An Integrated Associative Processing System

by

Frederick Paul Herrmann

S.B., Electrical Engineering S.B., Mathematics
Massachusetts Institute of Technology (1987)

S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology (1989)

Submitted in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy

in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September, 1994

Signature of Author [illegible]
Department of Electrical Engineering and Computer Science
July 28, 1994

Certified by [illegible]
Professor Charles G. Sodini
Thesis Supervisor

Accepted by [illegible]
Professor Frederic R. Morgenthaler
Chairman, Department Committee on Graduate Students

Eng.
WITHDRAWN
FROM
MIT LIB
NOV 18 1994

© Massachusetts Institute of Technology, 1994
All rights reserved

An Integrated Associative Processing System

by

Frederick Paul Herrmann

Submitted to the
Department of Electrical Engineering and Computer Science
on July 28, 1994 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Abstract

The design and implementation of an integrated associative processor is described. The chip is intended for use as a coprocessor in a system for accelerating pixel-parallel image processing tasks on desktop workstations and personal computers. Each of the chip's 256 processing elements includes 64 trits of three-state memory, a function generator, an activity register, and connections to a reconfigurable mesh network and a response resolution subsystem. The device-intensive design style makes extensive use of dynamic logic in dense pitch-matched circuits.

A novel dynamic associative memory cell supports fully-parallel match and write operations. The cell uses five MOS transistors, including two overlapping dual-gate devices available in double-polysilicon processes. The necessary sense and driver circuits are discussed, with particular attention to switching noise considerations.

The processing elements communicate over an Asynchronous Reconfigurable Mesh network which provides a mechanism for simultaneous broadcasting over multiple connected regions. The area-efficient implementation requires only one driver circuit per processing element. The network is readily extendible across chip boundaries, so that large arrays can be constructed with multiple chips.

A new response resolution circuit is used to count and prioritize selected processing elements. A compact pass transistor chain is used within the confines of the array, and results from multiple arrays are combined in a tree structure.

Results are presented which verify the full functionality of all chip subsystems. The performance evaluation of the design is complicated by the fact that the chip was fabricated in a process other than that for which it was intended, but experimental results are shown to be consistent with measured process parameters. A 6-V supply is used to compensate for the high (2.1 V) second-polysilicon threshold voltages. Write and match instructions are performed in 175 ns, and 220 mW are dissipated in typical applications. The results of storage time experiments examining charge pumping, spooning, and leakage effects are presented.

A sample image processing application is demonstrated on a prototype system with four chips and 1024 processing elements. Several ideas for further work are discussed in the conclusion.

Thesis Supervisor: Charles G. Sodini
Title: Professor of Electrical Engineering

Acknowledgements

I am grateful to my thesis advisor, Professor Charles Sodini, for his encouragement and guidance, both technical and personal. I have been privileged to be a member of his technically diverse research group, and I appreciate the opportunities I have had to broaden my experience under his supervision.

Jon Wade is gratefully acknowledged as my predecessor in the Smart Memory Project and as the inventor of the dynamic CAM cell upon which the cell used in this work is based. I learned a great deal while sharing office space with Jon, and I am particularly appreciative of his patience in going over the DBA design with me. Jeff Gealow brought a wealth of memory design knowledge to this same office, and I gratefully acknowledge his collaboration in the Associative Processing Project. I have enjoyed working with him on the system and software aspects of our projects, and I wish him and his circuits well.

I would like to acknowledge my readers, Professor Jon Allen and Professor Greg Papadopoulos, and thank them for their interest in this work. Their suggestions have helped me produce a better and more readable document.

I further acknowledge past and present members of the Associative Processing Project: Larry Hsu designed and built the system controller, which proved invaluable for chip testing. Chad Mikkelson was of great help to me in the lab, gathering most of the charge transfer data presented herein. Daphne Shih constructed the demonstration I/O system, and she is now working on a full-scale implementation. Keisuke Ishio designed the first cell test circuits; I thank him and Sharon Britton for sharing their real-world design experience. Thanks also to Phil Chu, John Maeda, Ahmed Mitwalli, and Shih-Jih Yao, all of whom contributed to the early associative processor software.

Craig Keast developed the MTL CCD-CMOS process and graciously accommodated the cell test circuits on his wafers. My thanks to him and to Jeff Gross, Kathy Krisch, and Joe Lutsky for answering my device questions, and to Ken Szajda for watching over my wafers in the ICL. I am also grateful to Steve Decker and Chris Umminger for help with measurements and much useful information, and to Curtis Tsai for convincing me that learning more about UNIX would be worthwhile.

I have greatly benefited from my association with the Vision Chip Project, and I appreciate their willingness to consider digital solutions. I particularly thank Professor John Wyatt for his leadership of the group and Professor Harry Lee for his always well-considered advice.

I was fortunate to accompany Prof. Sodini to UC Berkeley in the spring of 1990. Much of the architecture was still crystallizing during this period, and I thank Krste Asanović for his helpful comments on my partially formed ideas.

Pat Varley and Carolyn Zaccaria are skilled navigators of the M.I.T. bureaucracy, and I have been glad of their assistance on many occasions.

I appreciate the work of our computer system managers, both official and pro tempore: Vince Loh, Thomas Lohman, Giampiero Sciutto, and "Fletch" Freeman. I thank Professor Donald Troxel for access to the instructional laboratories, and I thank the MTL staff for fabrication, packaging, and measurement services.

A friend observes that nothing affects one's job satisfaction so much as the people with whom one works, and my experience confirms that this applies to grad school as well. This place abounds in talented graduate students and all-around good folks. Along with those already mentioned, I am glad to have known and worked with Michael Ashburn, Monica Choi, Kamyar Eshghi, Rod Hinman, Merit Hong, Jarvis Jacobs, Andrew Karanicolas, Jen Lloyd, Dave Martin, Vince McNeil, Ignacio McQuirk, Shujaat Nadeem, Ken O, Patrice Parris, Gee Rittenhouse, Julie Tsai, Bing Wang, and Paul Yu.

While at Berkeley I had the pleasure of meeting Eric Boskin, David Cline, Cormac Conroy, Dave Helman, and Sherry Lee. I remain grateful to them and the rest of the Cory Hall crowd for making me feel welcome and sparing me some space in front of a workstation.

I thank my parents for their unfailing love, support, and encouragement throughout my student career. Their prayers for me are truly appreciated. I also thank my in-laws for their kindness and for their patience with my nocturnal habits during these last difficult months.

Finally and most importantly, my thanks and love to Julie, who means everything to me.

This work was supported by grants from NSF and ARPA (Contract MP-9117724), Digital Equipment Corporation, and IBM. Integrated circuit fabrication was provided by MOSIS.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Associative processors are parallel computing machines that perform both memory and processor functions in a single unit. They are therefore somewhat unconventional, since the two components are usually viewed as being completely separate. This perception affects the way computers are understood at every level, from design and manufacturing up through architecture and programming. In the field of circuit design, for example, memory and processor specialists have segregated themselves into distinct subdisciplines. Even in companies that make their primary business in processors, it is common practice to have one team of expert memory designers develop on-chip caches or control stores for several products. These memory subsystems are then "thrown over the wall" for other groups to incorporate into their own chip designs.

Given that memory subsystems can be integrated onto processor chips, it is natural to ask whether processing functions could be added to memory chips. This idea is especially attractive to memory designers, who look to the precedent established by video memories [1] and see that the addition of a few shift registers can turn a price-competitive commodity product into a highly profitable specialty part. Video memories represent an incremental approach to smart memory, but processor and memory can be combined more effectively when starting from scratch. Original designs afford greater flexibility to implement processor functions in the memory design style, which emphasizes large arrays of hand-crafted circuits. Because memory designers optimize each transistor of their key circuits, they must repeat these circuits many times over in order to justify their attention to detail. The design style is well matched to the fine-grained organization of an associative processor, in which every memory word acts as one processing element (PE) of a highly parallel system.

1.1 An Image Processing System

In the early years of associative processing, the pioneers of the field recognized picture processing as a promising source of applications for their new machines [2, 3]. Many image processing problems are rich in parallelism, with thousands of pixels receiving

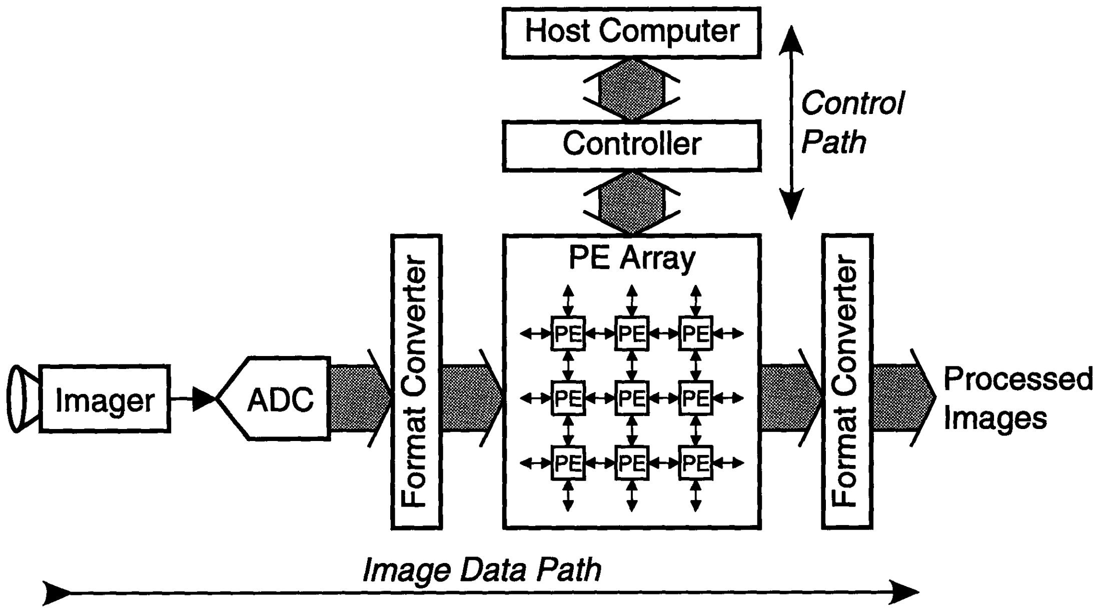


Figure 1.1: Image processing system using array of associative processing elements.

identical processing steps. The low precision of image data (typically 8-bit integers) and the often modest computational requirements at each pixel match the capabilities of very simple associative processing elements. Modern integrated circuit technology provides the density necessary to produce large arrays at low cost, with each PE assigned to a single pixel.

Similar arguments have been made in favor of analog circuits for early machine vision [4, 5]. Parallel arrays of relatively simple circuits can achieve adequate precision for many applications, and analog circuits have desirable speed, power, and density characteristics. The only drawback to these systems is their lack of generality; most are designed to solve only a single problem [6, 7, 8]. In contrast, massively parallel supercomputers provide all the flexibility of programmable digital systems, but these million-dollar machines are overkill for all but a few select applications. Associative processors occupy a middle ground, achieving massive parallelism and appropriately fine granularity while retaining the integration benefits of the device-intensive design style.

Figure 1.1 shows a block diagram of an image processing system based on an associative processor. A two-dimensional network connects the processing elements and also handles image input and output. Image data flow from left to right in the diagram, while the vertical control path carries instructions from the host computer to the array. Several conversion steps are necessary to prepare images for processing. An imager, such as a video camera, first converts from the (typically optical) acquisition domain to the analog electronic domain. The analog-to-digital converter then produces a digital signal representation, and a format converter reorders the image bits before loading them into the associative processor. A similar format converter may be necessary at the output, depending on the destination of the processed images.

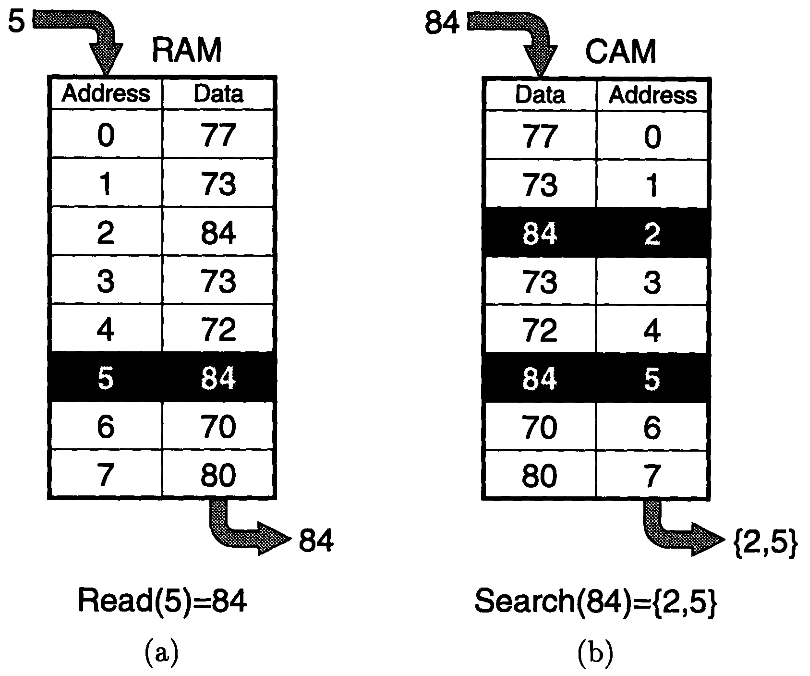


Figure 1.2: RAM read (a) and CAM search (b) operations.

Because the system of Fig. 1.1 requires very little interaction between the data and control paths, the design and construction tasks can be cleanly partitioned into manageably sized projects. This thesis concentrates on the design and integrated circuit implementation of the associative processor array. As of this writing, the control path is complete and the remainder of the data path is in the early planning stages. When the entire project is finished, the three components will constitute a coprocessor system for accelerating image processing tasks. It is believed that something very similar to this demonstration system may one day be standard equipment on workstations and personal computers, just as digital signal processing chips are now used for audio applications.

1.2 Associative Memories and Processors

If one wanted to define *associative memory* as broadly as possible, one could go so far as to claim that all memories are actually associative. The familiar random access memory (RAM), for example, is a special case in which each word is partitioned into separate address and data fields. As shown in Fig. 1.2(a), the read operation selects the word matching an input address (5), and reports the contents (84) of its associated data field. Each RAM address is hard-wired with a unique value corresponding to its physical location. It is therefore guaranteed that exactly one word will be selected for any valid input address. In contrast, a content addressable memory (CAM) can examine many words simultaneously. Figure 1.2(b) shows a CAM in which two words

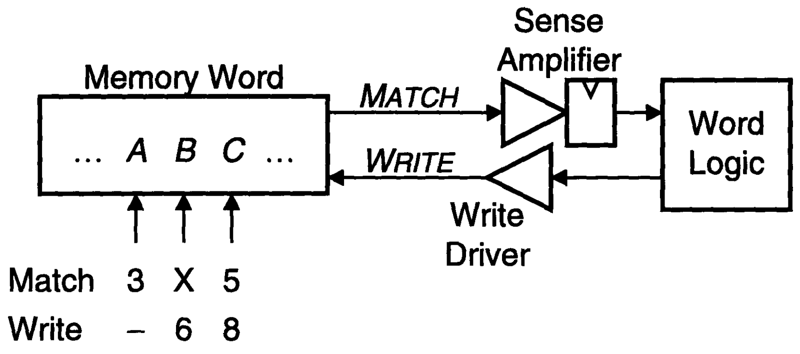


Figure 1.3: Associative processing element.

match the input value, and so two address are produced. Because the CAM's data contents are not hard-wired to predetermined values, it is not constrained to match only a single word.

There is an obvious symmetry between the RAM and CAM operations. Both compare an input value to a search field, and both produce output from a result field. Since all memories perform some restricted version of this associative match operation, all could reasonably be described as associative memories. Such a definition would be uselessly inclusive, however, and so the term *associative memory* is usually taken to exclude RAMs and other memories without modifiable search fields. Established usage accepts *content addressable* as a near synonym for *associative*, although the second adjective seems to be gaining favor.[1]

In systems with associative memory hardware, the central processor can use the search operation to find a few records of interest without needing to access many uninteresting records. Traffic from the memory to the processor is reduced, but the flow of data in the opposite direction is not much affected. Updates and modifications to the associative memory contents must still be performed sequentially, under control of the central processor. To take full advantage of the parallelism available in associative memories, the memory words must be endowed with some processing capabilities.

Figure 1.3 is a generalized diagram of a single associative processing element (PE), with its memory word divided into several fields (A, B, C, ...). The match operation finds PEs that match a presented pattern, with unused fields marked with *don't cares* (Xs). For example, the operation Match(3, X, 5) would identify all PEs with $A = 3$ and $C = 5$, and would set their sense amplifier outputs to 1. (As indicated in the figure, it is assumed that the sense amplifier includes a register or some means to maintain its state until the next match operation.) Each PE's match result passes to its word logic, which in turn controls the write driver. In this way, match results can condition

[1] *Content addressable* seems inappropriate in the absence of physically addressable words. Such is the case in certain neural and holographic systems [9, 10, 11]. However, these implementations will not be given further consideration in this work, and so the terms *associative memory* and *content addressable memory* will be used interchangeably.

later write operations. If the write driver is not enabled, the PE is masked, and its memory remains unchanged. Fields within a word can also be masked, so that some fields are modified while others are preserved. The Write$(-, 6, 8)$ operation sets $B = 6$ and $C = 8$ while leaving the contents of the A field unchanged. These match and conditional write operations turn out to be surprisingly powerful primitives, which can be combined to perform more complex tasks. Section 2.1 demonstrates bit-serial addition procedures on single-trit fields, where a *trit* is a ternary digit of value 0, 1, or X.

The example PE of Fig. 1.3 illustrates the two defining features that distinguish an *associative processor* or *content addressable parallel processor* from an associative memory [12]:[2]

- The feedback path allows match results to control later writes, and
- The masked write feature allows some parts of the word to be preserved while others are modified.

Computing systems incorporating associative processors as essential components are termed *associative computers* [14, 15].

The discussion so far has assumed *fully-parallel* operation, meaning that multiple bits of all PEs may be examined or modified simultaneously. Other organizations are also possible [16], such as bit-parallel/word-serial or bit-serial/word-parallel. The first approach allows several memory words to share the same word logic and sense amplifier [17]. The second and more common organization requires that match and write operations examine or modify only one bit of each memory word at a time [18, 19, 20]. Multibit operations take several cycles, and combining results of the single-bit matches requires relatively complex word logic. Both of these organizations give up some of the available parallelism, but they have the advantage of requiring only conventional RAM cells in their implementations. In contrast, fully-parallel systems must use more specialized associative memory cells, which historically have been relatively expensive compared to standard RAM. However, this work describes a dense new dynamic cell using only five n-channel transistors. This technology should make fully-parallel systems competitive, especially in applications requiring many relatively simple PEs.

1.3 Comparison to Other Image Processors

Pixel-parallel image processors can solve many problems in constant time, but their hardware requirements vary directly with image area. Although associative processing elements can be very compact when designed in the smart memory style, one might still wonder whether the pixel-parallel approach is too expensive. To answer the question, it is worth skipping ahead a bit to present some experimental results here in the introductory chapter.

[2]Compare also the less precise definitions of [13, 14].

Table 1.1 compares the associative processor to several other systems. The entries are arranged in order of approximate flexibility, with an application-specific analog processor [8] on the top line and a general-purpose microprocessor [21] at the bottom. In between are two pipelined convolution chips from LSI Logic's family of image processing products [22, 23, 24], the four-PE Image Signal Multiprocessor (ISMP) from Matsushita [25], and the prototype associative processor of this work. Two image processing problems are used in the comparison: a 3×3 Laplacian convolution and the nonlinear smooth and segment (S&S) operation described in Section 5.6. Per-image execution times are given for the pixel-parallel machines; per-pixel values for the others.

Two figures of merit are useful for comparative evaluations. The first of these can be understood as

$$\left(\frac{\text{Area}}{\text{Pixels per second}}\right), \tag{1.1}$$

the silicon area required to achieve a given rate of performance. Equivalently, as

$$\left(\frac{\text{Time} \times \text{Area}}{\text{pixel}}\right), \tag{1.2}$$

it reflects the resources utilized for each pixel processed. The rightmost column of the table presents the energy required for each pixel computation.

The analog focal plane processor is the clear winner by any numerical measure, but it achieves its efficiency at the expense of generality, as it performs only the smooth and segment operation. At the other end of the flexibility spectrum, the general-purpose Alpha microprocessor is by far the most resource-hungry of the systems. The remaining four systems obtain comparable scores on the time-area and energy metrics. For a given level of performance, the associative processor should not be substantially more expensive than the the other approaches, but it will be considerably more general. While the specialized pipelined parts might be preferred in certain embedded applications, the associative processor's flexibility makes it more attractive as a general-purpose image processing accelerator.

1.4 Organization of Thesis

This chapter introduced associative processing as a model of computation utilizing smart memories, and it described an image processing system based on the chips produced in this work. The remaining chapters discuss the primary contributions of this thesis, which include: 1) a new dynamic associative processor cell using dual-gate devices, 2) compact word logic for use with the cell, 3) an asynchronous reconfigurable mesh network for inter-processor communication, 4) a pass transistor network for responder prioritization and counting, and 5) the implementation and demonstration of a fully functional integrated associative processing system.

Chapter 2 describes the specific features of this associative processor design, at a level intended to familiarize potential users with the system's capabilities. Examples

Table 1.1
Comparison of Image Processing Systems

	Description	Time per $\sqrt{N}\times\sqrt{N}$ image		Time × Area per pixel		Energy per pixel	
		S&S	Laplacian	S&S	Laplacian	S&S	Laplacian
Application Specific	Focal Plane Processor [8] Pixel-parallel analog 1.5 μm CCD-CMOS	200 ns		310 μs μm^2		187 pJ	
	LSI 64243 3 × 3 Multi-Bit Filter [24] Pipelined digital 1.5 μm CMOS		$N\times$25 ns		—		25 nJ
	LSI 64240 8 × 8 Multi-Bit Filter [22, 23] Pipelined digital 1.5 μm CMOS		$N\times$50 ns		11 s μm^2		125 nJ
	ISMP [25] 4-PE Multiprocessor 1.2 μm CMOS		$N\times$98 ns		19 s μm^2		283 nJ
	Associative Processor **Pixel-parallel** **1.5 μm CCD-CMOS**	**90 μs**	**98 μs**	**26 s μm^2**	**28 s μm^2**	**69 nJ**	**76 nJ**
General Purpose	Digital 21064 μP "Alpha" [26] 64-bit RISC at 150 MHz .75 μm CMOS	$N\times$1.5 μs		340 s μm^2		33 μJ	

of simple arithmetic algorithms provide motivation for the use of a two-input function generator in the word logic, and the operation of the reconfigurable mesh network and the response resolver is also described.

The next two chapters discuss the most important circuits in greater depth. Chapter 3 reviews previously reported associative memory cells and then presents the dynamic associative processor cell used in this design. The necessary sense and driver circuits are also discussed, with particular attention to switching noise considerations. The remaining circuits of the processing element are presented in Chapter 4, including the special-purpose cells used in the network and response resolver implementations.

Chapter 5 reports experimental results from the integrated associative processor and related test circuits. Both measured and simulated results are used in performance evaluation, which is complicated by the fact that the chip was eventually fabricated in a process other than that for which it was designed. A demonstration image processing application is also presented.

The body of the thesis concludes with a summary of major accomplishments and ideas for future research. Several appendices present detailed documentation, including complete schematic diagrams.

Chapter 2

An Integrated Associative Processor

This chapter provides an architectural overview of the integrated associative processor and describes the chip's capabilities and use. The first three sections cover the basic functionality of the word logic, the inter-processor communication network, and the response resolution subsystem. Section 2.4 describes the chip's image input/output facility, and Section 2.5 sketches the design of the system controller. Implementation details are touched on only lightly in this chapter; the interesting aspects of the circuit design are reserved for Chapter 4.

2.1 Word Logic

In an area-efficient design the processing element's word logic must match the vertical pitch of its memory word. This constraint is a strong incentive to reduce word logic complexity. Consider, as a design exercise, the simplest logic element imaginable: a wire.

Figure 2.1 is a fully-parallel associative PE with the sense amplifier output fed directly back to the write enable driver. Match and write patterns are presented to the associative memory word on the trit lines, which run vertically through the PE array. Surprisingly, even this simple PE can perform useful operations, through the use of a sequential state transformation process [27, 2]. Table 2.1 presents a truth table for the destructive single-bit full add, $A + B + C \rightarrow A, C$. The least significant bit of the sum replaces the value of A, and the carry bit C receives the most significant bit. A quick examination of the table reveals that no work needs to be done in the checked states; the new values of A and C are the same as the old values. The other four states need to be transformed, as indicated by the arrows. Each transformation requires one match and one write operation, as shown in Table 2.2.

The first match operation selects all PEs with $ABC = 001$ and sets their sense amplifier contents to 1. Only these PEs participate in the following write operation, which sets $A = 1$ and $C = 0$ to transform state 1 into state 4. Steps 3 and 4 transform

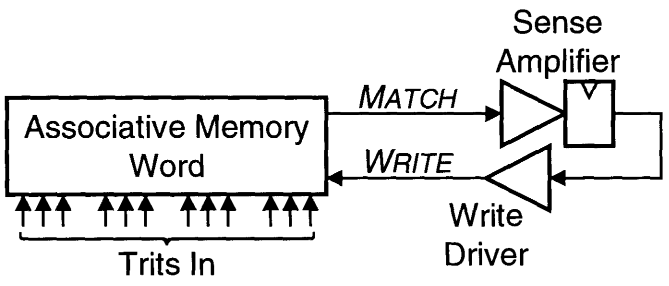


Figure 2.1: Associative processing element with direct match-write feedback.

Table 2.1
Full-Add Truth Table and Transformations

State Number	Previous State A	B	C	New State A	C	Transform
0	0	0	0	0	0	$\surd$
1	0	0	1	1	0	
2	0	1	0	1	0	
3	0	1	1	0	1	$\surd$
4	1	0	0	1	0	$\surd$
5	1	0	1	0	1	
6	1	1	0	0	1	
7	1	1	1	1	1	$\surd$

Table 2.2
Full-Add Procedure

Step	Instruction	Pattern A	B	C	Sense Amplifier Contents
1	Match	0	0	1	$\overline{A}\,\overline{B}\,C$
2	Write	1	–	0	$\overline{A}\,\overline{B}\,C$
3	Match	1	0	1	$A\,\overline{B}\,C$
4	Write	0	–	1	$A\,\overline{B}\,C$
5	Match	1	1	0	$A\,B\,\overline{C}$
6	Write	0	–	1	$A\,B\,\overline{C}$
7	Match	0	1	0	$\overline{A}\,B\,\overline{C}$
8	Write	1	–	0	$\overline{A}\,B\,\overline{C}$

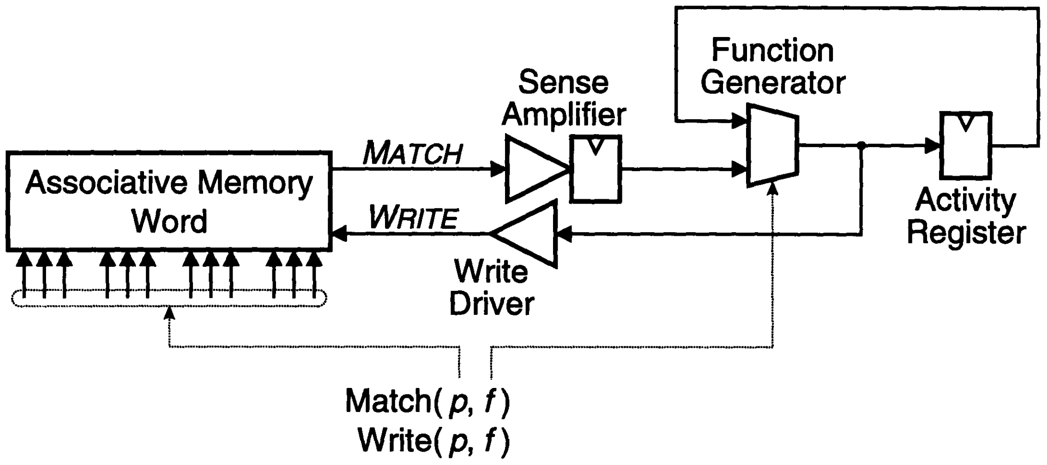


Figure 2.2: Associative processing element with improved word logic.

state 5 into state 1, and subsequent operations handle the remaining transformations. The order of the transformations is important. For example, if the transformation from state 5 to state 1 were performed first, then the transformation from state 1 to state 4 would affect some PEs that had started out in state 5. Ordering the transformations is easy in this case because there are no cyclic dependencies. If present, dependency cycles can be broken by adding a flag bit to indicate which PEs have been transformed. Initializing the flag to 0 requires another write instruction.

