COMPUTERSYSTEMSLABORATORY

DEPARTMENTS OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
STANFORD UNIVERSITY · STANFORD, CA 94305

Studies in Microprocessor Design

Donald Alpert

TechnicalReQortNo.232

June 1982

The work herein was supported in part by Stanford University.

Studies in Microprocessor Design

Donald Alpcrt

Technical Report NO. 232

June 1982

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Abstract

Microprocessor design practice is briefly surveyed. Examples are given for high-level and low-level tradeoffs in specific designs with emphasis on integrated memory functions. Some relations between architectural complexity and design arc discussed, and a simple model is presented for implementing a RISC-like architecture. A direction for microprocessor architecture is proposed to allow flexibility for designing with varying processing technologies, cost goals, and performance goals.

Key Words and Phrases: microprocessor, microprocessor architecture, microprocessor design

Table of Contents

List of Figures

List of Tables

1. Introduction

This study is concerned with the practice of microprocessor design. It is written from the perspective of a computer architect who has spent the past two years working with designers of commercial microprocessors. The reader should have some background in computer architecture, computer design, and VLSI technology.

One purpose of writing this report is to help architects understand better the considerations and problems of microprocessor designers. Those readers who are neither architects nor designers of microprocessors may gain insight into design issues.

Chapter 2 describes 3 examples of recent designs and discusses some observable trends. Chapter 3 presents an overview of the general problem of design: implementing an architecture using a processing technology. Chapter 4 examines the high-level tradeoffs of the Zilog Z8* microcomputer design. Chapter 5 discusses some low-level design tradeoffs, specifically for ROMs. Chapter 6 provides data on the relative costs of memories in several designs. A simple model is presented relating architecture to design tradeoffs. Based on this examination of design practice a direction for architecture is proposed.

I wish to thank the designers at Zilog who have patiently answered questions on their trade and whose responses are reported here explicitly or implicitly. Pat Lin, Gary Proscnko, and Jack Taylor are the designers of the Z8. Peter Ashkin, Dean Carberry, Bill Carter, Ross Freeman, Jackson Mu, C. N. Patel, J. K. Tsai, and Mike Yamamura deserve acknowledgement along with others, who with apology, are not mentioned. The views presented in this study do not represent Zilog or any other individuals at Zilog.

Professor Michael Flynn at Stanford University has supported this study, and coined the term *differential architecture*. Mark Horowitz at Stanford provided helpful comments about integrated circuit design. The work of Professor David Patterson and the designers of the RISC microprocessor at U.C. Berkeley provoked many of the thoughts on issues of architecture and design presented here.

---

*Z8 is a registered trademark of Zilog, Inc.

2. Approaches to Qesign

This study of microprocessor design begins with 3 examples: U. C. Berkeley RISC, Intel iAPX432*, and Hewlett-Packard 32-bit VLSI CPU.

2.1 Berkeley RISC

The designers of the Berkeley RISC contend that a Reduced Instruction Set Computer is more appropriate than a Complex Instruction Set Computer (CISC) as an architecture to support compiled code for VLSI processors [24, 25]. The RISC architecture reduces the instruction set to the level of vertical microcode. All instructions are 32 bits, and all except data memory referencing instructions execute in 1 machine cycle. Instructions that reference data memory require 1 extra machine cycle. Only register load and store instructions reference data memory. Many of these ideas in RISC appeared earlier in the 801 minicomputer developed at IBM Thomas J. Watson Research Center [28].

In order to diminish the number of data memory references RISC uses an innovative register file. At each procedure call a new set of 16 registers is allocated on a stack. These 16 registers, 6 of the caller's registers, and 10 registers global to all procedures can be accessed using 5-bit fields in the instructions. This scheme allows most parameters and local variables to be held in registers. The overhead of loading and storing between registers and memory that occurs with conventional general-purpose register architectures is usually avoided.

The RISC implementation buffers up to 8 of the most recently allocated register sets on-chip. When the nesting of procedure calls and returns causes the buffer to overflow or underflow, an exception is raised to allow software maintenance of the buffer using an overflow stack in memory. The size of the register buffer on the chip is not specified in the architecture; it is implementation dependent.

A chip plan for RISC, sketched from a photograph, is shown in Figure 2-l. The performance for this design is reported to be 70% better than for VAX 11/780** on a sample of 5 programs [12]. RISC achieves its performance with low-latency (250 ns), high-bandwidth (80 Mbps) instruction memory rcfcrenccs off-chip, intensive use of data storage on-chip, and littlc use of control storage on-chip.

---

* iAPX432 is a registered trademark of Intel Corporation.

** VAX is a registered trademark of Digital Equipment Corporation.

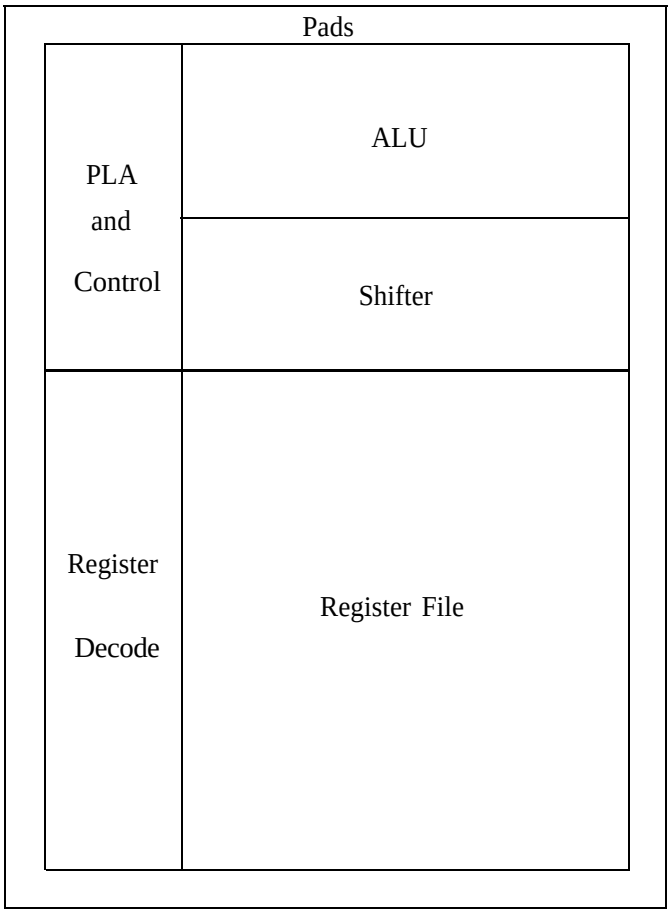


Figure 2-1: RISC Chip Plan

2.2 Intel iAPX432

The Intel iAPX432 [14], in contrast to RISC, raises the level of the instruction set close to system software. Instructions arc encoded on bit-variable boundaries. The operations defined in the architecture include language run-time support (e.g., heap allocation and garbage collection) and operating system kernel primitives (e.g., message passing and process scheduling). The architecture also defines an object-oriented memory management mechanism that promotes allocation and protection of individual structures.

The implementation of the iAPX432 is a 3-chip set. The chip plans are shown in Figure 2-2, reproduced from a recent article on microprocessors [4]. One chip, the IDU, decodes the machine instructions to transmit a sequence of vertical microinstructions to another chip, the MEU, for execution. A third chip, the IP, serves as interface between the protected iAPX432 memory and the external environment (generally I/O controllers).

