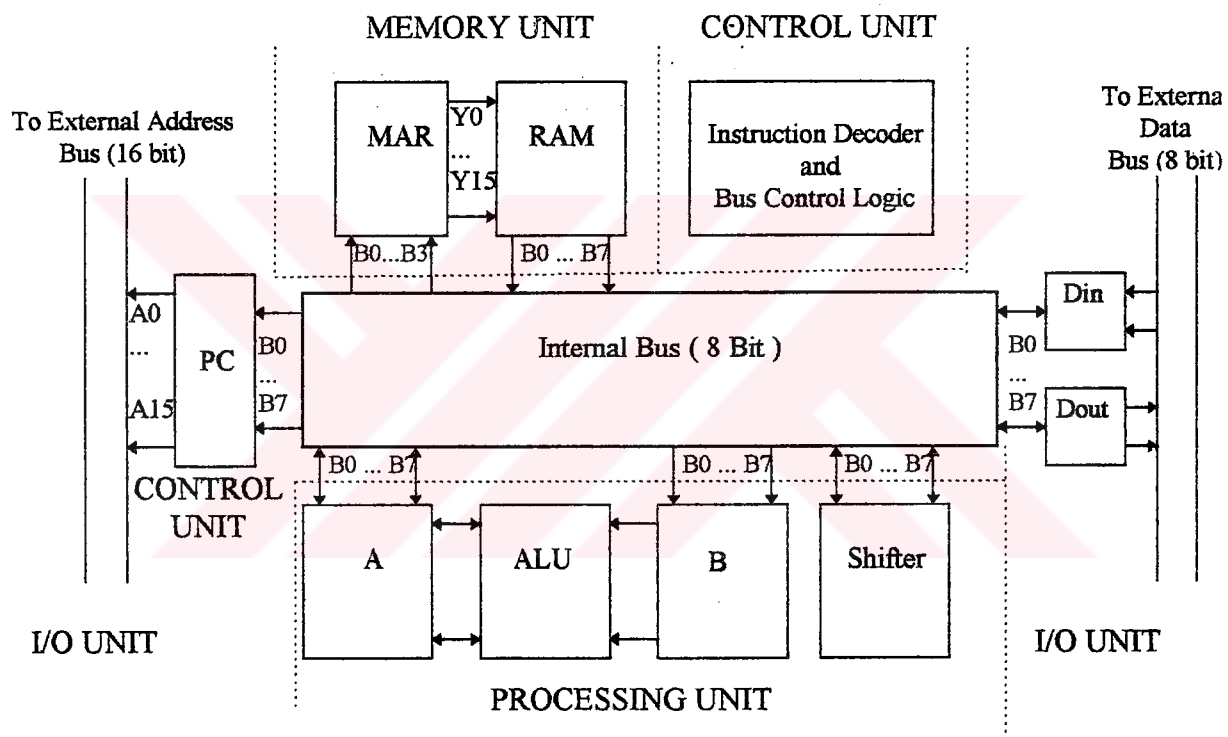


Figure 4.1 - Block Diagram of The Microcontroller

The architecture of the microcontroller is bus oriented which means that it has a main data bus that connects all subunits. Thus, the data flow among the subunits are faster than a microcontroller that has a unidirectional bus, because all operands and results are easily transferred to their destinations by the internal data bus.

Basically, the subunits of the microcontroller can be grouped into four main blocks. These are processing unit, memory unit, input/output unit and the control unit (Figure 4.1).

The processing unit consists of arithmetic and logic unit (ALU), shifter and two data registers A and B. The 8 bit inputs to ALU are from A and B registers and the result is written to A register. The shifter takes its data from any of the external memory, internal memory and registers through the internal data bus and the result is loaded to source again.

The memory unit includes memory address register (MAR) and RAM block. The memory address register is four bits wide and latches the internal RAM address from the bus in order to read from or write to memory. The RAM block consists of a 4 to 16 decoder and 16x8 bit static memory cells.

2. Operation of The Microcontroller

The operation of the microcontroller can be explained in the following way: The instructions are read from the external EPROM through external data bus and the input latch, Din. Then, the instruction is decoded by the control logic circuitry and it produces control signals for the microoperations. Additionally, bus control logic controls the access of the subunits to internal data bus. After decoding, one of the possible operations is followed according to the type of instruction. For the transfer, arithmetic-logic and shift operations, the operand that is needed for the instruction is read from the external program memory in immediate addressing mode case, or from internal memory in direct addressing mode case, or from registers in register addressing mode case. Then, the desired operation is performed upon the operand in the processing unit and finally the result is written into the destination. For branch instructions, the least significant byte of the branch address is read from the external program memory and it is written to least significant byte of the program counter. Additionally, if the branch instruction is conditional, the status flags are controlled to determine the next operation. For output instructions, the data is transferred to the out of the microcontroller through data output latch and I/O port.

3. Subsystems

As described earlier, microcontroller consists of four subunits: Processing unit, memory unit, control unit and input/output unit. The regularity is a key factor in subsystem design. If a standard cell could be found, then by replicating these cells the desired function is implemented. Therefore, in this design, the leaf cells are searched and realized as a first step. After the simulation of the standard cells they are connected to build the subsystem units if the results are acceptable. Finally, the subsystem is simulated. In the following sections the design steps and the results will be introduced.

3.1 Processing Unit

The processing unit is the unit that achieves data manipulations on the operands. These operations could be transfer, arithmetic, logic and shift operations. The subunits of the processing unit are the Arithmetic and Logic Unit, Data Registers and Shift Register.

3.1.1 Arithmetic and Logic Unit

The most important part of the ALU is a full adder. It is used to implement both arithmetic functions, such as addition and subtraction, and logic functions, such as AND, OR, NOT functions. The other parts are the status flags Carry and Zero, arithmetic output latch and logic output latch.

A one-bit standard full adder that adds I0 and I1 with carry C0 and gets the output S and the output carry C1 can expressed with the following equations;

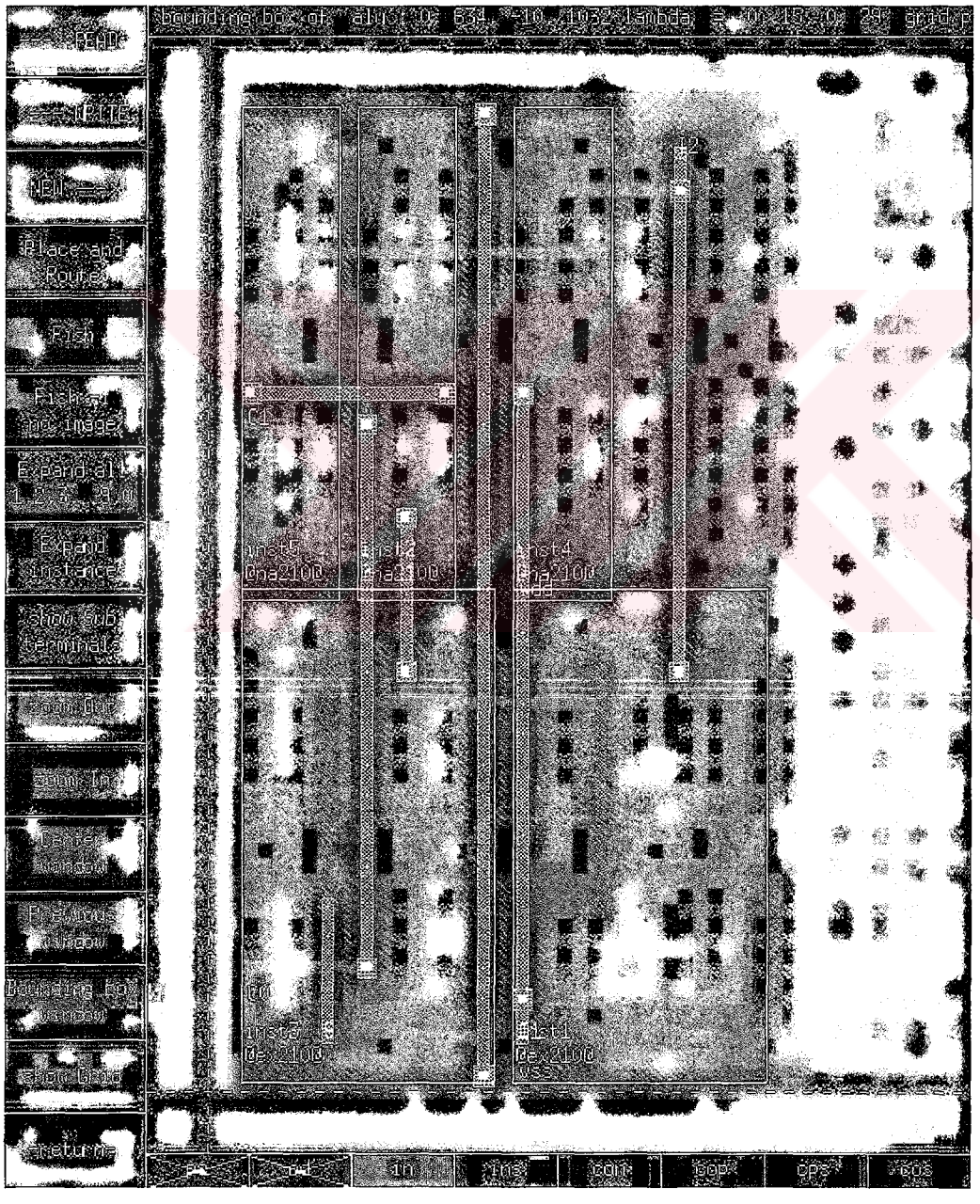
Sum $S = H\overline{C0} + \overline{H}C0$

Carry $C1 = I0I1 + HC0$

where $H = \overline{I0}I1 + I0\overline{I1}$.

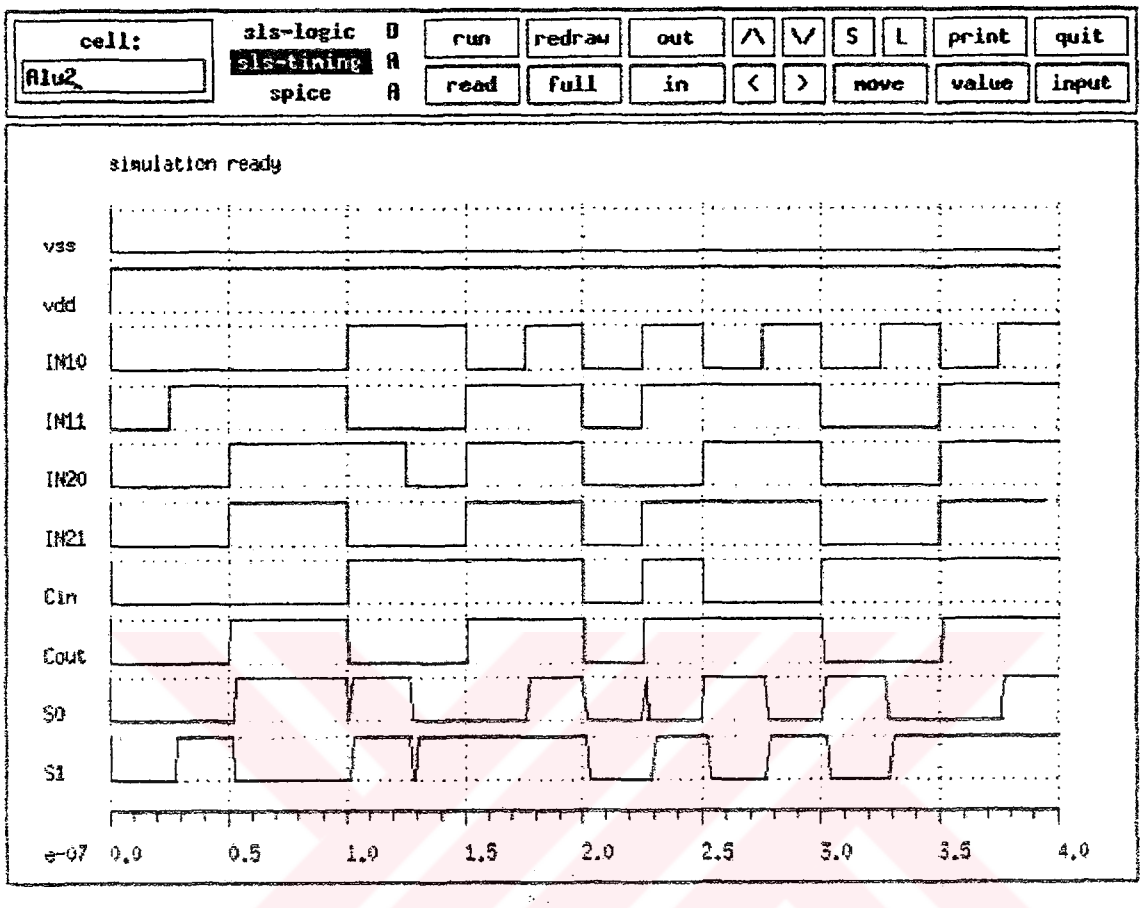
As can be seen from the equations, S and H can be obtained with exclusive-or gates. These equations only express the arithmetic operations. However, if the input carry is set to "1" or "0" level externally, the logical functions can be obtained [15].

$$C0 = 0 \Rightarrow C1 = A \bullet B$$
$$C0 = 1 \Rightarrow C1 = A + B$$
$$C0 = 0 \Rightarrow S = H = I0 \oplus I1$$
$$C0 = 1 \Rightarrow S = \overline{H} = \overline{I0 \oplus I1}$$



(a) Layout of the adder cell

Figure 4.2- The Layout and Simulation Results for Adder



(b) Simulation results

Figure 4.2- The Layout and Simulation Results for Adder (cont.)

Therefore, an adder element that realizes the above functions is designed with the standard gate elements of the Ocean database. The layout and the simulation results for this adder is given in Figure 4.2.

Eight bit ALU is constructed by cascading eight "alu" cell. Terminals of the cell are as follows;

Terminals: (IN10, IN11,IN12, IN13, IN14, IN15, IN16, IN17, IN20, IN21, IN22, IN23, IN24, IN25, IN26, IN27,Cin, Cout, S0, S1, S2, S3, S4, S5, S6, S7, Zero, vss, vdd)

where IN1x,IN2x : inputs

Cin : Carry in

Cout : Carry out

Sx : Result

Zero : Zero status bit, active high.

The simulation results for 8 bit adder is shown in Figure 4.3.