The procedure of Table 2.2 requires eight operations, but some of these are clearly redundant, as the four write instructions use only two write patterns.[1] One could imagine delaying the write of step 4 to merge it with the write in step 6, but this simple word logic can not support such reordering. The problem is that each write depends on the previous match, but each new match operation causes the sense amplifier to forget earlier match results. This suggests that the total number of operations could be reduced by adding state to the word logic [28, 29].

The word logic of Fig. 2.2 maintains an additional bit of state in an activity register (AR), and includes a function generator capable of computing any of the sixteen binary Boolean functions of two inputs (Table 2.3). Two instructions are defined, each with function and pattern parameters:

- *Match*(p, f). The function generator computes $f(\mathrm{SA}, \mathrm{AR})$ using the old values of the activity register and sense amplifier (SA). At the same time, pattern p is presented to the associative memory for matching. The AR takes the function generator output as its new value, and the SA value is replaced by the match result.

[1]The number of unique write patterns provides an upper bound on the number of writes needed to compute a function given arbitrarily complex word logic, but better solutions can often be obtained by recasting the problem. To give a trivial example, sixteen write patterns are required for the computation $(A, B, C, D \rightarrow \overline{A}, \overline{B}, \overline{C}, \overline{D})$, but only eight are needed to perform the four separate operations $(A \rightarrow \overline{A})$, $(B \rightarrow \overline{B})$, etc.

Table 2.3
The Sixteen Binary Boolean Functions of Two Inputs

0000_2	False	0100_2	$\overline{\mathrm{SA}} \wedge \mathrm{AR}$	1000_2	$\mathrm{SA} \wedge \mathrm{AR}$	1100_2	AR
0001_2	$\overline{\mathrm{SA} \vee \mathrm{AR}}$	0101_2	$\overline{\mathrm{SA}}$	1001_2	$\overline{\mathrm{SA} \oplus \mathrm{AR}}$	1101_2	$\overline{\mathrm{SA}} \vee \mathrm{AR}$
0010_2	$\mathrm{SA} \wedge \overline{\mathrm{AR}}$	0110_2	$\mathrm{SA} \oplus \mathrm{AR}$	1010_2	SA	1110_2	$\mathrm{SA} \vee \mathrm{AR}$
0011_2	$\overline{\mathrm{AR}}$	0111_2	$\overline{\mathrm{SA} \wedge \mathrm{AR}}$	1011_2	$\mathrm{SA} \vee \overline{\mathrm{AR}}$	1111_2	True

The functions are numbered such that function f is true if and only if $(f \,\&\, 2^{\mathrm{SA}+2\mathrm{AR}} > 0)$, where & is the bitwise AND operator.

- *Write*(p, f). The function generator is evaluated and used to enable the write driver. If $f(\mathrm{SA}, \mathrm{AR}) = 1$, then unmasked cells of the memory word are written in parallel, their contents replaced with the corresponding trits of pattern p. The AR and SA values remain unchanged.

Table 2.4 shows an add procedure for the improved word logic. In the first step, the processor performs a match with the pattern (X X 1), thereby loading C into the sense amplifier. In step 2, the function generator passes the sense amplifier result to the activity register, while the sense amplifier takes on the value B. The exclusive-or $(B \oplus C)$ is computed in the third step, while the value of A is loaded into the sense amplifier.

After three matches, the sense amplifier and activity register contain the information necessary to control the writes. Step 4 uses the function $(\overline{\mathrm{SA}} \wedge \mathrm{AR})$ to enable the instruction Write $(1-0)$, transforming states 1 and 5 of Table 2.1. The last write transforms states 2 and 6.

The function generator and activity register reduce the number of instructions required for a full add from eight to five. Table 2.5 presents results for other arithmetic operations (Appendix F) and shows that the improved logic can increase performance by a factor of two or more. (These figures represent per-bit requirements, excluding constant-time initialization and cleanup instructions.)

The performance benefits provided by the word logic must be weighed against its cost in silicon area. Fortunately, this cost is small. The activity register is almost cost free, since it shares its transistors with a shift register required for testing. The function generator occupies about 4600 square microns in the final implementation, or a little more than four memory cells. Overall, the function generators account for less than four percent of the array area.

One might consider further increasing word logic complexity to improve performance. For example, adding an additional register and using a three-input function generator would trim one more instruction from the addition algorithm, but it would also more than double the word logic area. The two-input function generator provides a more appropriate trade-off between performance and area, consistent with the goal of very fine granularity.

Table 2.4
Full-Add Procedure for Improved Word Logic

Step	Instruction	Function	Pattern A	Pattern B	Pattern C	SA Contents	AR Contents
1	Match		X	X	1	C	
2	Match	(SA)	X	1	X	B	C
3	Match	(SA $\oplus$ AR)	1	X	X	A	$B \oplus C$
4	Write	($\overline{\text{SA}} \wedge$ AR)	1	–	0	A	$B \oplus C$
5	Write	(SA $\wedge$ AR)	0	–	1	A	$B \oplus C$

Table 2.5
Cycles Required for Arithmetic Operations

Operation	Notation	State Transformation	Improved Word Logic
Destructive Add	$A + B + C \rightarrow A, C$	8	5
Non-Destructive Add	$A + B + C \rightarrow \Sigma, C$	12	5
Scalar Add	$A + s \rightarrow A$	4	3
Absolute Value	$\lvert A \rvert \rightarrow A$	6	3

Figure 2.3 shows the processing element as actually implemented. The memory word has 64 general-purpose associative memory cells, organized as two half-words of 32 cells each. Each half-word has its own write driver, but they share a single sense amplifier through a multiplexer (Section 3.3). This design doubles the vertical pitch available to lay out the word logic, but has the disadvantage of requiring two cycles to match or write the full 64 trit word. It also complicates the programming of the associative processor, but this is mitigated by the inclusion of the special carry (C) cell. This is a dual-port associative memory cell which stores a single value but can be written or matched from either the A or B half-word (Section 3.2). It is particularly useful for storing carry bits for addends that span both half-words.

There are five pairs of special cells used for inter-PE communication and response resolution. Each pair has one match-only A cell and one match/write B cell. (*e.g.*, cells HA and HB make up the H cell pair). The functionality of these cells is described in the next two sections.

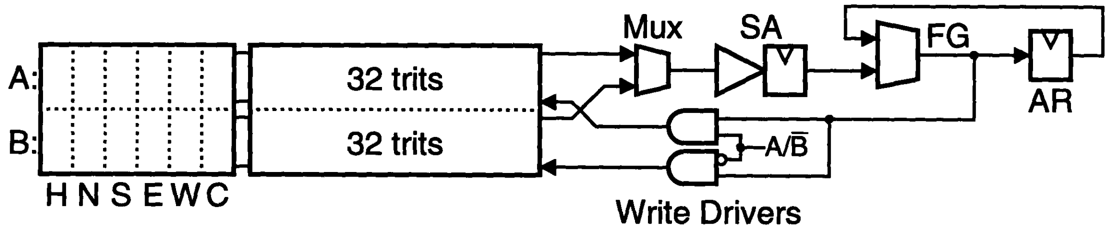


Figure 2.3: Processing element used on integrated associative processor.

2.2 Interconnect

Interprocessor communication is an important consideration in the design of any parallel computer, and this associative processor is no exception. However, the associative processor's fine grain creates a different set of constraints than that which might limit a coarse-grain design. All physically realizable networks are necessarily folded into three spatial dimensions. When many processing elements are integrated on a chip, however, many connections must be made in only the two dimensions available on the surface of the silicon die. Hypercubes and other high-dimensionality topologies rapidly become impractical as the number of PEs per chip increases.[2]

2.2.1 The Rectangular Mesh

Fortunately, many image processing algorithms have modest communication requirements. A two-dimensional rectangular mesh topology provides a natural representation for image data and maps readily into silicon. The large diameter of the network is not a limitation when communication is restricted to small neighborhoods [31, 32]. However, even relatively low-level algorithms such as computing histograms or labeling connected components can require non-local communication. Several authors have proposed augmenting the basic mesh-connected computer with additional communication paths, such as broadcast busses [33, 34, 35] and tree or pyramid structures [36, 31]. (The Connection Machine [37] uses both mesh and hypercube networks, but their relationship is more complementary than augmentative.) The integrated associative processor of this work retains the simple mesh topology but supports an asynchronous propagation mode to speed near-local communication. It also includes a response resolver (Section 2.3) to report global information on the state of the array.

An oft-repeated maxim says that in modern VLSI systems wires are expensive but transistors are cheap. In pitch-matched layout, however, transistors are expensive too, and efforts to reduce network wiring through multiplexing can result in false economies. If n driver circuits are connected to a common bus, then $n-1$ of them must be inactive at all times. In a dense layout the idle driver circuits may require more area than would the extra wires. The designer must also find room to route the control signals necessary to enable and disable the drivers. Extensive multiplexing is more appropriate for interchip communication than for on-chip networks in very fine-grain systems.

Receiver circuits tend to be smaller than drivers, so a good strategy is to equip each PE with multiple receivers but only a single driver. The driver is always enabled, so no additional control signals are required. Figure 2.4 shows how five special network cells are used to implement this scheme. When the PE writes to its HB cell, the

[2]Leiserson has shown that in asymptotic terms, fat-tree networks can simulate any other network without additional hardware and with at worst a polylogarithmic slowdown [30]. In practice, however, the bookkeeping overhead involved with such a simulation severely taxes the resources of a very fine-grained processing element. The associative processor's mesh network is sufficient for its intended applications.

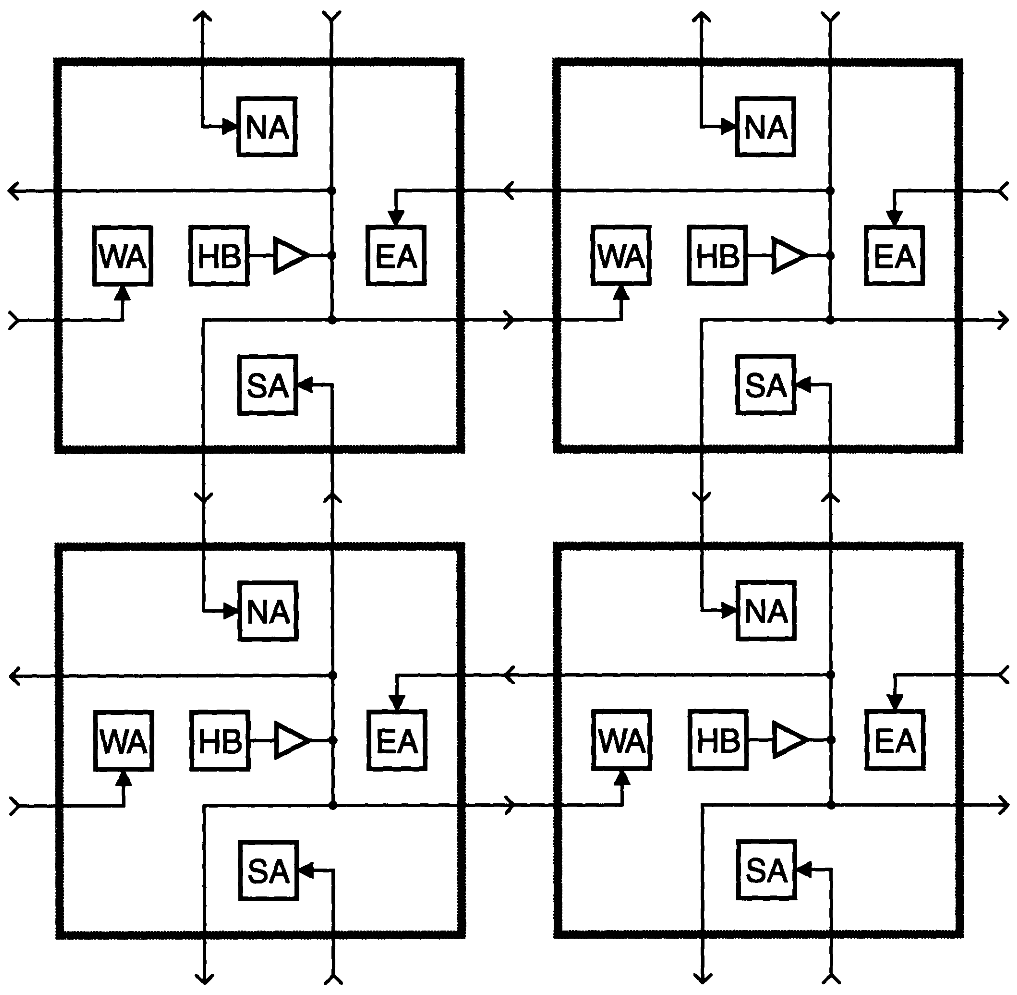


Figure 2.4: Four processing elements with mesh connections and special network cells. The HB cell controls the driver, and the match-only NEWS A cells are used as receivers.

driver transmits the bit to the PE's four nearest neighbors. (The network cells can not store the X state.) The value is then matchable in the neighbor's NEWS A cells, which function as receivers. A PE can examine all four of its NEWS A cells with a single match operation.

Bit-serial arithmetic algorithms can incorporate network operations. Table 2.6 presents the move-and-add procedure $A + B_N \rightarrow A$, in which the north neighbor's B field is added to the local field A. The first three instructions copy $\overline{B}$ to the special cell HB. If the first match is successful, the next write will set HB to 1. Otherwise, the write in step 3 will set HB to 0.

The remainder of the algorithm duplicates the destructive add discussed earlier, except that step 5 matches the net cell NA instead of the local cell B. In this way, the PE obtains the complement of its neighbor's HB cell, which in turn contains the complement of B_N as written in the first three steps. (Details of circuit implementa-

Table 2.6
Move-and-Add Procedure

Step	Instruction	Function	Pattern					SA	AR
			A	B	C	HB	NA	Contents	Contents
1	Match		X	0	X	X	X	$\overline{B}$	
2	Write	(SA)	–	–	–	1	–	$\overline{B}$	
3	Write	($\overline{\text{SA}}$)	–	–	–	0	–	$\overline{B}$	
4	Match		X	X	1	X	X	C	
5	Match	(SA)	X	X	X	X	0	B_N	C
6	Match	(SA $\oplus$ AR)	1	X	X	X	X	A	$B_N \oplus C$
7	Write	($\overline{\text{SA}} \wedge$ AR)	1	–	0	–	–	A	$B_N \oplus C$
8	Write	(SA $\wedge$ AR)	0	–	1	–	–	A	$B_N \oplus C$

tion necessitate the double complement; see Section 4.2.) Combining arithmetic and communications operations in this way is faster and more memory efficient than the naive approach of copying the B field and performing a local addition.

2.2.2 Asynchronous and Reconfigurable Communication

The example in Table 2.6 used the network's synchronous mode of operation, which requires several instructions to move information from one PE to its neighbor. This method works well for local communication, but results in unacceptable delays over long distances. Better performance may be achieved by providing an asynchronous communication path where gate delays—not clock cycles—limit propagation time. Illiac III [38] was an early machine to use the technique; its network provided a "flash-through" mode. Other examples include CLIP4 [39], BAP [40], DAP [41], and ASP [42].

Asynchronous propagation is especially powerful in systems that also provide some degree of *connection autonomy*, the ability of individual processing elements to configure their net connections independently. The polymorphic torus [43] is an early and representative example. As illustrated in Fig. 2.5(a), the base network (BNET) is a two-dimensional rectangular torus, with an internal network (INET) at each node. As originally conceived, the INET is a completely-connected switching network allowing each of the four BNET branches to be connected to one or more of the other branches. The processing element controls the six switches. The ideal INET proved too costly to implement in the "YUPPIE" prototype, and so a much more limited switch was used [44].

The Gated Connection Network (GCN) was developed jointly by researchers at Hughes and the University of Massachusetts at Amherst [45, 46]. (The Amherst authors seem to prefer the term *coterie network*.) While conceptually similar to the polymorphic torus, the GCN implements a more capable subset of the ideal full connectivity. Figure 2.6 shows a pair of GCN nodes, each with eight switches. Four switches correspond to the compass directions (N,S,E,W), while two more are used to

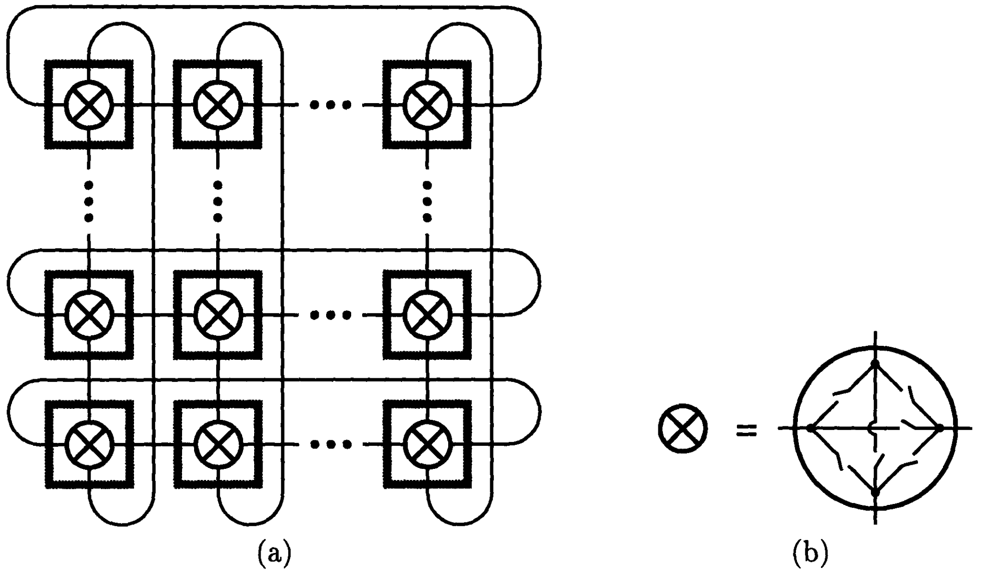


Figure 2.5: Polymorphic torus network [43]. (a) Base net with internal net $\otimes$ at every node. (b) Ideal completely-connected INET.

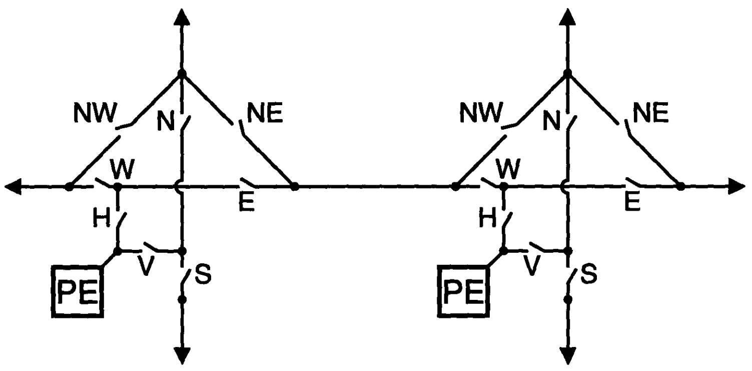


Figure 2.6: Gated Connection Network switches for two PEs [46].

bypass processing elements on the diagonals (NE,NW). The H and V switches were omitted in one of the group's earliest publications [47], but were added later to permit north-south and east-west signals to cross without interacting.

Reconfigurable networks like the GCN allow processing elements to group themselves into connected regions of arbitrary shape. Two such *coteries* are shown in Fig. 2.7. Each coterie's network functions as a wired-OR broadcast bus. A simple bit-serial procedure can be used to find the PE with the minimum address in each coterie, broadcasting its address and labeling connected PEs in the process [47]. The minimum and maximum x and y coordinates may be found by similar procedures, in order to compute a bounding rectangle for the coterie. Note that these operations are performed on all coteries in parallel. The real power of reconfigurable networks is their ability to perform parallel data-dependent operations on SIMD machines. The Amherst-Hughes group has coined the term *multiassociative* to

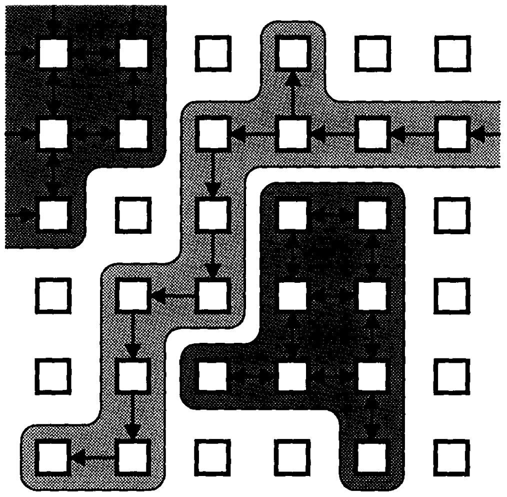


Figure 2.7: Net configuration with two coteries and a unidirectional path.

describe this capability [48].

More sophisticated algorithms use the coterie network to find means, medians and convex hulls [48]; to compute Hough transforms [47]; to detect straight lines [49]; and to route messages through the array [50]. However, the published GCN algorithms require neither the diagonal bypassing of PEs nor the orthogonal crossing of vertical and horizontal wires [51]. One goal of this project was to implement enough of the GCN functionality to perform the most important image processing tasks without compromising the associative processor's dense, pitch-matched design style. The resulting Asynchronous Reconfigurable Mesh (ARM) logic is shown in Fig. 2.8. A processing element establishes a communication link from a neighbor by writing a 1 to the appropriate NEWS B cell. When a 1 is received from a neighbor, the HB cell is set, and the signal propagates through the cell to all four neighbors. Simultaneous broadcasting in coteries is accomplished as follows:

- Processing elements configure their NEWS B cells to establish connectivity.
- All PEs write 0 to their HB cells.
- Senders write 1 to their HB cells, and the network is allowed to settle.
- Each PE examines its HB cell. If it finds a 1, it knows it is connected to a sender.

In this way, the ARM network computes the logical OR of the HB cells in a connected region.

One consequence of allowing processing elements to configure their own network connections is that the length of the longest path is not necessarily known to the controller, and therefore the worst-case propagation time may be unpredictable. However, the controller can poll the array to determine whether any unsettled coteries exist. The Amherst/Hughes group has developed hybrid algorithms that handle the

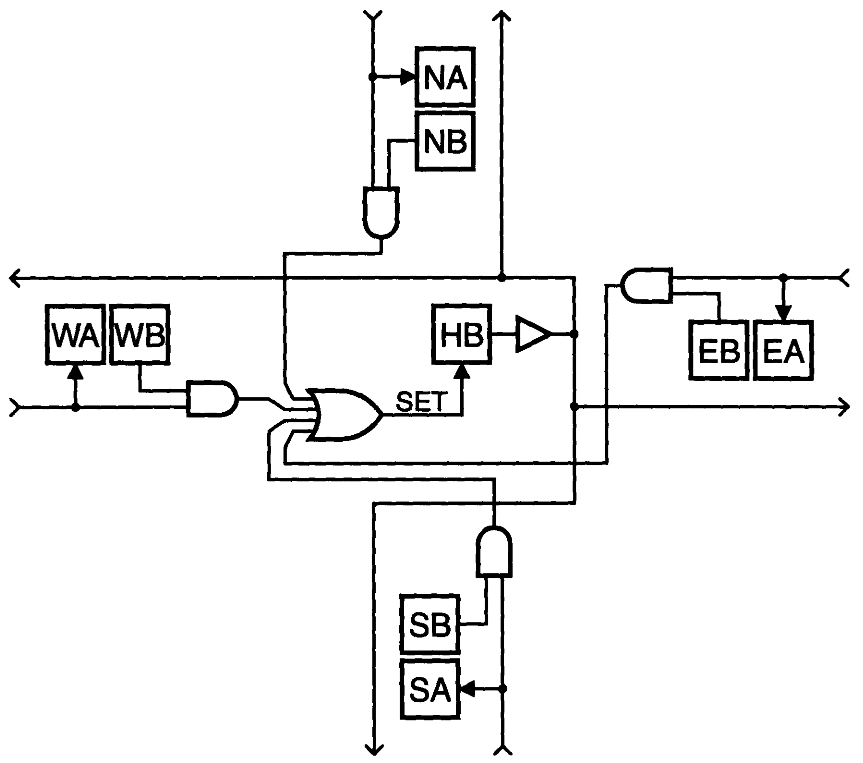


Figure 2.8: Asynchronous reconfigurable mesh logic. The NEWS B cells are used to enable the network inputs. The HB cell is set when a 1 is received on an enabled channel. Match-only NEWS A cells are used to examine the state of the connections, as in Fig. 2.4.

majority of coteries using their GCN network, but that timeout to handle the occasional troublesome cases by alternate means [52].

The ARM circuit implementation is discussed in Section 4.2, but it should be emphasized here that the AND and OR gates of Fig. 2.8 are drawn only to show logical function. The actual implementation uses *n*-channel precharged logic and is quite dense.

Unlike the GCN, which was implemented as a separate communication subsystem, the ARM shares its wiring and driver circuits with the associative processor's synchronous network. The two network designs provide the basic functionality necessary for a large class of applications, but the ARM does not support the GCN's little-used diagonal bypassing and orthogonal crossing capabilities. Similar features could be implemented in the ARM by providing four additional cell pairs for the next-nearest neighbors, but the potential benefits were deemed insufficient to justify a 10 percent increase in area. However, because the ARM uses separate wires for transmitted and received data on each link, it does allow unidirectional communication paths to be configured (Fig. 2.7). It remains to be seen whether either design's unique features provide significant algorithmic utility.

At the present level of integration, the chip's 16×16 array of processing elements is too small to hold an interesting image. Practical systems will therefore require arrays of many chips, with the communication network extended across chip boundaries (Section 2.4). The on-chip array has perimeter 64, but a 4:1 multiplexing scheme reduces the network pin count to 16 inputs and 16 outputs. Separate input and output pins are required since the direction of information flow in the ARM is not known *a priori*. This fact also precludes clever wiring schemes such as BLITZEN's X-Grid [53].

2.3 Response Resolution

When a content addressable memory performs a match operation, the memory words that match the presented data are termed *responders*. In an associative processor, the responder set may be identified by the logical combination of several match results. In either case, if more than one responder can occur, the system should have a multiple response resolution circuit. The response resolver produces summary information about the state of the array and feeds it back to the host computer and/or the individual PEs.

The ARM network of the previous section performs simple some/none response resolution on each coterie. The HB cell will match 1 if there are some coteries in the region and will match 0 if there are none. Responder prioritization and counting are examples of more sophisticated resolution tasks. The first task selects a single responder from a set of many. Repeated operations can be used to count the responders, or additional hardware may be provided to perform the count in time independent of the responder set's size.

Once again, the need to fit circuits on the memory pitch bounds the space of available solutions. The fastest response resolvers use tree and shower topologies [54, 55] that do not lay out well in memory arrays. Linear chains are easier to lay out, but their delays increase with length. A good compromise solution is to use linear chains within each subarray, combining the chain results in a tree structure.

2.3.1 Chains for Prioritizing and Counting

Figure 2.9 shows a logic diagram for a prioritizing chain using OR gates. Each input on the left side of the chain corresponds to one PE and is asserted if that PE is a responder. The output on the right side is asserted only if the PE is a responder and there are no higher priority responders. In the figure, PE1 is the first responder; it passes a 1 down the chain to inhibit PE2 and lower priority responders. The bottom of the OR chain produces a some/none result for all PEs in the chain.

A chain of exclusive-OR gates can count responders in $\log_2 N$ steps, with N equal to the length of the chain. Fig. 2.10 shows the three steps of the procedure for a chain of seven PEs. At the beginning of step 1, there are six 1 inputs on the left indicating six initial responders. The exclusive-OR chain computes the parity, and the last gate outputs a 0. This is the least significant bit of the count. Then the responders receiving a 0 from the chain are instructed to turn themselves off. This

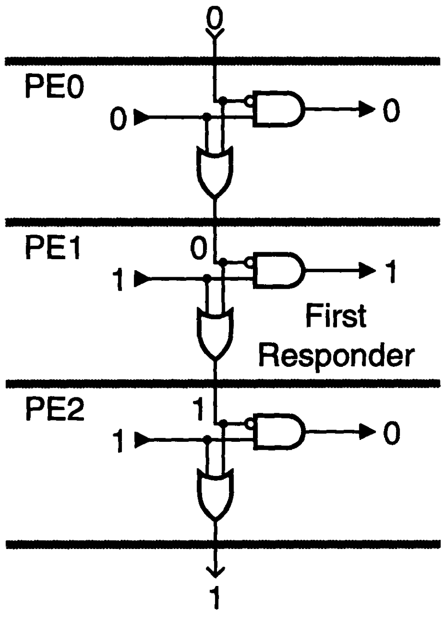


Figure 2.9: Prioritizing chain of OR gates.