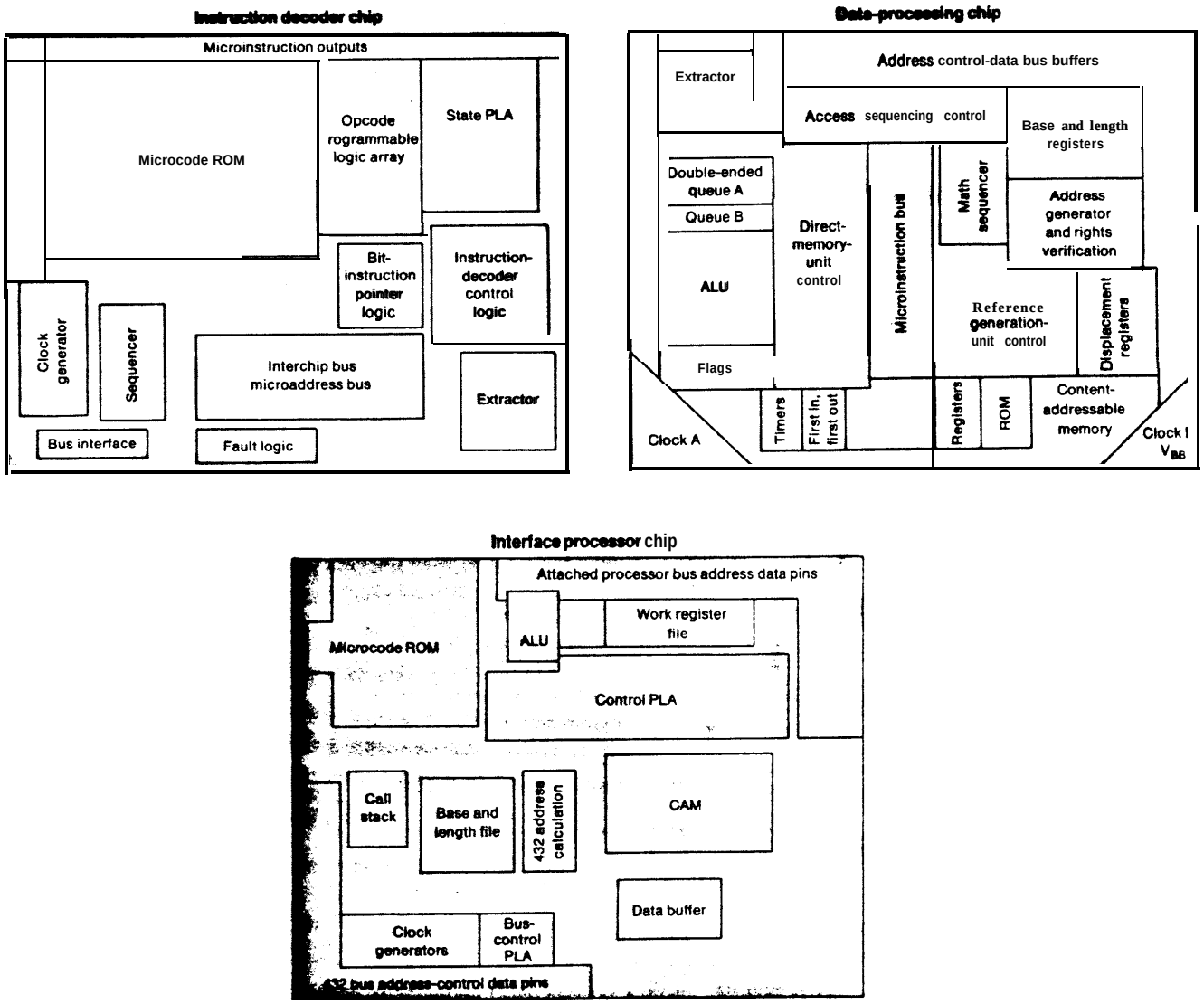


Figure 2-2: iAPX432 Chip Plans [4]

The iAPX432 design uses extensive control storage. The IDU has 64Kb of ROM and 10Kb of PLA [29]. The IP has 32Kb of ROM [3]. Little on-chip storage is provided for data: only two 80-bit microcode temporary registers and one 16-bit top stack buffer in the MEU [8].

2.3 Hewlett-Packard 32-bit VLSI CPU

Hewlett-Packard has developed a 32-bit VLSI CPU as one component in an experimental VLSI system using advanced processing technology [5, 6]. The architecture for this processor emphasizes high-level languages and operating systems. Numerous data types are supported, including logical values, integers, floating point formats, and strings. Functions to assist multiprogramming and multiple processors are in the architecture.

A chip plan for the CPU is shown in Figure 2-3, reproduced from the same article as the iAPX432 chip plans [4]. The chip contains 450,000 transistors. The microcode ROM stores 350Kb. Four 32-bit top of stack buffers are located in the ALU section. The register file holds 28 32-bit data values.

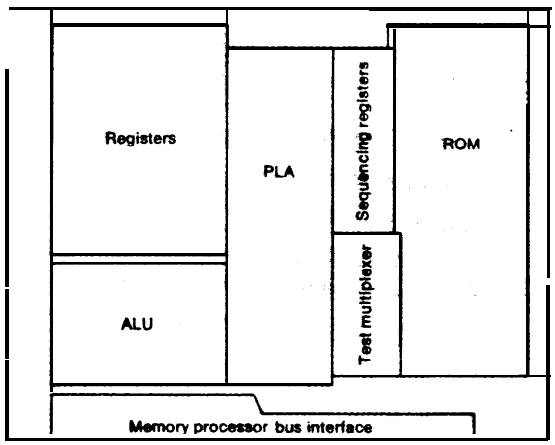


Figure 2-3: Hewlett-Packard 32-bit VLSI CPU Chip Plan [4]

2.4 Observations

This brief examination of contemporary microprocessor designs allows several observations to be made.

2.4.1 Regular Structure

The designs are composed of several function blocks that are largely replications of a single cell or a structure of cells. There is no other way to manage VLSI design. This is distinct from older designs that contained large portions of random logic and interconnection. (Examining photographs of chips designed in the old and new styles may remind one of the paintings by modern artists Jackson Pollock and Piet Mondrian.)

2.4.2 Memory

A trend of increasing use of on-chip resources for memory functions is becoming apparent. This characteristic was identified by Patterson and Sequin [26].

The memory types of interest here arc ROM, RAM, and CAM. ROM is used for control storage and for program storage in single-chip microcomputers. PLA, when used for control storage, is similar to ROM.

RAM is used in registers and for storage of other alterable data. Registers are often highly specialized RAM that may allow multiple-port access. Content Addressable Memory (CAM) is used for address translation either integrated with the CPU, as in iAPX432, or external to the CPU, as in Zilog's 28015 [34] and National Semiconductor's NS16082 [23]. CAM is also used for address tags in cache memories; an instruction cache is included in MC68020, Motorola's recently announced 32-bit implementation of the M68000 architecture [21].

There are costs and benefits for each type of storage. Memory integrated with the CPU must be designed carefully to achieve an effective, balanced design. The design of memories in VLSI processors is discussed in several later sections.

2.4.3 Architecture and Design

The choice of architecture has strong influence on the quantities of ROM, RAM, and CAM needed to meet performance goals.

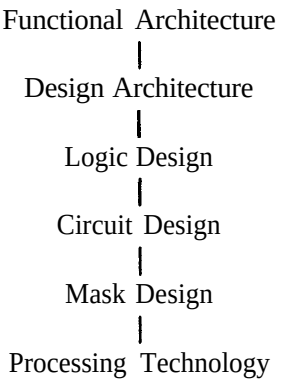
The RISC architecture limits the use of ROM control storage because the instruction set is so simple. The RISC architecture, with its register stack, encourages the use of RAM for data storage. The RISC design devotes 47% of chip area to the register file and decoder [12]. On the other hand, the iAPX432 architecture promotes the use of ROM to implement its complex functions. More than 25% of total device placements in the 3-chip implementation are located in control storage arrays. The iAPX432 architecture discourages the use of registers for storing data; it emphasizes protection of data through the memory management mechanism.

In chapter 6 architecture and design considerations for different memory functions are discussed. Now, however, we address the question "How is it that starting with basically the same raw materials such different designs result?"

3. Microprocessor Design Practice

3.1 Hierarchy of Disciplines

Microprocessor design involves a hierarchy of disciplines.



Functional architecture is responsible for writing a document, often called the *Principles of Operation,* specifying the common interface between software and hardware. Roth software and hardware designers refer to this specification to determine the behavior of the processor when particular bit patterns appear in instructions, data, or control registers. The issues decided at this level of design include USC of stack vs. general-purpose registers, variable-length vs. fixed-length instruction encoding, and linear vs. segmented addressing.

Design architecture is responsible for specifying the major function units of the implementation, the paths connecting those units, and the general timing rcquiremcnts. Dccisions at this level include multiplexed vs. demultiplexed address and data buses, width of the ALU, and degree of pipelining. (When the term *architecture* is used in this study without the qualification *functional* or *design,* functional architecture is implied. I believe this terminology originated at Amdahl Corporation.)

Logic design is responsible for implementing the function units defined by design architecture using the techniques of switching theory. Decisions at this level include PLA vs. random logic and carry-lookahead vs. carry-propagate adders.