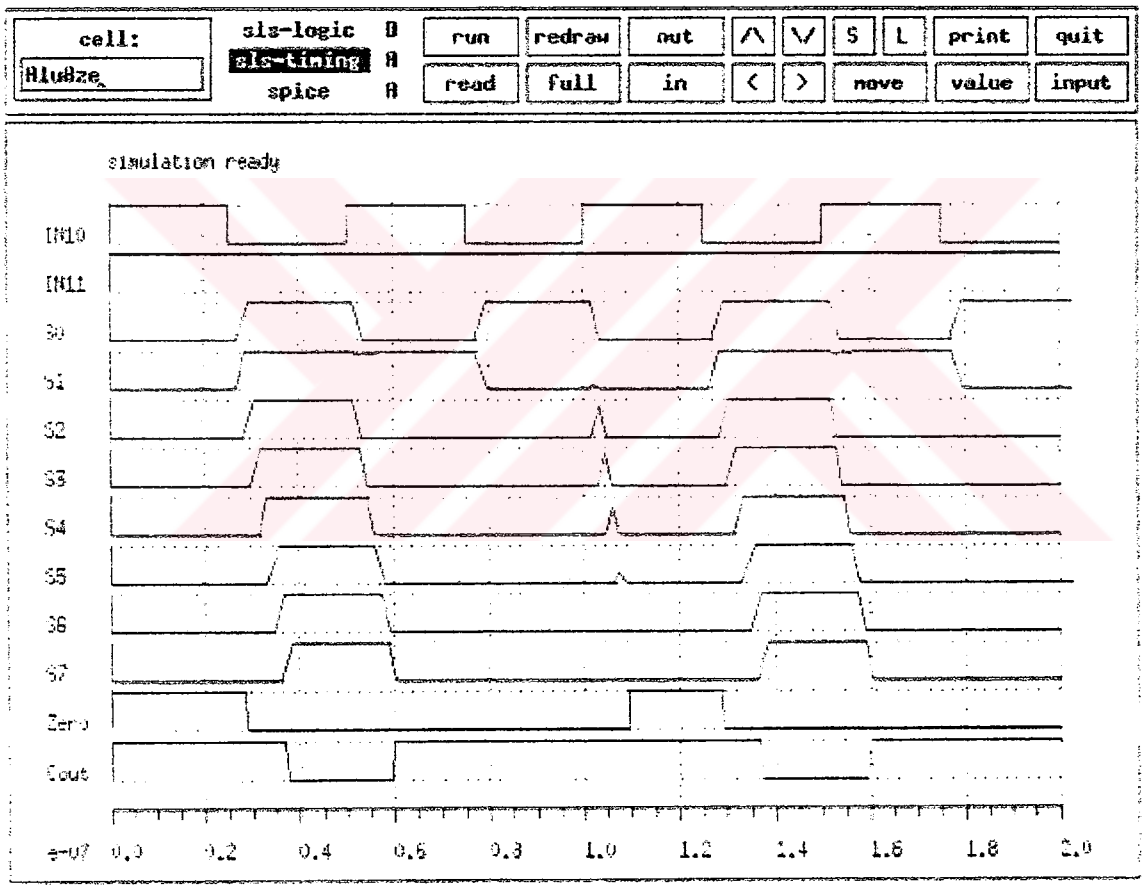


Figure 4.3- The Simulation Results for 8-Bit Adder

If the simulation results are considered, it is seen that the longest delay occurred in the output carry. Therefore, delay of the output carry is measured. When the change rate of the input signals is 30Mhz, delay from the input change to the rising edge of the Cout is 11.6ns (%72 of half period); similarly delay from the input change to the falling edge of the Cout is 12ns (%75 of half period). For 20Mhz and 40Mhz delay times are not different. Thus, this results could be considered as acceptable for 20Mhz. However, some spikes are observed in waveform during the simulation. Figure 4.3 shows the simulation results. Therefore, several place and route operation is performed. But, the results are not changed so much. As a result, an alternative adder element is designed.

The general idea behind the adder is not different, but the realization is modified to have a symmetrical structure for CMOS realization. Consequently, the delays that caused spikes are reduced. The expressions of the sum and output carry are the followings;

Sum $\quad S = \overline{C1}\cdot(I0 + I1 + C0) + I0 \cdot I1 \cdot C0$

Carry $\quad C1 = (I0 + I1)\cdot C0 + I0 \cdot I1$

By arranging these expressions logic functions can be obtained.

$C0 = 0 \Rightarrow C1 = A \bullet B$
$C0 = 1 \Rightarrow C1 = A + B$

The circuit that realizes these functions is shown in Figure 4.4 [15].

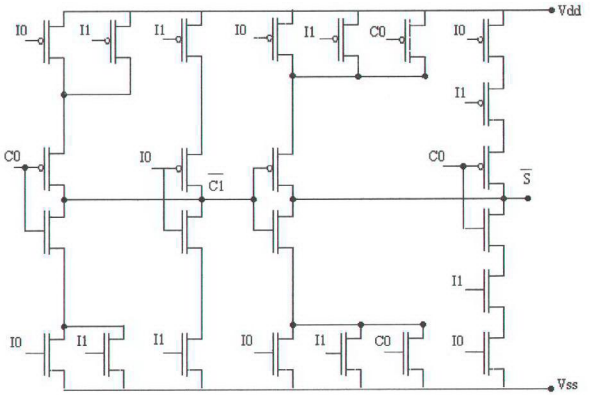


Figure 4.4- One Bit Full Adder Circuit

The layout of this cell is drawn manually in the layout editor Seadali. The layout of the adder cell is shown in Figure 4.5.

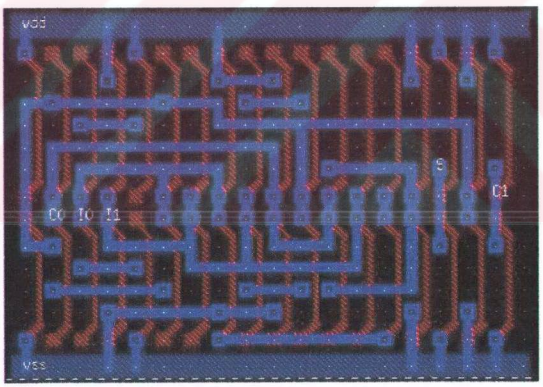
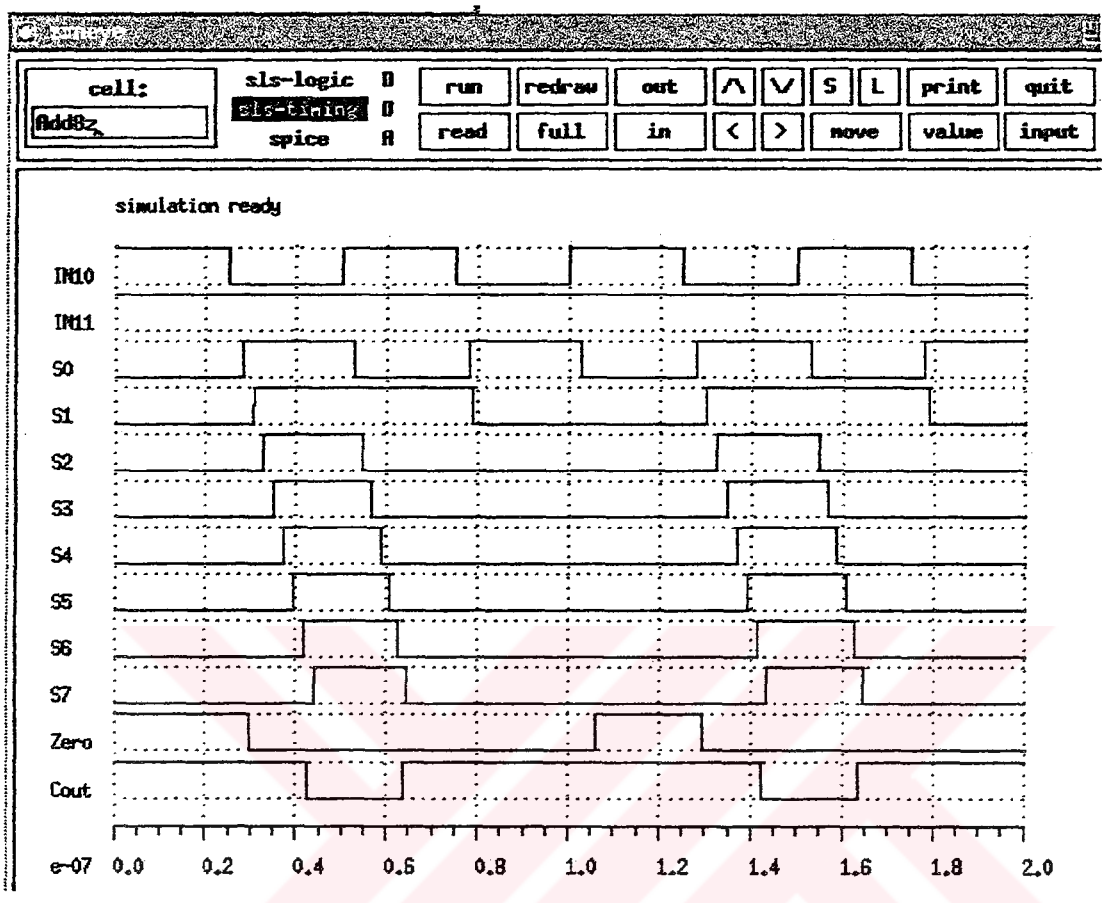


Figure 4.5-The Layout of One Bit Adder Cell

The one-bit adder cells are cascaded to construct the 8-bit adder block. Additionally, sum bits are NOR'ed in order to get Zero status flag. The simulation results are shown in Figure 4.6. If compared with the previous 8 bit adder, this one is faster and has a much clean signal waveform than the other. Therefore, this adder is used in the arithmetic logic unit.




```
IN10 = h*1 l*1 h*1 l*1 h*1 l*1 h*1 l*1
IN11 = h*8
IN12 = h*1 l*1 h*1 l*1 h*1 l*1 h*1 l*1
IN13 = h*8
IN14 = h*8
IN15 = h*8
IN16 = h*8
IN17 = h*8
IN20 = h*8
IN21 = l*2 h*2 l*2 h*2
IN22 = l*1 h*1 l*1 h*1 l*1 h*1 l*1 h*1
IN23 = l*8
IN24 = l*8
IN25 = l*8
IN26 = l*8
IN27 = l*8
Cin = l*8
```

Figure 4. 6 - The Simulation Results for 8-Bit Adder

Since the ALU is a combinational circuit, it is required to have two sets of output latches, one for arithmetic output and one for logic output. The inputs are from A and B registers. When the inputs are changed the output is set accordingly. If the selection bits of the output cells, arithmetic output selection bit (asel) and logic output selection bit (lsel), are active the result is send to output with the rising edge of the clock. Additionally, one D type flip flop is used for zero bit, so the zero flag is refreshed with clock signal. The output carry "Cout" is also latched to a D type flip flop and its output is both sent to control unit for conditional instructions and connected to the input carry with a combinational circuit. Thus, an operation can be implemented with or without carry. When the "carrysel" is high the output carry is sent to input carry. Furthermore carry can be resetted with the active high "cres" signal and can be set with active high "cset" signal. The block diagram of the ALU is shown in Figure 4.7.

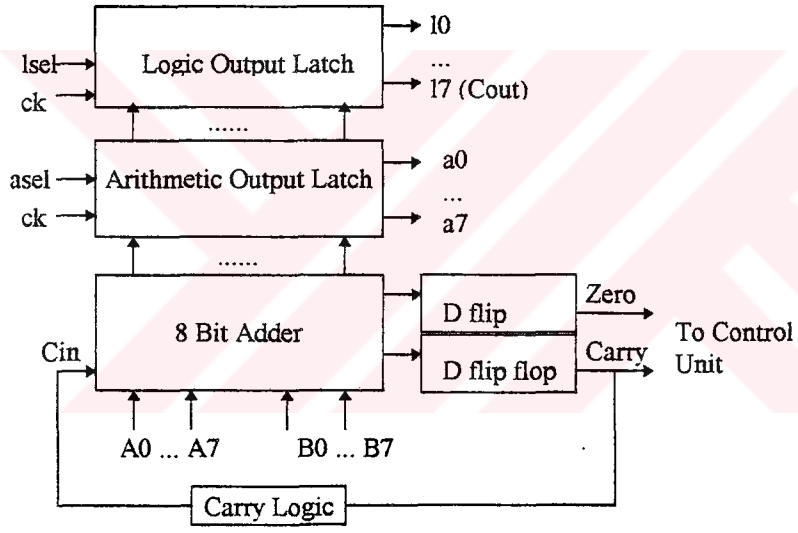


Figure 4.7-Block Diagram of ALU

3.1.2 Data Registers

The processing unit consists of two registers, A and B. This registers are used to hold the operands for ALU operations. The basic element of the registers are memory cells. A register cell is shown in Figure 4.8 [15].

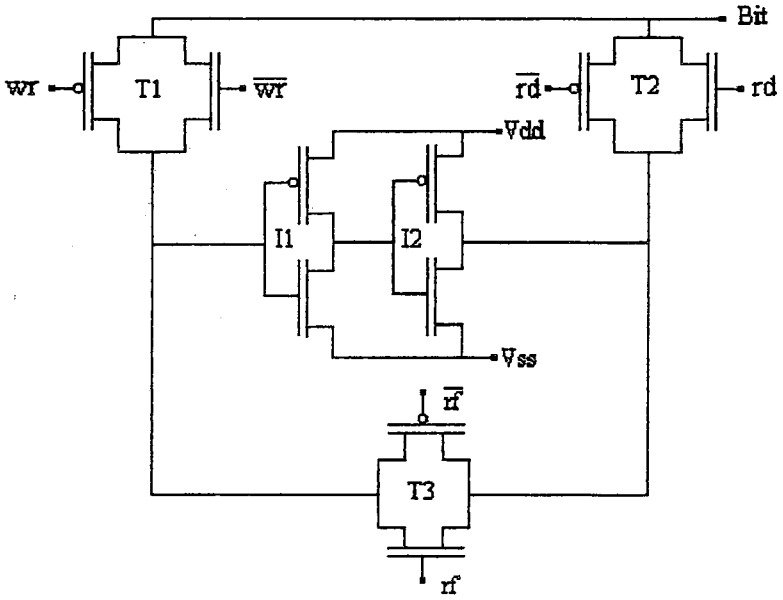


Figure 4.8-One Bit Register Cell

The bit is stored in two inverter stages, I1 and I2. T1, T2 and T3 are transmission gates. The first transmission gate T1 is activated by write signal, wr. When the "wr" is high, the bit is stored in the gate capacitance of the first inverter. This bit is reproduced by complementing the output of the inverter. The cell has a feedback path to refresh the stored bit. The stored bit is read from the cell with "rd" signal. After the cell is designed, it is tested by arranging the circuit in Figure 4.9.

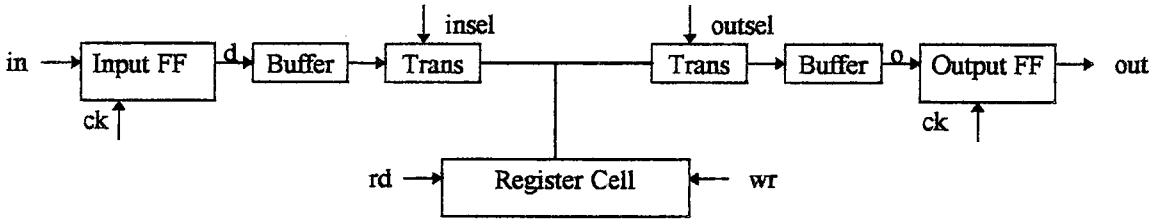


Figure 4.9- The Test Circuit For Register Cell

The test circuit contains two D type flip flops, two buffers, two transmission gates (trans) and a register cell. The input bit is copied to the output of the input FF in the rising edge of the clock. If the input select signal "insel" is active, the buffered signal arrived to the register cell. The "insel" and "wr" signals are activated simultaneously. Thus, the bit is written to the register. When, the "outsel" and "rd" signals are active, the stored bit is read to the output buffer. The simulation results for register cell tests is shown in Figure 4.10.