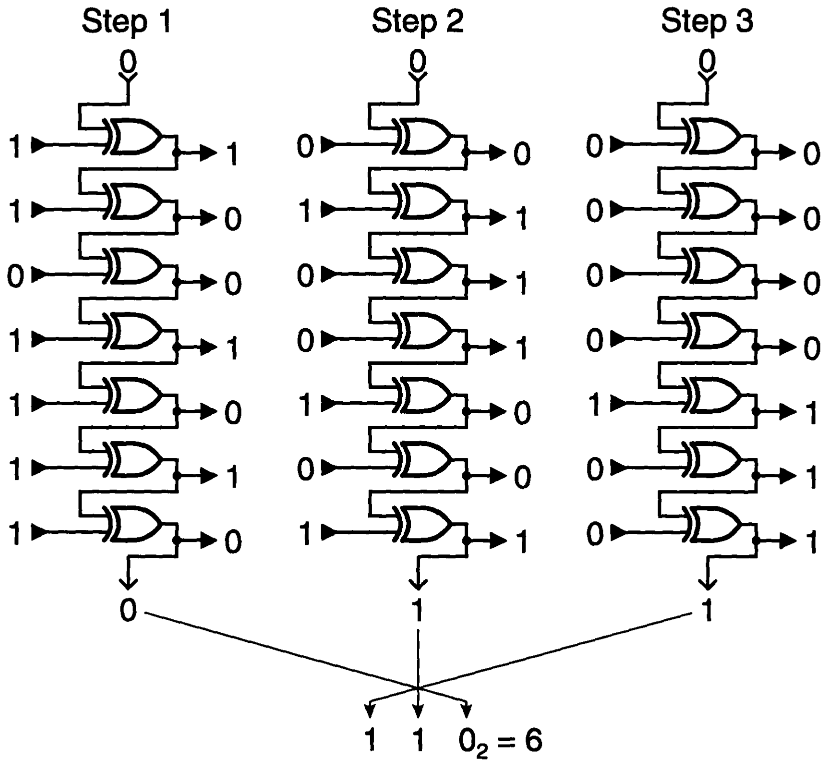


Figure 2.10: Counting responders with an XOR chain.

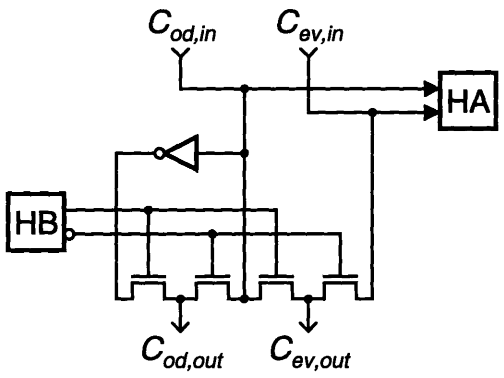


Figure 2.11: Pass transistor logic for responder chain.

Table 2.7
Responder Chain Behavior

$C_{od,in}\,C_{ev,in}$	$C_{od,out}\,C_{ev,out}$ HB = 0	HB = 1
00	00	10
10	10	01
01	01	10

action disables every other responder (not every other PE), so that three remain at the beginning of step 2. The bottom of the chain now produces a 1. Finally, half the responders are again disabled, and the most significant bit 1 is obtained. Taking the bits in reverse order gives $110_2 = 6$, the number of initial responders.

Both the OR and XOR chains can be implemented with a single pass transistor network, as shown in Fig. 2.11. The odd and even lines (C_{od} and C_{ev}) pass from one PE to the next, and PEs can examine the state of the chain by matching their HA cells. The HB cell serves as a responder flag. As shown in Table 2.7, the C_{od} and C_{ev} lines pass through non-responding PEs unchanged. However, if HB contains a 1, then $C_{od,in}$ will cross over to $C_{ev,out}$ while $C_{od,out}$ takes the inverter output $\overline{C_{od,in}}$. This is equivalent to the XOR function, if the tokens are identified with bits such that

$$\{00, 01\} \leftrightarrow 0 \qquad \text{and} \qquad \{10\} \leftrightarrow 1. \tag{2.1}$$

That is, if the top of the chain is loaded with a 00 or 01 token, then $C_{od,out}$ will produce the exclusive-OR of all the HB responder flags at the bottom of the chain.

To prioritize responders, one begins by loading the top of the chain with two 0 bits. The 00 token propagates down to the first responder, which converts it to a 10. Lower priority responders will exchange the 10 and 01 tokens, but they cannot restore the 00. The chain is equivalent to a cascade of OR gates under the identification

$$\{00\} \leftrightarrow 0 \qquad \text{and} \qquad \{01, 10\} \leftrightarrow 1. \tag{2.2}$$

Any PE receiving a 00 token on its $C_{od,in}$ and $C_{ev,in}$ lines knows there are no higher priority responders. Note that no additional circuitry is necessary to implement the AND gate of Fig. 2.9. After a PE examines its HA cell with a match operation, it can compute the AND in its function generator.

2.3.2 Prioritization with Multiple Chains

The prototype chip's 256 processing elements are serviced by four chains of length 64, organized as shown in Fig. 2.12. At the bottom of each chain, an OR gate computes a

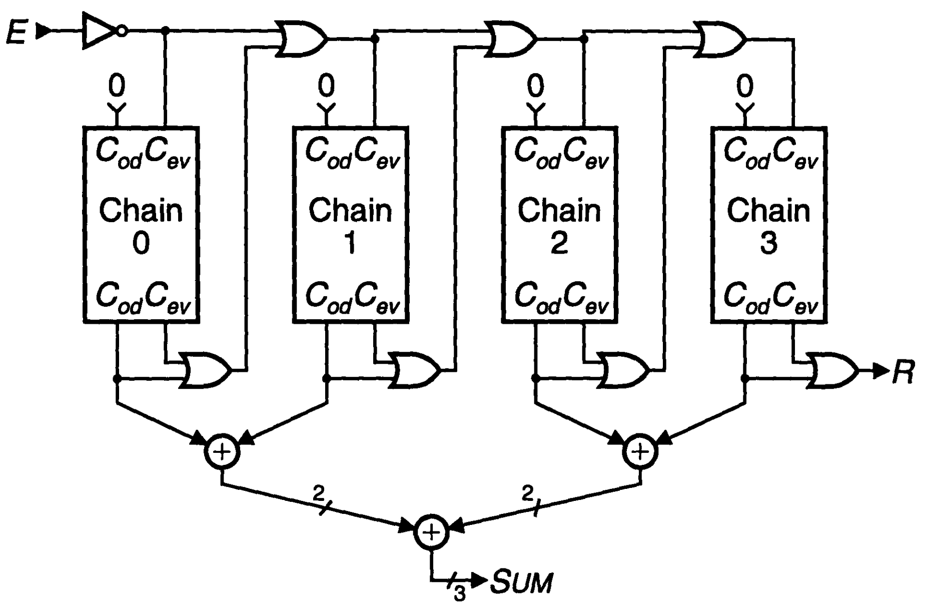


Figure 2.12: Response resolver with four chains. Count results are combined by a tree of adders. The top row of OR gates performs prioritization look-ahead.

some/none result from the C_{od} and C_{ev} outputs. This signal passes to the C_{ev} inputs of the following chains, where it can inhibit lower-priority responders. The count operation is unaffected by the responder propagation, because the XOR map (2.1) interprets both 00 and 01 tokens as 0 bits.

The chip enable signal E indicates that no responders are present in any higher-priority chips. It is computed as the NOR of all previous chips' responder (R) outputs, using the conventional tree structure of Fig. 2.13. (The R signal could be taken from the last array's OR gate, but a faster circuit is described in Section 4.3.1.) A binary tree with N chips at its leaves requires $N - 1$ nodes and settles in $O(\log N)$ time. A thorough summary of optimized responder designs may be found in [56, §4.3]. At higher levels of integration it may be desirable to use a tree for on-chip prioritization, but with only four subarrays the tree offers little advantage over the arrangement of Fig. 2.12.

2.4 System Organization and Image I/O

A digitized image is a three-dimensional array of bits, with height and width corresponding to the size of the image in pixels, and depth equal to the number of bits used to represent each pixel. Because interesting images contain far too many bits to be contained on a single present-generation associative processing chip, the bits must be parceled out to several chips for processing. One consequence of integrating processor and memory on the same chip is that systems can not be organized into bit planes, but must instead be partitioned in the spatial dimensions. This may be seen in Fig. 2.14, in which system (b) illustrates on a small scale the planar memory

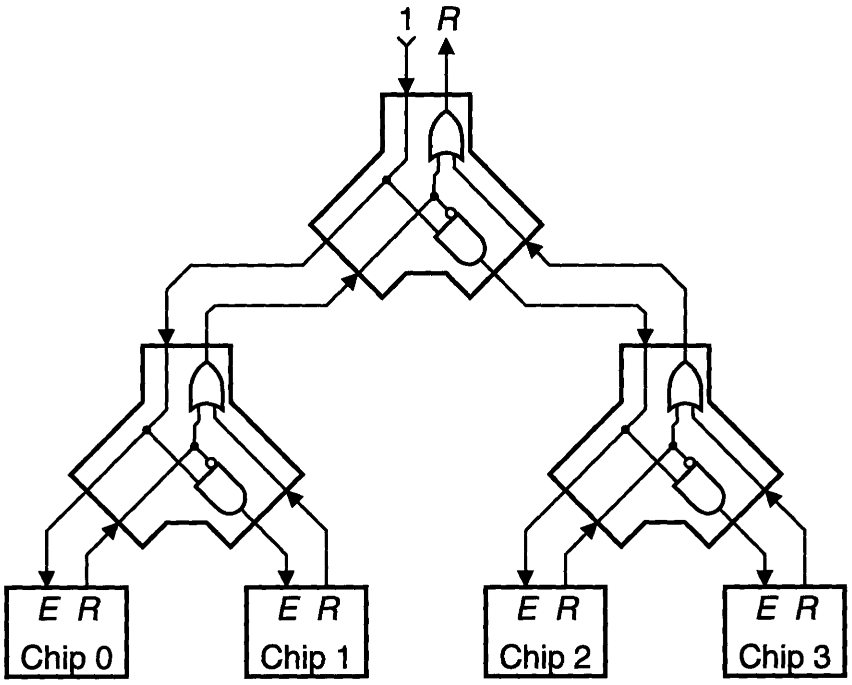


Figure 2.13: Responder prioritization tree. The leaves of the tree are four associative processing chips, each with an enable input E and a some/none resonder output R.

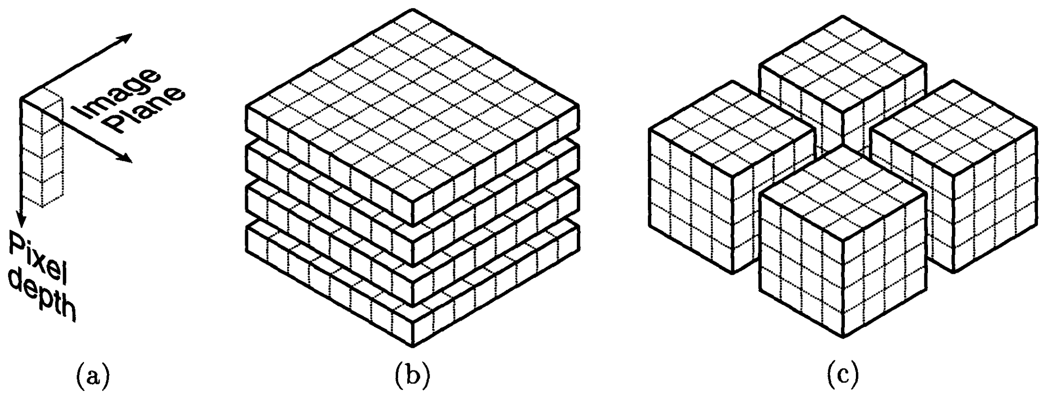


Figure 2.14: Multi-chip system organization. (a) A single four-bit pixel in the image plane. (b) Planar partitioning, with one chip per bit plane. (c) Spatial partitioning, with one chip per quadrant of the image.

organization typically used for frame buffers in graphics applications. There are four chips in the example system, each devoted to one bit plane of the 8×8 image. Since no pixel is stored entirely in one chip, any processing must be performed outside the memory system where the bits can be brought together. Pixel-parallel image processing requires a different memory organization, as shown in Fig. 2.14(c). This design also uses four chips, but each chip handles one quadrant of the image plane, rather than one plane of the image depth. That is, 16 complete four-bit pixels are stored in each chip, instead of 64 one-bit quarter-pixels. The actual associative processor design is similar but larger, with each chip performing pixel-parallel processing on a 16×16 array of 64-trit processing elements.

Considering only bandwidth limitations, the two example systems have equivalent input/output (I/O) capabilities. For example, if every chip had one I/O pin, then both systems could move four bits in each transfer. The planar organization of Fig. 2.14(b) has a decided practical advantage, however, because it can easily transfer whole pixels in scan line order, the most convenient format for most imagers and displays. The transfer operation makes effective use of the parallelism available in the pixel dimension by dividing the pixel stream into separate bit streams for each plane.

To achieve similar I/O parallelism in an associative processor, the image data must be reformatted to match the system's spatial partitioning. The early associative processor STARAN and its successor the MPP both had memory systems designed to perform bit reordering [57, 58], and more recent publications have described specialized memory chips for this purpose [59, 60]. These designs share certain similarities, but each is specific to its intended application. The associative processor chip described in this work includes a shift register for communicating with its own format converter, the design of which is still in progress [61]. As shown in Fig. 2.15(a), the shift register runs along the south border of each chip and functions independently of the network during normal operation. In load mode (b), however, the northbound network connections of each column are wrapped into a loop with the shift register forming an extra row. The shift register contents are then copied into the southernmost row of processing elements, to be replaced by the contents of the top row's HB cells. By repeatedly shifting and loading rows of bits, the system can output a processed image while simultaneously reading in the next raw image. Note that the bits of each plane are transferred in conventional scan line order—the first bit to enter from the southeast ends up in the northwest corner.

2.5 Controller Design[3]

The associative processing elements are voracious consumers of instructions. Designed to execute an 83-bit instruction every 100 ns, the chips require close to 1 Gbit

[3]The ideas presented in Section 2.5 are the products of collaborative work with other members of the M.I.T. Associative Processing Project. More thorough treatments of these topics may be found in [62] and [63].

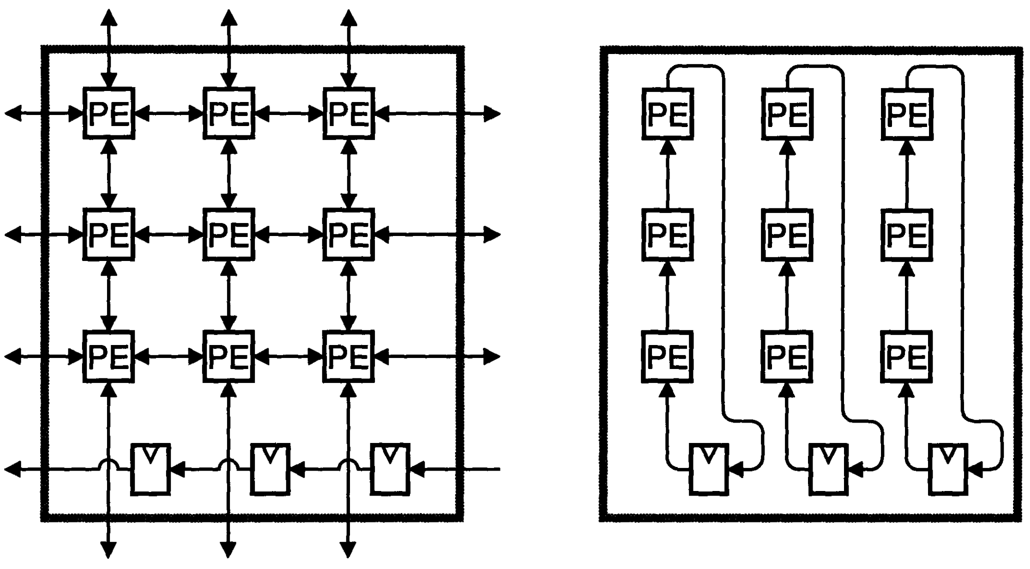


Figure 2.15: Shift register for image I/O, shown running along the south edge of an associative processor chip. A small 3×3 array of PEs is used to illustrate the two modes of operation: (a) Normal mode, with network and shift register functioning independently, and (b) Load mode, in which each column is connected in a loop.

of instructions every second. The fastest workstations would be hard pressed to sustain this data rate over their system busses; they would hardly have time to decide what the associative processing elements should do next. To ease the burden on the host computer, many designers of parallel machines have used two- or sometimes even three-level hierarchical control structures [64, 37, 65, 66, 67]. As shown in Fig. 1.1, a controller is placed between the host and the associative processor array, where it receives macroinstructions from the host and produces array instructions for the PEs. For example, one macroinstruction might specify that the fields A and B should be added, and another could instruct the controller to find the PE with the minimum sum. Because each macroinstruction may correspond to several tens of array instructions, the traffic through the host interface is reduced by a comparable factor.

Previous designers have used complex controllers that are in fact specialized, dedicated computers. This processing power is needed to interpret the incoming macroinstructions and keep the processing elements supplied with array instructions. Designers of parallel supercomputers can afford the complexity, but the associative processor is intended for low-cost applications in which a dedicated control computer would be prohibitively expensive. There are software costs as well, since separate programs are required for the host and the controller. Getting these two programs to work together can be more than twice as hard as writing a single program. The complex controller also requires its own set of software development tools, including a compiler and a debugger.

A simpler controller model is clearly desirable. One idea is to generate all the necessary array instructions at compile time and store the prepared code sequences in the controller. The macroinstructions are then reduced to subroutine calls telling the controller which sequence to send to the PEs. The controller can be simplified because it does not have to interpret parameterized macroinstructions. Figure 2.16 shows a block diagram. The sequencer receives a starting address from the host and begins stepping through the control store, producing one array instruction every clock cycle. This uncomplicated design is very attractive, but efficient implementation requires further refinement.

Unfortunately, the compiler does not have enough information to predict every sequence a program might require. Consider the task of adding a scalar s to a field A, $A + s \rightarrow A$, and assume that s is an n-bit value unknown at compile time. Because the instructions needed to perform this operation depend on s, a total of 2^n different sequences are possible. It is not necessary, however, to store all 2^n sequences in the controller; one can instead store just n sequences for $s \in \{1, 2, \dots 2^{n-1}\}$ and use these to construct the others. An even better decomposition exists, requiring $2n$ sequences but handling carries more efficiently. Because many important operations can be similarly decomposed, the controller includes hardware to select appropriate sequences for scalars supplied at run time [62].

Another optimization allows the system to handle responder feedback more efficiently. In the conventional scheme, the sequencer uses the responder flag and other array status bits to perform conditional branches. Following path 1 in Fig. 2.16, the state of the array determines the sequencer's flow of control, which in turn determines which instructions will be read from the control store and delivered to the array. This responder feedback path is critical to many search algorithms, so there is a strong incentive to reduce its delay. The associative processor chips are designed to accept responder feedback directly, following path 2 and bypassing the sequencer and control store. Support for this feature requires only one additional transistor per processing element (Section 4.3).

2.6 Summary

The first sections of this chapter described the architectural features of three associative processor subsystems. Each design involves a compromise between capability and complexity. In Section 2.1, a compact word logic design using a two-input function generator and a single-bit activity register was shown to improve arithmetic performance substantially. The next conceivable increment in complexity, a three-input function generator, would not deliver sufficient performance improvements to justify its cost in area.

The Asynchronous Reconfigurable Mesh supports the basic functionality of the Gated Connection Network, but the present design does not implement the seldom-used features of diagonal bypassing and orthogonal crossing. Special memory cells allow the network circuits to communicate with the word logic over the memory match and write lines. This design avoids complicating the word logic with additional

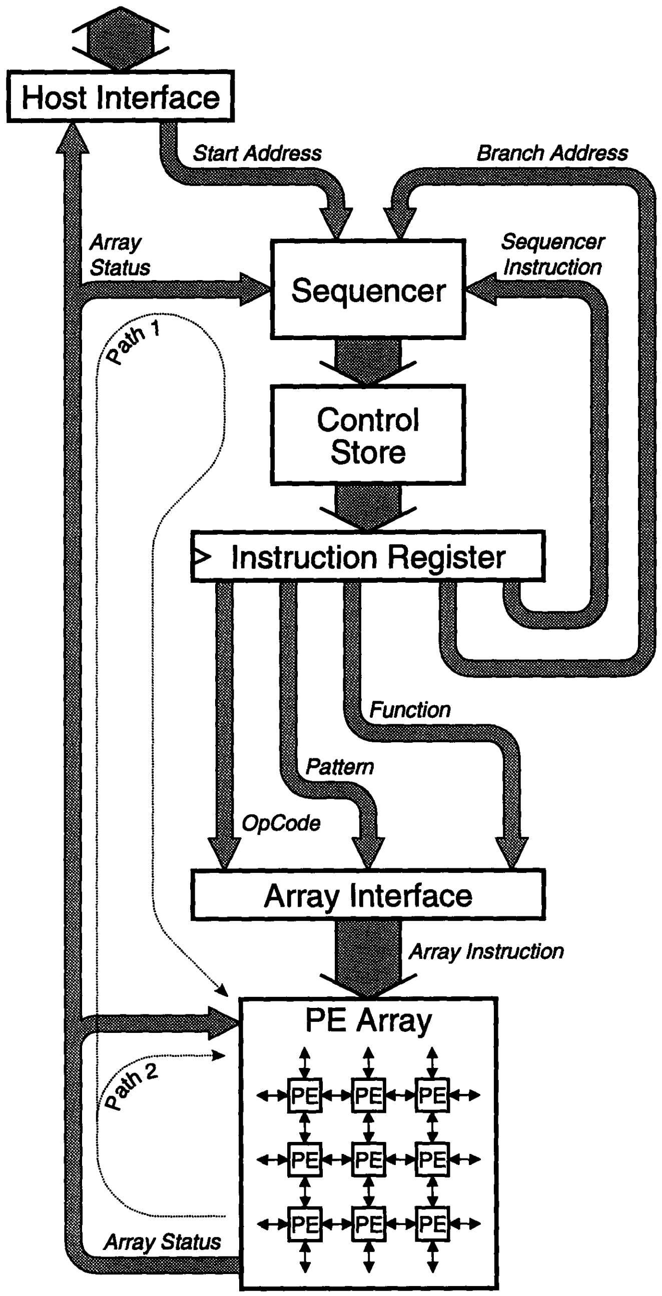


Figure 2.16: System control path.

registers, and does not require expanding the instruction set to include operations other than match and write.

The response resolver performs both prioritization and counting tasks, using a simple pass transistor circuit to compute both the OR and XOR functions. Linear chains are used in the pitch-matched environment of the PE arrays, but tree structures are used to combine responder results from the many chips in a system.

Section 2.4 described the organization of a multi-chip system and explained the need for a format converter in the image I/O path. Section 2.5 discussed the benefits of the simplified controller design used in the associative processor system.

Chapter 3

Associative Memory

The first electronic content addressable memory was reported by Slade in 1956 [68]. His "catalog memory" used cryotrons, the superconducting switches thought to be very promising at the time. Since then, associative memories have been implemented or proposed in most of the technologies that have been used for conventional memories, from cryotrons to Josephson junctions, magnetic cores to magnetic bubbles [13]. Of course, the dominant memory technology today is the integrated circuit, with no immediately competitive alternatives. In particular, MOS (Metal-Oxide-Semiconductor) technology achieves the high density and low power that is especially desirable for memory arrays, both conventional and associative.

Without undertaking an exhaustive survey, the first section of this chapter reviews some representative implementations of MOS CAM cells. This brief history prepares an appropriate context for Section 3.2, which describes the dynamic associative processor cell used in this design. The remaining two sections describe the circuits used to sense and control the memory array, with particular attention to switching noise considerations.

3.1 Content Addressable Memory Cells

Like their random access cousins, most content addressable memory cells can be classified as either *static* or *dynamic.* The static cells use active bistable circuits to maintain their state and generally offer a speed advantage. Dynamic cells forgo the active circuitry and store their state by charging or discharging high-impedance nodes. Periodic refresh operations are required to replace any charge that leaks away. This refresh overhead is the price one pays for the dynamic cells' greater density.

Pseudo-static cells make up a third class, storing charge dynamically but also supporting a self-refreshing operation. Although beyond the scope of this discussion, most of these cells can be understood as variants or simplifications of the static cells described below [69, 70].

The basic associative operation is the match, in which the state of the cell is compared with a presented value. Most CAMs allow a *don't care* (X) to be presented,

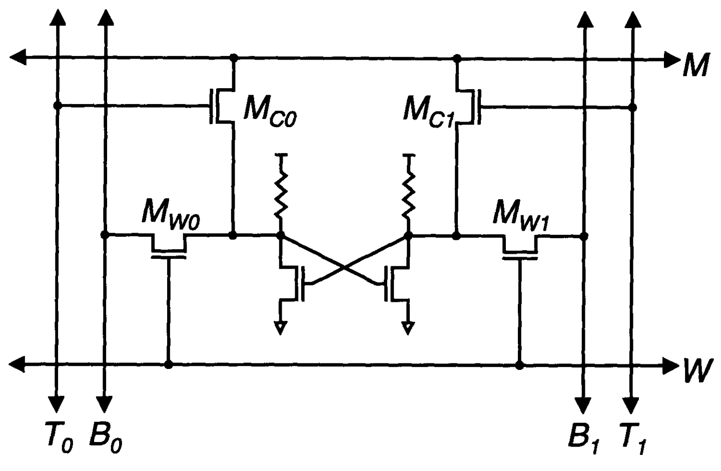


Figure 3.1: First MOS CAM cell, by Igarashi *et al.* [73].

which will match any stored state. Some cells are also capable of storing the X state, to match any presented value. This feature is particularly valuable in logic programming applications and quadtree image encoding [71, 72].

All the CAM cells described in this section happen to be RAM cells as well, meaning that they support the familiar read and write operations. The read is not strictly required of a CAM cell, since the match operation provides adequate means to examine its state. Even if the application does not use it, however, the read operation is a necessary step in the refresh procedure of many dynamic cells.

3.1.1 Static Cells

Perhaps the most straightforward way to build an associative memory cell is to start with a standard RAM cell and add a comparator. This was the approach taken by Igarashi *et al.* in the first reported MOS CAM cell [73]. Figure 3.1 shows the NMOS dual of the original PMOS circuit.[1] The inverters and write transistors $M_{W0,1}$ make up the SRAM cell, and the transistors $M_{C0,1}$ form the comparator. The match operation begins with the match line M precharged high, and a 0 or 1 is presented by driving one of the test lines $T_{0,1}$ high. If the presented value does not match the state of the cell, then current will flow through one of the comparator transistors $M_{C0,1}$, and the match line will be discharged. The X value is presented by holding both test lines low, in which case both comparator transistors will be off and no mismatch current will flow. The designer must take special care that the match line potential does not drop too far below the switching threshold of the inverters, especially in the case where all but one of the cells in a word mismatch. The mismatching cells might then sink enough current to "write" the matching cell through the comparator transistor,

[1]The dual form is more readily compared to other cells in the chapter, and avoids the confusion of negative mismatch current. The Koo cell of Fig. 3.4 is also presented in dual form, without further comment in the text.

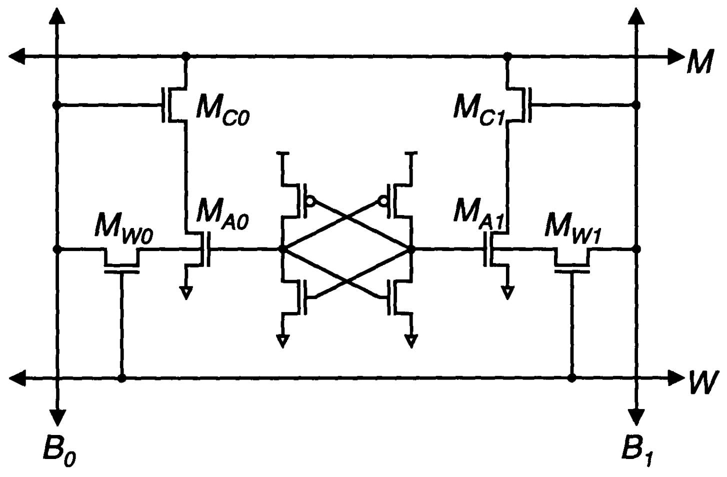


Figure 3.2: Improved static CAM cell by Kadota *et al.* [74].

thereby changing its state erroneously. This danger is avoided in the improved cell designs described below.