Circuit design is responsible for the electrical realization of logic functions defined by logic design. Decisions at this level include dynamic vs. static buses and l-bit per cell vs. 2-bit per cell ROMs.

Mask design is responsible for creating the geometric patterns used during fabrication to realize circuit elements defined by circuit design.

Processing tcchnolgy is responsible for specifying thc layout rules for mask design, thc ranges of device parameters for circuit design, and the manufacturing costs for design architecture to evaluate. The layout rules include spacing between conductors on the same level, overlap for conductors on different levels, and contacts between levels. The device parameters include sheet resistivity for conductors, capacitance per unit area between conductors, threshold voltages for transistors, and maximum current densities.

The job of microprocessor design is to implement a functional architecture using a processing technology while minimizing a cost/performance measure, subject to constraints. The following sections discuss the design considerations of cost, performance, and constraints when given a functional architecture.

3.2 Cost Considerations

There are many costs tc consider in planning a microprocessor product: design, fabrication, packaging, testing, marketing, support software, etc. This study concentrates on the technical costs of area, power, and pin count. The next two sections explain the relations between these technical costs and thc economic costs of fabrication and packaging.

The other costs that concern microprocessor designers, such as design and testing, are significant. These are, however, not treated here because they are less quantitave than the technical costs identified, and they seem less clearly relatcd to the issues of architecture and design that are discussed later.

3.2.1 Fabrication Cost

During integated circuit fabrication a circular wafer containing many copies of the same chip is processed through several steps. The number of working chips per wafer is known as *yield.* Yield depends on the number of chips processed per wafer and on the distribution of defects that occur during processing. Dcfccts are caused by many factors including imperfections in a mask, impurities in the wafer, and pinholes in the oxide layer.

The number of chips processed on a wafer (gross dice per wafer) is approximately $\pi(R\text{-}A^{1/2})^2/A$, where R is the radius of the wafer and A is the area of thc chip. The term $R\text{-}A^{1/2}$ compensatcs for thc uselcss portions of chips that lie on the perimeter of the wafer.

A number of statistical models have been proposed for determining the probability that a fabricated chip will bc defect-free [22, 33]. The model presented here is the iso-defect relation based on Bose-Einstein statistics [27]. This model is reported to overestimatc yield for large chips [13].

The Bose-Einstein statistics predict that at each step of processing the probability of a chip's being defect-free is $(1+DA)^{-1}$, where D is the defect density measured in defects per unit area, and A is the active area of the chip. (In this presentation assume that A is the total area of the chip, and D is adjusted so that DA is the expected number of defects per chip.) The iso-defect model assumes independent processing steps, each with the same defect density. The probability of a chip's being defect-free after n processing steps is $(1+DA)^{-n}$. In practice the value of n is adjusted to compensate for the type of circuit (e.g., ROM, RAM, logic), maturity of the process, and design rules used.

The formula for yield (Y) predicted by the iso-defect model is given below.

$$Y = (1 + DA)^{-n}\pi(R\text{-}A^{1/2})^2/A$$

The value of R is typically 5.0 cm (2.0 in) in the early 1980's. Larger values of R do not allow the entire wafer to be in focus during exposure because the wafer warps during fabrication.

Table 3-1 shows yield for several values of A, n, and D. (For those more comfortable with English units: 1 mm is approximately 40 mil, and 1 cm^{-2} is approximately 6.5 in^{-2}.) Some of these values are displayed graphically in Figure 3-l.

n	D (cm^{-2})	$A^{1/2}$ (mm)		
		4	6	a
0		415	168	86
5	0.5	282	73	21
5	1.0	198	36	7.2
10	0.5	192	32	5.3
10	1.0	94	7.8	0.6

Table 3-1: Yield (R = 5.0 cm)

As chip area increases, yield decreases because there are fewer gross dice per wafer and because relatively fewer of the fabricated chips are defect-free. The values of n, D, and wafer fabrication cost for commercial processing are highly proprietary. For purposes of discussion only, assume n = 10, D = 0.5, and wafer fabrication cost of several hundred dollars. For these values when chips grow from 4 mm on a side to 8 mm on a side, a factor of 4 increase in area, then yield decreases by a factor of 36. In considering a design that is approximately 6 mm on a side, the yield changes by 3.4% for a 1% change in area. The marginal cost per area can be very high for a large chip.

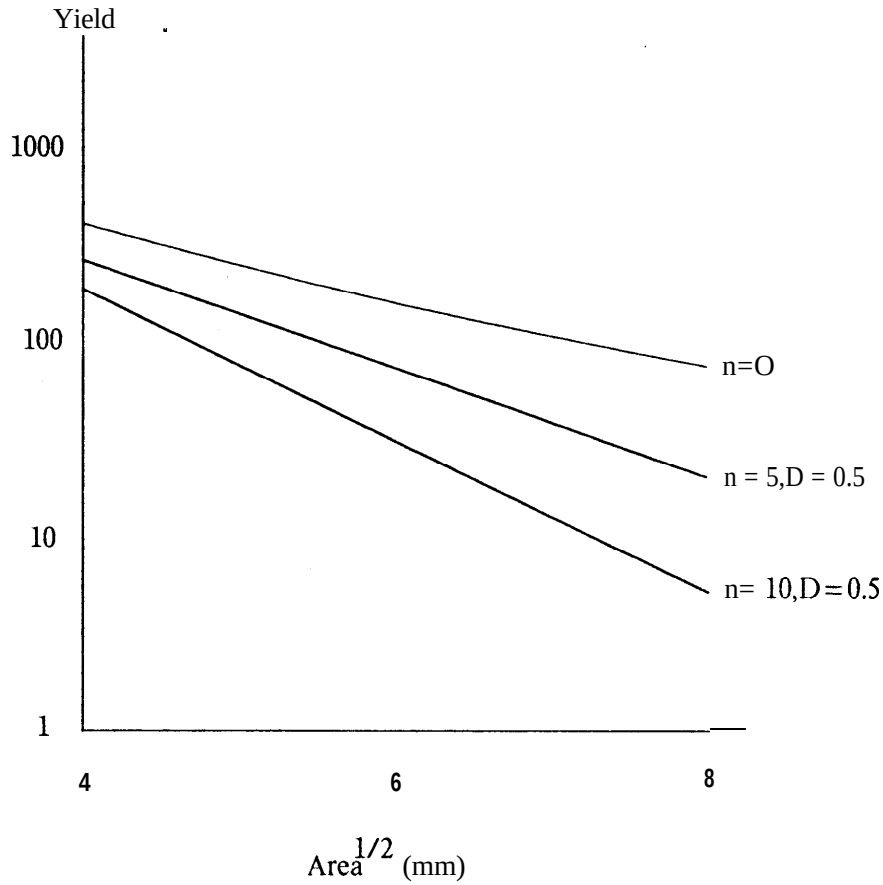


Figure 3-1: Yield vs. Area (R = 5.0 cm)

3.2.2 Packaging Cost

A number of techniques have been developed for packaging in tegratcd circuits. Until recently, dual-in-line packages made of plastic or ceramic with 40, 48, or 64 pins were almost exclusively used for commercial microprocessors. Newer methods, such as quad-in-line packages and leadlcss chip-carriers, have been developed to reduce package size and to increase the number of external electrical connections to the microprocessor. (The term *pin count*, as used in this study, means the number of external electrical connections whether or not pins are used.)

Table 3-2 gives representative 1982 values for the materials cost of dual-in-line packages as a function of pin count and thermal resistance θ_J. This information was provided by Mark Brodsky of Zilog's component packaging department [7]. Thermal resistance is the rise in temperature of the semiconductor junctions above the package exterior when 1 W of power is dissipated. Standard practice is to design for correct circuit operation within some limits of ambient temperature, typically 0 degC to 70 degC for commercial applications. The maximum allowed junction temperature is about 140 degC.

Pin Count	θ_J (degC/W)	cost ($)
40	90	0.8
40	45	3.0
48	90	1.0-1.5
48	40	4.0
64	90	1.5-2.0
64	40	4.5-5.0

Table 3-2: Package Costs

The number of pins is directly related to the area and power required for bonding pads and interface buffers. The implications of power and pin count arc not limited to packaging, but extend to the system using the microprocessor. The power dissipated by the chip must come from a supply and must be removed to the environment as heat. The number of pins is related to requirements for printed circuit board space, number of support circuits (e.g., electrical buffers and decoders), and width of the memory data path. It may be interesting to note that IBM has sclcctcd the Intel 8088, with an 8-bit data path, rather than the binary compatible 8086, with a 16-bit data path, for use in its personal computer.