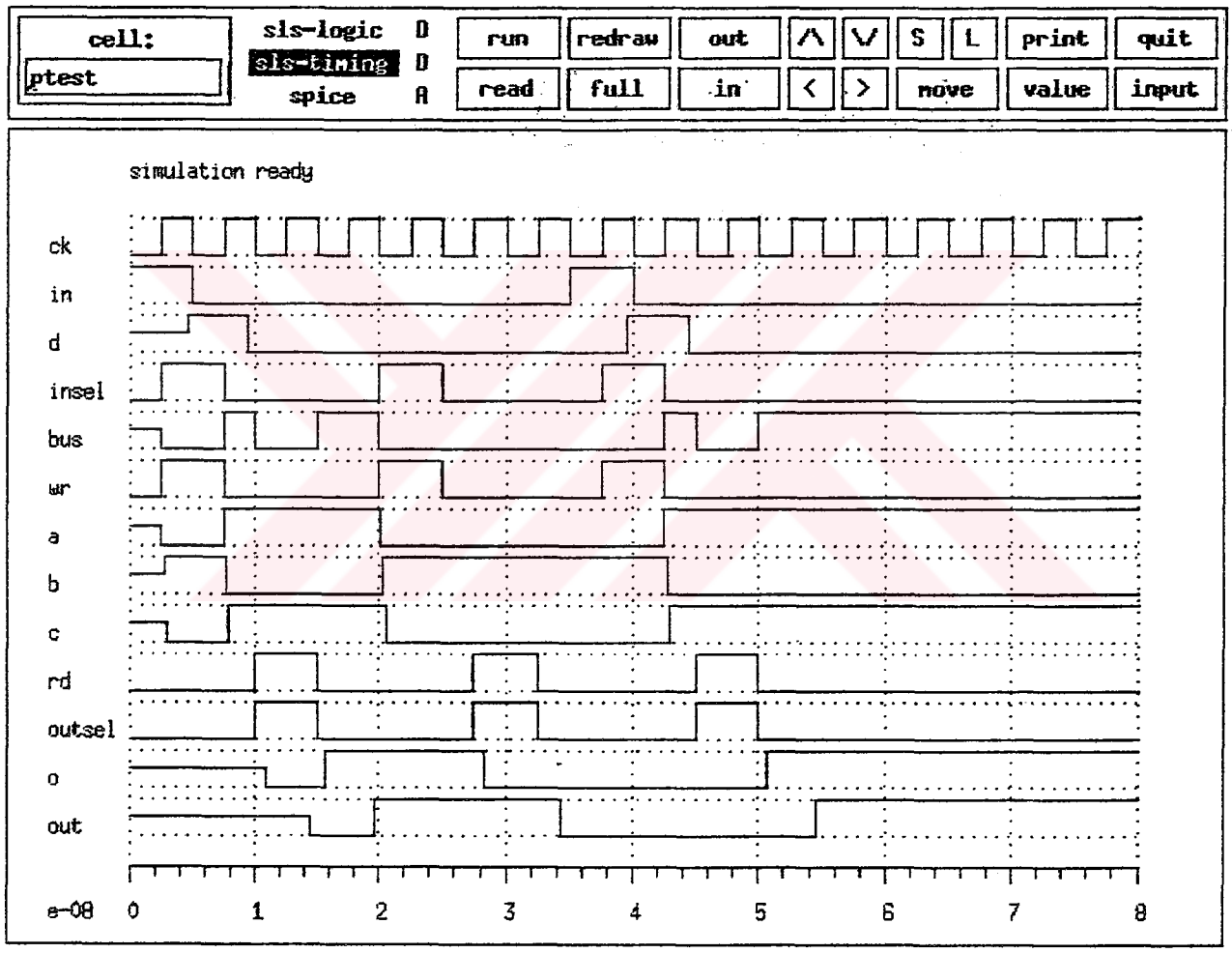


Figure 4.10- Simulation Results For Register Cell Test

The symbols a, b and c stands for respectively input of the first inverter, output of the first inverter and the output of the second inverter inside the register cell. As can be shown

from the simulation results, the following delay times are observed for the clock period of 5ns.

clock to output of input flip flop	:	4.6ns
insel or wr to a	:	5ns
insel to c	:	5.4ns
rd to output of the output ff.	:	9.6ns
data setup time	:	7.5ns

The register are connected to form A and B registers in layout manually in layout editor. The control signals of the registers are "wra", "wrb", "rda" and "rdb".

3.1.3 Shift Register

The shift register should have parallel loading, left and right shift and rotate capability. In order to achieve these goals it is formed by D type flip-flops, multiplexers and transmission gates. A cell of the shift register is shown in Figure 4.11.

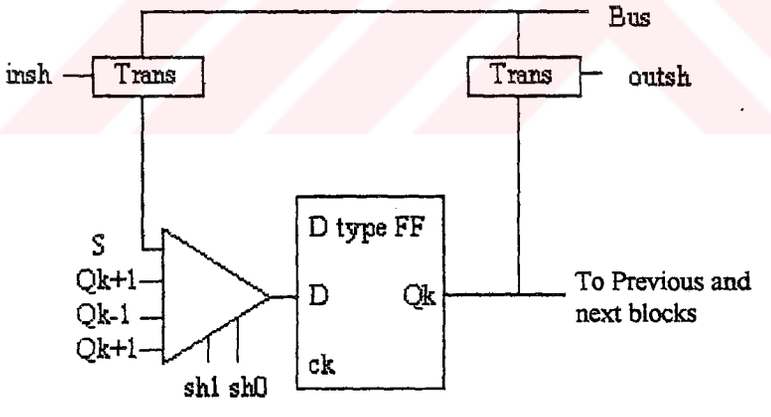


Figure 4.11- Shift Register Cell

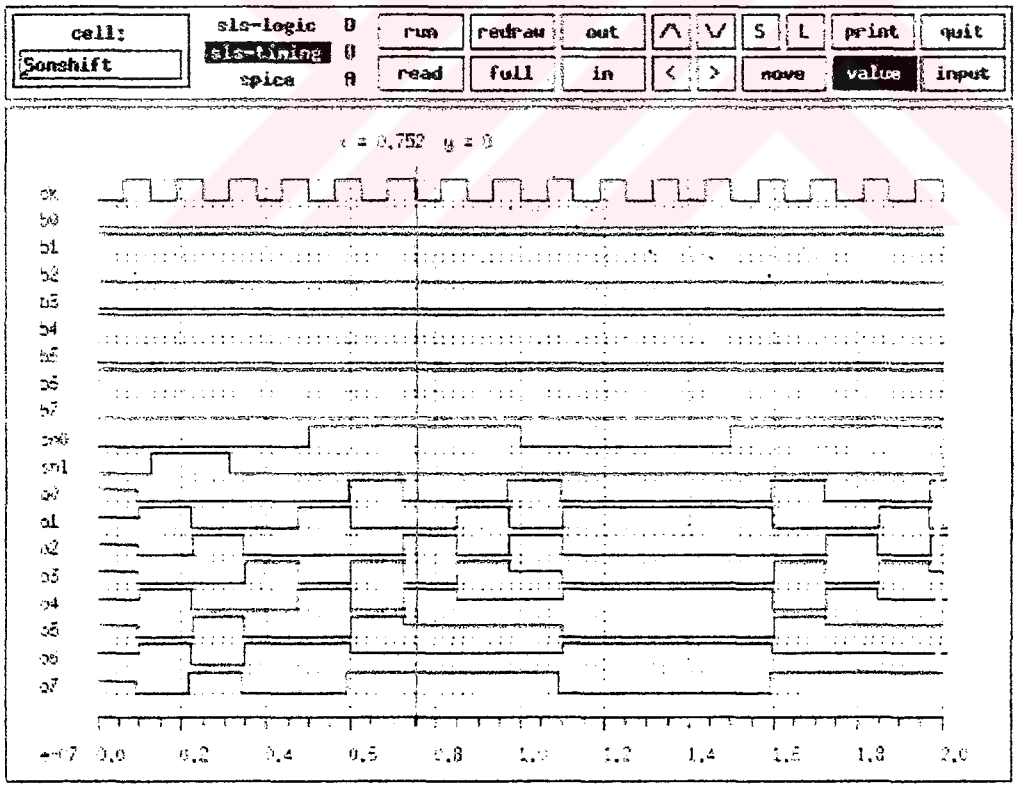
It has four control signals: "sh0", "sh1" , "insh" and "outsh". The first two of them defines the operation of the shifter by selecting the inputs of the multiplexer. "insh" and "outsh" connects the internal data bus to the input and the output respectively. While "Qk" is the output of the cell that is under consideration, the "Qk+1" symbolized the output of the next stage and the "Qk-1" symbolizes the output of the previous stage. Since there is

not a previous output for the first stage, it is pulled down. Similarly, the first "Qk+1" input of the last stage is connected to the ground rail. In other words, the fill in bit is "0" . In addition, the second "Qk+1" input of the last stage is supplied by the output of the first shift register cell to realize rotate right operation. The function table of the 8 bit shift register is shown in Table 4.1.

Table 4.1 - Shift Register Function Table

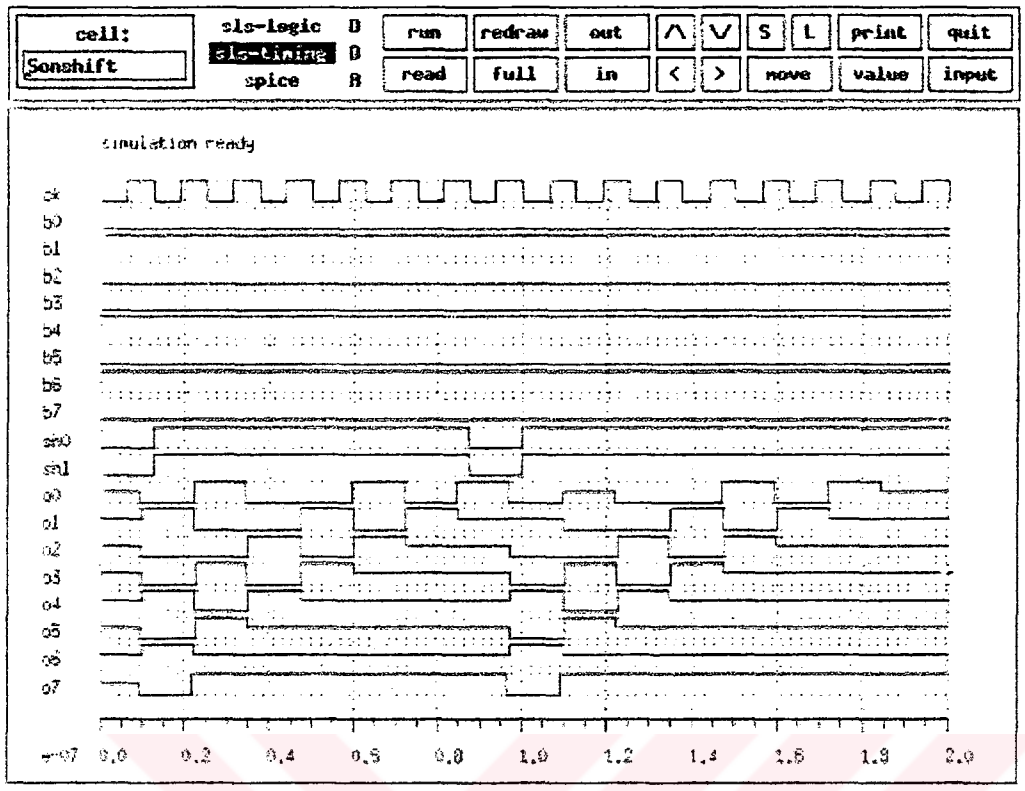
ck	sl1	sl0	Function
↑	L	L	Load
↑	L	H	Shift Right
↑	H	L	Shift Left
↑	H	H	Rotate Right

The simulation results for different operations of the shift register are given in Figure 4.12 .



(a) Shift Left and Right Operation

Figure 4.12 - Shifter Operations



(b) Rotate Right Operation

Figure 4.12 - Shifter Operations (cont.)

3.2 Memory Unit

Memory unit of the microcontroller is a static read write memory array. It is designed for modification of the data bits stored in the memory array and for retrieval on demand during the program flow. Memory unit includes 16x8 RAM matrix, a 4-to-16 decoder and memory address register.

3.2.1 Random Access Memory

The most common static memory cell is used for the random access memory (RAM) array. This cell consists 6 transistors or in other words, cross coupled two inverters and two pass transistors. The schematic of the RAM cell is shown in Figure 4.13 ([9] and [21]).

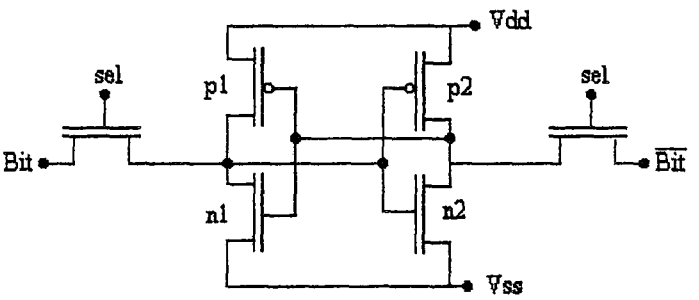


Figure 4.13 - CMOS Static RAM Cell

The NMOS pass transistors are used to access the memory cell during the "write" and "read" operations. These transistors are activated by the select signal "sel". The "Bit" and "$\overline{Bit}$" lines of each memory cell column are pulled up with an NMOS transistor. When the select line is "low" the pass transistors are turned off, thus the data is held. At this point, if the all selects are inactive the large column capacitances Cbit and C$\overline{bit}$ are charged-up by the PMOS pull-up transistors as shown in Figure 4.14. The voltage of the "Bit" and "$\overline{Bit}$" lines are equal during the hold phase. Assuming the "sel" line is activated, the "write" and "read" operations are implemented as the following manner: At the beginning of data-read operation, assuming a logic "0" stored in the cell, the voltage of the "$\overline{Bit}$" line is Vdd while the voltage of the "Bit" line voltage is approximately Vdd. The transistors p1 and n2 are turned off, meanwhile p2 and n1 are in linear operation region. The node voltages V1 =0V and V2= Vdd before the pass transistors are turned on. After the cell is made accessible by turning the pass transistors on, there will be no change in the voltage level of the bit line. However, there would be a current flow through m1 and n1, and the voltage stored in Cbit is going to start to drop. Since the gate capacitance is relatively large, this voltage drop would be a few milivolts. Thus, the data-read circuitry will have detected this drop and amplified it as a "0" level. The V1 voltage level naturally would increase during the read operation. If this voltage exceeded the threshold of n2, it could turn on the transistor. As a result, the stored bit would have been changed. Thus, this point will have carefully been considered.

If

$$V_{1\max} \le V_{t,n2} \qquad \text{[Eq.4.1]}$$

where $V_{t,n2}$ is the threshold voltage of the n2, the stored bit will not be changed during the read operation. In this case the aspect ratio of m1 to n2 will have been as follows

$$\frac{\left(\frac{W}{L}\right)_{m2}}{\left(\frac{W}{L}\right)_{n2}} \le \frac{2\left(V_{dd} - 1.5V_{t,n2}\right)V_{t,n2}}{\left(V_{dd} - 2V_{t,n2}\right)^2} \qquad \text{[Eq.4.2]}$$

In the "1" read operation, the case is similar except the conducting transistors.