The Kadota cell [74] shown in Fig. 3.2 demonstrates three variations on the Igarashi cell. (The separate innovations were described earlier by others [75, 76].) First, Kadota *et al.* use a full CMOS static RAM cell, which dissipates no standby power and provides a faster write cycle than do cells with resistive loads [77]. The penalty for these advantages is the increased cell area necessary to accommodate complementary devices. Resistive loads remain an appropriate choice when higher density is required [78].

Second, the Kadota cell uses one pair of bit lines $B_{0,1}$ for both writing and matching, eliminating the Igarashi cell's test lines. This reduces the horizontal pitch of the cell and saves area, but it precludes simultaneous match and write operations, which are required in some applications [78, 79].

The third and most important advantage of the Kadota cell is its use of transistors $M_{A0,1}$ to sink the mismatch current, instead of sinking it through the inverters. This isolates the inverters during the match operation, and prevents the false writes possible in the Igarashi cell. The memory and comparator portions of the cell can be optimized separately, making it easier to balance write, match, and read performance. The drawback to this design is the additional area required by the two added transistors.

Figure 3.3 shows the cell of Ogura *et al.* [80], which uses the three-transistor comparator configuration first described by Zehner [81]. Mismatch current flows through only one transistor, M_D, which is controlled by one of the test lines T_0 or T_1, depending on the state of the cell. Unfortunately, this cell suffers from a double threshold drop problem. Because the high-resistance loads can not be relied upon to pull the storage nodes above $V_{DD} - V_T$, the gate of M_D can not be charged to greater than $V_{DD} - 2V_T$. The reduced gate drive limits the mismatch current and results in a slower match cycle. Bootstrapped write line drivers [82] could alleviate this difficulty

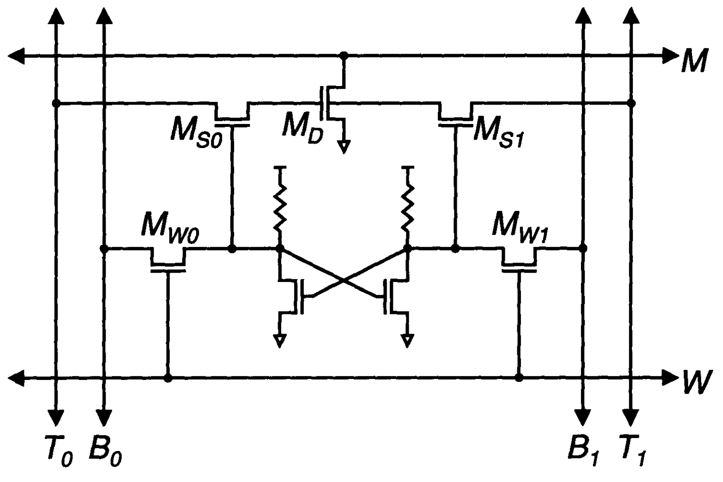


Figure 3.3: Static CAM cell with three-transistor comparator, sinking mismatch current to supply. By Ogura *et al.* [80].

by increasing the gate-source drive on the write devices, but the authors of [80] do not mention this possibility. Of course, CMOS designs can use complementary write devices to eliminate the threshold drops entirely. A BiCMOS version has also been reported, using a bipolar transistor in the place of M_D and achieving a 4-ns match time [83].

Another three-transistor comparator design is shown in Fig. 3.4. The Koo cell [75] avoids the double threshold drop problem, and has the notable feature of sinking mismatch current to the bit lines, thereby eliminating a supply connection. Of course, power must still be routed to the inverters, so substantial area savings may not be realized. Designers must weigh any density advantage against the increased demands this comparator places on the bit line drivers.

3.1.2 Dynamic Trit Cells

The selection of comparators appropriate for dynamic cells is somewhat limited. Igarashi's design is immediately excluded because it does not present a high impedance to the storage node. The remaining comparators are all potentially workable, but only the Koo configuration does its job without a supply connection. This advantage makes it especially suitable for dense dynamic designs.

Figure 3.5 shows the dynamic CAM cell invented by Mundy [84, 85], which is recognizable as the Koo cell sans inverters. The gates of M_{S0} and M_{S1} serve as the storage nodes, and a 0 or 1 is stored in the cell by charging exactly one of these nodes. Since there are no inverters, it is also possible to discharge both nodes, turning off both M_{S0} and M_{S1} and putting the cell in the X state. Thus the cell stores a single ternary digit, or *trit.* The hypothetical state in which both nodes would be charged is not used.

Any of the three trits can be presented to the cell for matching, with result that

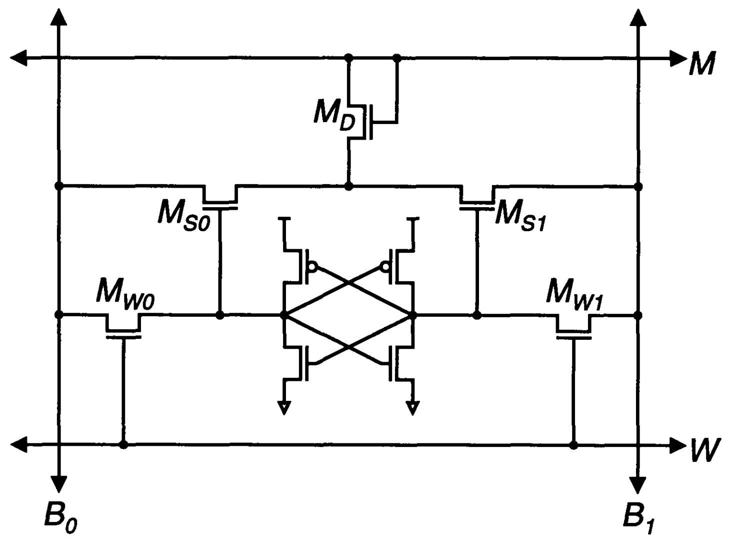


Figure 3.4: Static CAM cell with three-transistor comparator, sinking mismatch current to the bitlines. By Koo [75].

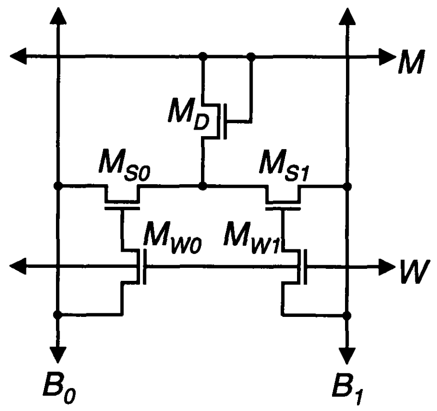


Figure 3.5: Five-transistor dynamic CAM cell by Mundy [84].

Table 3.1
Summary of Mundy Cell Match Operation

Presented Value	Bit Line Potential B_0	Bit Line Potential B_1	Matching Trits
X	High	High	0 1 X
0	High	Low	0 X
1	Low	High	1 X
–	Low	Low	X

each trit matches itself, and every trit matches the X trit. A fourth *don't match* (–) value can be presented but not stored. It matches stored X trits, but mismatches the 0 and 1 states. Table 3.1 summarizes the match operation.

Because it has no inverters to actively drive the bit lines, the Mundy cell can not be read in the same way as the SRAM-based static cells. Instead, the comparator is used to perform a non-destructive read. The bit lines are first initialized to a low potential, and then the match line is driven high. Current will flow to the bit line B_0 if transistor M_{S0} is on, or to B_1 if the cell is in the 1 state. In the X case, both transistors will be off and both bit lines will remain low.

The write operation would seem to be quite simple: just assert the write line W and drive the bit lines to the appropriate potentials to charge the storage nodes. Of course, a *node* can not really store charge. That is just a convenient way of saying that charge is stored on some *capacitors*, and that these capacitors each have one plate connected to the node. To understand the cell, however, one must examine both plates of each capacitor, as well as the capacitors themselves. Figure 3.6 models the capacitors attached to the storage node S_0, and Table 3.2 presents typical capacitance values for a 1.5 μm process.

The gate capacitance of M_{S0} is highly nonlinear. When the device is in inversion, electrons in the channel form the bottom plate, so the effective capacitance is C_{ox}. However, when the device turns off then the gate charge is matched by ionized dopant charge in the depletion region. The capacitance looking into the gate is then C_{GB}, the series combination of C_{ox} and C_b.

Suppose a 0 is written to the cell. B_0 will be driven high during the write, and the storage node will be pulled up to about 3.5 V (assuming $V_{TN} = 1.5$ V with the back gate effect [86]). M_{S0} will be off, offering only the small gate capacitance C_{GB} to store charge.

Now present a 1 for matching. M_{S0} turns on as B_0 is driven low, and the gate capacitance increases to C_{ox}. To satisfy charge conservation, the voltage at the storage node must decrease as the capacitance increases. Table 3.2 shows how charge redistributes among the several capacitors, and also reflects the coupling through the overlap capacitances $C_{ol,S0}$ and $C_{ol,W0}$. The resulting storage node potential is only 1.4 V during the mismatch. The transistor M_{S0} will be turned on only weakly, limiting the current and making for a slower match cycle.

Two solutions to this problem have been reported, one requiring advanced devices and the other relying on circuit techniques. The device technology solution focuses on changing the configuration of capacitances at the storage node. Note that in Fig. 3.6 both C_j and C_{ip} have one plate tied to the substrate, making them immune to coupling effects and independent of the transistor operating point. Ironically, both capacitances would be considered undesirable parasitics in most other circuits, but in this cell they contribute to the beneficial component of the storage node capacitance. Yamagata *et al.* [87] further increase this component by adding stacked capacitors $C_{S0,1}$ to the storage nodes, as shown in Fig. 3.7. The sizes of M_{S0} and M_{S1} may then be reduced, which has the additional benefit of reducing the C_b component of bit line

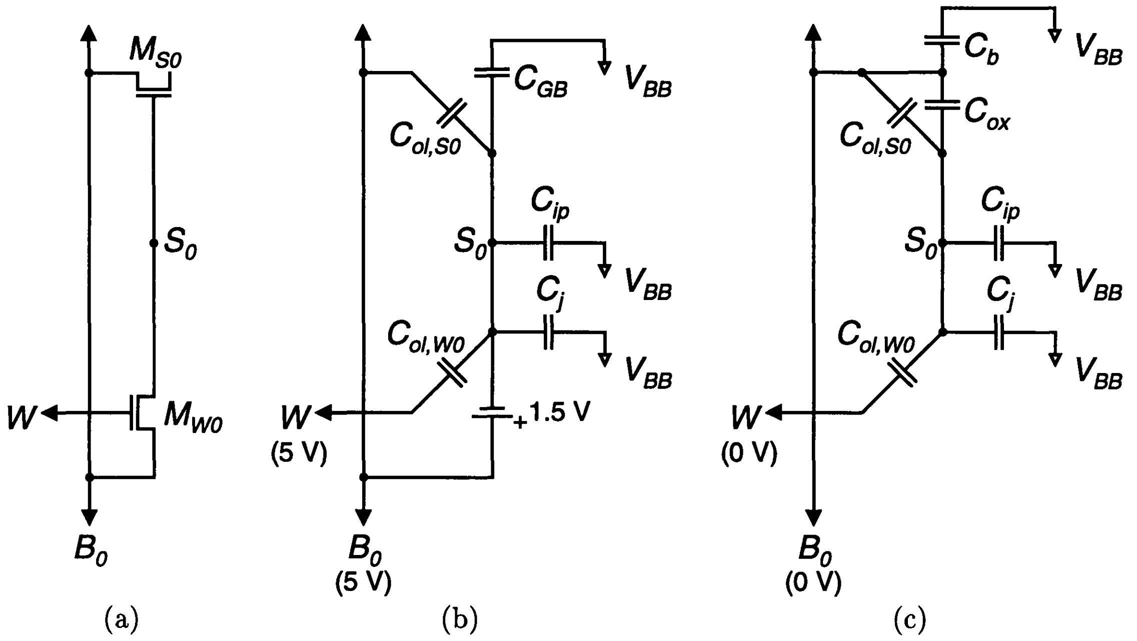


Figure 3.6: Capacitance model of Mundy cell storage node. (a) Portion of cell to be modeled. (b) Model for write 0 operation. (c) Model for subsequent match 1 operation.

Table 3.2
Charge at Mundy Cell Storage Node

Capacitance		Typical Value (fF)	Charge During Write (fC)	Charge During Mismatch (fC)
$C_{ol,S0}$	Gate-source overlap	21	−31.5	30.4
C_{GB}	Gate-substrate	63	220.5	
C_{ox}	Gate-channel oxide	135		195.3
C_b	Channel depletion region	117		
$C_{ol,W0}$	Gate-drain overlap	2	−3.5	2.9
C_j	Drain-substrate junction	10	35.0	14.5
C_{ip}	Interconnect parasitic	11	38.5	15.9
	Total charge:		**259**	**259**
	Storage node potential:		$V_{S0} = 3.5$V	$V_{S0} = 1.4$V

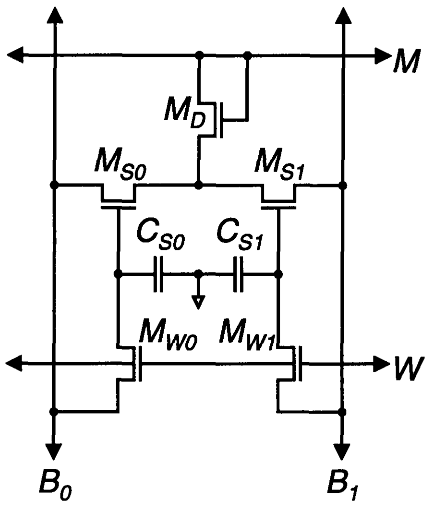


Figure 3.7: Improved dynamic CAM cell using stacked capacitors at the storage nodes. By Yamagata *et al.* [87].

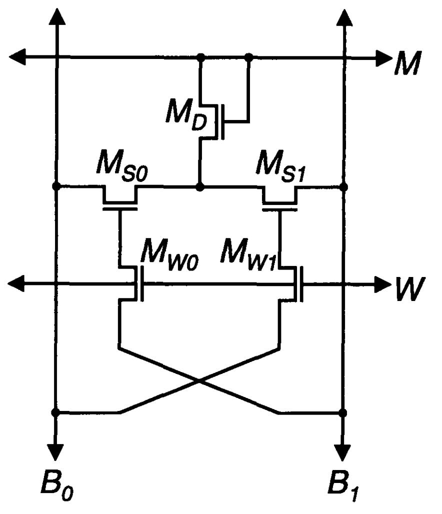


Figure 3.8: Improved dynamic CAM cell with crossed bit lines. By Wade and Sodini [88].

capacitance. The resulting cell is very compact, only 8.8 μm × 7.5 μm in a 0.8 μm process.

All these advantages are won at the cost of increased process complexity—the cell requires three levels of polysilicon, one of which is used to distribute the plate supply. Unfortunately, stacked capacitor processes are not offered by the largest prototyping services, and they may be incompatible with other enhancements required by particular applications.

Wade and Sodini have demonstrated an improved cell requiring only standard MOS technology [88]. As shown in Fig. 3.8, the cell uses a crossed bit line configuration to solve the Mundy write problem.[2] Each storage node is now written from the opposite bit line, so it is B_1 that must be driven high when writing a 0 trit, while B_0 is held low to provide electrons to the channel of M_{S0}. The storage node sees the full C_{ox} gate capacitance during both the write and the match operations. The crossed bit lines also eliminate the coupling associated with $C_{ol,S0}$.

3.2 A Dynamic Associative Processor Cell

In the cell schematics of the previous section, all of the match and write lines are drawn horizontally, while the bit and test lines are shown running vertically. This convention reflects the two-dimensional structure of the memory array. Each word comprises a row of cells sharing the same match and write lines, and the cells in

[2]An earlier dynamic cell using crossed bit lines was disclosed by Schuster in 1984 [89], but the inventor seems not to have been fully aware of the configuration's advantages to the write operation. The cell uses the same comparator as the cell of Fig. 3.3, and it therefore suffers from the same double threshold problem.

Table 3.3
Summary of Representative MOS CAM Cells

Reference	Storage	Transistor Count	Comparator (Figure #)	Separate Test Lines	Notes
Carr [76]	SE	10	3.2	No	
Crandall [69]	PS	4	3.1	No	
Igarashi [73]	SR	6	3.1	Yes	1
Jones [70] A	PS	8	3.2	No	2
Jones [70] B	SC	10	3.2	No	2
Kadota [74]	SC	10	3.2	No	
Koo [75]	SC	9	3.4	No	
McAuley [90]	SC	12	3.2	No	
Mundy [85]	DY	5	3.4	No	
Nogami [78]	SR	8	3.2	Yes	
Ogura [91]	SC	11	3.3	Yes	
Ogura [92, 80]	SR	7	3.3	Yes	
Schuster [89] 1	DY	5	3.3	No	
Schuster [89] 2	DY	5	3.3	No	3
Shin [79]	SC	11	3.3	Yes	
Tamura [83]	SC	10	3.3	No	4
Wade [88]	DY	5	3.4	No	3
Yamagata [87]	DY	5	3.4	No	5
Zehner [81]	SD	9	3.3	No	

DY	Dynamic	1	First MOS CAM
PS	Pseudo-static	2	Two match lines per word
SC	Static, CMOS inverters	3	Crossed bit lines
SD	Static, depletion-mode MOS loads	4	BiCMOS
SE	Static, enhancement-mode MOS loads	5	Stacked capacitors
SR	Static, resistor loads		

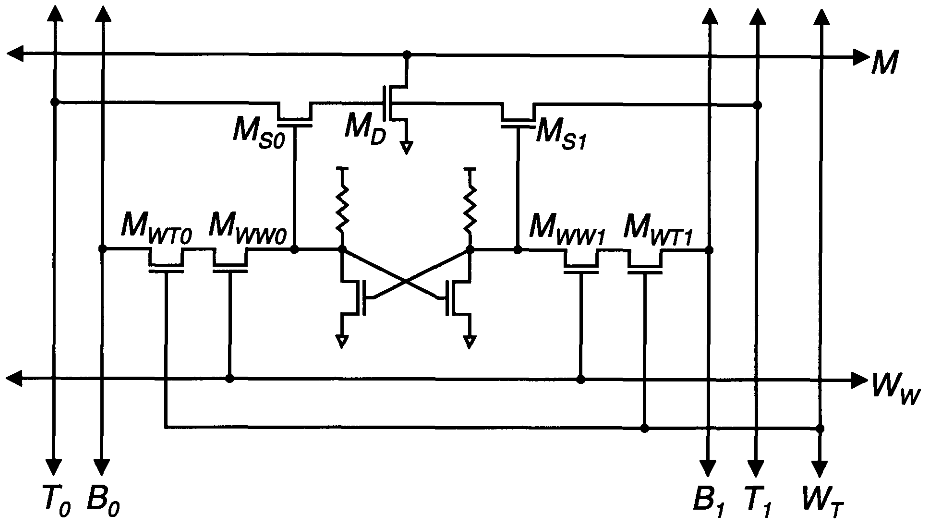


Figure 3.9: Static associative processor cell [80].

each column share common bit lines (and test lines, if any). One consequence of this organization is that is impossible to modify individual cells without affecting others. When a write line is asserted to enable one cell, the same signal necessarily enables all the other cells in the word. An associative processor, however, requires the ability to modify some cells in a word while leaving other "masked" cells unchanged.

Writes to static and pseudo-static CAM cells can be masked by driving both bit lines to the same potential. The write transistors then act as enhancement-mode load devices, and the cell maintains its original state. However, this mode of operation typically requires a great deal of current. Although one author accepts the increased power dissipation to achieve a modest area reduction [70], not all designers can afford this bargain.

A better solution is to run a vertical write enable signal through each column. Figure 3.9 shows the implementation by Ogura *et al.*, who add a second pair of write transistors to their CAM cell [80]. The cell is written only if both the W_W and W_T lines are asserted. This implementation requires no additional power, but the horizontal pitch must be increased to fit the W_T line.

Unfortunately, the same approach does not work with dynamic cells, because of the charge sharing problem illustrated in Fig. 3.10(a). Ideally, no charge will be transferred between the bit line and the storage node unless both transistors are turned on at the same time. But suppose only the write-trit line W_T is asserted, allowing electrons to move from the bit line to the parasitic junction capacitance C_j. If W_W is enabled sometime later, these electrons can move to the gate of the storage device M_{S0}. Only a few such *spooning* cycles are needed to completely discharge the storage node and change the state of the cell. The process can run in reverse as well, charging the storage node by spooning electrons to the bit line.

Dual-gate CCD transistors (Fig. 3.10(b)) greatly reduce the charge spooning problem by largely eliminating the parasitic junction capacitance. When W_T is raised, the

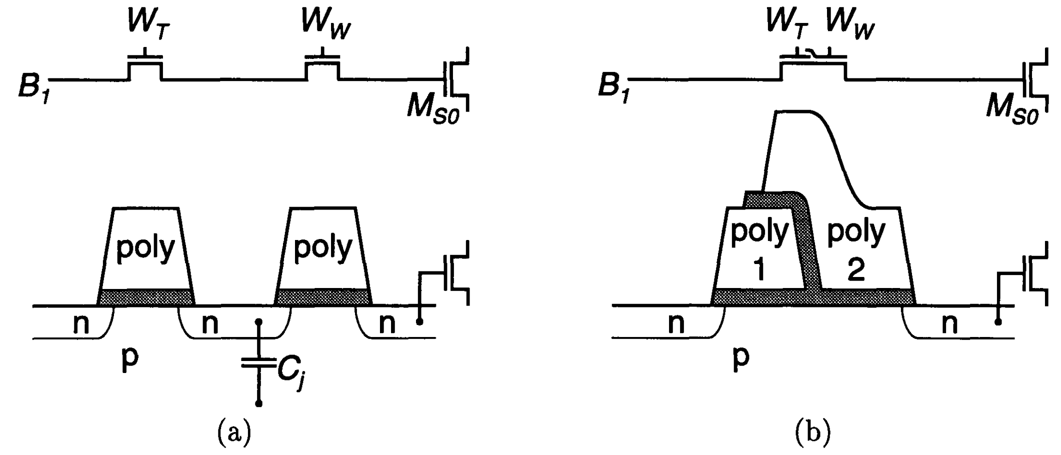


Figure 3.10: Cross sections of write transistors. (a) Two FETs in series. (b) Dual-gate device.

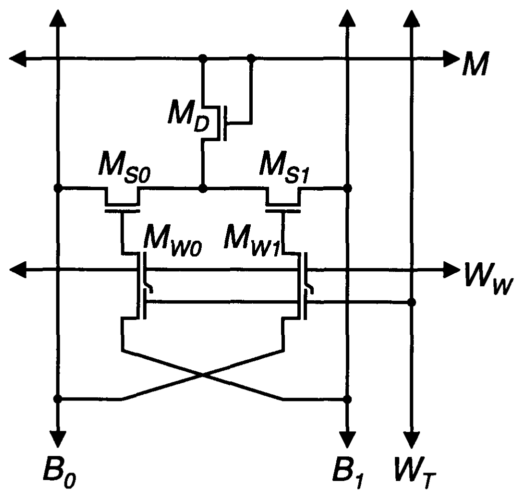


Figure 3.11: New dynamic associative processor cell used in this work.

left half of the channel will fill with electrons. Most of them will return to the bit line when the gate is lowered. However, some may stay behind in traps, and eventually diffuse to the storage node and discharge it. The same mechanism is responsible for transfer inefficiencies in charge-coupled devices [93]. Charge spooning and the related pumping phenomenon are discussed in Section 5.5. Proper operation requires that these non-ideal currents be small enough so that the cell can maintain state through one refresh cycle.

Figure 3.11 shows the dynamic cell used in this work. The circuit is based on Wade and Sodini's CAM cell, but it adds dual-gate write transistors and a vertical write enable line. A similar cell based on the Yamagata CAM is also conceivable, but a practical implementation would require a process with stacked capacitors, local interconnect, and dual-gate devices. The crossed bit line configuration was a better choice for the fabrication technology available to this project. As was discussed in

Table 3.4
Circuit and Process Data for Cell Test Chip

Technology	CCD-CMOS
Minimum device length	2 μm
Gate oxide thickness	220 Å
Cell dimensions	$33.5 \times 44.5 \mu m^2$
Storage capacitance (half trit)	90 fF

Section 2.1, each of the associative processor chip's PEs has 64 dynamic trit cells, organized into two 32-trit half-words. The PE is also equipped with one dual-port carry cell, which is implemented by simply wiring together the storage nodes of two cells from different half-words. Complete schematics for both the standard and dual-port cells are provided in the appendices.

Early in the course of this work, experiments were conducted to verify the functionality of the cell and to assess the feasibility of using it in an integrated associative processing system [94]. The M.I.T. CCD-CMOS process [95] was then in the final stages of its development, and a single-cell test circuit was included on a wafer of process characterization structures (Table 3.4). Pass transistors were used to isolate the match and bit lines, and source followers were used to observe the high-impedance nodes. Although the test circuit used only *n*-channel transistors, the wafer was fabricated in the complete CMOS process.

Figure 3.12 is an oscilloscope output showing match and mismatch operations. The first write line pulse stores a 0 in the cell. After precharging the match line high, a 0 is presented by driving B_1 low. The match line source follower output *MS* remains high, indicating a successful match. In the second half of the test a 0 is written and a 1 is presented; the falling match line indicates a mismatch.

Read results of the three ternary digits are shown in Fig. 3.13. BS_0 and BS_1 are the bit line source follower outputs. After each trit is written, the bit lines are precharged low and the match line is driven high. Rising bit lines indicate the state of the cell.

The effect of charge spooning on storage time was also investigated, with encouraging results. These are presented in Section 5.5, along with data from other chips.

3.3 Match Sensing

The match operation compares a presented datum to a datum stored in a CAM word, with several possible outcomes. If the match is successful, then the precharged match line will remain high. Unsuccessful matches produce a mismatch current related to the number of mismatching trit positions. Only a small current will flow if only one or two cells mismatch, but a complete mismatch will discharge the match line rapidly. The match sense amplifier must distinguish the successful match results from all the possible mismatch results, and must do so in the presence of undesirable noise signals. This task is quite different from that of read sensing in a random access memory, in

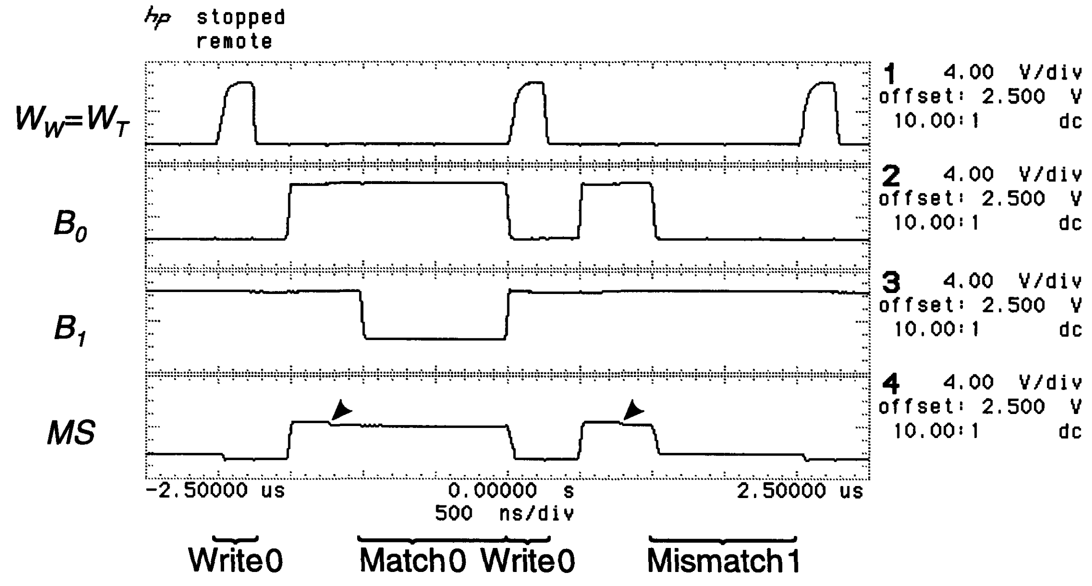


Figure 3.12: Oscilloscope output of match operation on single-cell test structure. The arrows indicate the coupling from the gate of the n-channel pass transistor to the match line at the end of the precharge phase.