3.3 Performance Considerations

To evaluate the performance of a microprocessor design it is first necessary to select a workload characteristic of the intended application. The workload can be a single program, a weighted combination of programs, a synthetic program, or a synthetic mix of instructions. For programs written in a high-level language a compiler must also be selected to determine the workload. It may be necesary to specify more than one workload when very different applications are expected.

A performance measure (P) for the design and workload is defined to be the reciprocal of the execution time.

$$P = (\text{execution time})^{-1}$$

For synchronous designs the performance is dctcrmincd by the cycle time and the number of cycles required to execute the workload.

$$P = (\text{cycle time} * \text{no. cycles})^{-1}$$

3.3.1 Cycle Time

One way to improve performance is to reduce the cycle time. Some of the many factors determining cycle time are technology, power, width of data paths, and degree of pipelining.

A change in technology can reduce the inherent delay through switching elements. The change may be dramatic, as from MOS to biploar processing, or modest, as varying a given process. For example, the Zilog Z80 and Z80L* are nearly identical in design, but during processing the transistors arc implanted differently. The Z80L has approximately twice the cycle time and one-fifth the power dissipation of the Z80 [35, 36].

When the technology is fixed, cycle time can be traded for power. In nMOS a critical path can usually be made faster by scaling up device geometries, which also increases power dissipation. Such a change cannot be made without considering its effects on electrical loading and chip area.

Narrower data paths can reduce the cycle time, although this may be of secondary importance to the design. The delay in gating data onto buses is less, especially with current MOS processes where charging long polysilicon control lines can be a large fraction of the cycle time. The ALU can have smaller delay because carry determination is simpler. Narrower data paths use less area; a more compact design can have shorter communication delays.

Pipclining can help reduce the cycle time by dividing delay into separate, smaller time units. The designers of RISC assert that a simple instruction set architecture allows a faster cycle time [12]. When instruction decoding and execution are pipelined, however, the effect of instruction set architecture on cycle time seems negligible.

---

*Z80 and Z80L are registered trademarks of Zilog, Inc.

3.3.2 Number of Cycles

Computer designers have developed numerous techniques to reduce the number of cycles required to execute instructions. Many of these methods are specific to particular architectures and designs. Among the general factors affecting the number of cycles are number and width of data paths, function of data paths, number of execution units, degree of pipelining, memory access time, and use of cache or other buffer memory.

Increasing the number or width of data paths can decrease the cycles required for data movement, address computation, arithmetic, etc. Increasing the functionality of the data path can reduce the number of cycles to execute particular instructions, for example multiply or memory block move. Shustek's measurements of IBM System/370 implementations show that such instructions, although infrequent among total instructions, are often performance bottlenecks [30]. Pipclining and multiple execution units both allow reduction in the number of cycles by simultaneously executing several instructions or portions of a single instruction. (Heavy pipelining can also increase cycle count due to resource interlocks and branch delays.) Caches and buffer memories eliminate cycles used for memory access.

3.4 Design Const raints

There are many constraints on a commercial microprocessor design. Two extreme cases serve as examples.

The design of a high-performance microprocessor has an upper bound on chip area determined by manufacturing economics, and upper bounds on power and pin count determined by the sclcctcd package. The goal of the design is to maximize performance.

The design of a low-cost microprocessor has a lower bound on performance determined by the market of intended applications, and upper bounds on power and pin count determined by the selected package. The goal of the design is to minimize area.

Of course, there are numerous other constraints on the design. The design must work correctly over a range of operating conditions and processing variations. The external interface usually must be compatible with standard electrical levels (TTL) and bus protocol used by other chips. Future versions of the design, such as *shrinks* with improving technology, must be considcrcd. The process of establishing goals and constraints during the design is iterative. The effort spent on the design, itself, must be limited.

4. Case Study: Zilog Z8 Microcomputer

The Zilog Z8 8-bit microcomputer integrates program memory, register file, clock generator, 2 counter/timers, 1 UART, and 32 bits of I/O on a single chip [35]. The Z8 is intended for low-cost applications, such as computer terminals, printers, and toys.

The Z8 architecture supports 3 address spaces: program memory space, data memory space, and register file. Program and data memory spaces are each 64K bytes. The register file is 256 bytes. The first 4 bytes of the register file are used to address I/O ports. The last 16 bytes of the register file arc used to address control registers. One control register is used as a pointer to a set of 16 working registers in the register file. This allows working registers to be addressed using only a 4-bit code.

An instruction is 1, 2, or 3 bytes long. There are 43 operations oriented toward simple byte processing (e.g., move, arithmetic, logical, and shift) and simple program control (e.g., conditional jump, call, and return). The addressing modes allow registers to be used as accumulators, as pointers into the register file or data memory, and as index values. Other addressing modes are immediate, PC relative, and absolute address.

The Z8 is flexible in defining the function of the counter/timers, UART, and I/O ports. The ports are programmed in groups as input, output, or bidirectional (open-drain). The I/O lines can be programmed to use for external memory access. The interrupts from the counter/timers, UART, and I/O ports are programmably maskable and prioritized.

Pat Lin, one of the designers, provided the information presented here about the Z8601 implementation of the Z8 architecture [19]. A chip plan is shown in Figure 4-l. In this implementation the first 2K bytes of program memory are addressable in a maskable ROM on the chip: the remaining program memory is addressable externally. The first 2K bytes of data memory are not addressable; the remaining data memory is addressable externally. The first 128 bytes and last 16 bytes of the register file are addressable on the chip; the other registers are not addressable.

The instruction path, data path, and ALU arc each 8 bits wide. The processor is controlled with a microcode ROM that has a word width of 60 bits. Each microcode word controls 2 machine cycles and the selection of the next microaddress. A pipeline allows the fetch of 2 instruction bytes to overlap execution.

A typical instruction adds one working register to another and stores the result into the second register. This instruction is 2 bytes long. Each byte of the instruction requires 3 cycles to fetch. The instruction requires 5 cycles to execute: 1 cycle to fetch each working register into ALU temporaries, 1.5 cycles to

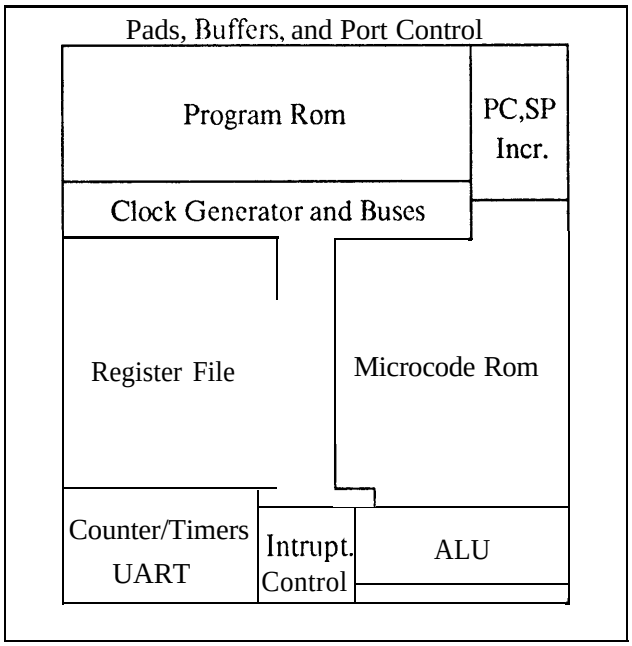


Figure 4-l: Z8 Chip Plan

perform the addition, and 1.5 cycles to store the result. The execution is therefore completely overlapped with the fetch, and the instruction effectively requires 6 cycles. The average instruction execution time is approximately 9 cycles.

The die size is 5.61 mm by 5.82 mm (221 mil by 229 mil) for an area of 32.7 mm^2 (50.6 $Kmil^2$). The maximum power consumption is 950 mW. This occurs under conditions of 0 degC temperature, 5.25 V power supply, and worst-case processing variations. The cycle time is 250 ns.

The technology is depletion load nMOS with 2.5 levels of conduction (metal, polysilicon, and diffusion). The minimum channel length is 5 um (worst-case) and the metal pitch is 12 urn.

Several packages are used with this design. The one that most constrains the design is a 40-pin, plastic, dual-in-line package with thermal resistance of 60 degC/W. The 40 pins arc allocated as follows.