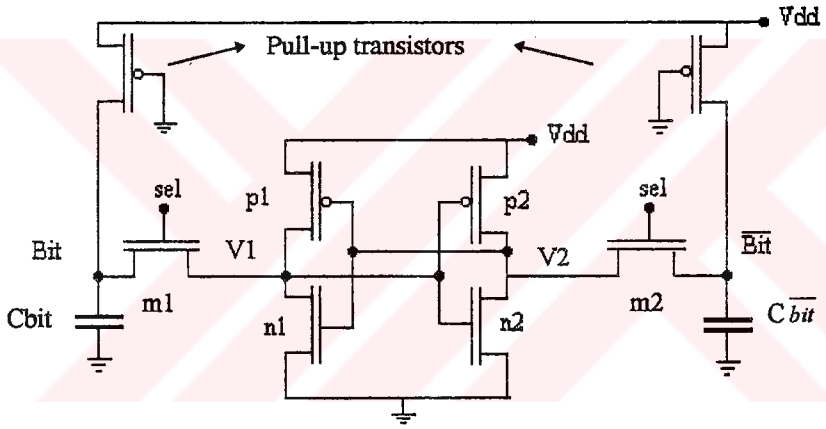


Figure 4.14- The Operation of SRAM Cell

While writing "0" to the memory when the stored bit is "1" the following events occurred: n2 and p1 transistors are turned on, so the voltage levels of the nodes are V1= Vdd and V2=0, before the write operation. "Bit" line is forced to logic "0" level by the data write circuitry. In order to change the stored information n2 should be turned off by reducing the gate voltage below its threshold. Then, n1 is turned on.

By keeping the operation principles in mind, a data write circuitry should have been a circuit that forces the "Bit" or "$\overline{Bit}$" line to the voltage level of ground. That could be accomplished by two pass transistors that could be activated by data write circuit.

A data read circuitry should sense the small voltage difference between the "Bit" and "$\overline{Bit}$" line. A suitable circuit is shown in Figure 4.15. This circuit is called as current mirror differential sense amplifier [9].

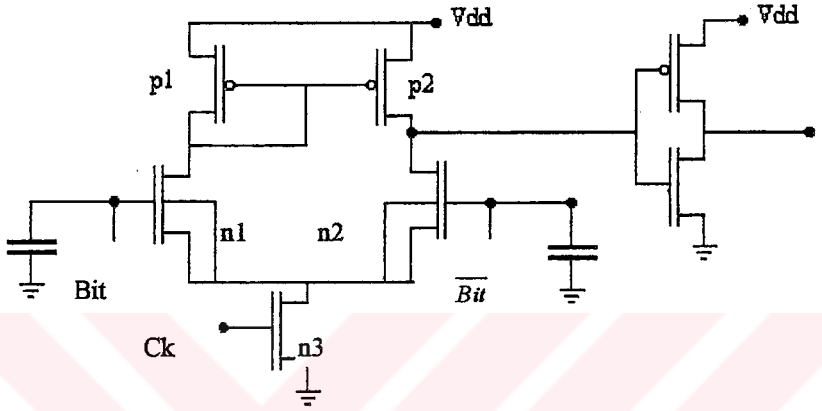


Figure 4.15 - Current Mirror Differential Sense Amplifier

When the memory cell is selected, the voltage of the "Bit" or "$\overline{Bit}$" line will drop slightly, meanwhile the n3 will be turned on by the Ck signal. If the stored data forces the "Bit" line to decrease slightly and n1 will turn off. Therefore, the output of the differential amplifier will drop immediately. The output of the inverter will be logic "1". Similarly, when the stored bit is "0", n2 will turn off and the output of the inverter will not change from Vdd by keeping the output of the inverter in "0" logic level.

As explained before, the aspect ratio of the transistors in memory cell configuration should have a certain value. But the gate width and lenghts of all n type transistors of Fishbone image are the same. Thus, the designed memory cell and read-write circuit did not work. The simulation results for this cell is in Figure 4.16. Instead of this design, the configuration that is designed for data registers is used for memory unit.

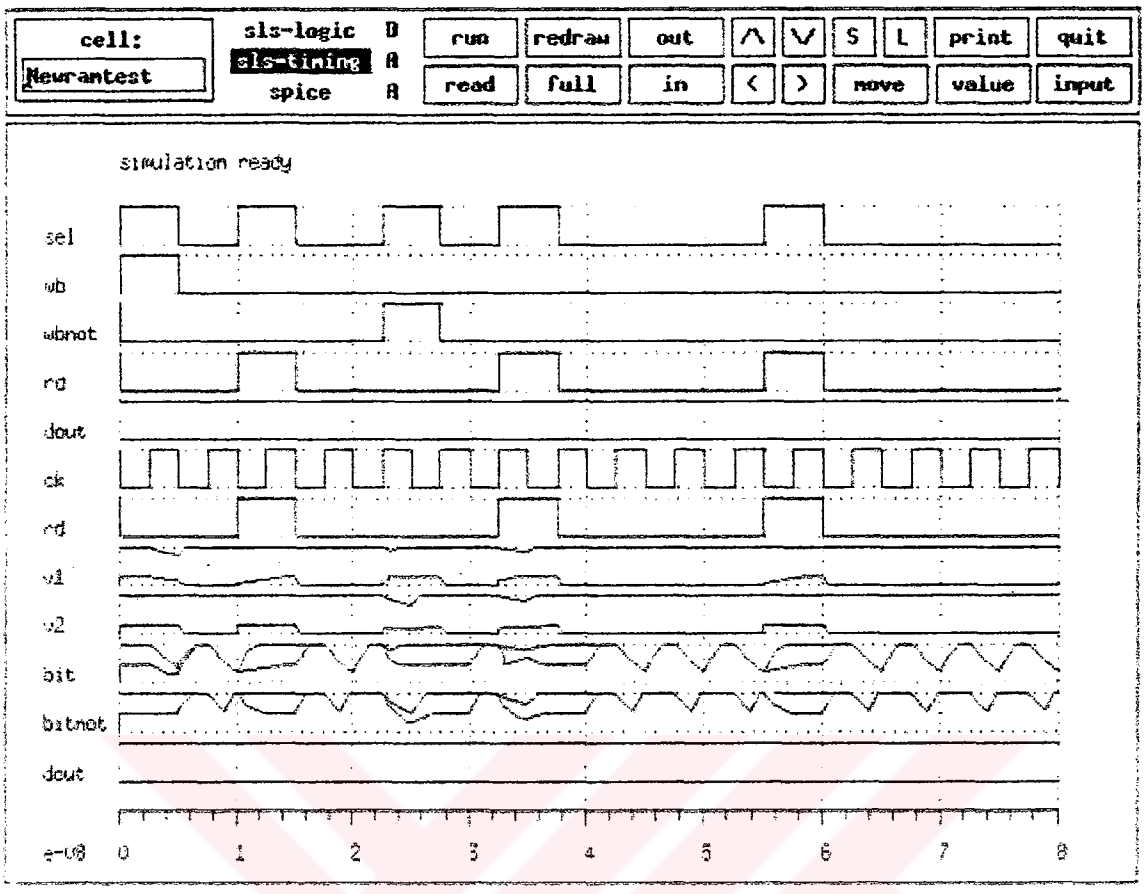


Figure 4.16- The Simulation Results for Cross-Coupled Ram Cell

After the layout of one bit memory cell is designed, RAM cells are connected to form an memory array of 16x8 bits. The resultant RAM matrix has a clock input, 16 row select, data, write and read signals.

3.2.2 Memory Decoder

In order to select any one of the 8 bit row of the memory, a row decoder is needed. Thus, a 4 to 16 decoder is designed. The decoder has sixteen active high outputs. It is designed by using the standard gate elements of the Ocean database. The layout and the simulation results are shown in Figure 4.17.

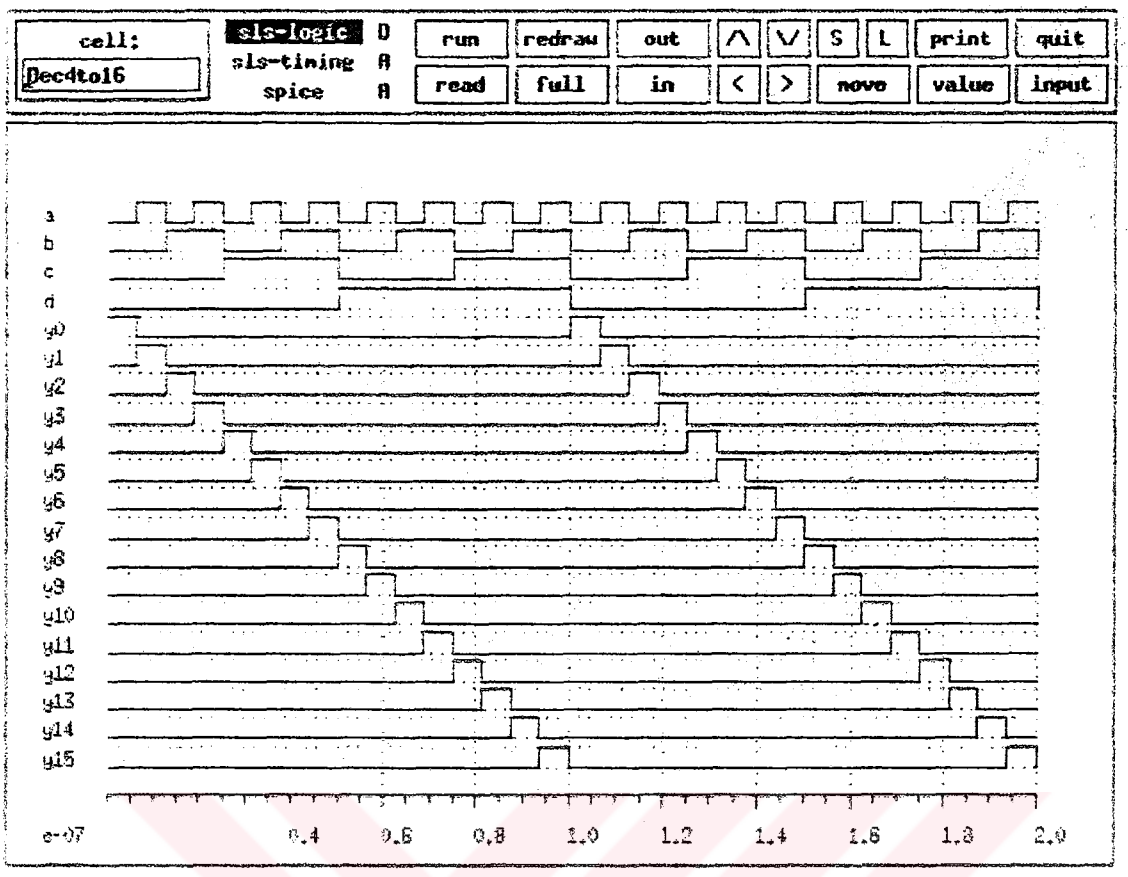


Figure 4.17- The Simulation Results of Memory Decoder

3.2.3 Memory Address Register

The address of the random access memory is a part of the instructions. Therefore, the address is read from the external program memory and it is sent to the memory unit through the internal data bus. During the read and write operations, the address o the RAM should have been stable. Thus, a memory address register should have taken the address from the bus and keeps the bits during the access to the RAM. It is actually a gated latch and called as MAR. D type flip flops and transmission gates are used to design MAR. When the active low "marsel" signal is high, the address is transferred to the output of the MAR and it is hold until next read or write operation. The layout of MAR is shown in Figure 4.18.

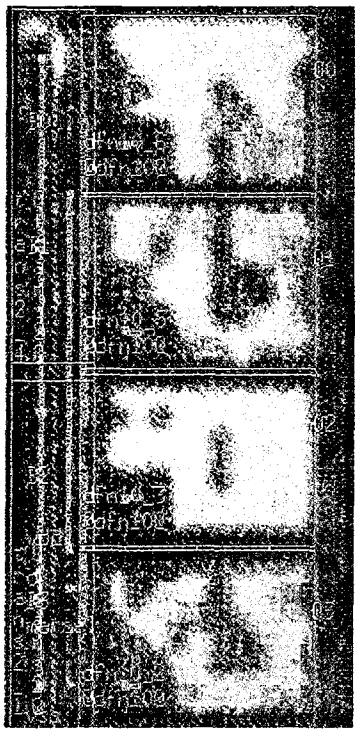


Figure 4.18-The Layout of Memory Address Register

The block diagram of the memory unit is shown in Figure 4.19.

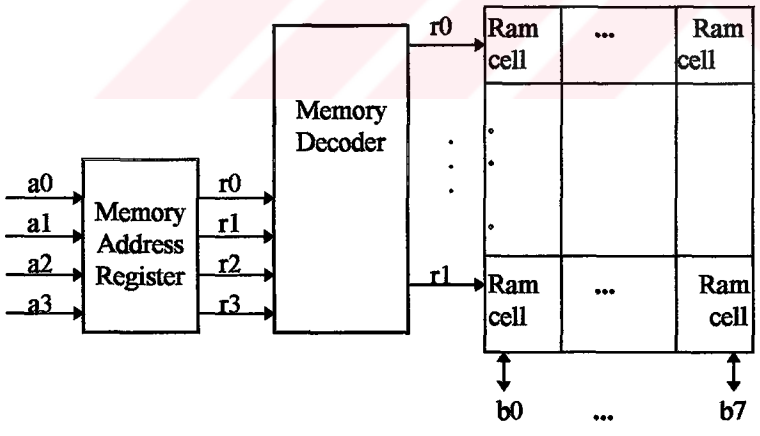


Figure 4.19 -The Block Diagram of The Memory Unit

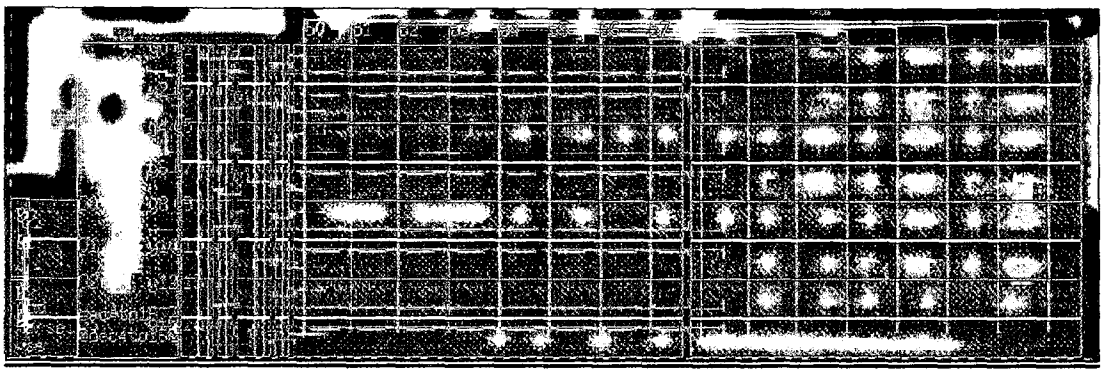


Figure 4.20- The Layout of Memory Unit

3.3 Input/ Output Unit

The program that directs the operation of the microcontroller is fetched from the external program memory. Furthermore, the operands could be taken from the external memory. Furthermore, data may be sent to the other devices with output instructions. The Input/Output Unit of the microcontroller is used to provide an interface between the external world and the chip. This unit includes the data input and output latches and the port structure.