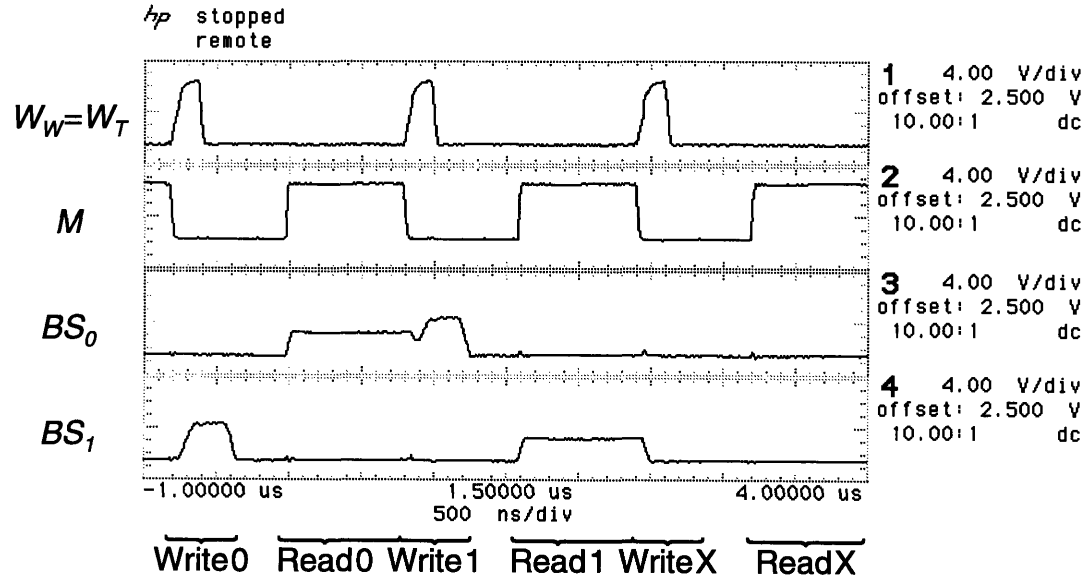


Figure 3.13: Oscilloscope output of read operation on single-cell test structure.

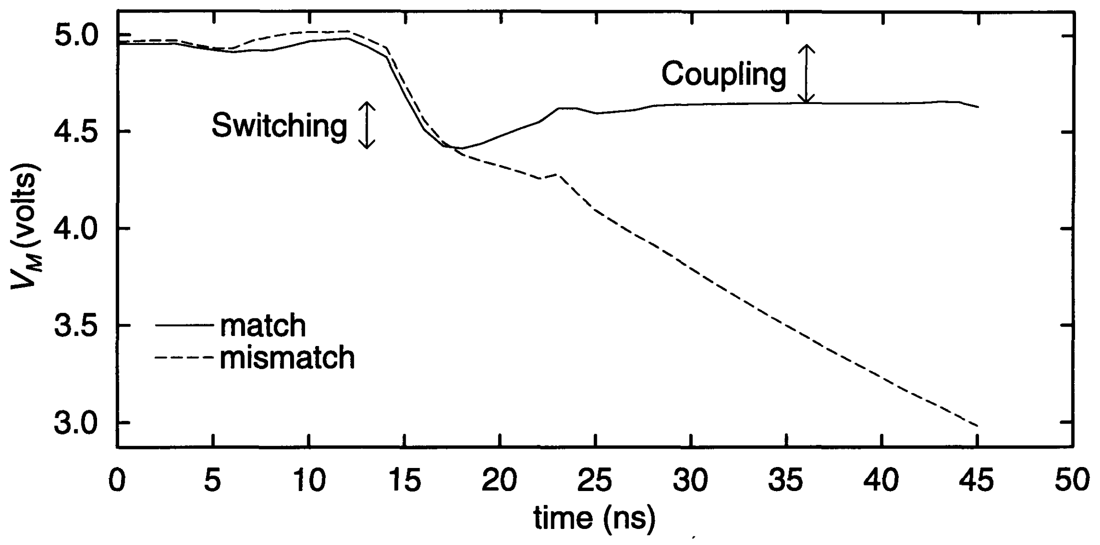


Figure 3.14: Simulated match line potential with switching and coupling noise.

which only two outcomes are possible. The RAM sense amplifier takes a differential input and determines whether $B_0 < B_1$ or $B_0 > B_1$. In contrast, the CAM sense amplifier must sense a single-ended signal that may fall quickly, slowly, or not at all. Some authors have proposed measuring the mismatch current to determine the Hamming distance between presented and stored data [76], but to do so with any accuracy requires well-controlled current sources in the cells, which are difficult to achieve in dense memory arrays.

Figure 3.14 presents simulation results of successful and unsuccessful match operations, with the mismatch case demonstrating a moderate discharge rate. Note that even when the operation is successful, the match line can be pulled down by capacitive coupling to the bit lines. This is the dominant noise signal which the sense amplifier must tolerate. A second source of noise is the transient switching pulse generated when the bit line drivers are enabled. The current surge produces an $L\frac{dI}{dt}$ drop on the supply rails, which in turn couples to the match line. The pulse dies away as the bit lines settle.

3.3.1 Time-Differential Sense Amplifier

The first sense amplifier considered for this work was the time-differential design used on the Database Accelerator chip [96]. As shown in Fig. 3.15, the circuit samples the match line M twice: initially on the rising edge of ϕ_1 and later on the edge of ϕ_2. If the two samples differ by more than a fixed offset voltage, then a mismatch will be indicated when the comparator is latched by ϕ_3. The positive offset voltage is built into the amplifier by intentionally mismatching M_{N1} and M_{N2}, and it must be chosen greater than the magnitude of the possible coupling noise. If set too large, however, a longer signal development phase will be required and the match operation will be slower. The simulation results of Fig. 3.14 might lead the conservative designer to

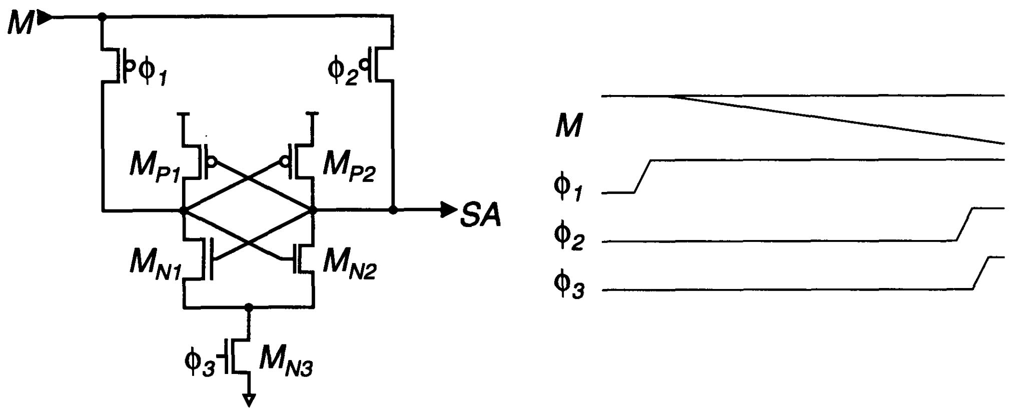


Figure 3.15: Time-differential sense amplifier and control signal timing diagram. The transistors M_{N1} and M_{N2} are intentionally mismatched to build an offset into the comparator.

choose an offset voltage of approximately 600 mV, about twice the coupling noise.

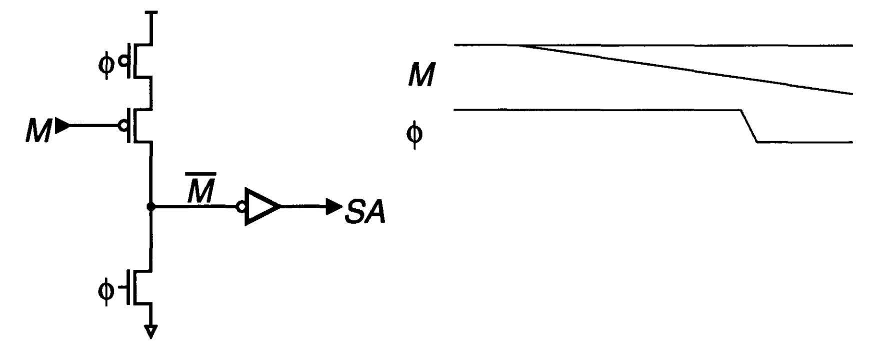
Unfortunately, noise considerations make it difficult to design for a predictable offset voltage. Early simulations of the associative processor indicated that its larger memory array would experience significantly more noise than was present in the Database Accelerator. The danger is that the combination of coupling and switching effects (Fig. 3.14) might pull the match line low enough turn on M_{P1}. The first sample of the match line potential would then be replaced with an unpredictable sample of the noisy supply.

3.3.2 Single-Ended Sense Amplifier

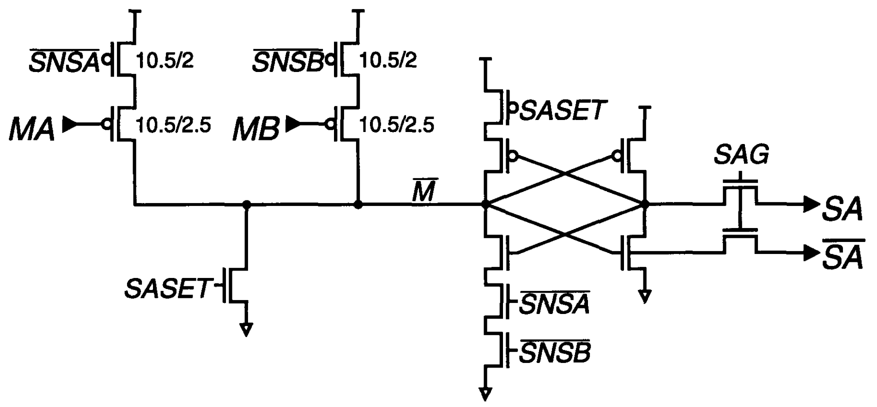
An alternative sensing scheme compares the match line potential with a fixed reference voltage. A switching threshold one p-channel threshold below the positive supply is particularly convenient to implement; Fig. 3.16(a) shows the basic idea. The match cycle begins with ϕ high and the internal node $\overline{M}$ precharged low. After the switching transients have settled, ϕ may be safely brought low to begin sensing. If M falls more than a p-channel threshold voltage below V_{DD}, then $\overline{M}$ will be pulled high and the inverter output will go low to indicate a mismatch.

Compared with the 600-mV offset of the time-differential amplifier, the fixed threshold of this design is more conservative and requires more time for signal development. Fortunately, however, the single-ended approach places fewer constraints on the control signal timing and thereby reduces the control logic overhead. In the case of this associative processor design, the choice of fixed-threshold sensing approximately doubled the signal development phase without increasing the overall match cycle time.

The actual sense amplifier must interface with other components of the processing element, so it is necessarily more complex than the concept design discussed above. The circuits of Figs. 3.16(a) and (b) differ in three important ways. First, the actual



(a)



(b)

Figure 3.16: Single-ended sense amplifier. (a) Simplified design with timing diagram. (b) Circuit as implemented.

circuit uses two separate pull-up stacks to support duplex match lines MA and MB, which correspond to the PE's two half-words of associative memory. Separate control signals $\overline{SNSA}$ and $\overline{SNSB}$ are used for the two stacks, while a third signal $SASET$ controls the pull-down device. Large sensing devices are used to minimize sensitivity to short- and narrow-channel effects, but the other transistors are near minimum size. Second, the simple inverter of the concept design is replaced with a static latch. The five-transistor feedback gate is disabled during the precharge and sensing phases of the match, but it provides active drive for the node $\overline{M}$ at other times. Finally, the function generator requires that the sense amplifier's true and complementary outputs remain stable during the precharge phase of the match. Two pass transistors provide the necessary isolation, and are enabled by the signal SAG.

3.4 Array Driver Circuits

In the previous section, noise tolerance was seen to be an important consideration in sense amplifier design. This section addresses the other half of the noise problem, that of minimizing noise generation in the array driver circuits. Several different noise reduction techniques are used, as appropriate for the various control signals.

The easiest way to reduce switching noise is simply to avoid fast switching. When a signal is not in the critical path, there is no advantage to driving it harder than necessary. As a case in point, asserting the W_W signal too early in the write cycle can only redistribute charge between the cells' storage nodes and the bit lines. The speed of the write will still be limited by the bit line driver current.

The write-word driver circuit of Fig. 3.17 uses an undersized pull-up device M_{P1} to produce the slow rising edge shown in Fig. 3.18. However, such leisurely settling can not be afforded to the falling edge, as it is critical to the write cycle timing. The write-word line must be driven completely low before the bit lines can be recharged for the next operation.

The pull-down devices $M_{N1,2}$ are large enough to satisfy the timing constraints, and they provides sufficient output conductance to prevent coupling problems in the array. Unfortunately, they are also large enough to generate significant switching noise; the driver circuits of 256 PEs could produce a noise pulse of a few 100 mV if turned on suddenly. To prevent this, the small transistor M_{N1} is turned on first, while the larger device is inhibited by the feedback transistor M_{P2}. Once W_W falls more than two threshold drops below V_{DD}, then M_{P2} begins to conduct and pull up the internal node $\overline{WWX}$. An inflection point appears in the W_W waveform as M_{N2} turns on (Fig. 3.18).

A similar noise reduction strategy is used for the bit line driver circuit, with small and large inverters connected in parallel. Instead of a feedback circuit, however, the two control signals $BLXP$ and $\overline{BLXN}$ are used to enable the large driver. This is done so that different timing can be used for the match and write operations. The large driver can be turned on earlier during the match, because the worst-case capacitive load is smaller.

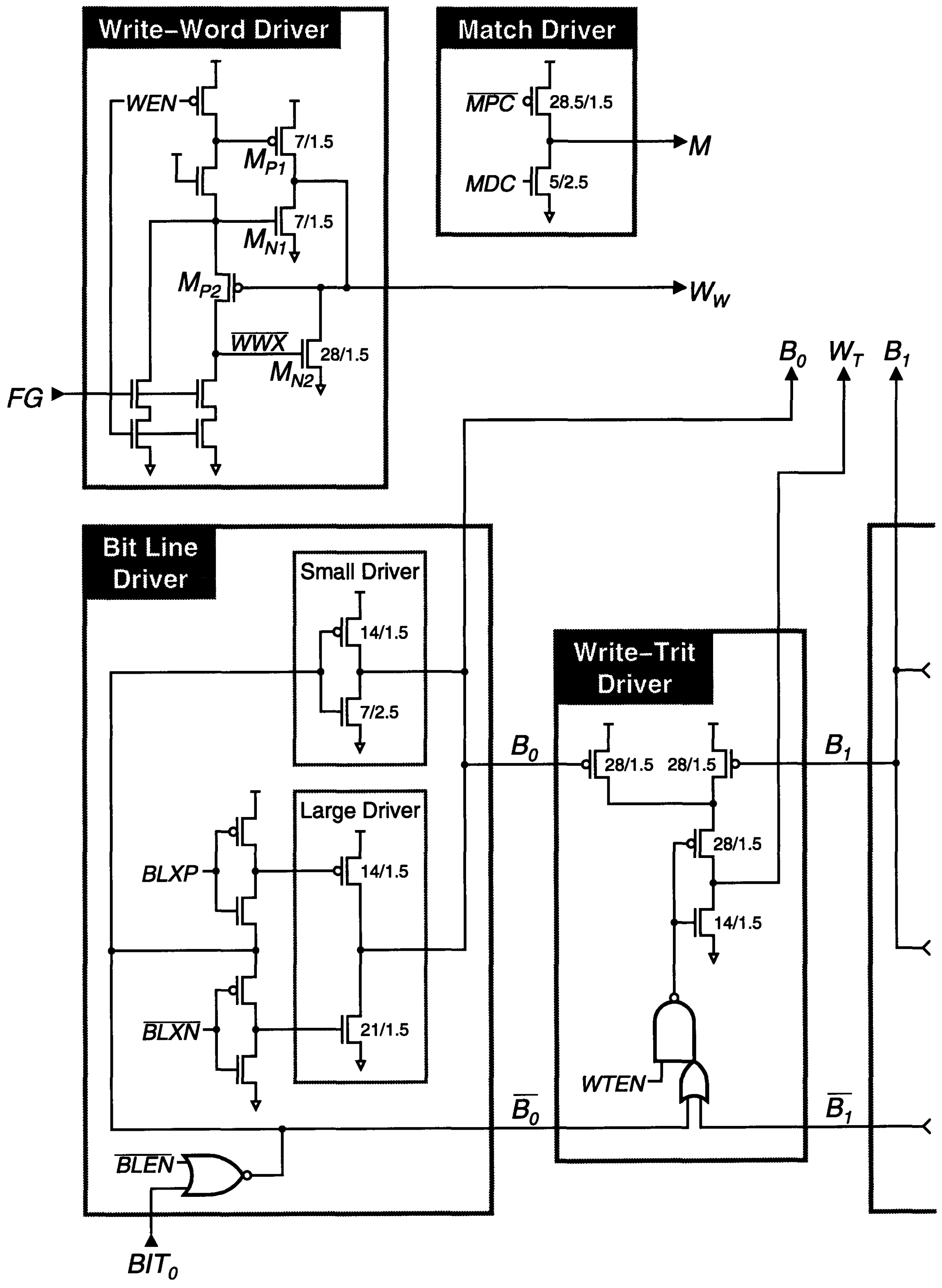


Figure 3.17: Array driver circuits. Each processing element has two rows (half-words) and therefore requires two match drivers and two write-word drivers. Two bit line drivers (only one is shown) and a write-trit driver service each column.

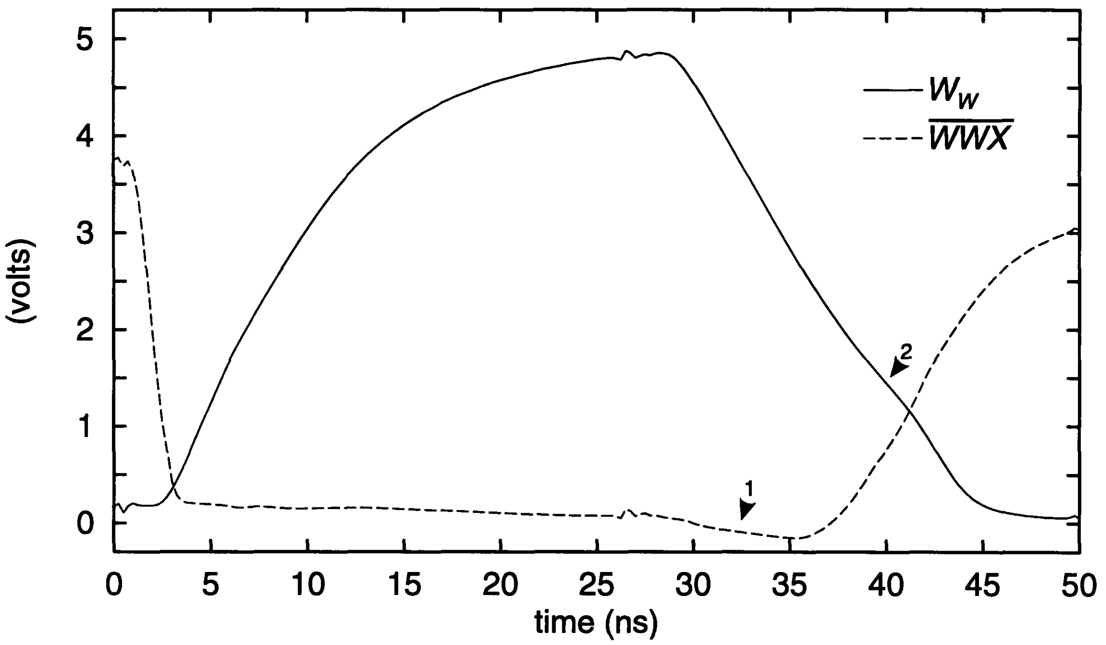


Figure 3.18: Simulated write-word line (W_W) potential. Note the coupling (1) to $\overline{WWX}$. The inflection point (2) occurs when M_{N2} turns on (see Fig. 3.17 and text).

The bit lines are also used to delay the write-trit driver, so that its turn-on transient will not add to that of the bit line driver. The W_T line will not be pulled up until either B_0 or B_1 has fallen more than a *p*-channel threshold voltage below the positive supply. One disadvantage to this design is that the *p*-channel transistors must be drawn twice as wide as would be necessary for a single pull-up device. The area is available, however, because both the horizontal and vertical pitches are already limited by unrelated circuits.

Last, and in fact least, the two-transistor match line driver is the simplest of the circuits in Fig. 3.17. The *p*-channel device precharges the match line at the beginning of the match operation. No aggressive noise reduction measures are necessary because this is the quietest phase of the cycle. The *n*-channel pull-down transistor is used to discharge the line after a successful match operation. This reduces the load on the bit line drivers should the next instruction be a write or a match on the opposite half-word. Because the pull-down transistor has half of the cycle in which to do its work, a weak device can be used.

3.5 Summary

This chapter began with an abridged history of content addressable memory cells, leading up to the five-transistor dynamic cell by Wade and Sodini. The second section described an enhanced dynamic associative processor cell and presented experimental results demonstrating its functionality. The cell uses dual-gate write transistors so that write enable signals can be run both vertically and horizontally. Without these devices, charge sharing problems would render the dynamic cell inoperable. The designer would then face three alternatives:

- Accept the area penalty and use a static or pseudo-static cell to avoid charge sharing problems. Although less dense, the resulting chip would have capabilities similar to the present design.
- Eliminate the write-trit line, giving up the masked write capability. The resulting chip would be a content addressable memory rather than an associative processor.
- Eliminate the write-word line, leaving only write lines running perpendicular to the word organization. The bit lines would then be run through the words, producing a bit-serial/word-parallel design (Section 1.2).

The M.I.T. Associative Processing Project pursued the second approach in its Database Accelerator project [56], and is investigating the third alternative in a project parallel to this one [97].

Two sense amplifier designs were considered, but the time-differential comparator was rejected because of its sensitivity to switching noise. The single-ended design uses a fixed threshold and is more tolerant of noise, but it does require a longer signal development phase. Fortunately, the amplifier's simpler control timing allows this phase to be extended without increasing the overall match cycle time.

The array driver circuits use several techniques to reduce switching noise. The write-word driver and the bit-line driver are the most complex; both use a pair of small and large drivers in parallel. The small driver is turned on first, thereby reducing the initial current surge and the inductive switching noise. The larger driver is enabled later, so that the output conductance of the circuit is not compromised.

Chapter 4

The Processing Element

One advantage of fully-parallel associative architectures is that they require very little word logic in each processing element. The first section of this chapter is accordingly brief, describing the circuit implementation of the word logic's two components: the function generator and the activity register. The second section discusses the special memory cells used for interprocessor communication, and Section 4.3 presents wired-OR and pass transistor circuits for response resolution.

4.1 Word Logic

A binary Boolean function of n inputs is completely specified by a truth table with 2^n entries. As shown in Fig. 4.1, a function generator is essentially a truth table selector. The signals $F_{0...3}$ specify the function to be computed; one of these will be selected by the SA and AR inputs and passed to the FG output. Note that this reverses the usual roles of multiplexer data and control signals. Data bits SA and AR are generated locally at each processing element, while the control signals $F_{0...3}$ are broadcast to all PEs as part of the instruction word.

The function generator is implemented with the precharged multiplexer circuit of Fig. 4.2. Each of the signals $F_{0...3}$ controls the top of a NAND stack with three

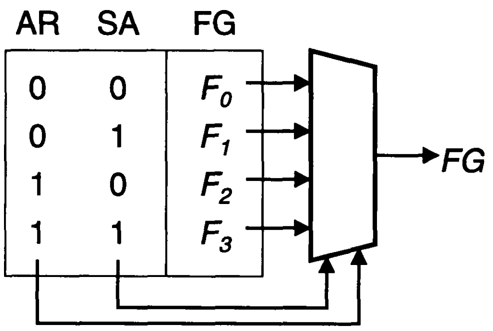


Figure 4.1: Function generator as truth table selector.

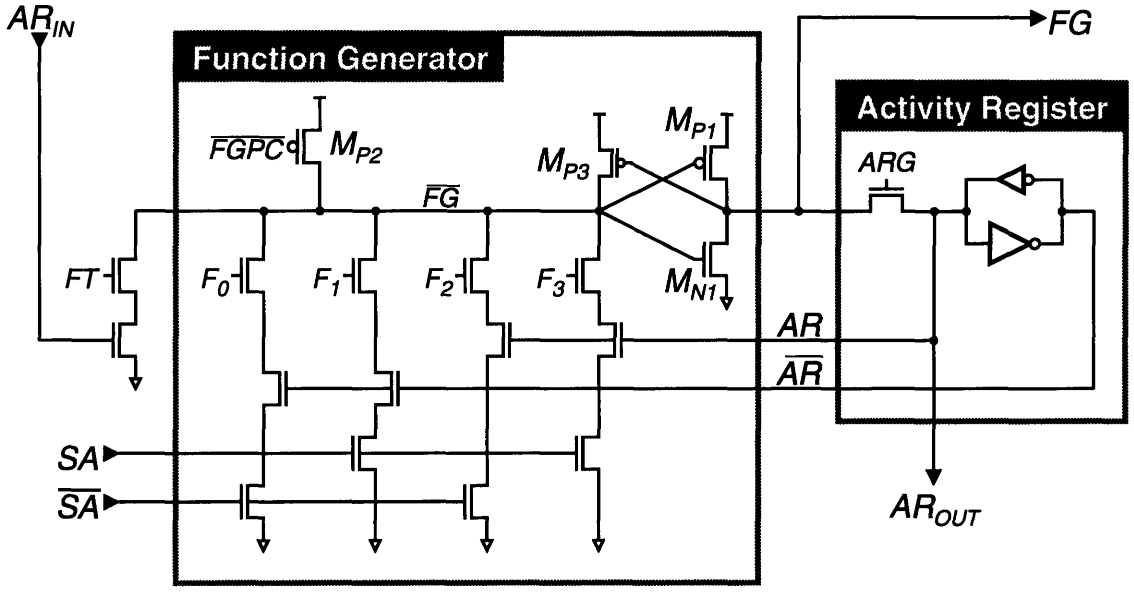


Figure 4.2: Circuit implementation of word logic.

n-channel transistors. The lower transistors in the stacks are controlled by the SA and AR signals and their complements. Exactly one NAND stack is enabled for each possible combination of sense amplifier and activity register states. Asserting the corresponding F_n signal will pull down the node $\overline{FG}$ and cause the output FG to go high. Even if F_n remains low, however, there may be some charge shared between the internal nodes of the other NAND stacks and the precharged node $\overline{FG}$. The weak feedback device M_{P3} ensures that this effect can not turn on M_{P1} and burn unnecessary power in the inverter.

Other multiplexer designs were also considered, including NMOS and CMOS pass transistor circuits [98, 56]. The NAND design has a decisive advantage in this application, however, because the bottom of each stack is tied to ground. This prevents coupling to the SA and $\overline{SA}$ nodes through the gate-source capacitance of the bottom transistors, which in turn allows the function generator to accept the dynamic sense amplifier signals without input buffers.

A simple transparent latch circuit is used for the activity register, with a weak feedback inverter designed to be easily overdriven by the function generator. To facilitate testing, the activity registers of neighboring processing elements are connected together to form a shift register chain through each subarray. Only two additional transistors are required to support the shift operation, because the existing function generator circuit doubles as part of the shift path. The NAND stack on the far left of Fig. 4.2 is enabled by the control signal FT, and it pulls down $\overline{FG}$ if the neighboring PE's activity register is set.

Section 2.1 discussed some of the reasons for preferring very simple word logic. The design of Fig. 4.2 is quite compact, but it would expand rapidly with any increase in complexity. Choosing a 3-input function generator, for example, not only doubles the

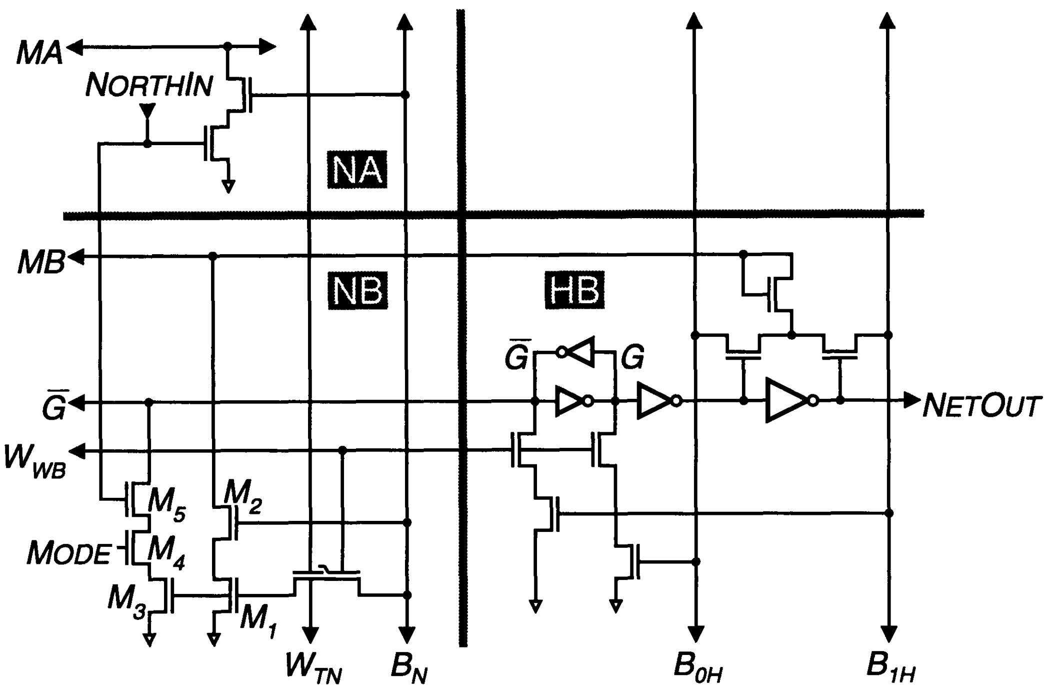


Figure 4.3: Network cell circuits, showing the home cell (HB) and one of the four direction cell pairs (NA,NB).

number of NAND stacks, but also adds one transistor to each stack's height. Another five-transistor register could maintain additional state, but its outputs would have to be routed back to the function generator, further crowding the tight horizontal pitch. The modest performance gains to be obtained by these enhancements can not justify the necessary increase in area.