Pin Function	Noi n s
vcc	1
GND	1
Timing XTAL	2
Reset	1
External memory timing	3
I/O ports	32

8 of the I/O port pins can be time-multiplexed for address and data to refcrcncc external memory. 4 or 8 additional pins can be used for external address.

The different function units on the chip were carefully balanced for area, power, and speed. The tradcoffs are shown in Table 4-1. The numbers are accurate within 10%. The values for power are based on a total consumption of 950 mW. The function units include associated decoders, latches, and control. The function unit for "PC,SP" includes the 16-bit program counter, 16-bit stack pointer, and incrementing logic. "Other" includes the clock generator, buses, dead area not in the pad boundary, and small pieces of random logic not included in any other function unit. The cycle time reported for "ALU" and "PC,SP" is actually the delay through combinational logic. The access time listed for memory is the cycle time less the column precharge time.

Function	Area			Power			Speed	
	Total		Per Bit	Total		Per Bit	Cycle	Access
	mm^2	%	um^2	mW	%	u w	ns	ns
Program	4.6	14	280	30	3	2	750	500
Register	3.5	11	3500	180	19	180	250	125
Microcode	3.8	12	620	110	11	170	500	375
ALU	1.6	5	200000	110	11	13000	375	
PC, SP	1.5	5	96000	120	13	7700	500	
UART	2.3	7		130	14			
Interrupt	1.0	3		25	3			
Pads	9.5	29		220	23			
Other	4.9	15		25	3			

Table 4-l: Z8 Area/Power/Speed Tradeoffs

The 3 memory functions have different cycle times. The most significant factor in the design is the program ROM. The memory cycle time of 750 ns matches the requirements of implementing a large ROM on-chip and accessing standard memories off-chip. The processor cycle time of 250 ns is determined by the register file. It is necessary to use sense amplifiers on the register file to achieve a read access time of 125 ns. The microcode ROM is accessed every other processor cycle rather than every cycle. This undesirably wastes a portion of some microcode words, and increases the microprogram branch penalty. Thcsc disadvantages are outweighed by the advantages of fewer bits to specify the next microaddrcss and lower power consumption. Details of the memory designs are specified below.

The program ROM is logically organized as 2K by 8 bits. It is physically organized as 2 banks with 64 rows and 128 columns. Each bank stores 4 of the 8 referenced bits. The size of a bit cell is 12 um by 16 um (192 um^2). Sense amplifiers are not used. The bandwidth available from the program ROM is 10.7 Mbps. The layout allows expansion to 4K bytes.

The register file is logically organized as 124 by 8 bits. (The other addresses are mapped to control registers.) It is physically organized as 31 rows by 32 columns. The size of a bit cell is 48 um by 52 um (2496 um^2). The bit cell is single-ported. Sense amplifiers arc used for read access. Approximately 40% of the reported power is used by the decoder and sense amplifiers; the remainder is used in the storage cells. The bandwidth available from the register file is 32 Mbps.

The microcode ROM is logically organized as 104 by 60 bits. It is physically organized as 52 rows by 120 columns. The microcode ROM uses the same bit cell as the program ROM. Sense amplifiers arc not used. The bandwidth available from the microcode ROM is 120 Mbps.

The area and power per bit for the different memory functions are distorted by including the decoders, latches, sense amplifiers, and associated control logic. This is particularly the case for the microcode ROM function, which includes not only the ROM array but also address decoders, field decoders, next address generation, and related control. Despite these distortions it is not misleading to observe that the program ROM and register file differ in area per bit by 1 order of magnitude and in power per bit by 2 orders of magnitude.

5. ROM Design

Each processing technology offers to circuit designers a range of low-level tradeoffs for area and power. An example of these tradeoffs is presented by 3 circuits used to implement ROMs.

5.1 NOR Structure

Figure 5-l shows a block diagram for one output bit in a ROM similar to the Z8 program ROM. The address is separated into two portions to access the ROM array. The row address of x bits is decoded into 2^x row select lines, and the column address of y bits is decoded into 2^y column select lines. A bit is programmed at the intersection of each row and column by the presence or absence of a transistor. The array would be replicated for each output bit of the ROM.

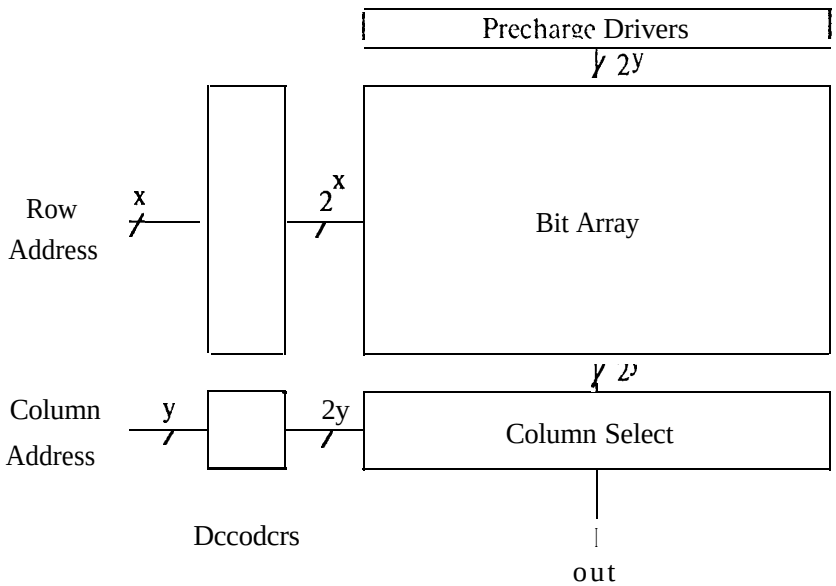


Figure 5-l: NOR ROM Block Diagram

Figure 5-2 shows a circuit diagram for column i of the ROM. Figure 5-3 shows the layout for a bit cell. When the cross-hatched portion of the layout is masked for diffusion, then a 0 is programmed.

The circuit operates by first driving all the row selects to a low voltage, and prccharging all of the columns

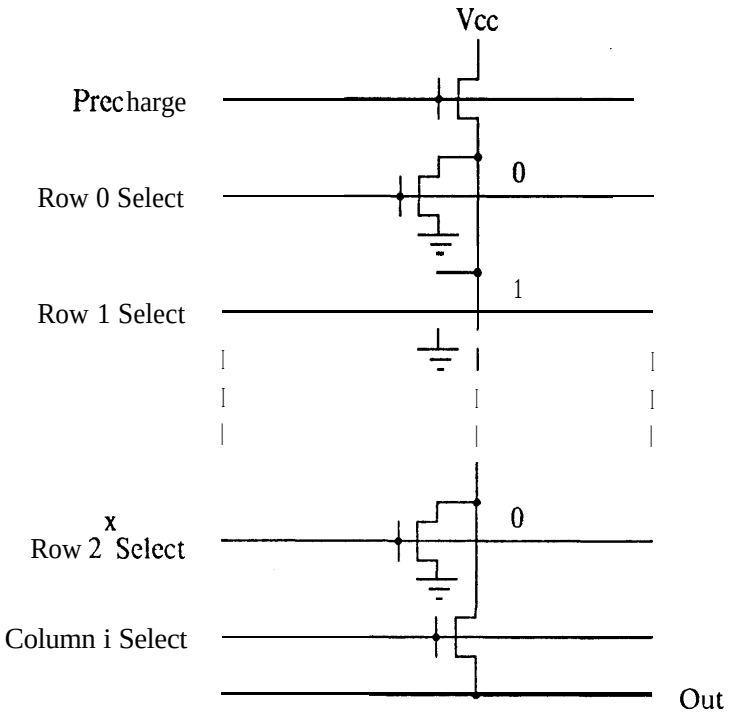


Figure 5-2: NOR ROM Circuit Diagram

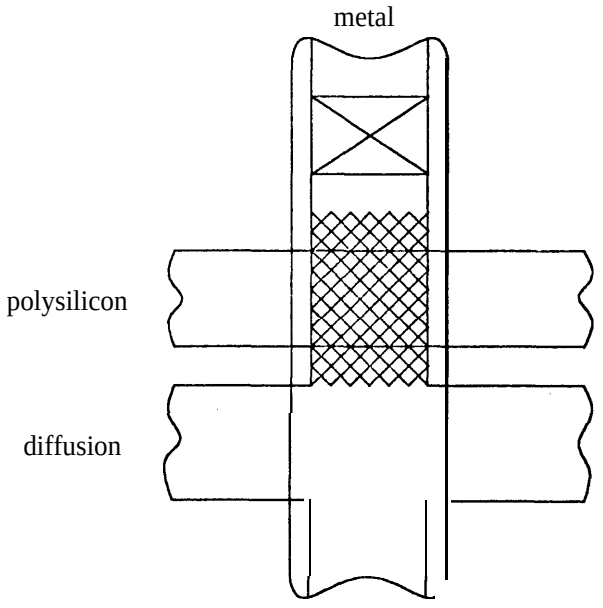


Figure 5-3: NOR ROM Bit Cell Layout

to a high voltage. The precharge driver is then turned off, and the selected row is driven high. Where a transistor is located on the sclcctcd row, the corresponding column is discharged. The sclcctcd column is gated to the ROM output. This structure is called *NOR* because driving any row to I can force a column to 0.