3.3.1 Data Input and Output Latches

The data input and output latches are used to store the data that is transferred between the external data bus and the internal bus. Data output latch, Dout, takes its data from the internal data bus and holds it during the output operation. Similarly, the data input latch, Din, holds the data that it is taken from the external bus. The structure of these two latches are same. They are constructed of transmission gates and D type flip flops. The latches transfer their inputs to the outputs in the rising edge of the clock. The control variables that controls the transfer operation are the "dinsel" and the "doutsel" for Din and Dout, respectively. The layout of Dout is shown in Figure 4.21.

Figure 4.21- The Layout of Dout

3.3.2 Input/ Output Ports

While designing an integrated circuit, the interface between the internal circuitry and the external components are important. Thus, the off-chip driver circuits and the input circuits are investigated carefully. The data port of the designed microcontroller is bidirectional that means it can take data from outside of the IC as well as it can send its data to the external world. This provided by the following bidirectional port circuitry of Figure 4.22 as introduced in [21].

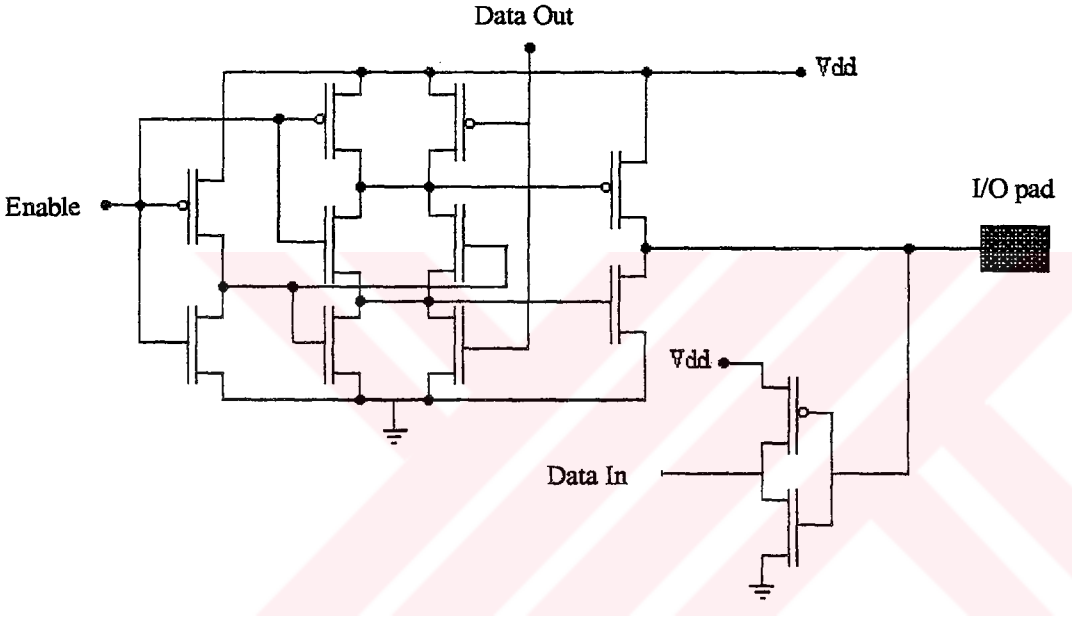


Figure 4.22- Bidirectional I/O Port

When the "Enable" signal is high, the output circuit sends the "Data Out" to the I/O pad by turning on one of the output transistors. Otherwise, both of the output transistors are in the cut-off, thus the output of the circuit is in High-Z state. Input from the I/O pad is provided by an inverter circuit. The layout of the I/O port cell is shown in Figure 4.23.

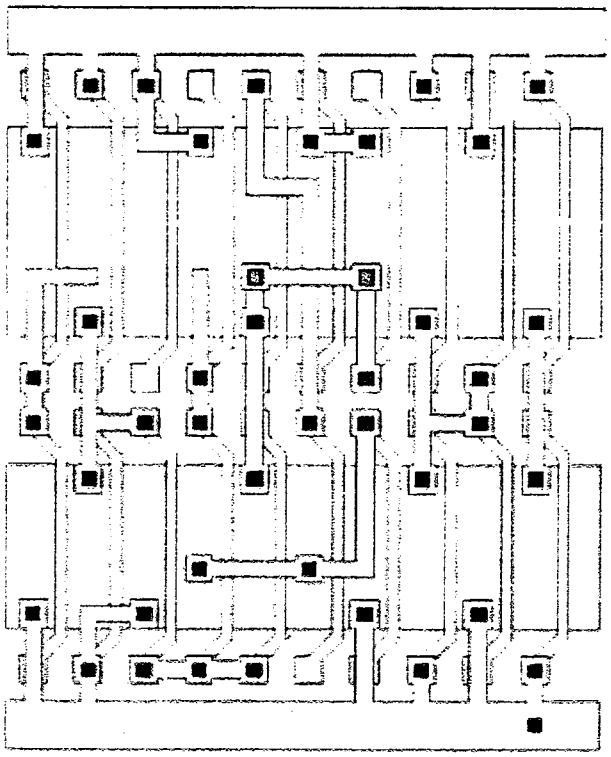


Figure 4.23- The Layout of The I/O Port Cell

3.4 Control Unit

The control unit is the unit that controls all the operations that are performed by the microcontroller. The control method that is used by the control unit is microprogram method. In this method, a memory called control memory holds the micro instructions which would produce the control signals for the microoperations [11]. Microoperations are the operations that are achieved by the subunits of the microcontroller in one machine cycle.

Basically, there are three elements of the control unit. These are the control address register, control memory and the program counter. The block diagram of the control unit is shown in Figure 4.24.

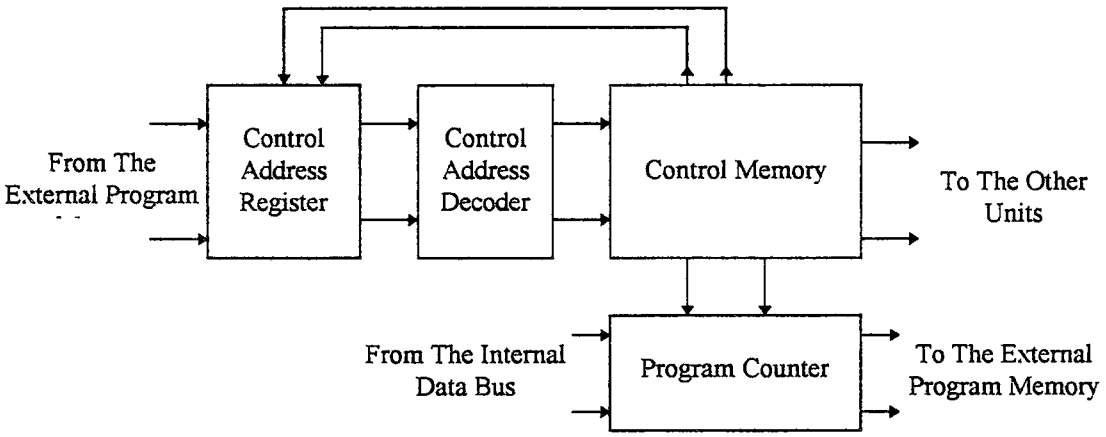


Figure 4.24- The Block Diagram of The Control Unit

The opcodes are read from the external program memory and these opcode entered to the Control Address Register. The Control Address Decoder that can be considered as a part of Control Memory. Finally, the control signals are distributed to the each unit of the microcontroller. The feedback path from the Control Memory to the Control Address Register defines the next operation of the Control Unit.

3.4.1 Control Address Register

The addresses of the microoperations that will be performed by the microcontroller is determined by the Control Address Register (CAR). The CAR is an 8 bit counter with parallel loading ability. Since the combinational circuit of the 8-bit counter is too complex, 4-bit counters are designed and then two of them cascaded to have an 16-bit counter. The four bit counter consists of D type flip flops and logic gates. The logic expressions of the four D type flip flops are below:

$$D0 = \overline{Q0}$$

$$D1 = Q0 \cdot \overline{Q1} + \overline{Q0} \cdot Q1$$

$$D2 = \overline{Q1} \cdot Q2 + \overline{Q0} \cdot Q1 \cdot ; Q2 + Q0 \cdot Q1 \cdot \overline{Q2}$$

$$D3 = \overline{Q0} \cdot Q1 \cdot Q3 + \overline{Q2} \cdot Q3 + \overline{Q1} \cdot Q3 + Q0 \cdot Q1 \cdot Q2 \cdot \overline{Q3}$$

$$Cout = Q0 \cdot Q1 \cdot Q2 \cdot Q3$$

where D0 symbolizes the least significant bit.

After the design of the circuit that realizes these expressions is designed, a control logic part is added to the circuit. This part executes the selected functions. The block diagram of the counter cell is shown in Figure 4.25.

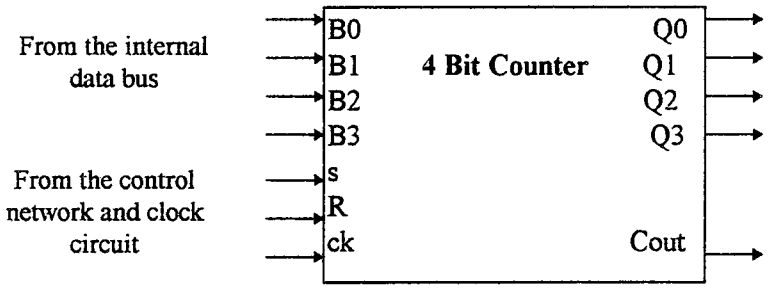


Figure 4.25- The Block Diagram of The 4-bit Counter

The selection signal "s" is used to select the function of the counter. For s=0, the counter is loaded from the internal data bus. If s is equal to "1", increment operation is performed. The reset signal, "R" is activated in the rising edge of the clock, the high value resets the counter.

The CAR is constructed using two 4-bit counters. Additionally, some combinational logic circuits are used to cope with the timing problems. The block diagram of the control address register is shown in Figure 4.26. The function table of the CAR is shown in Table 4.2.

Table 4.2- The Function Table of The Control Address Register

ck	Rcar	Cars	Function
↑	H	X	Reset
↑	L	H	Increment
↑	L	L	Load

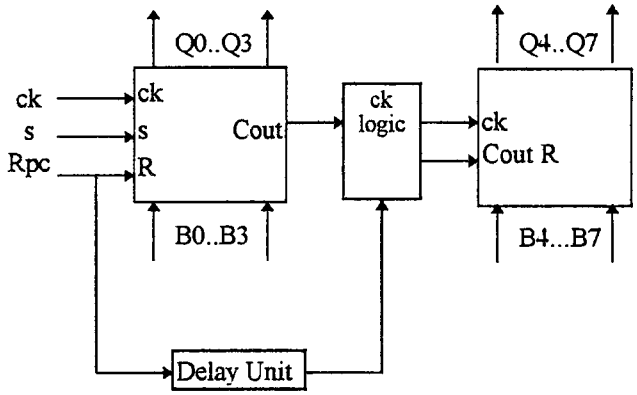


Figure 4.26- The Block Diagram of The CAR

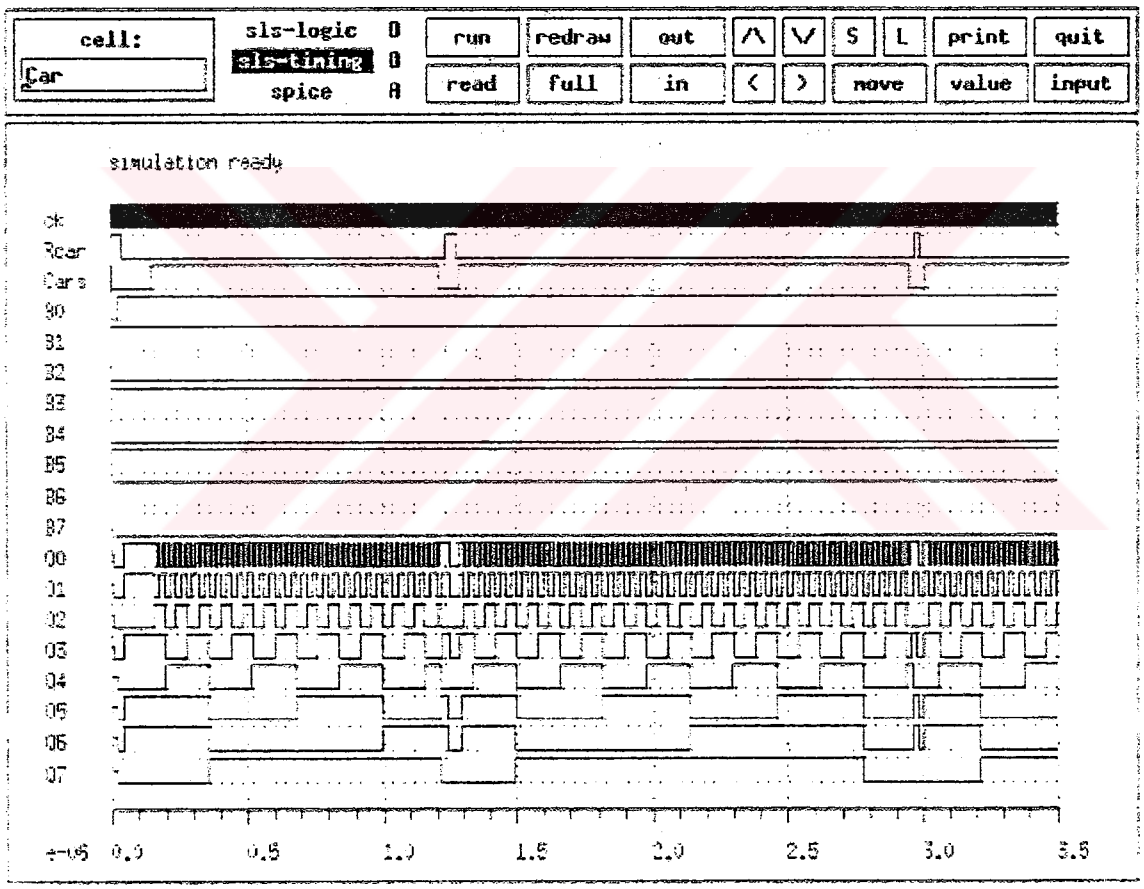


Figure 4.27- The Simulation Results for The Control Address Register

Figure 4.27 shows the simulation results for the Control Address Register. The 350 periods of simulation are performed and three different operation of CAR is demonstrated.

3.4.2 Control Memory

The Control Memory is a Read Only Memory (ROM) matrix. It could be considered a network which produces a specified output for each input combination. An example of a ROM array is shown in Figure 4.28 [9], [18].

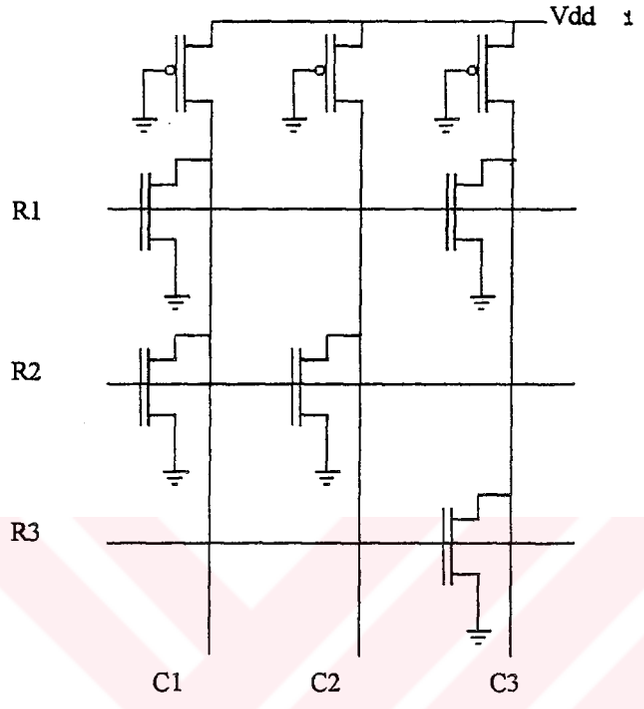


Figure 4.28- An 3x3 ROM Array

In this configuration, when one of the row select bits R1, R2, R3 is activated, the column bits will have a value depending on there is a path between each of the column and the selected row is present or absent. The important point is that only one of the rows can be activated at a time. The truth table of the array is given in Table 4.3.