4.2 Interconnect

Figure 4.3 shows the circuit implementation of the special cells used to interface the word logic to the interprocessor communication network. As discussed in Section 2.2, each processing element has a single output driver, which is controlled by the HB cell. The match-only NEWS A cells function as receivers (only NA is shown), while the NEWS B cells are used to configure the network.

The NA cell is the simplest. A two-transistor NAND stack discharges the match line *MA* when the both the north input and the bit line B_N are true. Because it has only one discharge path, the cell can not perform the complete ternary match operation—there is no way to mismatch a presented 1. One can usually work around this limitation, however, as the move-and-add procedure of Table 2.6 demonstrates. The slight inconvenience to the programmer wins a significant advantage for the circuit designer, in that the incoming signal need not be inverted locally. This permits a dense n-channel implementation and eliminates the need for wells. The NEWS A

Table 4.1
HB Cell Write Behavior

Trit Presented	Trit Written	Network Mode
0	0	Synchronous mode set
1	1	Mode unchanged
X	1	Asynchronous mode set
–	–	Mode unchanged

and B cells are in fact slightly smaller than the associative trit cells.

The HB cell uses a static register and a Koo comparator configuration. It supports full ternary matches, but can not store the X state. The flip-flop inverters are designed with weak *p*-channel devices, so that the cell can be written by pulling one side of the flip-flop down to V_{SS}. Masked writes are performed by holding both bit lines low. This design draws less bit line current than the series pass transistor design of Fig. 3.9, and it does not require a vertical write line.

Writes to the HB cell are decoded by the control logic to set the network operating mode, as summarized in Table 4.1. Writing a 0 trit to the HB column always puts the network into synchronous mode, even if no processing elements are enabled. This behavior prevents the pathological situation in which the bit lines try to reset the HB cell while the asynchronous net tries to set it. Writing the X trit enables asynchronous reconfigurable mode. Since the HB cell can store only binary values, the control logic converts the X to a 1. Masked writes and 1 trits have no effect on the mode.

The NB cell uses a three-transistor NAND stack to gate the network inputs. The left side of the HB flip-flop ($\overline{G}$) is pulled low when all three transistors conduct. That is, the HB cell is set when

$M_3 \Rightarrow$ the NB cell contains a 1 to enable north inputs, and

$M_4 \Rightarrow$ the network is in asynchronous reconfigurable mode, and

$M_5 \Rightarrow$ the network input is true.

With four NAND stacks pulling down a precharged node, this design is quite similar to the function generator described in the previous section. Once again, the NAND design was chosen over pass transistor-based alternatives. While pass transistor designs may provide faster network propagation, they have the disadvantage of capacitively coupling the signal being switched to the control signal at the gate. One must either drive the gate with an active circuit, such as a static register, or else ensure that the network is precharged to a known state before writing the control potential. The first option consumes too much area, and the second unduly complicates the timing of the write operation. In contrast, the NAND implementation allows the configuration state to be stored dynamically on the gates of M_3 and M_1. Since both transistors' sources are connected to the supply, coupling through C_{gs} is eliminated. As in the associative trit cell (Section 3.2), dual-gate transistors are used to support both horizontal and vertical write enables. Transistors M_1 and M_2 are used to examine the cell with the match operation. Like the NA cell, the NB cell can not mismatch a presented 1 trit.

4.3 Response Resolution

The pass transistor responder chain of Section 2.3 is capable of performing all the response resolution tasks required by the associative processor, namely some/none response, responder counting, and responder prioritization. However, it is only one of three response resolvers on the chip. The second computes the some/none response of the HB cells using a wired-OR circuit to produce a result in less time than the pass transistor chain requires. The third resolver uses a similar circuit to compute the OR of all the activity registers. Because the HB cell is used to store both the network state and the responder flag, the AR resolver is needed for polling during ARM settling (Section 2.2).

4.3.1 Wired-OR Some/None Resolver

The wired-OR circuit of Fig. 4.4(a) is used for both the HB and activity register response resolvers [99]. Transistors M_1, M_4, and M_5 form a current source to the node V_{in}. If none of the n-channel input transistors are on, M_5 will pull up V_{in} to just above the switching threshold of the inverter $M_{N3,P3}$. This drives V_{out} low and begins to turn on M_2, establishing equilibrium when the currents through M_1 and M_2 are equal. Any one of the n-channel input transistors can upset the equilibrium, pulling V_{in} low and thereby causing V_{out} to go high.

Figure 4.4(b) shows the relevant transfer characteristics. The solid line represents an ideal inverter, with β-matched square-law MOSFETs [100]. The dashed line is the positive feedback characteristic, in the simple case where

$$\left(\frac{W}{L}\right)_1 = \left(\frac{W}{L}\right)_2, \tag{4.1}$$

and

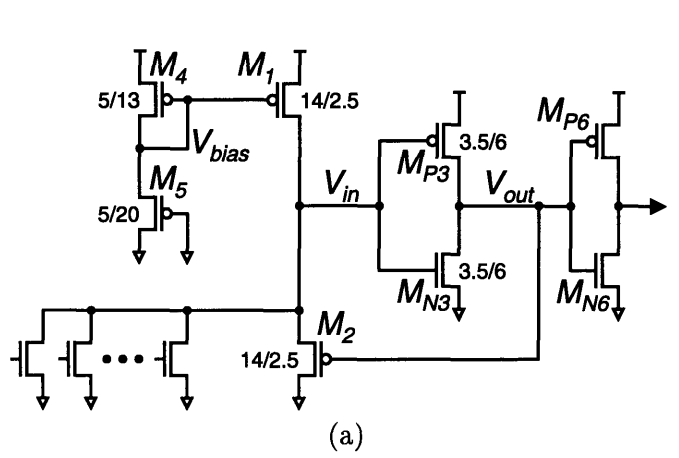
$$\left(\frac{W}{L}\right)_4 = \left(\frac{W}{L}\right)_5. \tag{4.2}$$

By symmetry, $V_{in} = V_{bias} = \frac{1}{2}V_{DD}$ when $V_{out} = 0$. Transistor M_2 acts as a source follower, so the ideal curve has slope unity for increasing V_{out}, until M_1 enters the linear region. In the actual circuit, the inverter is designed to have a switching threshold lower than $\frac{1}{2}V_{DD}$, which has the effect of moving the solid curve to the left. Mismatching the transistors M_4 and M_5 moves the dashed curve to the right, thereby lowering V_{out} at the equilibrium point and ensuring that M_{N6} will be off. The simulated equilibrium point occurs when $V_{in} = 2.6\text{V}$ and $V_{out} = 0.2\text{V}$.

Unlike more conventional precharged designs, this circuit does dissipate some static power

$$P = IV = \left(C_{in}\frac{\frac{1}{2}V_{DD}}{T_{rise}}\right) V_{DD}, \tag{4.3}$$

where T_{rise} is the time required to charge the input node to the equilibrium voltage, approximately $\frac{1}{2}V_{DD}$. Compare this to the worst-case dynamic power for a precharged



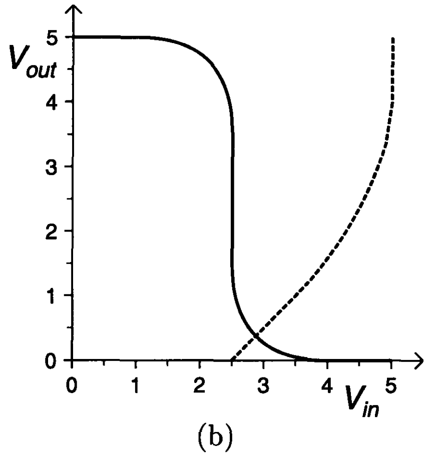


Figure 4.4: Some/none response resolver. (a) Wired-OR circuit. (b) Transfer characteristics for ideal inverter (solid) and source follower (dashed).

implementation,

$$P = \frac{C_{in} V_{DD}^2}{T_{period}}. \tag{4.4}$$

When designed to operate at the same speed ($T_{rise} = T_{period}$), the first circuit's static power is only half the precharged circuit's worst-case dynamic power. Assuming the precharged node is typically discharged on half of the cycles, the two circuits' power requirements are roughly equivalent.

Of course, this first-order analysis neglects several improvements that could significantly reduce power dissipation in both circuits. For example, one could adjust the input swings (subject to noise constraints) or add standby circuits to disable the response resolvers when not in use. These optimizations were not pursued, however, since the response resolver circuits account for only a small fraction of the chip's overall power dissipation. Instead, the static design was chosen for a more compelling advantage, its ability to operate without additional clocks or control signals. This helps simplify the control logic, thereby saving layout area and reducing the design time.

4.3.2 Pass Transistor Chain

Figure 4.5 shows the responder chain circuits used to implement the basic functionality described in Section 2.3. The non-restoring stage (a) differs from the logic of Fig. 2.11 only in the addition of the weak feedback device. Since the inputs are driven through the previous stage's *n*-channel pass transistors, M_{P23} is needed to pull the input above $V_{DD} - V_T$ and turn off M_{P21}. The inverter is ratioed for a low switching threshold to compensate for the limited input swing.

The circuit of Fig. 4.5(b) is functionally equivalent, but it includes additional inverters to restore full logic levels. The inverters are drawn the same size as in the non-restoring circuit, and all but inverter 4 incorporate weak *p*-channel feedback

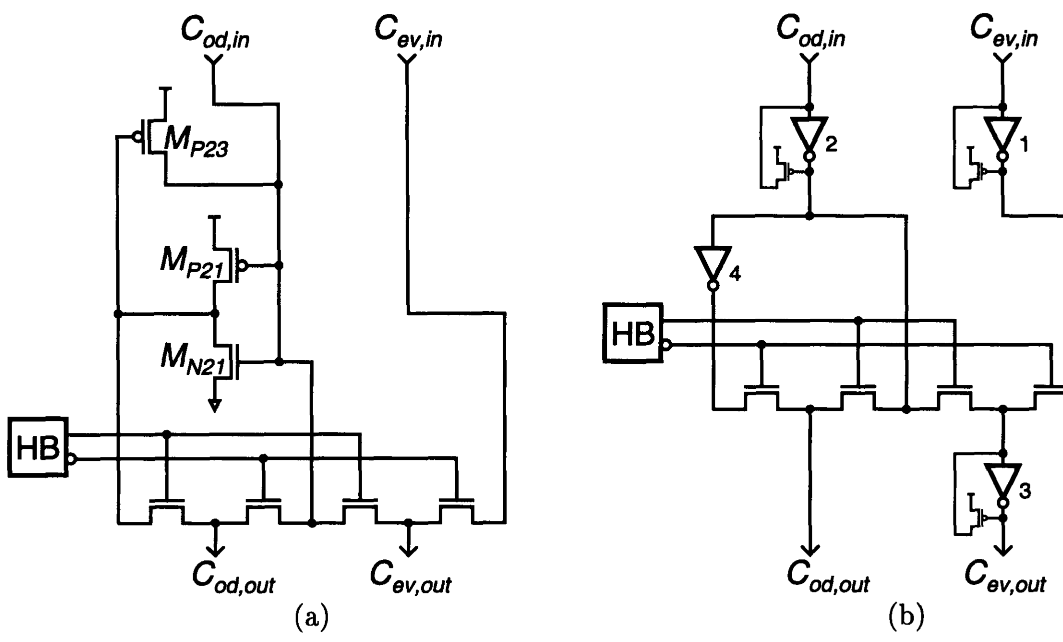


Figure 4.5: Responder chain pass transistor circuits. (a) Non-restoring version. (b) Restoring version.

Table 4.2
HA Cell Match Behavior

Presented	Bit Lines			Match Condition
Value	B_{0H}	B_{1H}	B_{EH}	
X	High	High	High	Match always
0	High	Low	High	Match if even ($\overline{C_{od}}$)
1	Low	High	High	Match if zero ($\overline{C_{od} \vee C_{ev}}$)
–	Low	Low	High	Match— no responders
–	Low	Low	Low	Mismatch— some responders

devices. The associative processor's responder chains are constructed of alternating restoring and non-restoring stages. The slowest signal path through two stages passes through three inverters, or 1.5 gates per stage. One could consider restoring less frequently, trading increased load on the inverters for fewer gates per stage.

The processing elements interrogate their HA cells to determine the state of the responder chain. Figure 4.6 shows the circuit, and Table 4.2 summarizes its behavior. Three discharge paths correspond to the presented values 0, 1, and –. A 0 is presented as part of the responder counting procedure (Section 2.3). The bit line B_{1H} is driven low, and transistors M_{16} and M_{17} will conduct if C_{od} is high. A mismatch indicates that the number of higher-priority responders is odd and the processing element should not be reset. The prioritization procedure makes use of the discharge path comprising $M_{14,15,18}$. A presented 1 will only match if both C_{ev} and C_{od} are low, indicating the absence of higher-priority responders.

The lone diode M_{19} forms the third discharge path, sinking mismatch current to

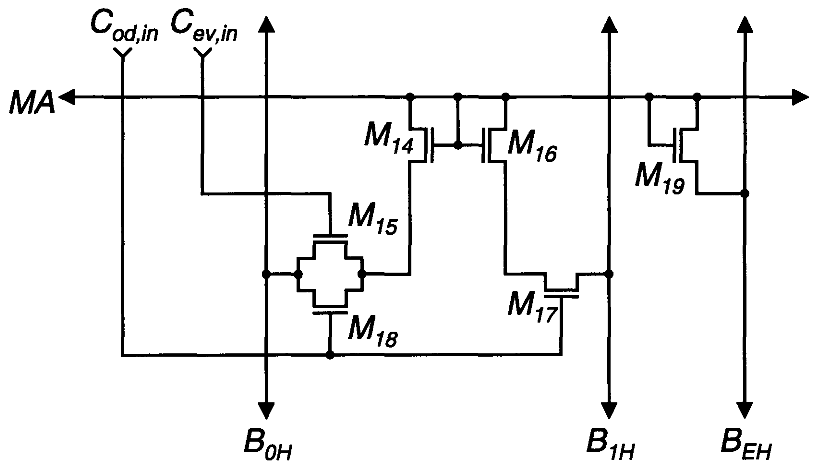


Figure 4.6: Circuit diagram of responder cell HA.

B_{EH}. This bit line is unique in that it depends not only on the instruction being executed but also on the state of the responder set. More specifically, B_{EH} is driven low only if a – is presented and the responder set is not empty, thus providing means to inform the individual processing elements of the global some/none responder state. This responder feedback feature proves to be of significant advantage in the design of the system control path (Section 2.5).

4.4 Summary

This chapter has described the circuit implementation of the word logic and of the specialized cells used in the network and response resolution subsystems. Together with the memory, sense amplifier and write driver circuits of Chapter 3, these components make up the associative processing element.

The circuits of this chapter are all designed for density, occasionally at the expense of performance or flexibility. Both the function generator and reconfigurable network designs used NAND logic instead of pass transistor circuits, thereby permitting more extensive use of area-saving dynamic storage. The NEWS network cells are implemented with all n-channel transistors, producing a compact layout but inconveniencing the programmer, who must deal with cells that can not mismatch the 1 trit. Finally, circuits are made to serve dual purposes wherever possible. The function generator is part of the shift register path, and the HB cell acts as both the responder flag and the network output register.

Chapter 5

Experimental Results

This chapter begins with a brief narration of the chip's design and fabrication history. Section 5.2 describes how the chip was tested, and the subsequent three sections present the bulk of the experimental results. A simple image processing application is described and demonstrated in Section 5.6, and the chapter concludes with a summary of the most significant results.

5.1 Design and Fabrication

Figure 5.1 is a photomicrograph of the associative processor chip. One can immediately observe that it looks much more like a memory than a microprocessor, with large arrays of repeated structures rather than numerous unique functional units. The chip integrates four arrays of 64 processing elements, and each PE has 64 trits of associative memory. The network is folded to connect the PEs in a 16×16 array, and the south edge of the logical network incorporates a shift register for image input and output.

An assortment of commercial and university CAD tools was used for design and verification. The most important of these was the layout editor Magic, part of a tool suite developed at the University of California at Berkeley [101, 102]. It includes a design rule checker and router, but its extraction capabilities are limited, and it is unable to perform physical-to-logical verification. The most important circuits, including the entire PE array and the memory drivers, were laboriously hand-extracted to produce input decks for HSPICE, a commercial circuit simulator. Non-critical control logic circuits were extracted to produce net lists for Irsim, the switch-level simulator distributed with Magic. Verification was performed by comparing the simulator output to a behavioral model coded in C.

The chip was designed for fabrication in the 1.5 μm CCD-CMOS process [95] at the M.I.T. Microsystems Technology Laboratories (MTL), with the Orbit Foresight service [103] to be used as a backup foundry. A common set of design rules was drawn up to provide compatibility with both processes (Appendix C). The completed chip design was submitted to MTL in October 1992. The first wafers were received the

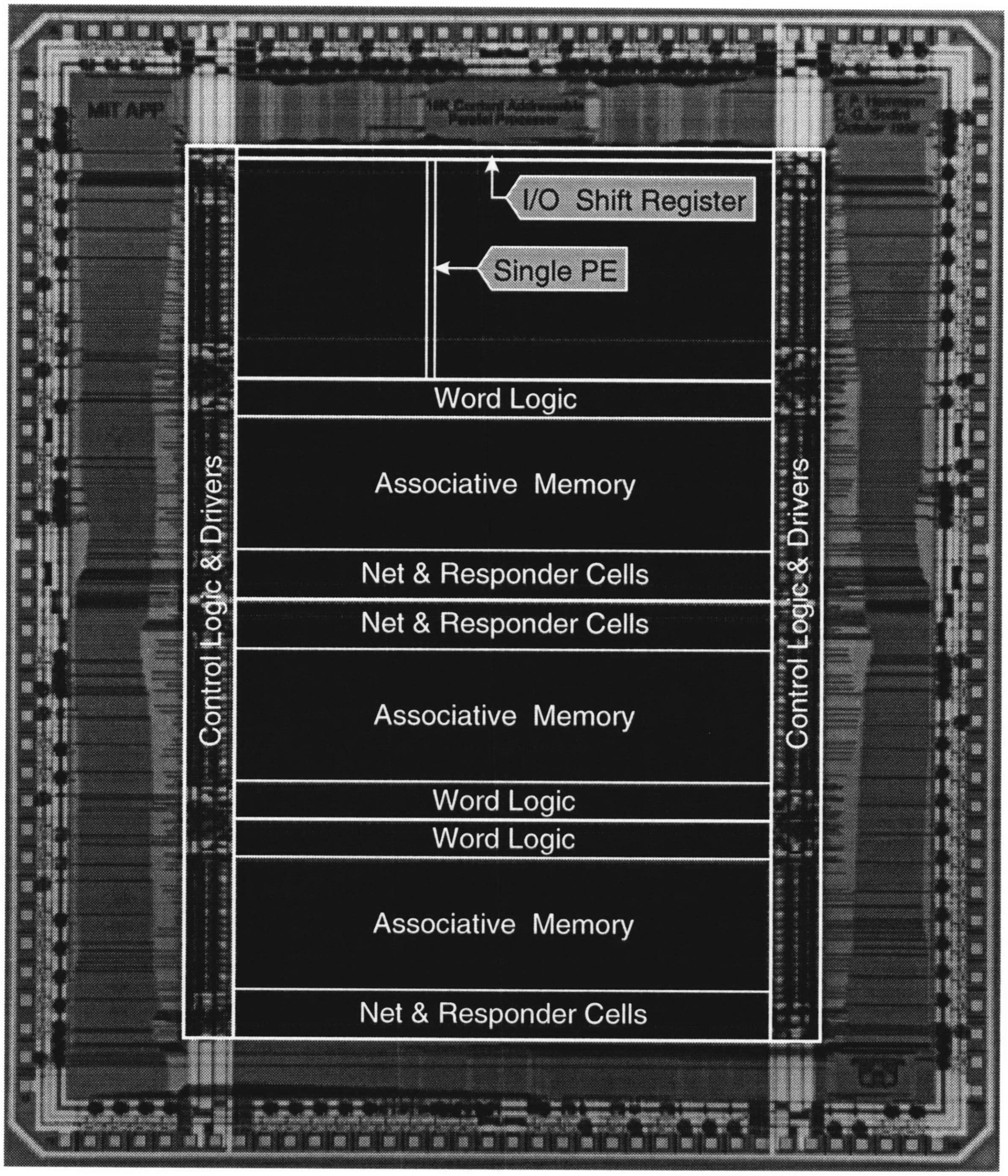


Figure 5.1: Photomicrograph of integrated associative processor.

following February, but the run exhibited anomalous high leakage currents which severely crippled the parts. The I/O shift registers were shown to function at the limit of the test setup (12.5 MHz), and the control logic showed some signs of life, but the greater part of the chip could not be tested. Leakage and metallization problems continued to plague later MTL runs, so that in the end no functional parts were obtained from the M.I.T. lab.

The backup plan to use the Foresight prototyping service was thwarted in December 1992 when Orbit announced the discontinuance of their 1.5 μm process. More encouraging news was received the following month, as MOSIS disclosed plans to offer Orbit's 1.2 μm process on a qualification basis. Qualification runs are free of charge, but MOSIS makes no promises regarding either the quantity of parts to be delivered (if any) or the date of delivery. Because space on the run was limited, it was imperative to ready the submission as soon possible. Time did not permit a redesign for the new process, but minor changes were made to the power wiring to satisfy Orbit's mechanical stress relief rules [103]. A test array of process characterization structures was submitted as a companion project to the associative processor chip. The finished parts were received from MOSIS in September 1993, eleven months after the original design was completed.

5.2 Test Setup and Strategy

Figure 5.2 shows the demonstration system used to test the the associative processing chips. The custom printed circuit board accommodates up to four chips, the minimum number necessary to fully exercise the two-dimensional inter-chip communication. Instruction vectors are received from one of two sources, the associative processor controller or the Hewlett-Packard 16500A Logic Analysis System. The system proved useful during the preliminary testing, when instruction vectors tended to be short and frequently modified, but the demands of later tests exceeded its capabilities. The AP controller [62] is much more powerful if somewhat less flexible: changing a test pattern usually requires recompiling a C program.

With the exception of the power supply, all system components operate under control of the Sun IPX, a UNIX workstation. Performance Technologies' SBus-VME interface connects the AP controller to the host, while the test equipment communicates over the IEEE-488 instrument bus. The experimental data path unit provides a high-speed interface for image I/O [104]. It has sufficient on-board memory to handle bursts of up to eight frames, but its sustained throughput is limited by the slow serial interface to the host. A future system is planned to incorporate an improved data path for sustained real-time I/O and on-the-fly image format conversion [61].

Considerable time was spent developing software to control the instruments, generate test vectors, and analyze results. The investment yielded a versatile automated test system operable from any networked workstation. Operator intervention is required only to insert chips, adjust supply levels, and change probe connections. Of course, there is no real substitute for physical presence, especially when one is si-

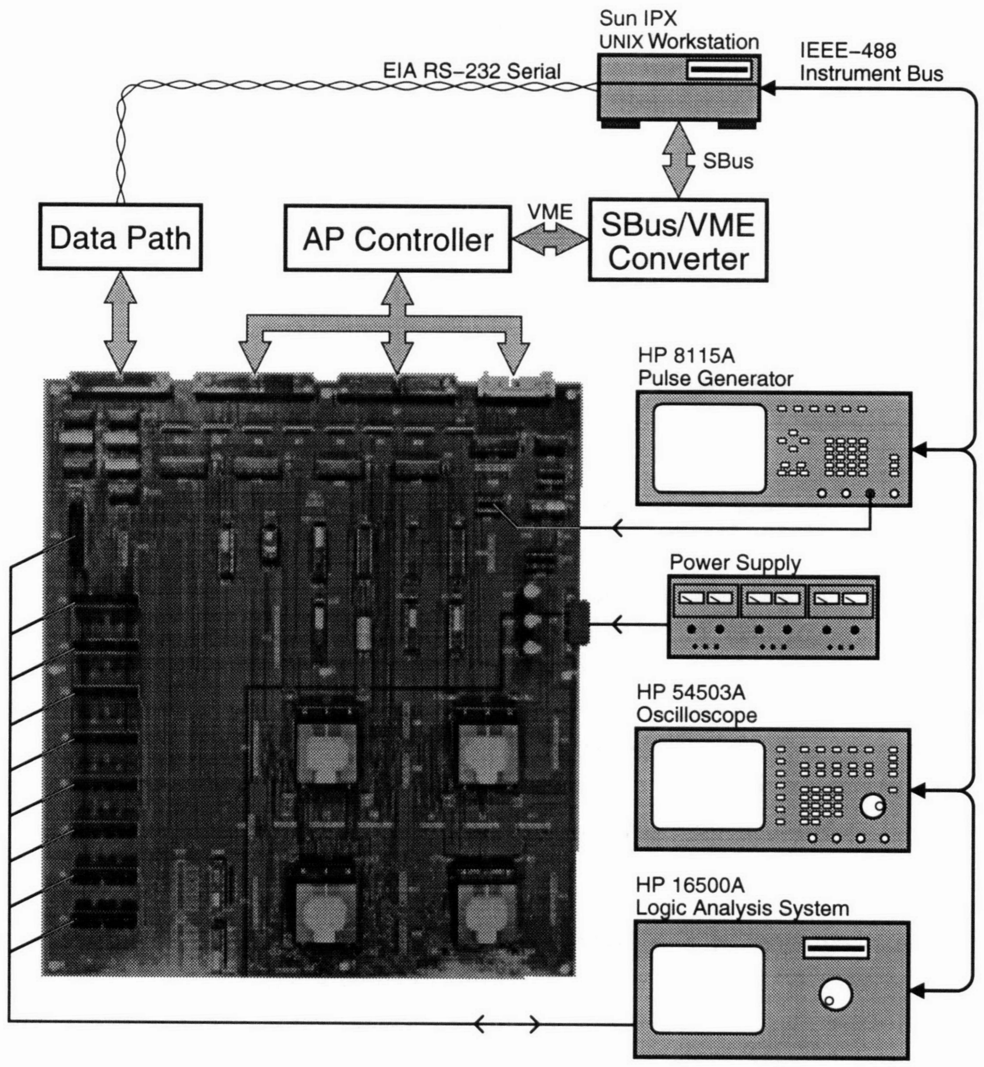


Figure 5.2: Associative processor demonstration system. The photograph shows the printed circuit board with four associative processor chips in the lower left corner.

multaneously engaged in debugging the test system, writing the test software, and (it seemed almost incidentally) testing the chips themselves. For the most part, the test board and AP controller worked as expected, although the test board's clock distribution did require some rework, and the AP controller was bedeviled by coupling problems. The latter were particularly insidious, appearing only at certain clock frequencies. Once diagnosed, they were easily corrected by rerouting signals and adding additional drivers.

Chip testing proceeded in two phases. The early experiments were designed to verify functionality of all chip subsystems. The typical methodology was to load the AR shift registers with a known state, issue some instructions, and then shift out the results. In this way, all the processing elements could be tested in parallel, with a single bit of state obtained from each. Of the seventeen chips received from MOSIS, seven demonstrated full functionality, with all subsystems operating as intended. Nine parts exhibited memory defects (rows, columns, or cells) but were otherwise operational. Only one part was completely lifeless.

These encouraging results established that the chip functions as designed. The purpose of the second phase of testing was to determine how well it works, in quantifiable terms such as power, speed, and storage time. These characterization results are discussed in the following three sections.