In designing this ROM, tradeoffs are made among speed, power, and decoder area. The number of rows determines the height of a column. This height is directly related to the capacitance that must be charged by the precharge driver, and discharged by the bit cell for each access. The number of columns determines the load on a row decode driver. With current MOS processes the row select is usually polysilicon, which has a relatively high resistivity. To improve access time, sense amplifiers can be used to detect changes in column voltage much less than between logic values. Cragon mentions that in single output ROMs the number of rows and columns is made equal, to minimize decoder area [9]. Mead and Rem present a theoretical treatment of these organizational tradeoffs [20].

5.2 NAND Structure

A form of NAND-structured ROM is used in the Hewlett-Packard microprocessor design of section 2.3. A bit is programmed at the intersection of each row and column by placing an enhancement-mode or depletion-mode transistor. Figure S-4 shows a simplified circuit diagram for one column of a typical NAND ROM, first described by Kawagoe and Tsuji [16]. Figure 5-5 shows the layout for a bit cell.

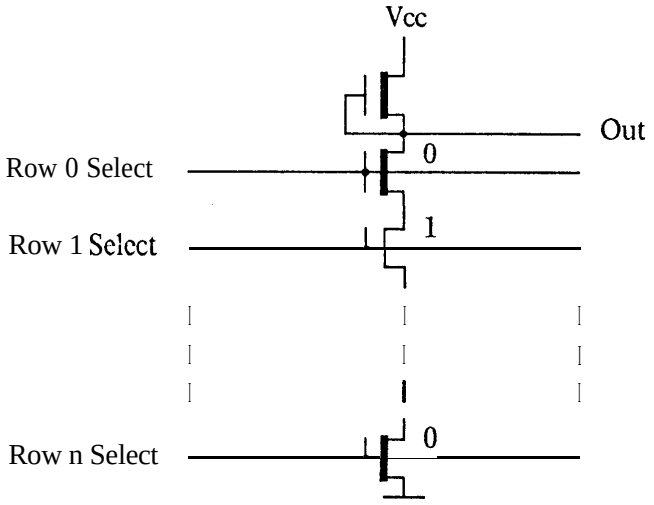


Figure 5-4: NAND ROM Circuit Diagram

The circuit operates by driving the selected row to a low voltage while the other rows are driven to a high voltage. The series transistors in the column conduct to ground only when the selected row has a depletion-mode transistor. This structure is called *NAND* because the output is 0 when all of the series transistors are conducting.

The bit cell for a NAND-structured ROM is usually smaller than the bit cell for a NOR-structured ROM because there is no ground line or metal contact. Lin reports the NAND cell area (9 um by 7 urn) to be 30%

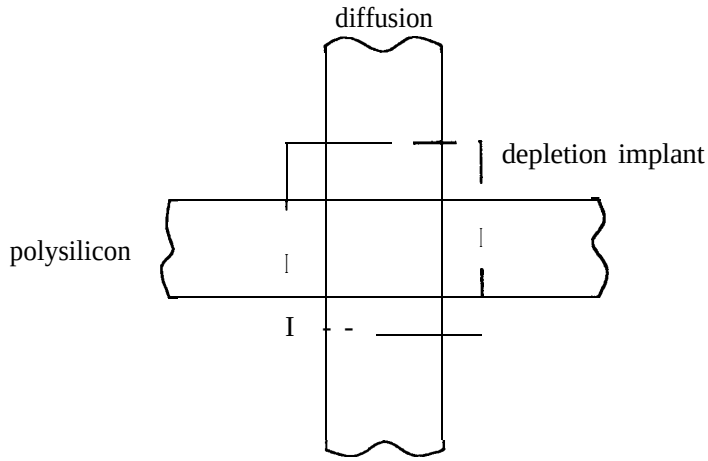


Figure 5-5: NAND ROM Bit Cell Layout

of the NOR cell area (14 urn by 14.5 urn) for a 4 um nMOS technology [18]. (See this reference for details of a practical design.) The NAND structure has generally slower access and limited number of bits per column, compared to the NOR structure, becuase of the series transistors. The NAND-structured ROM is also more sensitive to processing variations.

5.3 Multiple Bits Per Cell

A modification of the NOR-structured ROM allows more than 1 bit to be programmed in each cell. Instead of programming a single bit by the absence or presence of a transistor, 2 bits can be programmed by the absence of a transistor or the presence of a transistor with 3 possible sizes. Sensing the state of the multiple-bit cell requires more complex circuitry.

The iAPX432 Interface Processor uses this technique [2]. The designers report that the l-bit cell size is 15 urn by 14 urn, and the 2-bit cell size is 15 urn by 15 urn. The area savings per bit is 46%. The area savings for the 32Kb ROM, including decoders and sense amplifiers, is 41%.

The designers of the NS16032 microprocessor examined the choice of storing 1 bit per cell or 2 bits per cell in the microcode ROM [15]. They report the 2-bit cell is also 15 um by 15 um, but the l-bit cell is 12.5 um by 12.5 urn. The National Semiconductor and Intel technologies arc similar; the difference in area between the l-bit cells appears to be due to contact rules. The NS16032 designers decided that the area savings per bit of 25% was insufficient to justify the extra delay and complexity of the 2-bit cell.

6. Architecture and Design

6.1 Memory Costs

The trend in microprocessor design toward integrating increasing quantities of memory (ROM, RAM, and CAM) on-chip has been mentioned. In a balanced design the circuits for each type of memory have a cost in area and power that depends on technology and design goals. One of the major challenges of microprocessor design is to find the most effective ways of combining memory functions. Table 6-1 shows the difference in bit-cell area for different types of memory implemented in actual designs.

Design	Bit Cell Area (urn^2)		
	ROM	RAM	CAM
H-P	10	1155	
Z8	192	2496	
28015		3360	4760, 5320

Table 6-1: Bit Cell Area (um^2)

The Hewlett-Packard design [5] was described in section 2.3. The nMOS technology allows 1.5 urn channel length and 2.5 um first-level metal pitch, providing a very compact ROM cell. The RAM cell is actually a special purpose register cell using 11 transistors. The register provides dual access ports, and uses dynamic feedback to reduce power consumption. The difference in area between ROM and register cells is unusually large compared to other designs.

The Z8 design was described in chapter 4. The nMOS technology allows 5 urn channel length and 12 um metal pitch. The RAM cell in the register file is single-ported, uses 6 transistors, and has static feedback. A circuit diagram for the cell, shown in Figure 6-1, is typical of RAM used in nMOS microprocessor design.

The Zilog 28015 is a memory management unit used with the Z8000* microprocessor family. The information here is provided by J. K. Tsai, one of the designers [32]. The circuit translates a 12-bit logical page number into a 13-bit physical page frame. The logical page number includes a 7-bit segment number and 5 bits of segment offset; the segment number is valid earlier than the offset. The design uses the same technology as the Z8 design.

*Z8000 is a registered trademark of Zilog, Inc.