Table 4.3 - The Truth Table of 3x3 ROM Array

R1	R2	R3	C1	C2	C3
1	0	0	0	1	0
0	1	0	0	0	1
0	0	1	1	1	0

The microcontroller has 25 control variables, thus the ROM array for the control memory consists of 25 columns. The number of the rows are defined by the total number of the microoperations. The ROM array is designed using manually designed standard cells for "low" and "high" logic levels.

In order to access the selected row of the memory array the output of the Control Address Register must be decoded. This task is performed by the Control Address Decoder. The decoder is constructed of six 4-to-16 decoders. One of the decoders is used to enable the other five decoders. The least significant outputs of the CAR are connected to the five decoders, and the others are entered to the sixth one. Thus, this configuration permits to address larger spaces. The outputs of the decoder are connected to the rows of the control memory matrix and the columns are connected to control inputs of the microcontroller units.

3.4.3 Program Counter

The task of the program counter is to point the address of the next instruction during the program flow. The program that would be followed by the microcontroller is taken from the external program memory. Therefore, the program counter (PC) addresses the external program memory. It is an 16 bit counter that is constructed by D type flip flops and gates. It has reset and parallel loading capability. These operations are used for reset and branch instructions.

In order to have an 16 bit counter, the four bit counter of which design steps has explained in section 3.4.1 are cascaded by connecting the Cout signal of the previous block to the clock input of the clock signal. However, because of the some timing problems caused by the combinational circuits, the connection of counter blocks are made in the following way.

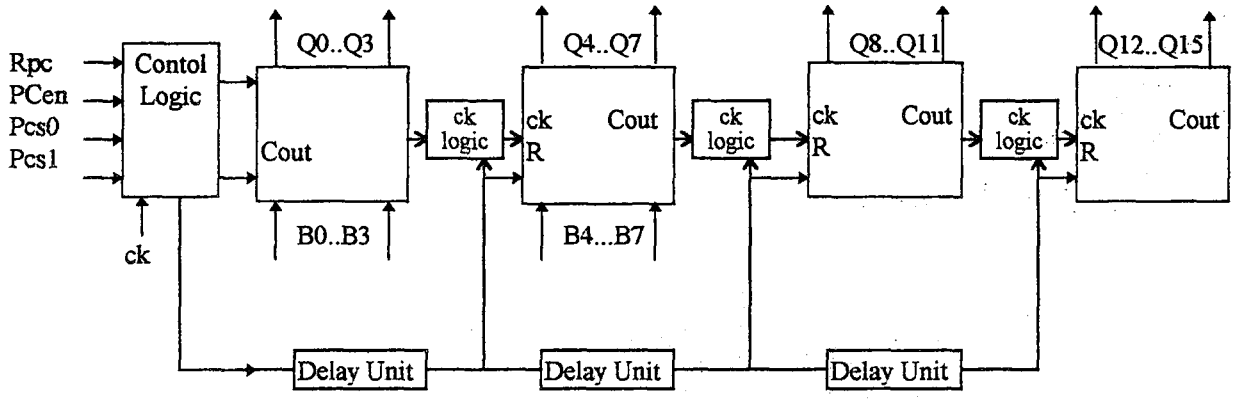


Figure 4.29- The Block Diagram of Program Counter

Since the internal data bus is 8 bits wide, the branch address is limited to 2^8 bytes wide pages. Thus, only the least significant 8 bits of the address is taken from the data bus. The clock logic includes a two input multiplexer that is controlled by the reset signal. The input signals are common clock and the carry out of the previous block. When the reset signal is active, the common clock is applied to all blocks. Consequently, the counter is reset at the same instance.

The Program Counter has 6 types of operations. The four of these are reset, increment, load and stop. The last two of them depend on the status flags carry and zero. The function table are shown in Table 4.4.

Table 4.4 - Program Counter Function Table

ck	Rpc	Pcen	Pcs1	Pcs0	Function
↑	0	0	0	0	Stop
↑	1	0	0	0	Reset
↑	0	1	0	0	Load
↑	0	1	0	1	Increment
↑	0	1	1	0	Depends on Zero
↑	0	1	1	1	Depends on Carry

The simulation results for the program counter is shown in Figure 4.31. Since, the completion of 16 bit count operation requires at least 64000 period (128000 simulation periods), the least significant bits and ck could not be seen clearly.

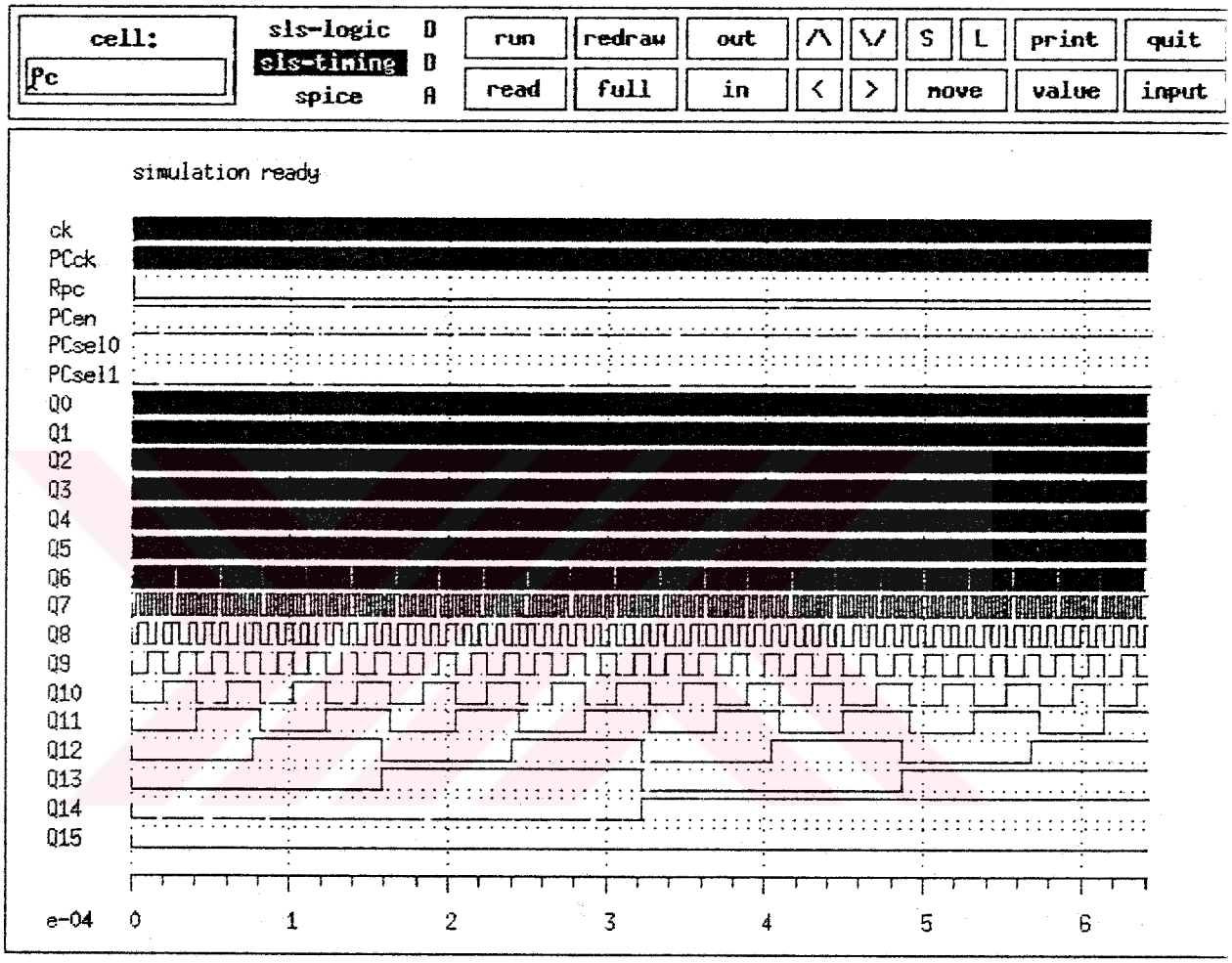


Figure 4.30- The Simulation Results For The Program Counter

4- Overall System Design

4.1- The Clock Distribution and Floorplan

An important aspect in integrated circuit design is to prepare the floorplan and distributing clock signal without causing clock skew and hazards. There are numerous approaches for clock distributions inside a VLSI chip [3], [21], [16]. After the structure of the microcontoller and avaliable clock schemes are considered, the distribution shown in Figure 4.31 is decided.

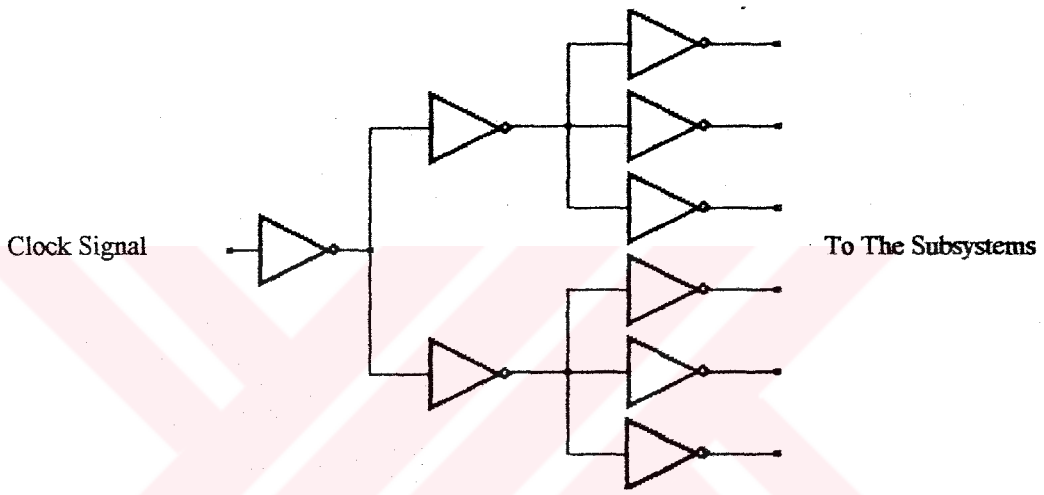


Figure 4.31- The Clock Distribution Tree

The clock distribution tree provides to reach the clock signals to each subunit with approximately same delay. Because, along the each path among the units there is same number of components. Additionally, the clock signal can drive the units since the distribution tree consists of buffers. The clock generator of the microcontroller is external. Thus, the received signal from the clock pin of the microcontroller is connected to the subunits.

The floorplan of the microcontroller is shown in Figure 4.32. In order to drive internal data bus, input and output of each subunit is buffered. By isolating each unit from the other, desired signal levels are obtained.

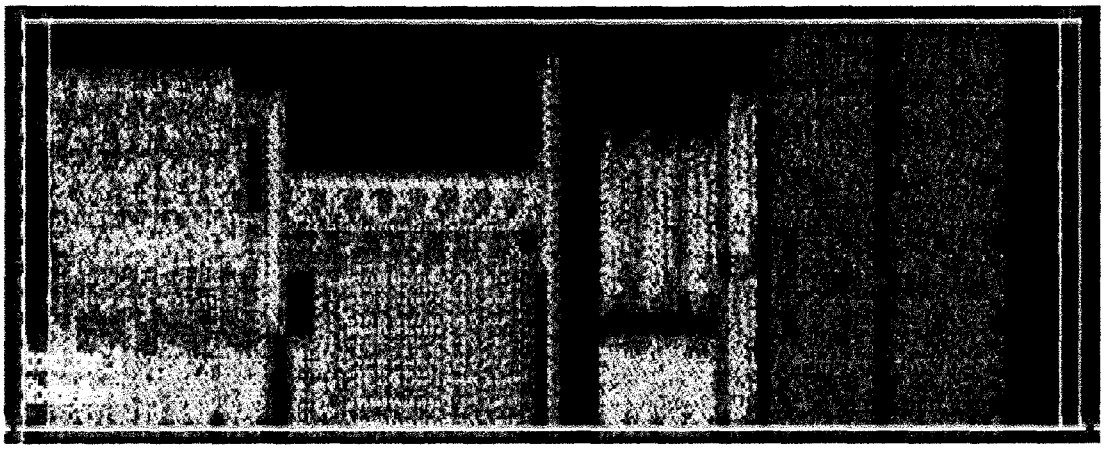


Figure 4.32- The Floorplan of the Microcontroller

4.2- Implementation of Instructions

The implementation of the instructions of the microcontroller are made in two steps. These are "fetch" and "execution". In the "fetch", the opcode of the instruction is taken from the external program memory and it is decoded to find the microoperations in the contol memory. This step is same for all instructions. The other microoperations which the microcontroller will perform for the instruction are implemented in execution phase. The execution part of each instruction is different from the other. Therefore, the microoperations of the "fetch" stage are placed in the first addresses of the control memory. After the execution of any instruction, the fetch phase can restart by resetting the control address register.

The opcode of the instruction defines the address of the related microinstructions in the control memory. Thus, at the end of the fetch operation the opcode of the running instruction is transferred to the control address register. Then, the control signals are produced from the control memory and the sequence of the microoperations are started. Until the end of the instruction, the control addres register are incremented by one in order to point the next microoperation in the sequence. The list of the microoperations are in Appendix C.

4.3- The Performance of The Microcontroller

After the subunits of the microcontroller are placed in the floorplan, the complete system are simulated to obtain the performance of the microcontroller. For the simulation, the control signal are generated in the input file of the simulator.

The first sample program is below. In this program the transfer instructions are implemented and the results are sent to the output pins by output instructions.

Mnemonic	**Comment**
YUK A, 46H	Loads 46H to A register
YUK B, B9H	Loads B9H to B register
CIK A	Outputs the content of A register
CIK B	Outputs the content of B register
DUR	Ends the program

The simulation results for the program is shown in Figure 4.33.