5.3 Threshold Effects and Supply Requirements

Orbit's design manual describes a "1.2 micron double poly double metal" CMOS process, not a CCD-CMOS process [103]. The absence of a buried channel implant mask is the most telling difference, although that is of no consequence to this design. A more relevant process characteristic is the difference in dielectric thickness between transistors formed with first and second polysilicon gates. As illustrated in Fig. 5.3, this results in an asymmetrical dual-gate structure, with the threshold voltage being greater on the right half of the device. Orbit does not specify minimum and maximum values for second-poly transistors thresholds, but does list a typical value of 1.55 V, somewhat less than the 2.1 V measured for this run.

Recall that the dual-gate structures are used as write transistors in the associative memory cell of Fig. 3.11. In this configuration they are subject to the back gate effect, which further increases the threshold. Experiments on the characterization structures fabricated along with the associative processor chip indicate that the cell's storage nodes can not be charged to greater than 1.5 V when $V_{DD} = 5\,\mathrm{V}$. Transistors M_{S0} and M_{S1} are turned on only weakly, resulting in reduced mismatch current and a slower match operation.

The problem is more severe in the NEWS B cells used in the ARM network (Fig. 4.3). The NAND stack $M_{3,4,5}$ is designed to fight the HB flip-flop and pull down the $\overline{G}$ line. With the write threshold limiting the drive of M_3, however, the NAND can not win the fight, and $\overline{G}$ remains high. Increasing the supply voltage V_{DD} from 5 to 6 V solves the problem, effectively compensating for the large second-poly threshold.

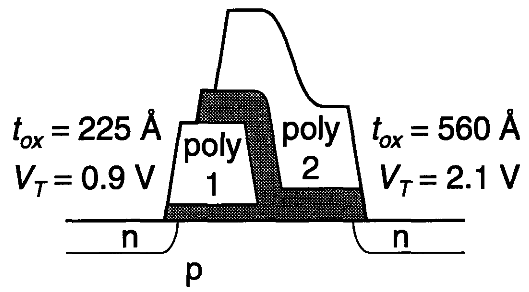


Figure 5.3: Asymmetrical dual-gate device. The thicker dielectric under the second-poly gate results in an increased threshold voltage. The t_{ox} values are Orbit nominal specifications; the threshold voltages are measured results.

Another difference between the NEWS B cells and the associative trit cells is that their write devices are reversed. That is, the second-poly gate is on the *inside* (near the storage node) of the trit cells, but on the *outside* (the bit line side) of the NEWS B cells. When the storage node is charging, the second-poly transistor turns off first because it has the higher threshold voltage. The first-poly device retains channel charge proportional to the difference in threshold voltages,

$$Q_1 = -W_1\,L_1\,C_{ox,1}\,(V_{T2} - V_{T1}), \tag{5.1}$$

which works out to about −7.9 fC for the device in question. This charge must go somewhere when the device turns off. In the trit cell it returns harmlessly to the bit line, but in the NEWS B cell it is injected onto the 210-fF storage capacitance, reducing the storage potential by about 38 mV. Compared to the 1.6-V threshold mismatch (with back gate effect), the charge injection is only a second-order effect, but it could be corrected easily by using the same gate configuration as in the trit cells.

Power dissipation is highly dependent on the activity of the processing elements and on the instructions being executed. With a 6-V supply and 175-ns clock period, an idle chip requires 48 mW to execute Nop (no operation) instructions, while a pathological program designed to move as much charge as possible burns a maximum of 690 mW. More typical image processing applications dissipate about 220 mW, with the network circuits accounting for 25% of that figure.

5.4 Cycle Time and Delay Characterization

Although the chip was designed to operate with a 10 MHz (100 ns) clock, the minimum measured cycle time was 175 ns. The result is disappointing but not particularly surprising, given that the chip was not fabricated in the process for which it was designed. Additional investigations were undertaken to determine whether process differences alone could account for the chip's slow operation, or whether some deficiency of design should be suspected. An electron beam prober would be ideal

Table 5.1
Measured and Simulated Delays

Model	Type	Ring Oscillator		Responder Chain	
		Period	Ratio^{-1}	Delay	Ratio^{-1}
Measured		44 ns	1.0	3.4 ns	1.0
MTL	Level 3	25 ns	1.8	1.5 ns	2.3
MOSIS	Level 3	35 ns	1.3	2.6 ns	1.3
MOSIS	BSIM	46 ns	.96	2.9 ns	1.2

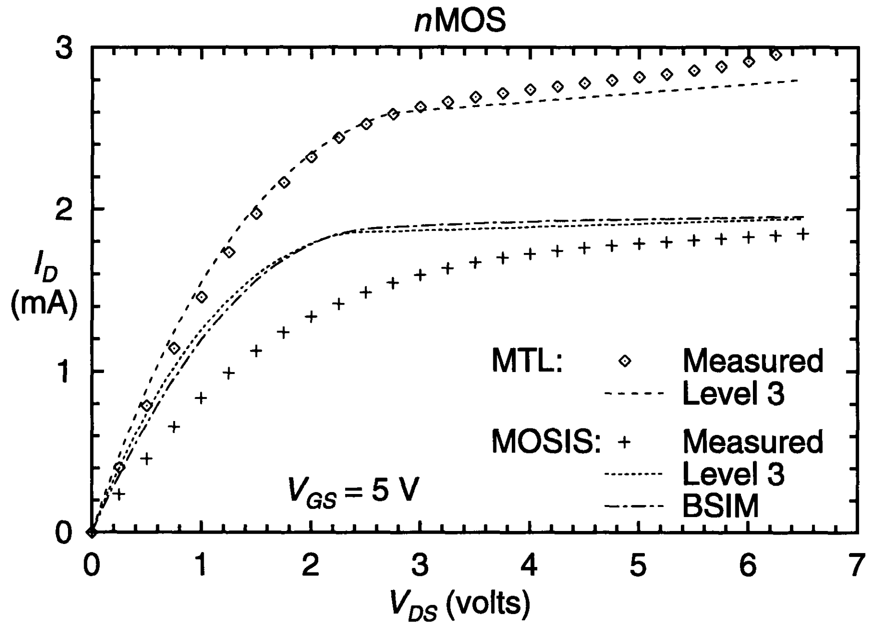
for this kind of work, permitting one to measure propagation delays through most any circuit on the chip. In the absence of such exotic equipment, a more realistic approach is to measure a few accessible circuits and make some general deductions about the chip delay characteristics.

One of the characterization structures on the companion chip is a 65-stage ring oscillator. Its period is a good measure of the delays to be expected in static logic. The responder chain is another easily tested circuit. Its per-stage delay can be computed by toggling the first and last PEs in the chain and measuring the difference in delay at the output. Table 5.1 presents measured and simulated results for both circuits, with the MTL model predicting much faster performance than was realized in silicon. This is the model that was used to design the chip; it was extracted from transistors fabricated at MTL, and its correspondence to the physical devices was verified. The other models were provided by MOSIS, extracted from measured devices on the Orbit qualification run. The MOSIS models are somewhat less optimistic, but even these underestimate delays in three of the four cases.

The I-V curves of Fig. 5.4 help explain the circuit simulation results. While the MTL models match the MTL measured data fairly well, they can not be expected to describe the Orbit devices. Indeed, the MTL models predict substantially more current than the Orbit transistors provide, especially in the p-channel case. The MOSIS models provide a reasonable estimate of the saturation current, but the n-channel curves do not fit the measured data for intermediate values of V_{DS}. This explains their less accurate modeling of the responder chain circuits (Fig. 4.5), in which weak p-channel feedback devices are pulled down through n-channel pass transistors.

The memory array design was simulated again using the MOSIS BSIM model, and the results were consistent with the experimental observations. Rising edges were particularly slow in the new simulations, due to the weakness of the Orbit p-channel devices. The write-word line rise time, for example, was 16 ns in the original design, but 22 ns in the MOSIS simulation. The bit line slowed even more dramatically, with rising transitions requiring 34 ns rather than 17 ns.

It is clear from these results that the associative processor chip was implemented with much weaker transistors than those for which it was designed. Experimental results with two easily measured circuits show that the circuits fabricated at Orbit are 1.8 to 2.3 times slower than the MTL simulations, and simulations with MOSIS models show some signals to be twice as slow as in the original design. One can not



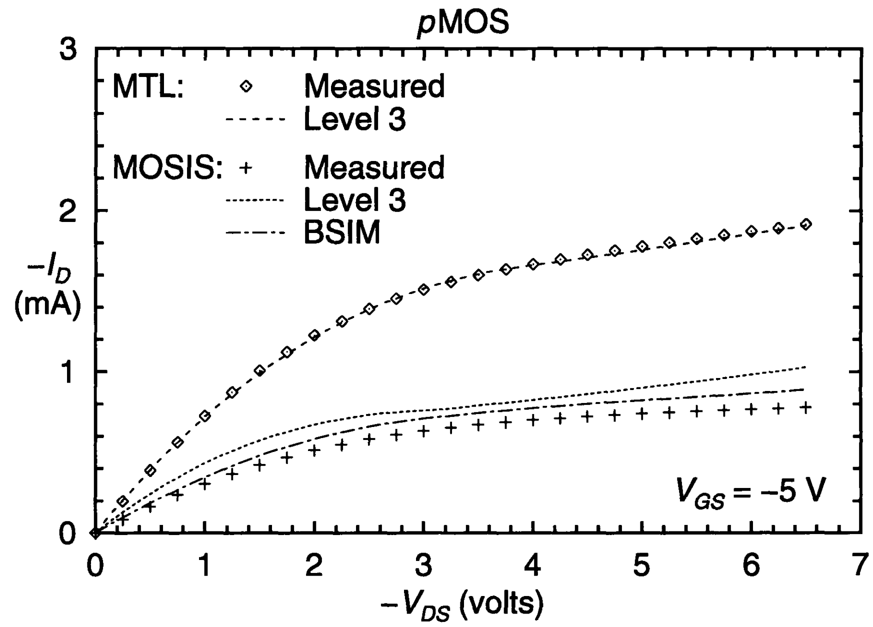


Figure 5.4: Measured and simulated transistor characteristics for n-channel and p-channel devices. Except for the MTL measurements, all data are for (10 μm/1.5 μm) devices. As no MTL devices of that size were available, (20 μm/1.5 μm) devices were measured instead, and half the drain current was reported.

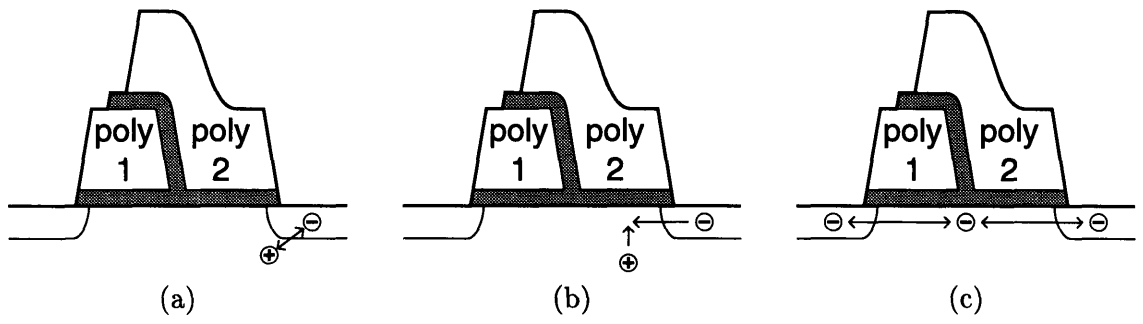


Figure 5.5: Non-ideal charge transfer mechanisms. (a) Leakage. (b) Pumping. (c) Spooning.

be entirely certain what performance might have been achieved had the chips been fabricated at MTL, but the difference between the designed (100 ns) and measured (175 ns) clock periods can be plausibly attributed to differences between the Orbit and MTL processes.

5.5 Charge Storage

If the dynamic memory cells behaved ideally, charge would be transferred on or off the storage node only when both W_W and W_T were asserted. Unfortunately, real memory cells also transfer charge by various non-ideal mechanisms, as diagrammed in Fig. 5.5.

Leakage The term is used collectively here to describe several paths by which current can flow in or out of the cell. These include reverse current across the source-substrate junction, subthreshold conduction through the write device, and possibly surface conduction in the field region. Diode leakage can only discharge the cell, but the other currents could conceivably flow in either direction. With the high n-channel transistor and field thresholds of the Orbit process (MOSIS measured $V_{T,field} = 15.6$ V), reverse diode current is the dominant leakage effect.

Pumping When the inside gate is raised, the channel fills with electrons from the storage node. Not all of the electrons will return to the storage node when the device turns off; some may be stay behind in interface traps. If the electrons detrap while the device is off, then they may recombine with holes from the substrate. The removal of electrons produces a current into the storage node [105, 106],

$$I_{CP} = q\, f\, WL\, \overline{D_{it}}\, \Delta E, \tag{5.2}$$

where q is the electron charge, f is the pumping frequency, WL is the area of the gate, and $\overline{D_{it}}$ is the effective density of contributing interface states for the range of energies ΔE.

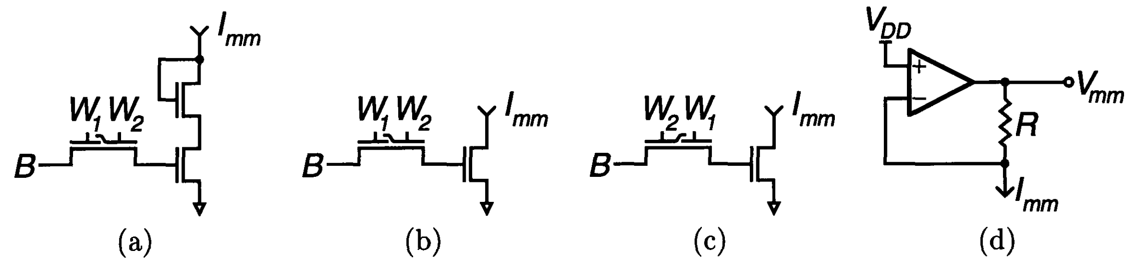


Figure 5.6: Charge storage test circuits. (a) Half of the single-cell test chip fabricated at MTL (see Section 3.2). (b,c) Similar circuits fabricated by Orbit on the same run as the associative processor chip. (d) External circuit using transimpedance amplifier and resistor to set the drain voltage for (a,b,c) while monitoring I_{mm}. The output voltage $V_{mm} = V_{DD} + R\,I_{mm}$, with useful values for R ranging from 1 to 30 kΩ.

Spooning If the electrons under the inside gate do not recombine after detrapping, then they may diffuse across the write transistor to the bit line. Similarly, electrons trapped under the outside gate may eventually find their way to the storage node. Thus charge spooning can work in both directions: either discharging the storage node by spooning electrons onto it, or charging the node by spooning electrons to the bit line.

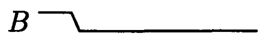
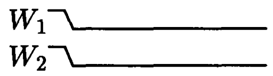
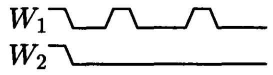
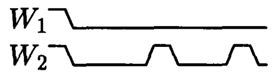
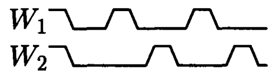
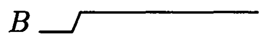
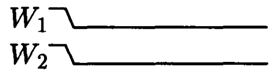
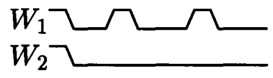
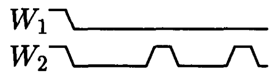
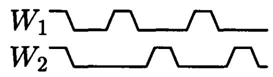
Of course, all three transfer mechanisms can operate simultaneously, making it difficult to measure the independent effects of each process. Charged pumped onto the storage node leaks away through the junction, while charge spooning can help or hinder either process. Whether the storage potential increases or decreases depends on which mechanism transfers charge most efficiently, which in turn depends on both the physical characteristics of the devices and on the test waveforms used.

Two sets of charge storage experiments were performed. The first used the small test circuits of Fig. 5.6, with off-chip drivers controlling the write and bit lines. The external circuit (d) was used to measure the mismatch current. The second set of experiments worked with the associative processor chips themselves, observing the cells in the environment of a large array.

5.5.1 Test Circuit Experiments

Table 5.2 presents experimental results for the three test circuits, using eight test procedures. Each experiment began by writing the storage node from the bit line. The write lines were then brought low, and the bit line was toggled to the opposite state. One, none, or both of the write lines were then pulsed repeatedly for the duration of the experiment. Observation of the mismatch current I_{mm} indicated whether the storage node was being charged or discharged, and the rise or fall time to $I_{mm} = 50\,\mu$A was recorded. (Early simulations of large arrays indicated that a mismatch current of $50\,\mu$A could be detected by the sense amplifiers, and that value was used in subsequent experiments in the interest of comparability.) The oscilloscope

Table 5.2
Results of Charge Storage Experiments on Test Circuits

Experiment Number	Procedure	MTL (a)	Orbit (b)	Orbit (c)	Units
	Discharging Experiments				
					
Exp. 1		3.2	1.4	1.5	seconds
Exp. 2		3.2	*	.59	seconds
Exp. 3		58 k	3.0 k	*	writes
Exp. 4		32 k	5.4 k	*	writes
	Charging Experiments				
					
Exp. 5		*	*	*	
Exp. 6		*	280 k	*	writes
Exp. 7		1.7 M	*	5.4 k	writes
Exp. 8		145 k	*	11.6 k	writes

* No charging or discharging was observed. In Exps. 2–4, the node was presumably held high by pumping effects.

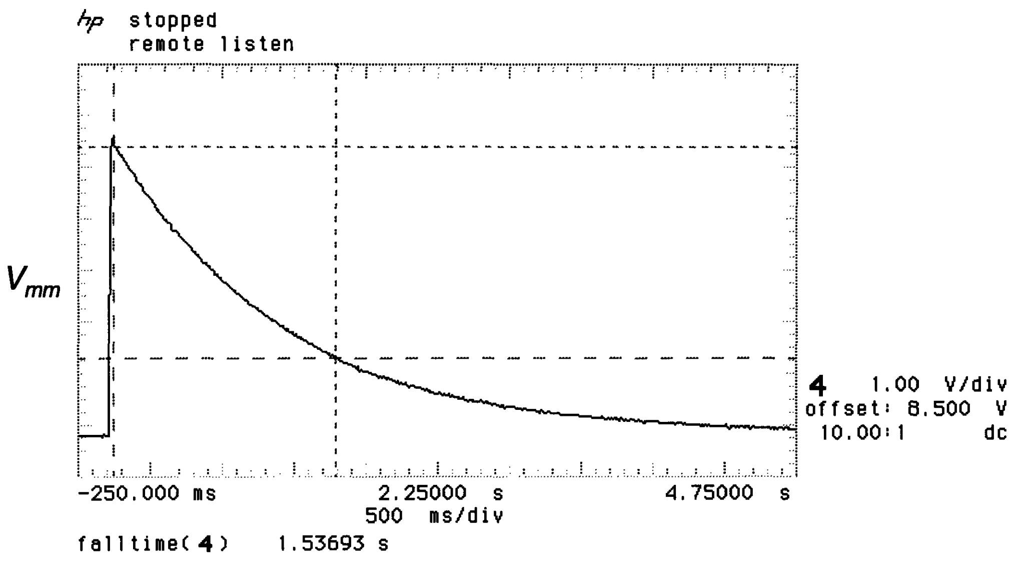


Figure 5.7: Oscilloscope output of leakage test (Exp. 1) for the circuit of Fig. 5.6(c). For this experiment, $V_{mm} = 5.28\,\text{V} + (30\ \text{k}\Omega)\,I_{mm}$. The resistance was chosen to maximize the output swing without saturating the measurement circuit's operational amplifier.

output of Fig. 5.7 shows the result of the leakage test (Exp. 1) on circuit (c), with a measured fall time of 1.5 seconds.

The MTL test chips containing circuit (a) were fabricated and measured in 1989, early in the course of this project. (Functional results were presented in Section 3.2.) Results of charge storage experiments indicated that spooning down (discharging) was the dominant non-ideal charge transfer mechanism. When both W_1 and W_2 were active (Exp. 4), the cell maintained its state through 32 000 write operations. Less efficient spooning was observed when W_1 was disabled and W_2 acted alone (Exp. 3). However, when only W_1 was pulsed then the storage time was limited by leakage current. This is not surprising, since it was the outside half of the device that had a source of electrons from the bit line.

No charge pumping was observed when pulsing the W_1 gate alone (Exp. 6); it may therefore be surmised that spooning was the operative charging process in Exp. 8. The slow charging of Exp. 7 is best by explained subthreshold leakage current—no charge pumping would be expected from the outside W_2 gate, but capacitive coupling between the write lines could conceivably pull W_1 high enough to permit some subthreshold conduction.

The Orbit circuits (b,c) demonstrated consistently shorter storage times than did the MTL parts, and they were especially efficient at charge pumping. This may be attributed to a lower-quality dielectric with a greater interface trap density, particularly in the unoptimized second-poly devices. Test circuit (c), which has the second-poly gate on the inside, pumped up in every experiment with W_2 active. Spooning effects were also observed, with the worst case being Exp. 3 on circuit (b). The discharge rate was slower in Exp. 4, probably because charge pumping subtracted from the spooning current.

These results give some indication of the relative magnitudes of the leakage, pumping, and spooning currents in the Orbit and MTL processes. The test circuits are of limited use, however, in predicting the behavior of memory cells in the array. A second set of characterization experiments was necessary.

5.5.2 Memory Array Experiments

Table 5.3 presents results of experiments on the associative processor chip's memory array, using both NEWS B cells and trit cells. The table's organization is similar to that of Table 5.2, but several differences of construction and experimental procedure should be kept in mind when comparing the results.

Consider first the effect that large arrays have on minority carriers in the substrate. In bulk p-type silicon, a minority electron can be expected to travel a diffusion length L_n before recombining. If it finds its way into the depletion region associated with the storage node, then the electric field will sweep it across to the other side of the junction, thereby contributing to the reverse diode leakage current. An isolated memory cell stands a good chance to collect any carriers generated within a diffusion length, but a cell in an array must compete with its neighbors for minority carriers. Each associative processor cell measures $32\,\mu\text{m} \times 33.5\,\mu\text{m}$, smaller than the typical electron diffusion length of $L_n = 60\,\mu\text{m}$ [86]. Other things being equal, one might expect the storage time to increase by a factor N equal to the number of cells fitting within a disc of radius L_n,

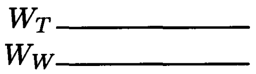
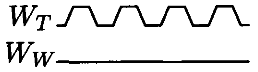
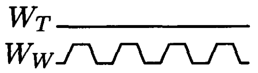
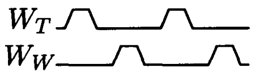
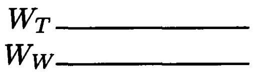
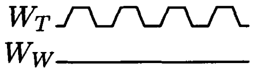
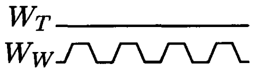
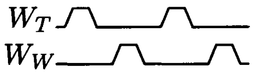
$$\begin{aligned} N &= \frac{\pi L_n^2}{A_{cell}} \\ &= \frac{\pi(60\,\mu\text{m})^2}{32\,\mu\text{m} \times 33.5\,\mu\text{m}} \approx 11\,. \end{aligned} \tag{5.3}$$

Realistically, however, other things are decidedly not equal: the bit line and match line diffusions can also act as collectors competing for minority carriers, the trit cells have two storage nodes (of which at most one is ever charged), and the actual diffusion length of the Orbit material is unknown. Still, the mechanism described above can help explain why the storage times of Exp. 9 are considerably greater than those of Exp. 1, and it also accounts for the observation that cells near the edge of the array tend to discharge faster.

The second difference between these experiments and those using test circuits is one of procedure. Because the mismatch current I_{mm} could not be measured directly, the sense amplifiers were used to discriminate between matches and mismatches. However, the match line signal depends not only on I_{mm} but also on the signal development time and the coupling noise (Section 3.3). The worst case for false matches is when the signal development is short, allowing less time to integrate the mismatch current. For this reason, the discharge experiments were conducted at the chip's minimum clock period of 175 ns. Conversely, the charging experiments used

Table 5.3

Results of Charge Storage Experiments on the Memory Array

Experiment Number	Procedure	NEWS B Cells[1]	Trit Cells[2]	Units
	Discharging Experiments			
Exp. 9		115	15	seconds
Exp. 10		*	330 k	writes
Exp. 11		17 M	1.6 M	writes
Exp. 12		*	30 k	writes
	Charging Experiments			
Exp. 13		*	*	
Exp. 14		16.5 k	*	writes
Exp. 15		*	1.8 k	writes
Exp. 16		44 k	6.4 k	writes

1 The NEWS B cells have the same gate configuration as the circuit of Fig. 5.6(b).
2 The trit cells have the same gate configuration as the circuit of Fig. 5.6(c).
* No charging or discharging was observed.

a 3-μs clock period, the maximum permitted by the associative processing system's host computer interface. The slow clock is appropriate for testing even though the intended operating speed is much faster. Allowing more time for signal development with a single mismatching cell simulates the case in which several weakly charged cells contribute mismatch current: one cell at 3 μs is equivalent to seventeen cells at 175 ns.

The third and last difference is that the memory array tests make use of on-chip circuits to drive the write and bit lines. The simple test patterns of Table 5.2 can not be used, because the control logic brings the trit cells' bit lines high between writes, preparing for a possible match instruction to follow. The NEWS B cells' lines are driven low between writes for the same reason.

In general, the cells in the array charged more quickly than might have been expected from the test circuit results. Charge pumping was observed only when the inside gate was active, with the worst cases for the NEWS B and trit cells occurring when the inside gate acted alone (Exps. 14 and 15, respectively). One disappointing result was the finding that trit cells charge after only 1.8 k writes. Since three writes are required to refresh each of the 64 trit cells, the system will need to devote 11 percent of its cycles to doing refresh. This is more than the one or two percent overhead anticipated, but it is workable in a prototype system.

The discharging results are less easily summarized, except in the observation that none of the discharge mechanisms proved as limiting as the charge pumping process. In Exp. 9, the NEWS B cells leaked much more slowly than did the trit cells, but it would be reckless to conclude overmuch from this one curious result. The currents involved are in the low tens and single digits of femtoamperes and may be sensitive to an assortment of second-order effects. Experiments 11 and 12 are notable in that the trit cells discharged in both cases, although test circuit (c) had remained charged in analogous experiments. The reverse is true of the NEWS B cells in Exp. 12 and the analogous circuit (b) in Exp. 4. In all three cases, charge pumping tends to oppose the discharge processes. Small differences in physical construction or experimental procedure may tip the balance and reverse the direction of net charge transfer.

5.6 Demonstration Application

Machine vision and image processing are two separate but overlapping fields of study. Both are subjects of active and ongoing research, and both are outside the scope of the present work. This section briefly describes a simplified application as it was demonstrated on the associative processor. The intention is to establish the utility of the associative processor in performing a pixel-parallel task, not to propose or defend a particular algorithm or method.

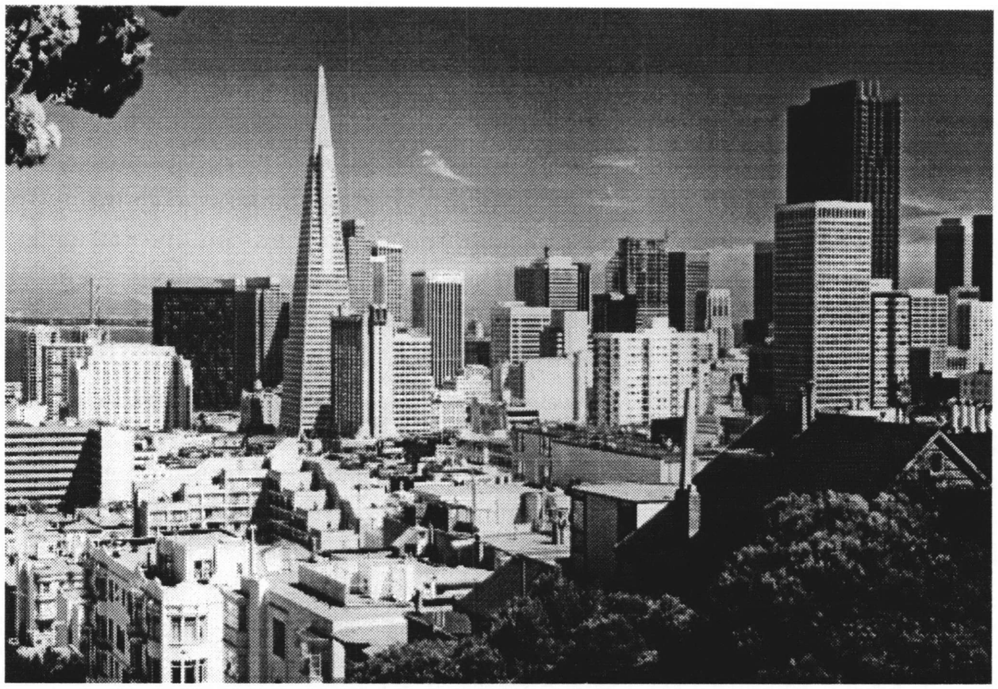
Figure 5.8(a) shows an unprocessed image of the San Francisco skyline, with the landmark Transamerica building and part of the San Francisco-Oakland Bay Bridge. A higher level vision algorithm (*e.g.,* identifying pyramidal buildings) might require a preprocessing step to remove unnecessary detail, such as the windows of the buildings. One could try a simple low-pass spatial filter, but this operation destroys not only the distracting details but also the useful edge information, so that the resulting image is merely blurred (Fig. 5.8(b)). An ideal preprocessing operation would filter only the connected regions while preserving segment boundaries.