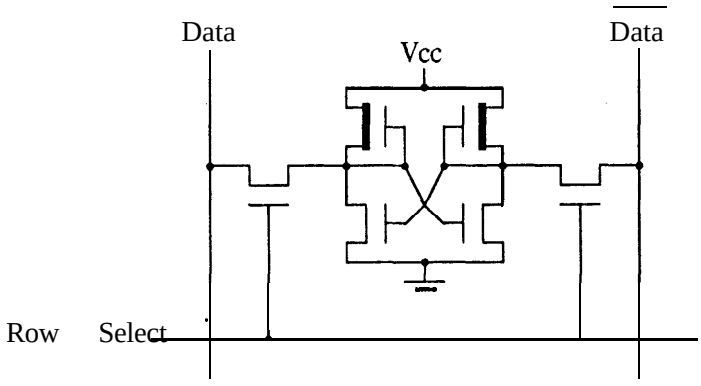


Figure 6-1: RAM Cell Circuit Diagram

The total number of logical-physical page translation pairs stored in the chip is 64. These are located in 2 arrays of 32 entries. The mapping is fully associative; a comparator is built into each entry. If the logical address matches the CAM content in any valid entry, then the corresponding physical address is read onto a bus.

Each entry is 70 um by 2033 urn, containing 12 CAM cells for the logical page number, 13 RAM cells for the physical page frame, 7 RAM cells for page attributes (e.g., valid bit and protection), and related match logic. The segment number CAM cell is 70 um by 68 urn, the segment offset CAM cell is 70 um by 76 urn, and the RAM cell is 70 um by 48 urn. (Notice that using the same technology and circuit topology but with different design goals, the areas of the RAM cells differ by 35% between the Z8 and Z8015.) Each entry uses approximately 10 mW of power. The delay from logical address valid in the CAM array to physical adress valid from the sense amplifiers is 90 ns. Figure 6-2 shows a circuit diagram for the 9-transistor CAM cell.

6.2 Differential Architecture

To see how the choice of architecture and technology influences design tradeoffs, consider the following idealized model. A RISC-like architecture and typical workload have been selected. A chip plan for the implementation is shown in Figure 6-3. The control storage section has N_{CS} words, each word occupies area a_{CS}, and the total area (including support circuitry) is A,. The cache has N_{CA} lines, each line occupies area a_{CA}, and the total area is A,.

A proposed change to the architecture would add an instruction. Consider only instructions that exactly replace a sequence of instructions, and have the same coding format as the original architecture.' For RISC examples arc bounds check with lower bound of 0 and memory to register add. (General memory-to-register

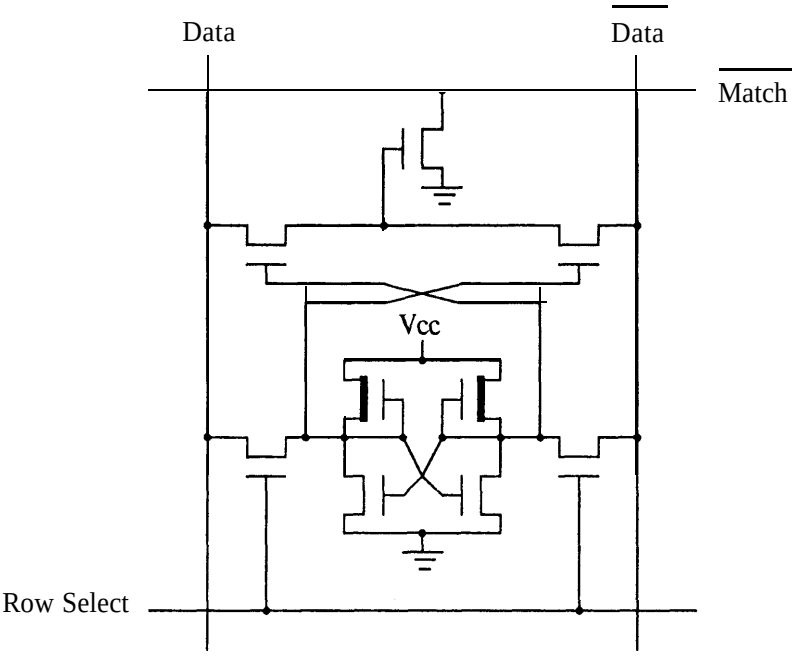


Figure 6-2: CAM Cell Circuit Diagram

architecture is no more troublesome than load/store architecture from the standpoint of page-fault recovery.) The original workload is transformed into a modified workload under the new architecture.

The original implementation is modified for the proposed architecture. No changes are made to the data path. By nature of the restriction on added instructions, the implementations require exactly the same number of cycles to execute their respective workloads. The new implementation, however, has n_{CS} additional words of control storage. Some modification of the control structure is necessary to support multiple-cycle execution.

The proposed change to the architecture reduces the program size of the workload. This reduction in program size improves the effectiveness of the cache because fewer misses are generated in executing sequential code before branching, larger loops fit in the cache, and more small loops simultaneously fit in the cache. (That fewer or equal number of misses are generated may be taken as obvious, or can be proved for stack replacement algorithms.) In the new implementation the cache is made smaller until the number of misses for the new workload is equal to that with the old implementation and workload. The number of lines eliminated is n_{CA}.

Assuming that speed-power tradeoffs remain the same, the original design and the modified design have the same cycle time, require the same number of cycles to execute, and have the same main memory

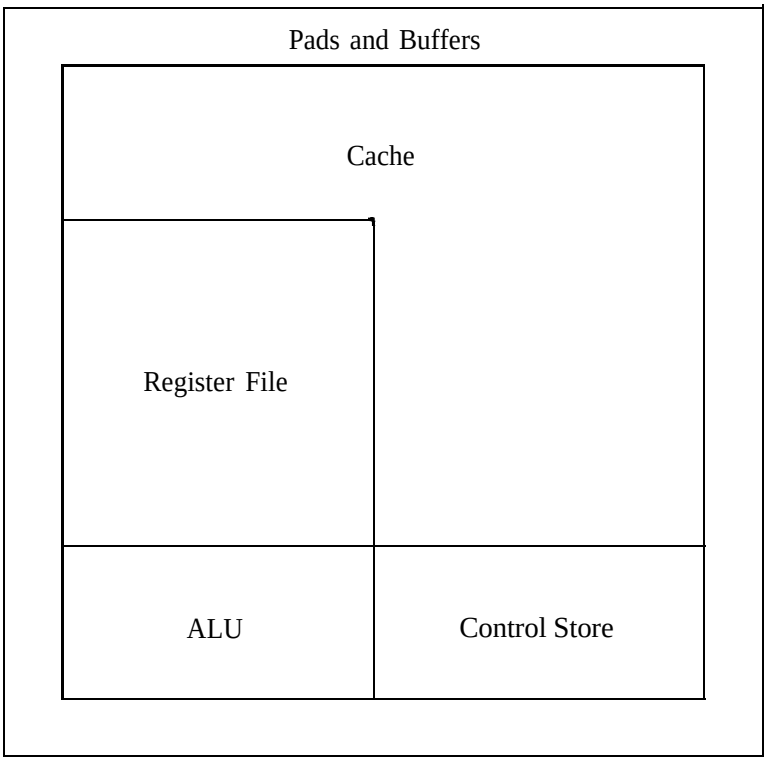


Figure 6-3: Implementation Chip Plan

bandwidth requirement. The difference is that the new design has n_{CS} additional words of control storage and n_{CA} fewer lines of cache. The proposed change is cost-effective if

$$n_{CS}a_{CS} < n_{CA}a_{CA}.$$

The preference for one design over the other depends on workload, technology, cost goals, and performance goal;. The workload determines the value of the proposed change in reducing program size and improving cache effectiveness. The processing technology defines a space of allowed circuit and mask designs for ROM, RAM, and CAM bit cells. The cost and performance goals constrain the size of cache and control store. This study has described the large range of choices available for high-level and low-level design tradeoffs. It is not possible to say, *apriori,* which of the two architectures allows a more cost-effective design.

Models similar to the one in this section can be developed to analyze the tradeoffs between register file and control storage or between register file and cache.

6.3 RISC vs. CISC

There are system-wide advantages to making code more compact than RISC-like instructions, even when variable-length instructions must be decoded, and multiple cycles are needed to interpret instructions. The bandwidth requirement for fetching instructions is reduced. This can allow longer memory access time and a narrower memory data path. The size of main memory needed to store programs (or the working set for demand virtual memory) is less. Generally, the cost of memory dominates the cost of the CPU in microprocessor systems. It may be argued that memory costs are declining. Nevertheless, processor performance is improving, and main memory size is directly related to instruction execution rate in a balanced system. RISC programs are reported to be approximately 50% larger than equivalent VAX programs [25].