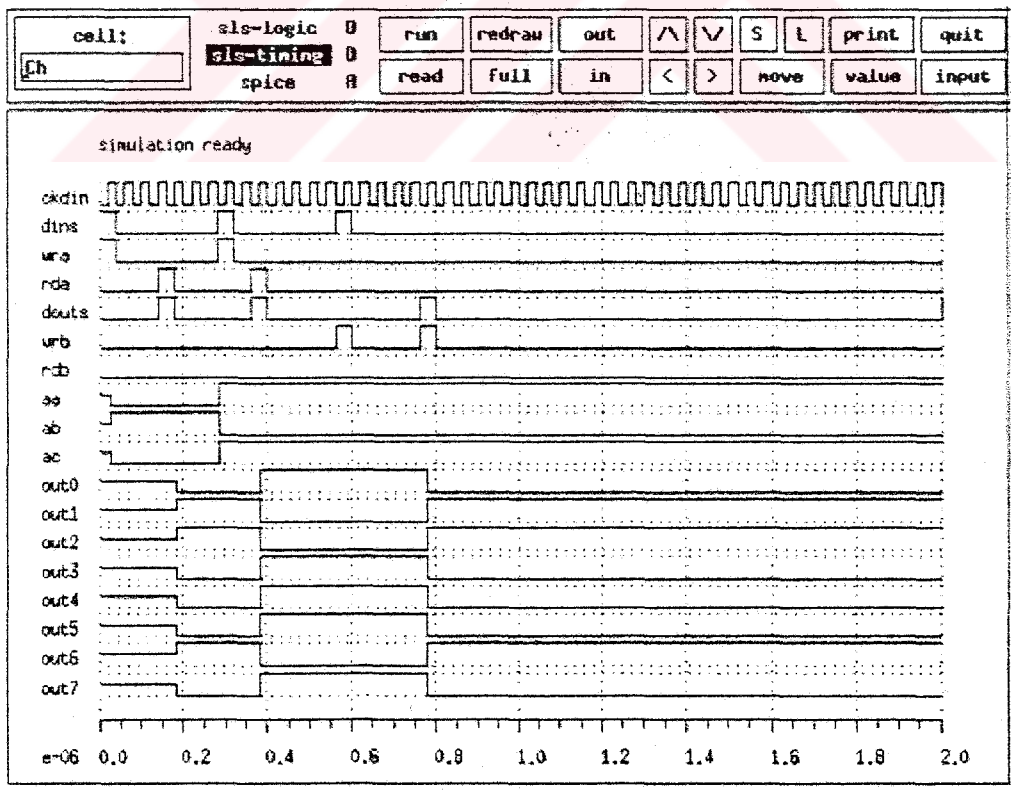


Figure 4.33- The Simulation Results for Transfer Instructions

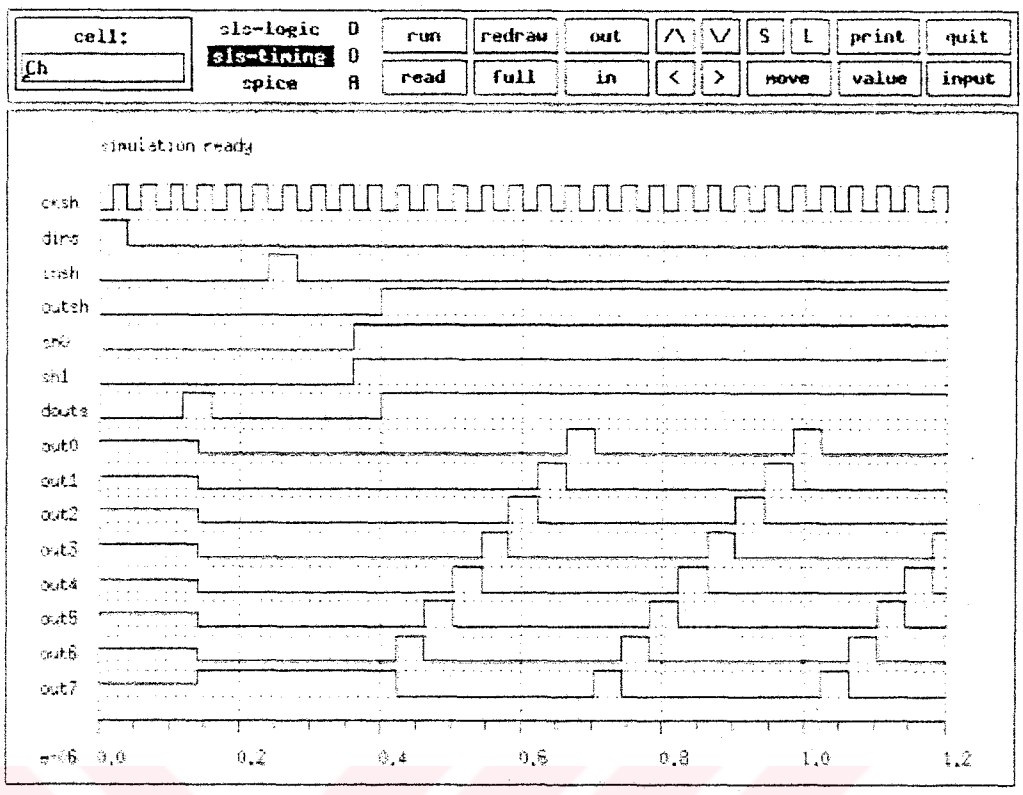


Figure 4.34- The Simulation Results for "DND A" Instruction

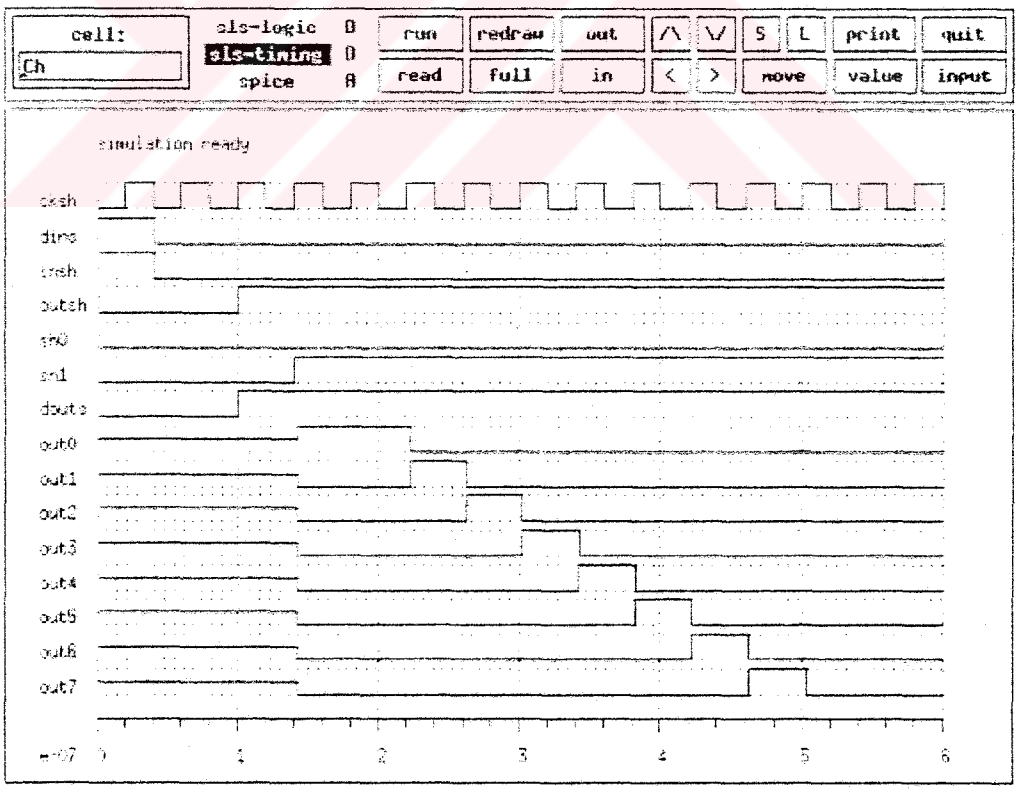


Figure 4.35- The Simulation Results for "SLK B" Instruction

The functions of the shifter are also experimented. In the Figure 4.34, 80H is loaded to A register and sequential rotate right operations are performed and 01H is loaded to B register and shift left operations are performed eight times in Figure 4.35.

The next program is about arithmetic operations. The program is given below.

Mnemonic	**Comment**
YUK A, 00H	Loads 00H to A register
YUK B, FFH	Loads FFH to B register
ETP A, B	Add the content of A to B with carry
YUK A, 22H	Loads 22H to A register
YUK B, FFH	Loads FFH to B register
TOP A, B	Add the content of A to B without carry
DUR	Ends the program

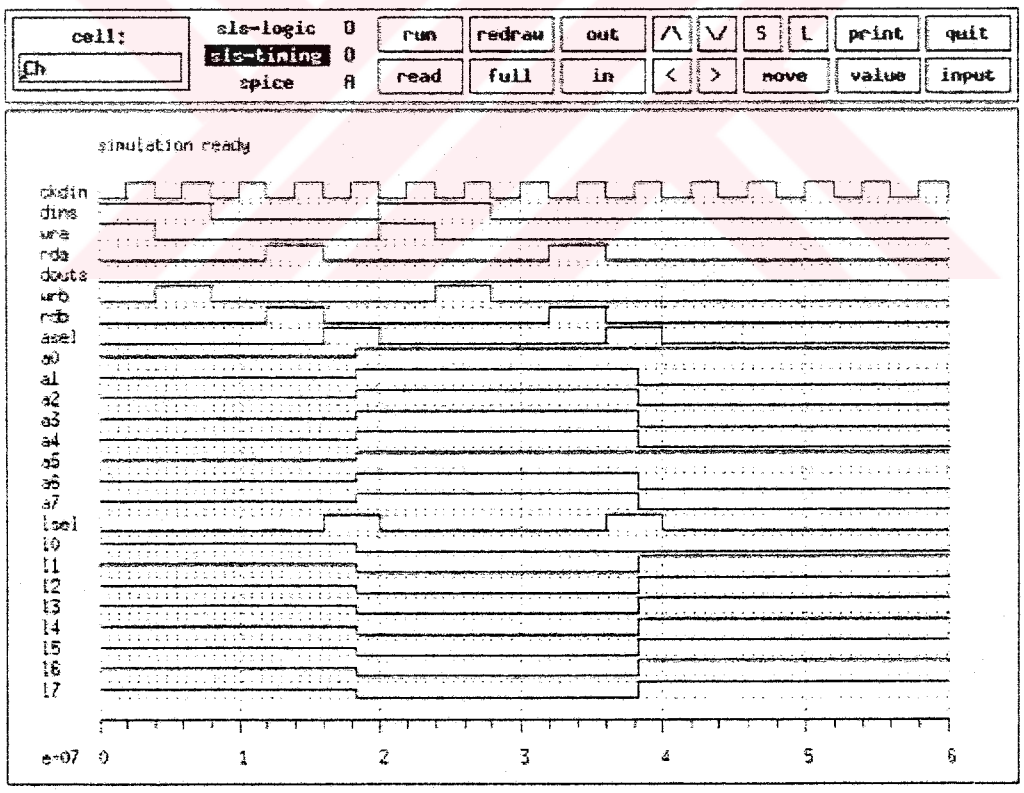


Figure 4.36- The Simulation Results for Arithmetic Operations

The last group of instructions that are considered are branch instructions. These instructions are performed by the program counter. Therefore, the output of the program counter for unconditional and conditional branch instructions are shown in Figure 4.37. First, reset operation is implemented. Then, after 8 periods of increment operation, an unconditional branch instruction to 01FFH is performed. Following the 4 periods of "no operation state" a zero conditional branch instruction is implemented when zero flag is equal to "0". Thus, increment operation is performed, the address is 0100H. Then, a second zero conditional branch instruction is implemented when the condition is fulfilled. As a result, branch occurs to 01FFH address (Figure 4.37).

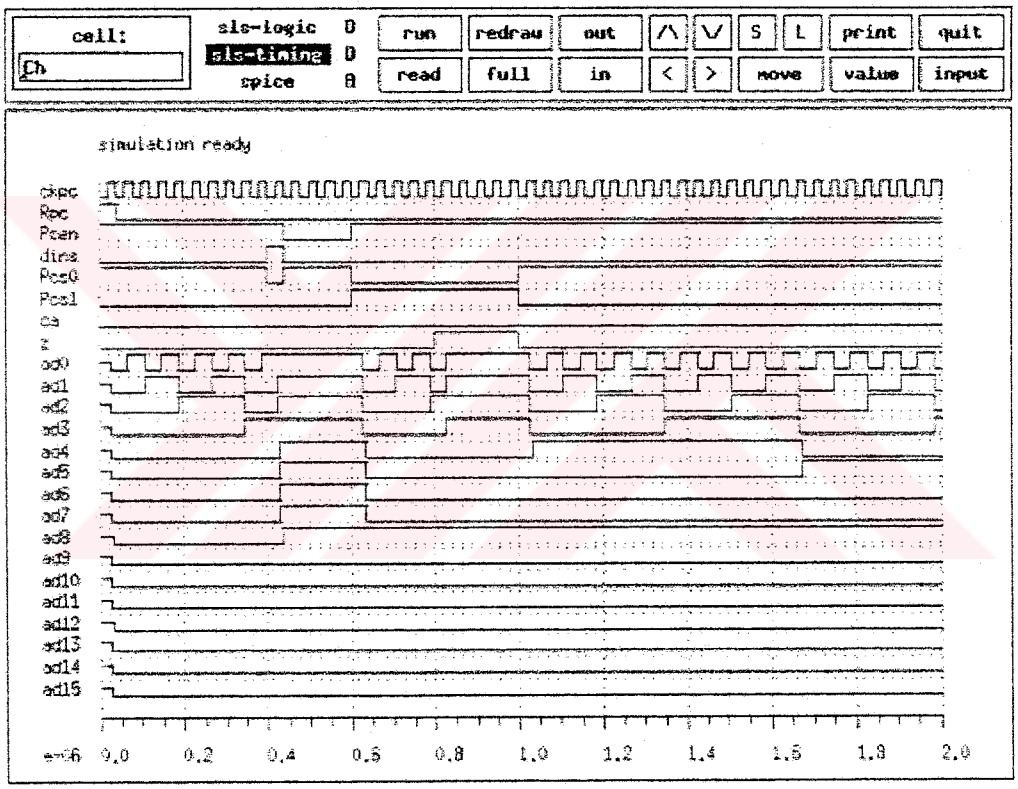


Figure 4.37- The Simulation Results for Branch Operations

CHAPTER FIVE
CONCLUSIONS

1- Conclusions

Minimizing the size of an electronic component while increasing its capability and reliability is one of the most important goals of the today's electronics world. Thus, there is a great trend to the design of very large scale and even ultra large scale integrated circuits. There is an increasing interest in integrated circuit technology also in our country and some design projects are performed by educational institutions. Therefore, in order to be a part of this progress, a general purpose 8 bit microcontroller designed with Very Large Scale Integrated circuit technology of 1.6μ double metal CMOS in this thesis.

The microcontroller includes four subsystems. These are processing unit, memory unit, input-output unit and control unit. In the processing unit, the arithmetic, logic and shift operations are implemented. The memory unit consists of 16 bytes of internal read-write memory and address decode unit. The aim of the input-output unit is to provide the interface between the external world and the microcontroller. Finally, all of the operations performed by the microcontroller are directed by the control unit. The microprogram control method is used for control of the microcontroller. Consequently, a control memory in which the microoperations are written is included for producing control signals.

The microcontroller has a capability of running 31 instructions including conditional and unconditional branch instructions. The other instructions can also be arranged to operate depending on status flags. Although the main instruction set base is established in the microcontroller, the instruction combinations can be increased with some hardware additions.

The designed microcontroller is a general purpose one. Thus, it can be used for many applications. Since the microcontroller has an external address bus of 16 bits, it can address 64 Kbytes of address space. Therefore, a 64 Kbytes long program can run on this microcontroller.

The CMOS technology that has some advantages compared to other technologies that are commonly used for integrated circuit design. These advantages are high component density, low power consumption and symmetrical structure.