The previous section considered highly restricted changes to a RISC-like architecture. Other changes may allow the implementation to reduce the number of execution cycles by increasing the functionality of the data path or increasing parallelism. A scaled-index addressing mode, as in VAX, can eliminate moves and shifts. A multiply instruction eliminates cycles when using multiple-bit retirement and special, loop-counting hardware. A memory block move instruction allows overlap of data memory access with modification of source address, destination address, and loop-counter. To reiterate, instructions that are infrequent among total instructions are often performance bottlenecks.

Some potential microprocessor applications, such as business processing, do not fit well into the RISC programming model of simple variables and procedural control. For these applications and a RISC-like processor the choices are the cost of storing very large programs or the performance degradation of software interpretation.

At some point adding to the function in the architecture can lead to a less cost-effective design. The location of that point is not clear; it depends in detail on workload, technology, cost goals, and performance goals. There is no general preference for a particular level of architectural complexity. There are, however, some basic principles to recognize. Compact code improves the effectiveness of cache memory and diminishes the system requirements for bandwidth and quantity of main memory. High-level functionality in the architecture allows optimizations by hardware and firmware. Increasing the functionality of the architecture may reduce the resources available for cache and register storage in a single-chip microprocessor.

6.4 Direction for Architecture

Following the principles above, some desirable characteristics for a microprocessor architecture are proposed. The architecture should provide a high level of functionality, and allow compact code. The architecture should have efficient trapping mechanisms to permit software interpretation of complex

operations, thus permitting an implementation-dependent quantity of control storage. In this way the architecture would be similar to indirect threaded-code systems, described by Kogge [17]. The architecture should support stack allocation of on-chip data storage with a mechanism similar to RISC or the C-machine stack cache [11], thus allowing an implementation-dependent quantity of data storage. An architecture with these characteristics has the properties of transparency to control storage size, transparency to data storage size, and uniformity of direct execution and software interpretation. An architecture with these properties may be called a Virtual Instruction Set Processor (VISP). The implementaion of a VISP is given great freedom in allocating resources to data storage and control storage when designing for different applications, processing technologies, cost goals, and performance goals.

One example of an architecture with these properties is Nebula, the standard architecture of the Military Computer Family [10, 31]. VISP architectures are being studied: the results will be reported later [I].

7. Summary

This study has described the practice of microprocessor design as viewed by one computer architect. I have attempted to explain the general problems confronting designers of commercial microprocessors, and also to provide sufficient detail that the reader can appreciate the tradeoffs available in design.

Designing a commercial microprocessor is a complex problem involving decisions that require expertise in several disciplines. The architecture most suitable for a design depends on the application workload, processing technology, cost goals, and performance goals. No particular architecture is best for all designs. Specifically, a RISC-like architecture may result in a less cost-effective design than an architecture with instructions of variable length and instructions requiring multiple execution cycles. One direction for architecture has been suggested, and is being studied.

References

[1] Donald Alpert.
A Virtual Instruction Set Processorfor VLSI.
PhD thesis, Stanford University, 1983.

[2] John A. Bayliss èt al.
The Interfacc Processor for the 32b Computer.
In *Digest of Technical Papers, 1981 IEEE International Solid-State Circuits Conference,* pages 116-117. February, 1981.

[3] John A. Bayliss et al.
The Interface Processor for the Intel VLSI 432 32-Bit Computer.
IEEE Journal of Solid-State Circuits SC-16(5):522-530, October, 1981.

[4] Robert Bernhard.
More Hardware Means Less Software.
IEEE Spectrum 18(12):30-37, December, 1981.

[5] Joseph W. Beycrs et al.
A 32-Bit VLSI CPU Chip.
IEEE Journal of Solid-State Circuits SC-16(5):537-542, October, 1981.

[6] Joseph W. Beyers et al.
A 32b VLST System.
In *Digest of Technical Papers, 1982 IEEE International Solid-State Circuits Conference,* pages 128-129. February, 1982.

[7] Mark Brodsky.
private communication, 1982.

[8] David L. Budde et al.
The Execution Unit for the VLST 432 General Data Processor.
IEEE Journal of Solid-State Circuits SC-16(5):514-521, October, 1981.

[9] Harvey G. Cragon.
The Elements of Single-Chip Microcomputer Architecture.
Computer 13(10):27-41, October, 1980.

[10] Department of Defense.
Instruction Set Architecture for the Military Computer Family
May 28, 1980.
MIL-STD-1862.

[11] David R. Ditzcl and H. R. McLellan.
Register Allocation for Free: The C Machine Stack Cache.
In *Proceedings, Symposium on Architectural Support for Programming Languages and Operating Systems,* pages 48-56. March, 1982.

[12] Daniel T. Fitzpatrick et al.
A RISCy Approach to VLSI.
Computer Architecture News 10(1):28-32, March, 1982.

[13] Anil Gupta, W. A. Porter, and Jay W. Lathrop.
Defect Analysis and Yield Degradation of Integrated Circuits.
IEEE Journal of Solid-State Circuits SC-9(3):96-103, June, 1974.

[14] Intel Corporation.
Introduction to the iAPX 432 Architecture
1981.

[15] Asher Kaminker et al.
A 32-Bit Microprocessor with Virtual Memory Support.
IEEE Journal of Solid-State Circuits SC-16(5):548-557, October, 1981.

[16] Hiroto Kawagoe and Nobuhiro Tsuji.
Minimum Size ROM Structure Compatible with Silicon-Gate E/D MOS LSI.
IEEE Journal of Solid-State Circuits SC-11(3):360-364, June, 1976.

[17] Peter M. Kogge.
An Architectural Trail to Threaded-Code Systems.
Computer 15(3):22-32, March, 1982.

[18] Chong Ming I-in.
A 4 um NMOS NAND Structure PLA.
IEEE Journal of Solid-State Circuits SC-16(2):103-107, April, 1981.

[19] Pat Lin.
private communication, 1982.

[20] Carver Mead and Martin Rem.
Cost and Performance for VLSI Computing Structures.
IEEE Journal of Solid-State Circuits SC-14(2):455-462, April, 1979.

[21] Motorola Semiconductor Products Inc.
The MC68020 Enhanced M68000 Microprocessor Product Preview
1982.

[22] B. T. Murphy.
Cost-Size Optima of Monolithic Integrated Circuits.
Proceedings of the IEEE 52(12):1537-1545, December, 1964.

[23] National Semiconductor Corporation.
NS16082 Memory Management Unit (MMU)
March, 1982.

[24] David A. Patterson and David R. Ditzel.
The Case for the Reduced Instruction Set Computer.
Computer Architecture News 8(6):25-33, October, 1980.

[25] David A. Patterson and Carlo H. Sequin.
RISC I: A Reduced Instruction Set VLSI Computer.
In *Conference Proceedings, The 8th Annual Symposium on Computer Architecture,* pages 443-458. May, 1981.

[26] David A. Patterson and Carlo H. Sequin.
Design Considerations for Single-Chip Computers of the Future.
IEEE Transactions on Computers C-29(2):108-116, February, 1980.

[27] John E. Price.
A New Look at Yield of Integrated Circuits.
Proceedings of the IEEE 58(8): 1290-1291, August, 1970.

[28] George Radin.
Th 801 Minicomputer.
In *Proceedings, Symposium on Architectural Support for Programming Laanguages and Operating Systems,* pages 39-47. March, 1982.

[29] William R. Richardson et al.
Thc 32b Computer Instruction Decoding Unit.
In *Digest of Technical Papers, 1981 IEEE International Solid-State Circuits Conference,* pages 114-115. February, 1981.

[30] Leonard Jay Shustek.
Analysis and Performance of Compu ter Instruction Sets.
SLAC Report 205, Stanford Linear Accelerator Center, Stanford University, May, 1978.

[31] Leland Szewerenko, William B. Dietz, and Frank E. Ward, Jr.
Nebula: A New Architecture and its Relationship to Computer Hardware.
Computer 14(2):35-41, February, 1981.

[32] J. K. Tsai.
private communication, 1982.

[33] R. M. Warner, Jr.
Applying a Composite Model to the IC Yield Problem.
IEEE Journal of Solid-State Circuits SC-9(3): 86-95, June, 1974.

[34] Zilog, Inc.
Z8015 Paged Memory Management Unit Product Specification
April, 1981.

[35] Zilog, Inc.
Microcomputer Components Data Book
1981.

[36] Zilog, Inc.
28300 Low Power Z80L CPU Central Processing Unit Product Specification
December, 1981.