The Ocean Sea-Of-Gates design system is used for layout design, extraction and simulation of the circuit. The sea-of-gates design style presents a base transistor image and the interconnections are made with two metal layers. Besides the basic cells are designed by the transistors, the Ocean layout editor also makes it possible to use hierarchical approach. Thus, the designed main cells are used for top level components as it is in semi-custom design. Although the automatic place and route option is possible for layout design, the automatically placed components were not suitable for regular design and not effective in terms of used area. Thus, the placement and routing operation was performed manually.

The use of sea-of-gates style reduced the flexibility of the design despite the fact that this approach is more suitable for a first design in VLSI. Thus, a low-level VLSI circuit is designed in the transistor level. Therefore, the design requirements, such as studying transistor structure; handling interconnection and placement issues, are deeply considered. Additionaly, main integrated circuit concepts are examined in order to use for further designs.

REFERENCES

[1] Burger, D. & Goodman, J. (1997), Billion Transistor Architectures, IEEE Computer, 30, 46-50

[2] Daga, A. & Birmingham, W. (1997), Interface Finite state Machines: Definition, Minimization and Decomposition, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 16, 497-505

[3] Fabricius, E., (1990), Introduction to VLSI Design, Mc Graw-Hill Publishing Company

[4] Geiger, R., Allen, P. & Strader, N., (1990), VLSI Design Techniques for Analog and Digital Circuits, Mc Graw-Hill Publishing Company

[5] Groeneveld, P. & Stravers, P., (1993), Ocean: The Sea-of-Gates Design System, Delft University of Technology

[6] Hammond, L., Nayfeh, B. & Olukotun, K. (1997), A Single-Chip Multiprocessor, IEEE Computer, 30, 79-85

[7] Ismailoğlu, N. & Köseoğlu, M.,(1997), 5. Sinyal İşleme ve Uygulamaları Kurultayı, 256x8 İkilik Sözde Gürültü Kodlu Sayısal Uyumlu Filtre Tasarımı ve 0.5μ Yarı Özel Çok Büyük Çaplı Tümleştirme Teknolojisi ile Gerçekleşmesi

[8] Jaggar, D., (1997), ARM Architecture and Systems, IEEE Micro, 17, 9-11

[9] Kang, S. & Leblebici, Y., (1996), CMOS Digital Integrated Circuits Analysis and Design, Mc Graw-Hill Publishing Company

[10] Kumar, A.,(1997), The HP PA-8000 RISC CPU, IEEE Micro, 17, 27-32

[11] Mano, M., (1979), Digital Logic and Computer Design, Prentice Hall International Editions

[12] Mc Calla, T., (1992), Digital Logic and Computer Design, Maxmillan Publishing Company

[13] Milne, G., Khan, A., Rayne,S. & Christensen, J., (1997), Microcontroller Design Advantages for Portable Computing, IEEE Micro, 17, 49-55

[14] NATO Advanced Study Institute Series, Design Methodologies for VLSI Circuits, Maryland, 1982

[15] Patt, Y., Patel, S., Evers, M., Friendly, D. & Stark, J., (1997), One Billion Transistor, One Uniprocessor, One Chip, IEEE Computer, 30, 51-57

[16] Pucknell, D. & Eshraghian, K., (1997), Basic VLSI Design, Prentice Hall International Editions, 3rd Edition

[17] Spruth, W., (1989), The Design of a Microprocessor, Springer Verlag

[18] Taub, H. & Schilling, D., (1983), Digital Integrated Electronics, Mc Graw-Hill PublishingCompany, 10th Edition

[19] Tellez, G. & Sarafzadeh, M. (1997), Minimal Buffer Insertion in Clock Trees with Skew and Slew Rate Constraints, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 16, 333-342

[20] Tredennick, N., (1987), Microprocessor Logic Design, Digital Press

[21] Uyemura, J., (1993), Circuit Design for CMOS VLSI, Kluwer Academic Publishers, 3rd Edition

[22] Vittal, A. & Sadowska, M. (1997),Crosstalk Reduction for VLSI, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 16, 290-298

[23] Wiatrowski, C. & House, H., (1980), Logic Circuits and Microcomputer Systems, Mc Graw-Hill Publishing Company

[24] Yazıcıoğlu, E., Dündar, G., Balkır, S. & Çağlar, H., 5. Sinyal İşleme ve Uygulamaları Kurultayı, Devşirimli Blok Dönüşümü ve Ters Dönüşümünü Gerçekleyen Tümdevrelerin VLSI Tasarımı

APPENDIX A
LINUX

Linux is a completely free reimplementation of Unix, but does not come from the same source code base, which is available in both source code and binary form. It is copyrighted by Linus B. Torvalds (Linus.Torvalds@Helsinki.FI) and other contributors, and is freely redistributable under the terms of the GNU Public License.Linux is not public domain, nor is it `shareware'. It is `free' software, commonly called freeware.

Linux is still free as of version 1.2, and will continue to be. Because of the nature of the GNU copyright which Linux is subject to, it would be illegal for it to be made not free.

Linux runs on 386/486/Pentium machines with ISA, EISA, PCI and VLB busses. There is a port in progress for multiple Motorola 680x0 platforms (currently running on some Amigas and Ataris), which now works quite well. It requires a 68020 with an MMU, a 68030, 68040, or a 68060, and also requires an FPU.Linux runs well on DEC's Alpha CPU, currently supporting the "Jensen", "NoName", "Cabriolet", Universal Desktop Box" and some other platforms.Linux is being rapidly ported to the Sun Sparc; most Sparcs now run Linux, but there isn't a distribution available yet.

Linux is developed using an open and distributed model, instead of a closed and centralized model like much other software. This means that the current development version is always public (with up to a week or two's delay) so that anybody can use it. The result is that whenever a version with new functionality is released, it almost

always contains bugs, but it also results in a very rapid development so that the bugs are found and corrected quickly, often in hours, as many people work to fix them.

In contrast, the closed and centralized model means that there is only one person or team working on the project, and they only release software that they think is working well. Often this leads to long intervals between releases, long waiting for bug fixes, and slower development. Of course, the latest release of such software to the public is often of higher quality, but the development speed is generally much slower.

Some of the features of Linux are multitasking, multiuser, multiplatform.It runs in protected mode on the 386. Additionally Linux has memory protection between processes, so that one program can't bring the whole system down. A unified memory pool for user programs and disk cache, so that all free memory can be used for caching, and the cache can be reduced when running large programs. All source code of Linux is available, including the whole kernel and all drivers, the development tools and all user programs; also, they are freely distributable. There are some commercial programs being provided for Linux now without source, but everything that has been free is still free.The kernel has 387-emulation so that programs don't need to do their own math emulation. Every computer running Linux appears to have a math coprocessor.

Linux supports several common filesystems, including minix-1, Xenix, and all the common system V filesystems, and has an advanced filesystem of its own, which offers filesystems of up to 4 TB, and names up to 255 characters long. It has transparent access to MS-DOS partitions (or OS/2 FAT partitions) via a special filesystem. VFAT (WNT, Windows 95) support has been added to the development kernel, and will be in the next stable release. Linux also has CD-ROM filesystem which reads all standard formats of CD-ROMs.

Linux supports TCP/IP networking, including ftp, telnet, NFS.Appletalk networking available in the current development kernel.

APPENDIX B

INSTRUCTION SET

The instructions of the microcontroller can be divided into six subgroups. The names, comments and binary representations of instructions are as following:

Transfer Instructions

1. YUK A, s	Loads the immediate data to Aregister
2. YUK B, s	Loads the immediate data to Bregister
3. YUK A, (n)	Loads the content of the direct address to A
4. YUK B, (n)	Loads the content of the direct address to B
5. YUK (n), A	Loads the content of A register to the direct
6. YUK (n), B	Loads the content of B register to the direct
7. YUK A, B	Loads the content of A register to the B register
8. YUK B, A	Loads the content of B register to the A register

Arithmetic and Logic Instructions

1. TOP A, B	Adds the content of the B register to A
2. ETP A, B	Adds the content of the B register to A with carry
3. VE A, B	ANDs the content of the B register with A
4. VY A, B	ORs the content of the B register with A
5. DGL A	Determines the complement of the content of A
6. ART A	Increments A
7. AZT A	Decrements A
8. SIL E	Clears carry flag
9. BIR E	Sets carry flag

Rotate and Shift Instructions

1. SLK A	Shifts the content of the A register left
2. SGK A	Shifts the content of the A register right
3. DND A	Rotates the content of the A register right
4. SLK B	Shifts the content of the B register left
5. SGK B	Shifts the content of the B register right
6. DND B	Rotates the content of the B register right

Branch Instructions

1. ATL addr	Jumps to the address unconditionally
2. ATL S, addr	Jumps to the address if Zero flag is set
3. ATL E, addr	Jumps to the address if Carry flag is set

I/O Instructions

1. CIK A	Outputs the content of the A register
2. CIK B	Outputs the content of the B register
3. CIK (n)	Outputs the content of the direct address

Control Instructions

1. SFR	Resets the microcontroller
2. DUR	Stops the microcontroller

APPENDIX C
LIST OF MICROOPERATIONS

In this section, the list of microoperations that constitutes the instructions of the microcontroller will be given. Each row shows the simultaneous actions. The microoperations are placed in control memory as listed below.

Instruction	**Microoperation**	**Next Address of CAR**
Fetch	PC ← PC, Din ← opcode	CAR ← CAR + 1
	PC ← PC +1	CAR ← Din
YUK A, s	PC ← PC, Din ← data	CAR ← CAR +1
	PC ← PC +1, A ← data	CAR ← 0
YUK B, s	PC ← PC, Din ← data	CAR ← CAR +1
	PC ← PC +1, B ← data	CAR ← 0
YUK A, (n)	PC ← PC, Din ← n	CAR ← CAR +1
	PC ← PC + 1, MAR ← Din	CAR ← CAR +1
	PC ← PC, A ← RAM	CAR ← 0
YUK B, (n)	PC ← PC, Din ← n	CAR ← CAR +1
	PC ← PC + 1, MAR ← Din	CAR ← CAR +1
	PC ← PC, B ← RAM	CAR ← 0
YUK (n), A	PC ← PC, Din ← n	CAR ← CAR +1
	PC ← PC + 1, MAR ← Din	CAR ← CAR +1
	PC ← PC, RAM ← A	CAR ← 0

Instruction	Microoperation	Next Address of CAR
YUK (n), B	PC ← PC, Din ← n	CAR ← CAR +1
	PC ← PC + 1, MAR ← Din	CAR ← CAR +1
	PC ← PC, RAM ← B	CAR ← 0
YUK A, B	PC ← PC, A ← B	CAR ← 0
YUK B, A	PC ← PC, B ← A	CAR ← 0
TOP A, B	PC ← PC, Carry ←0	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← a	CAR ← 0
ETP A, B	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← a	CAR ← 0
VE A, B	PC ← PC, Carry ←0	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← 1	CAR ← 0
VY A, B	PC ← PC, Carry ←1	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← 1	CAR ← 0
DGL A	PC ← PC, B ←FFH	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← 1	CAR ← 0
ART A	PC ← PC, B ←01H	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← a	CAR ← 0
AZT A	PC ← PC, B ←FFH	CAR ← CAR +1
	PC ← PC, ALU ← A+ B	CAR ← CAR +1
	PC ← PC, A ← a	CAR ← 0
SIL E	PC ← PC, Carry ← 0	CAR ← 0
BIR E	PC ← PC, Carry ← 01H	CAR ← 0

Instruction	Microoperation	Next Address of CAR
SLK A	PC ← PC, SH ← A	CAR ← CAR +1
	PC ← PC, SH ←Shift left	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
SGK A	PC ← PC, SH ← A	CAR ← CAR +1
	PC ← PC, SH ←Shift right	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
DND A	PC ← PC, SH ← A	CAR ← CAR +1
	PC ← PC, SH ←Rotate right	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
SLK B	PC ← PC, SH ← B	CAR ← CAR +1
	PC ← PC, SH ←Shift left	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
SGK B	PC ← PC, SH ← B	CAR ← CAR +1
	PC ← PC, SH ←Shift right	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
DND B	PC ← PC, SH ← B	CAR ← CAR +1
	PC ← PC, SH ←Rotate right	CAR ← CAR +1
	PC ← PC, A ← SH	CAR ← 0
ATL addr	PC ← PC, Din ← addr	CAR ← CAR +1
	PC ← addr	CAR ← 0
ATL S, addr	PC ← PC, Din ← addr	CAR ← CAR +1
	PC ← addr if Zero = 1 PC ← PC + 1 if Zero = 0	CAR ← 0
ATL E, addr	PC ← PC, Din ← addr	CAR ← CAR +1
	PC ← addr if Carry = 1 PC ← PC + 1 if Carry = 0	CAR ← 0
CIK A	PC ← PC, Dout ← A	CAR ← 0
CIK B	PC ← PC, Dout ← B	CAR ← 0

Instruction	Microoperation	Next Address of CAR
CIK (n)	PC ← PC, Din ← n	CAR ← CAR +1
	PC ← PC + 1, MAR ← Din	CAR ← CAR +1
	PC ← PC, Dout ← RAM	CAR ← 0
SFR	PC ← 0, Carry ← 0	CAR ← 0
DUR	PC ← PC	CAR ← CAR

The abbreviations used in microoperation table are listed below:

PC : Program Counter
CAR : Control Address Register
A : A Register
B : B Register
Din : Data Input Latch
Dout : Data Output Latch
ALU : Arithmetic-Logic Unit
SH : Shifter
MAR : Memory Address Register
RAM : Internal Memory Unit
a : Arithmetic Output Latch
l : Logic Output Latch
Carry : Carry Flag
Zero : Zero Flag
opcode : The opcode of the instruction from program memory
data : 8 bit immediate data
n : Internal memory address
addr : Branch address

APPENDIX D
CONTROL VARIABLES

The instructions of the microcontroller are implemented as a sequence of microoperations and each microoperation is placed in the control memory as row of binary numbers. These numbers called as control variables. The control variables are listed below as it is placed in the control memory.

Insruction	Rpc	Pcen	Pcs1	Pcs0	Rcar	Cars	dins	dos	wra	rda	wrb	rdb	cres	crset	asel	lsel	insh	oush	sh1	sh0	mars	rdra	wrra	bone	bset
Fetch	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
YUK A, s	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
YUK B, s	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
YUK A, (n)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
YUK B, (n)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0
YUK (n), A	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
YUK (n), B	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
YUK A, B	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
YUK B, A	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
TOP A, B	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
ETP A, B	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Instruction	Rpc	Pcen	Pcs1	Pcs0	Rcar	Cars	dins	dos	wra	rda	wrb	rdb	cres	crset	asel	lsel	insh	oush	sh1	sh0	mars	rdra	wrra	bone	bset
VE A, B	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
VY A, B	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
DGL A	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
ART A	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
AZT A	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
SIL E	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
BIR E	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
SLK A	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
SGK A	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
DND A	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0
	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0

Instruction	Rpc	Pcen	Pcs1	Pcs0	Rcar	Cars	dins	dos	wra	rda	wrb	rdb	cres	crset	asel	lsel	insh	oush	sh1	sh0	mars	rdra	wrra	bone	bset
SLK B	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
SGK B	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
DND B	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
ATL addr	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ATL S, addr	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ATL E, addr	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
CIK A	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
CIK B	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
CIK (n)	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
SFR	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
DUR	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0