A smooth and segment algorithm was implemented on a simulator of the associative processor before the hardware was built. The two-step procedure is the discrete-time analog of the fused resistor approach [7], and has also been implemented with CCDs in an analog focal plane processor [8].

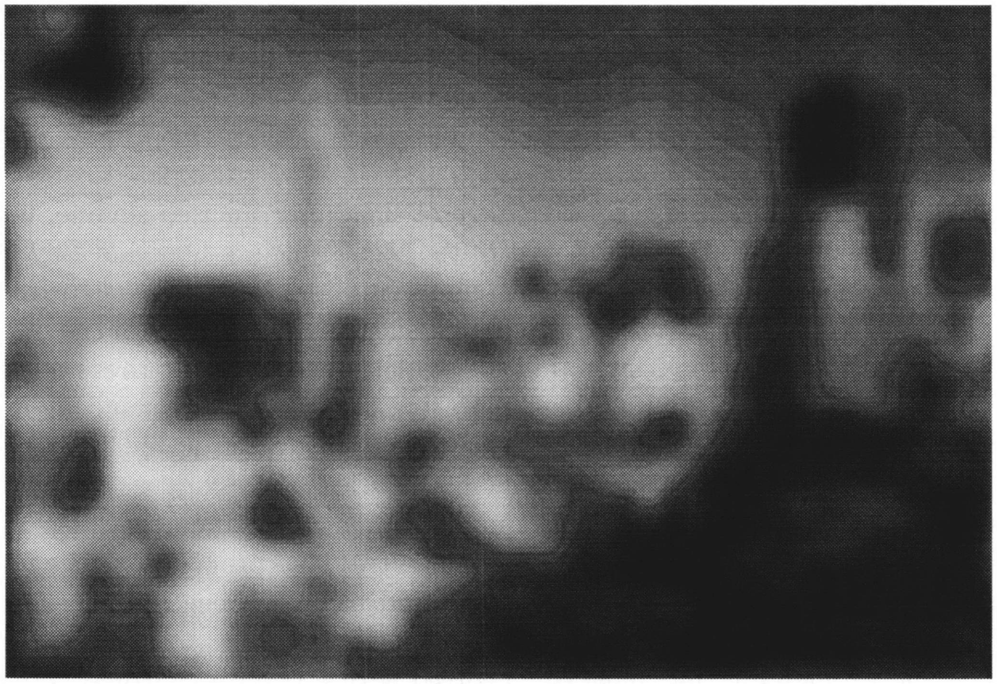
In the smoothing step, the image is convolved with a two-dimensional kernel such as the approximate Gaussian,

$$\frac{1}{8}\left(\begin{array}{ccc} & \boxed{1} & \\ \boxed{1} & \boxed{4} & \boxed{1} \\ & \boxed{1} & \end{array}\right). \tag{5.4}$$

In the segmentation step, each pixel compares its value with those of its four nearest

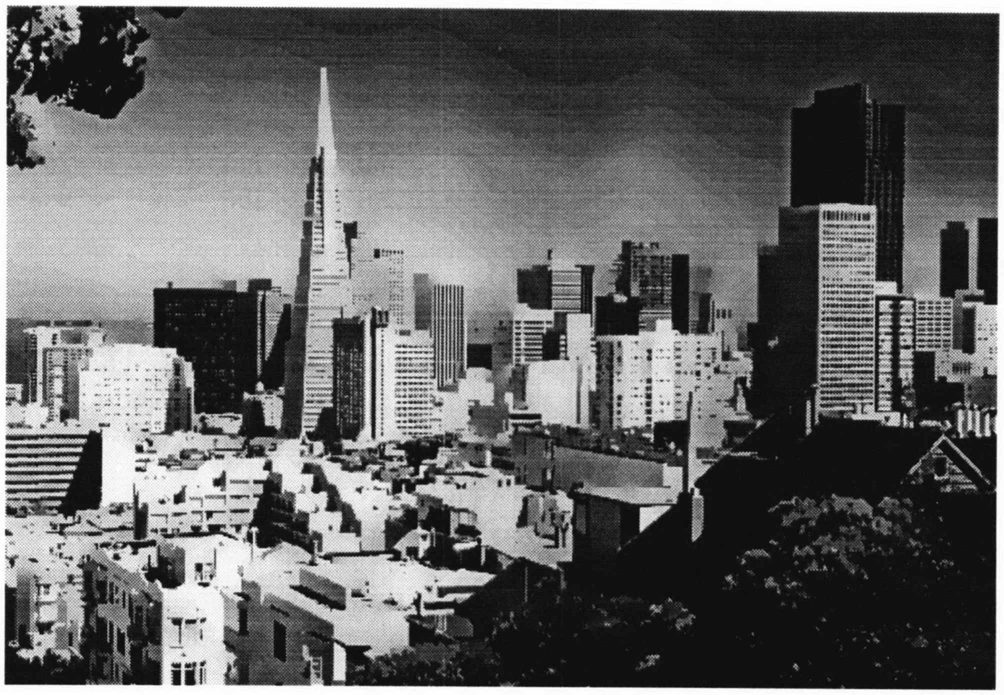


(a)

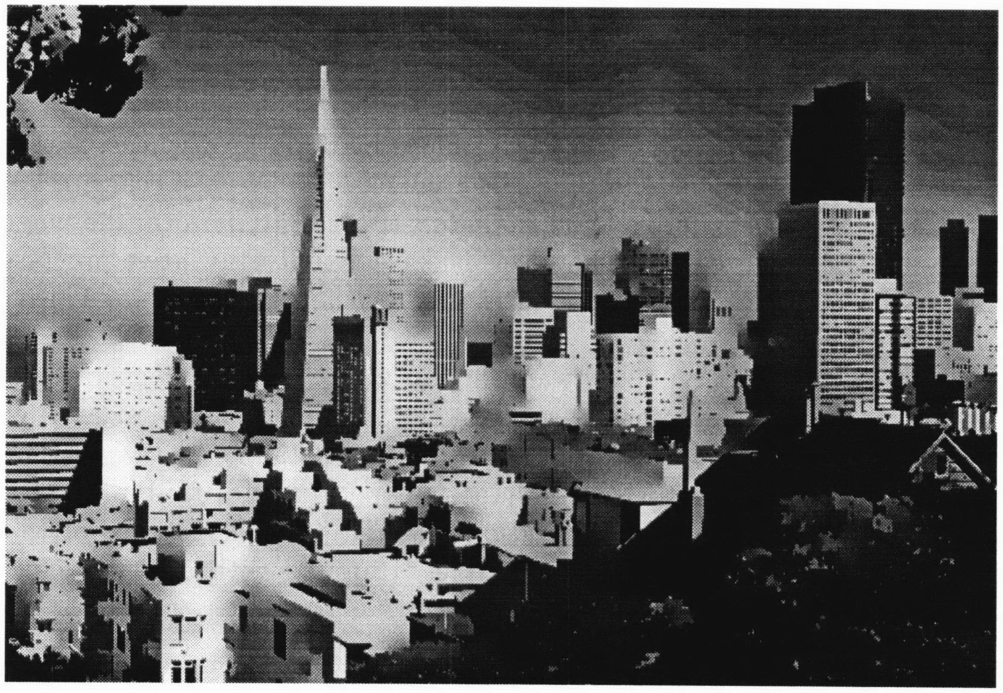


(b)

Figure 5.8: Original and processed images of San Francisco skyline. (a) Unprocessed image. (b) Low-pass filtered with 200 iterations of kernel (5.4). *Page 89:* (c) Result of 200 smooth and segment iterations, with threshold of 24/256. (d) Same, with threshold 32/256.



(c)



(d)

Table 5.4
Simulated Execution Times for Smoothing and Segmentation

Figure	Threshold (/256)	Iterations	Instructions per Iteration	Frames per Second 5 MHz	Frames per Second 10 MHz	Simulator Time (min:sec)
5.8(b)	256	200	439	56.9	114	52:57
5.8(c)	24	200	475	52.6	105	59:50
5.8(d)	32	200	463	54.0	108	57:52

neighbors, and a flag is set wherever the difference is greater than a given threshold. The next smooth step will not cross a boundary where a flag is set. For example, if a pixel's east segment flag is set, then the smooth step will use the modified kernel,

$$\frac{1}{8}\left(\begin{array}{|c|c|c|} \cline{2-2} \multicolumn{1}{c|}{} & 1 & \multicolumn{1}{c}{} \\ \hline 1 & 5 & 0 \\ \hline \multicolumn{1}{c|}{} & 1 & \multicolumn{1}{c}{} \\ \cline{2-2} \end{array}\right). \tag{5.5}$$

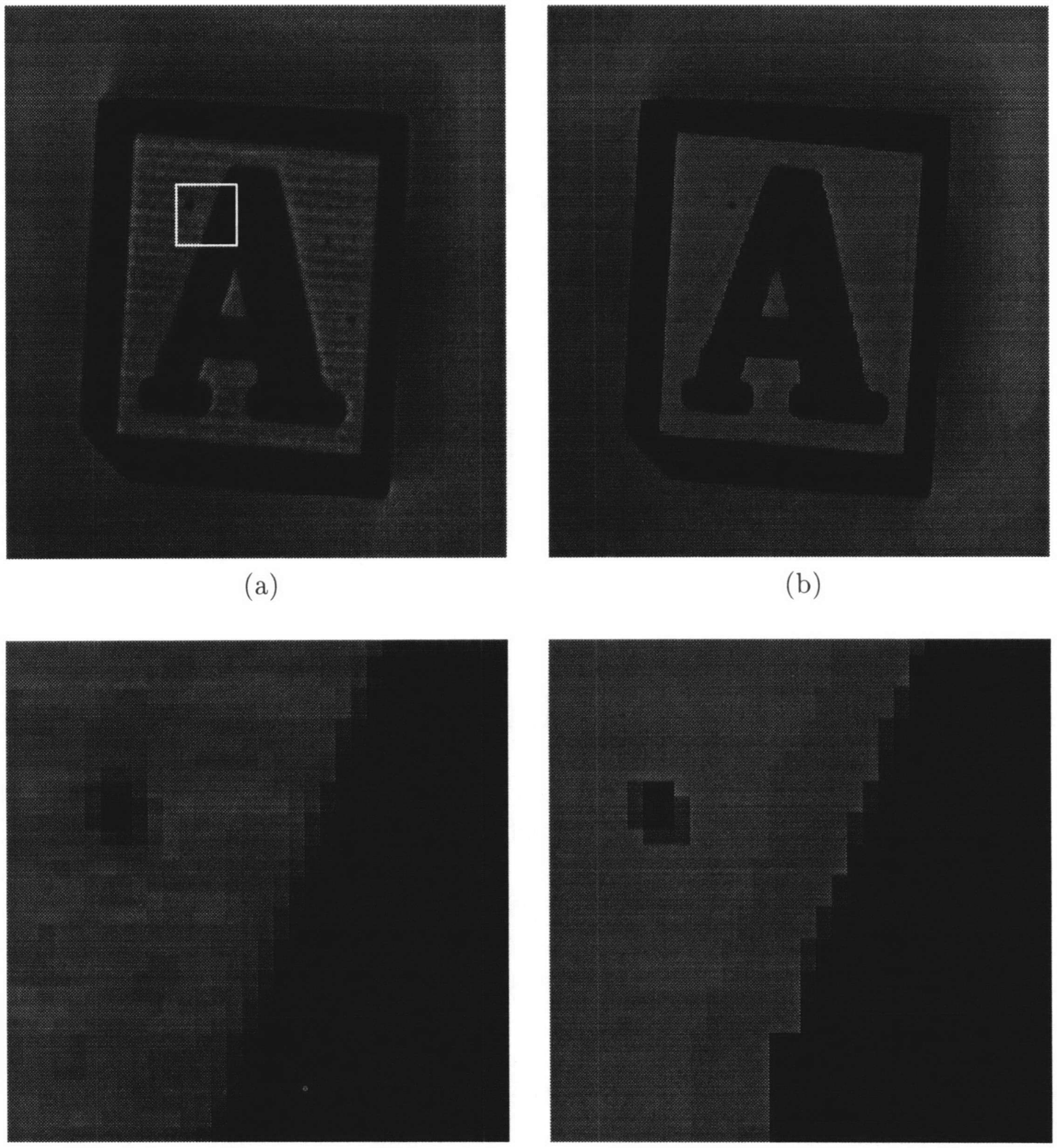
Figures 5.8(c) and (d) show the result of 200 iterations with threshold values of 24/256 and 32/256, respectively. The lower threshold permits about the right amount of smoothing, flattening the sides of buildings while preserving their edges. At the higher threshold, buildings begin to smear into one another. In some cases, it may be necessary to vary the threshold during processing to produce optimal results [107, §5.4]. The associative processor readily supports this flexibility.

Simulated execution times are presented in Table 5.4. The number of cycles required per iteration is somewhat dependent on the threshold used for segmentation. Case (b) is the easiest because no comparison need be performed when the threshold is at its maximum. Case (d) is slightly easier than case (c) because the threshold is a power of 2, and therefore requires fewer match instruction to perform the magnitude comparison [66]. In all three cases, however, it is clear that associative processor can perform real-time image processing at rates greater than the standard video frame rate of thirty frames per second.

Unfortunately, the prototype hardware can not work with images as large as those of Fig. 5.8. With four chips in the system, the maximum image size is 32×32 pixels. Figure 5.9(c) shows a suitable test image, a detail of the larger image in Fig. 5.9(a). The associative processor hardware performed 100 iterations with a segment threshold of 16/128. Each iteration required 439 instructions, of which 36 were used for an optimized refresh subroutine. Running with a 205-ns clock, the operation was completed in 9.3 ms.

5.7 Summary

This chapter has discussed the design, fabrication, and testing of the associative processor chip. Table 5.5 presents a summary of the chip's specifications. Although originally designed for the CCD-CMOS process at the M.I.T. Microsystems Technology Laboratories, the chip was eventually fabricated through the MOSIS service



(a) (b) (c) (d)

Figure 5.9: Original and processed images of toy block. (a) Unprocessed image, with box indicating area of detail. (b) Simulator output after 100 iterations with threshold of 16/128. (c) Detail of unprocessed image. (d) Result of processing (c) on associative processor hardware, using the same procedure as in (b).

Table 5.5
Associative Processor Chip Specifications

Associative memory	16 k	trits
Processing elements	256	
Transistors	134 k	
Minimum drawn channel length	1.5	μm
Package (PGA)	144	pins
Area		
Die	7.9×9.2	mm^2
Array	5.9×7.2	mm^2
Cell	32×33.5	μm^2
Minimum clock period		
Design	100	ns
Logic[1]	< 104	ns
Memory	175	ns
Power[2]		
Idle	48	mW
Typical	220	mW
Maximum	690	mW

1 Works to limit of test setup.
2 With 175-ns clock and $V_{DD} = 6\,\text{V}$.

using Orbit's double-poly CMOS process. An automated setup was used for testing, and most instruments were operated under computer control.

The first phase of testing demonstrated the full functionality of all chip subsystems, including the memory, word logic, network, and response resolver. Later characterization experiments indicated that the chip did not meet all design targets, but these results were shown to be consistent with the measured differences between the MTL and Orbit processes. The weak Orbit devices slowed the clock period to 175 ns, and a 6-V supply was required to compensate for the high second-poly thresholds. Experiments with test circuits and with the full memory array indicated that the Orbit parts pumped charge more efficiently than did the MTL circuits. This non-ideal charge transfer mechanism proved to be the dominant limit on storage time.

Finally, an application program to smooth and segment images was presented in Section 5.6. The program was first tested on a software simulator and then run on the actual hardware, thereby demonstrating the feasibility of real-time image processing with the associative processor.

Chapter 6

Conclusion

This thesis has described the design and characterization of an integrated associative processor intended for use in image processing and machine vision applications. Although the performance of the chip as fabricated by Orbit was somewhat disappointing, substantially all of the other project goals were met. This concluding chapter summarizes major accomplishments and presents some ideas for future work.

6.1 Summary

The associative processor chip uses a new dynamic associative processor cell supporting fully-parallel match and write operations. The cell is similar to a previously reported content addressable memory cell [88], but it adds dual-gate write transistors and a second write enable signal. The write-trit line runs parallel to the bit lines and is used to support masked writes. This allows some trits in a word to be preserved while others are modified.

The associative processor's masked write capability is one of two features that distinguish it from an associative memory. The other characteristic is a feedback path allowing match results to enable or disable the write-word driver. It was shown in Section 2.1 that useful work can be performed even when this feedback is implemented with a simple wire, but in practice it is usually desirable to provide more capable word logic. In this design, the combination of a single-bit activity register and a two-input function generator was found to improve arithmetic performance significantly but to require only a modest area penalty. Like the other processing element components, the word logic is designed in the device-intensive memory design style making extensive use of dynamic logic in dense pitch-matched circuits.

The processing elements communicate over an Asynchronous Reconfigurable Mesh network, which provides a mechanism for simultaneous broadcasting over multiple connected regions. The network implementation uses specialized memory cells to save area and preserve the simplicity of the word logic. Only one driver circuit is required per PE, and all but one of the network cells are implemented with only n-channel transistors. The network is readily extendible across chip boundaries, so that large arrays can be constructed with multiple chips.

A new response resolution circuit is used to count and prioritize selected processing elements. An area-efficient pass transistor chain is used within the confines of the array, but results from multiple arrays are combined in a tree structure. Each processing element is equipped with a specialized memory cell allowing it to examine the local state of the chain. The PE can then determine whether the number of higher-priority responders is even, odd, or zero, and it can use this information to change its own state in an iterative counting procedure.

The finished chip integrates 256 processing elements with a total of 16 k trits of associative memory. All chip subsystems are fully functional, and results from characterization experiments are consistent with the measured characteristics of the Orbit process. The test and demonstration system uses four chips to construct a 32×32 array of processing elements. An image smoothing and segmentation algorithm was presented as a sample pixel-parallel application, and the associative processing hardware was shown to perform the operation in under 10 ms, less than a standard video frame period. The demonstration proves that real-time performance is attainable.

6.2 Future Work

The first priority in pursuing further work with this associative processor chip should be to find a suitable CCD-CMOS process for its fabrication. The front end of the MTL process has several desirable properties, including matched first- and second-poly threshold voltages, good charge pumping performance (indicating low interface trap densities), and thin gate dielectrics. Once the current metallization and leakage problems are resolved, the MTL facility should be able to produce associative processor chips with significantly improved performance.

If the chip were to be redesigned for a more aggressive fabrication technology, then substantial density gains could be realized through the use of local interconnect [108]. The present cell design has three nodes which contact both active area and polysilicon gates. The metal used to make these connections obstructs the paths of the bit lines and widens the cell pitch. If truly ample resources were available for process development, then one could consider a stacked capacitor design similar to the Yamagata CAM cell (Section 3.1.2).

Hindsight and accumulated programming experience suggest that the chip would be significantly improved by the addition of an AndMatch instruction:

- *AndMatch*(p, f). The function $f(\mathrm{SA}, \mathrm{AR})$ is computed and the activity register is modified as in the Match instruction (Section 2.1). The sense amplifier value is preserved if the pattern p matches the associative memory word; it is reset otherwise.

That is, the AndMatch succeeds if and only if the associative memory word matches *and* the previous Match or AndMatch instruction succeeded. Of course, the same result can be computed in the function generator, but this operation modifies the activity register. With the present design, it is often necessary to write the register value to a scratch trit and then restore it after the match. The AndMatch instruction would be extremely useful in combining match results across the two half-words. Moreover, its implementation would not require any change to the associative processor array. One need only modify the control logic to prevent the assertion of $SASET$ during the precharge phase (Section 3.3.2).

The prototype associative processor chip includes several features that might be deemed unnecessary in a revised design. For example, to reduce the probability that a single design error would render the entire chip untestable, the responder and shift register signals for each of the four subarrays were made accessible from package pins. Another design concern was that subthreshold current through the write transistors might reduce storage time. The substrate bias (V_{BB}) was therefore separated from the negative supply (V_{SS}), in order to provide some means to increase the n-channel threshold voltages. Fortunately, neither of these fears were realized. A revised design could eliminate as many as 18 test and supply pins. As in the case of the AndMatch instruction, it would not be necessary to modify the core circuits of the array to implement these changes.

Working at the system level, the M.I.T. Associative Processing Project has begun the design of a data path to support sustained real-time image I/O. Once a complete prototype system has been assembled and demonstrated, the next natural step will be to improve physical construction and create a more polished final product. With presently available packaging technology, including multichip modules, the entire associative processing system should fit inside, or perhaps beside, a desktop workstation. Such a system would achieve massive parallelism on a personal scale and expand the availability of associative processing technology.

References

[1] Raymond Pinkham, Donald J. Redwine, Fred A. Valente, Troy H. Herndon, and Daniel F. Anderson. A high speed dual port memory with simultaneous serial and random mode access for video applications. *IEEE Journal of Solid-State Circuits*, SC-19(6):999–1007, December 1984.

[2] R. H. Fuller and R. M. Bird. An associative parallel processor with applications to picture processing. In *AFIPS Conference Proceedings: Fall Joint Computer Conference*, volume 27, pages 105–116, Washington, D.C., 1965. Spartan Books.

[3] M. A. Wesley, S.-K. Chang, and J. H. Mommens. A design for an auxiliary associative parallel processor. In *AFIPS Conference Proceedings: Fall Joint Computer Conference*, volume 41, pages 461–72, Montvale, New Jersey, 1972. AFIPS Press.

[4] Carver Mead. *Analog VLSI and Neural Systems.* Addison-Wesley, Reading, Massachusetts, 1989.

[5] John L. Wyatt, Jr., Craig Keast, Mark Seidel, David Standley, Berthold Horn, Tom Knight, Charles Sodini, Hae-Seung Lee, and Tomaso Poggio. Analog VLSI systems for image acquisition and fast early vision processing. *International Journal of Computer Vision*, 8(3):217–230, 1992.

[6] Mikko Hakkarainen, James J. Little, Hae-Seung Lee, and John L. Wyatt, Jr. Interaction of algorithm and implementation for analog VLSI stereo vision. In *Proceedings of SPIE Vol. 1473 Visual Information Processing: From Neurons to Chips*, pages 173–184, Bellingham, Washington, April 1991. Society of Photo-Optical Instrumentation Engineers.

[7] Paul C. Yu, Steven J. Decker, Hae-Seung Lee, Charles G. Sodini, and John L. Wyatt, Jr. CMOS resistive fuses for image smoothing and segmentation. *IEEE Journal of Solid-State Circuits*, 27(4):545–553, April 1992.

[8] Craig L. Keast and Charles G. Sodini. A CCD/CMOS-based imager with integrated focal plane signal processing. *IEEE Journal of Solid-State Circuits*, 28(4):431–437, April 1993.

[9] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *National Academy of Sciences*, volume 79, pages 2554–2558, April 1982.

[10] Teuvo Kohonen. *Self-Organization and Associative Memory.* Springer Series in Information Sciences. Springer-Verlag, New York, second edition, 1988.

[11] Yaser S. Abu-Mostafa and Demetri Psaltis. Optical neural computers. *Scientific American*, 256(3):88–95, March 1987.

[12] Caxton C. Foster. *Content Addressable Parallel Processors.* Computer Science Series. Van Nostrand Reinhold Co., New York, 1976.

[13] Teuvo Kohonen. *Content-Addressable Memories.* Springer Series in Information Sciences. Springer-Verlag, New York, second edition, 1987.

[14] Behrooz Parhami. Associative memories and processors: An overview and selected bibliography. *Proceedings of the IEEE*, 61(6):722–730, June 1973.

[15] Jerry L. Potter. *Associative Computing: A Programming Paradigm for Massively Parallel Computers.* Plenum Press, New York, 1992.

[16] R. Mike Lea and Ian P. Jalowiecki. Associative massively parallel computers. *Proceedings of the IEEE*, 79(4):469–479, April 1991.

[17] Ian N. Robinson. Pattern-addressable memory. *IEEE Micro*, 12(3):20–30, June 1992.

[18] Kenneth E. Batcher. Bit-serial parallel processing systems. *IEEE Transactions on Computers*, C-31(5):377–384, May 1982.

[19] Charles C. Weems Jr. The content addressable array parallel processor: Architectural evaluation and enhancement. In *IEEE International Conference of Computer Design*, pages 500–503, 1985.

[20] Robert Heaton, Donald Blevins, and Edward Davis. A bit-serial VLSI array processing chip for image processing. *IEEE Journal of Solid-State Circuits*, 25(2):364–368, April 1990.

[21] Daniel Dobberpuhl, Richard Witek, Randy Allmon, Robert Anglin, Sharon Britton, Linda Chao, Robert Conrad, Daniel Dever, Bruce Gieseke, Gregory Hoeppner, John Kowaleski, Kathryn Kuchler, Maureen Ladd, Michael Leary, Liam Madden, Edward McLellan, Derrick Meyer, James Montanaro, Donald Priore, Vidya Rajagopalan, Sridhar Samudrala, and Sribalan Santhanam. A 200MHz 64b dual-issue CMOS microprocessor. In *IEEE International Solid-State Circuits Conference*, pages 106–107,256, 1992.

[22] Peter A. Ruetz. The architecture and design of a 20-MHz real-time DSP chip set. *IEEE Journal of Solid-State Circuits*, 24(2):338–348, April 1989.

[23] LSI Logic Corporation. L64240 multi-bit filter. product description, 1991.

[24] LSI Logic Corporation. L64243 3×3 multi-bit filter. product description, 1991.

[25] Masakatsu Maruyama, Hiroyuki Nakahira, Toshiyuki Araki, Shirou Sakiyama, Yoshitaka Kitao, Kunitoshi Aono, and Haruyasu Yamada. An image signal multiprocessor on a single chip. *IEEE Journal of Solid-State Circuits*, 25(6):1476–1483, December 1990.

[26] Daniel W. Dobberpuhl, Richard T. Witek, Randy Allmon, Robert Anglin, Sharon Britton, Linda Chao, Robert A. Conrad, Daniel E. Dever, Bruce Gieseke, Soha M. N. Hassoun, Gregory W. Hoeppner, Kathryn Kuchler, Maureen Ladd, Burton M. Leary, Liam Madden, Edward J. McLellan, Derrick R. Meyer, James Montanaro, Donald A. Priore, Vidya Rajagopalan, Sridhar Samudrala, and Sribalan Santhanam. A 200MHz 64b dual-issue CMOS microprocessor. *IEEE Journal of Solid-State Circuits*, 27(11):1555–1567, November 1992.

[27] G. Estrin and R. Fuller. Algorithms for content-addressable memories. In *Proceedings of the Pacific Computing Conference*, pages 118–130, 1963.

[28] Richard G. Ewing and Paul M. Davies. An associative processor. In *AFIPS Conference Proceedings: Fall Joint Computer Conference*, pages 147–158, 1964.

[29] Anthony P. Reeves and John M. Bruner. Efficient function implementation for bit-serial parallel processors. *IEEE Transactions on Computers*, C-29(9):841–844, September 1980.

[30] Charles E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[31] Robert Cypher and Jorge L. C. Sanz. SIMD architectures and algorithms for image processing and computer vision. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):2158–2174, December 1989.

[32] M. J. B. Duff and T. J. Fountain. Enhancing the two-dimensional mesh. In *10th International Conference on Pattern Recognition*, volume II, pages 654–659, 1990.

[33] Quentin F. Stout. Mesh-connected computers with broadcasting. *IEEE Transactions on Computers*, C-32(11):826–830, September 1983.

[34] V. K. Prasanna Kumar and C. S. Raghavendra. Array processor with multiple broadcasting. In *Proceedings 12th Annual Symposium on Computer Architecture*, pages 2–10. ACM-SIGARCH, June 1985.

[35] Yen-Cheng Chen, Wen-Tsuen Chen, Gen-Huey Chen, and Jang-Ping Sheu. Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting. *IEEE Transactions on Parallel and Distributed Systems*, 1(2):241–246, April 1990.

[36] Steven L. Tanimoto. A pyramidal approach to parallel processing. In *Proceedings 10th Annual Symposium on Computer Architecture*, pages 372–378. ACM-SIGARCH, 1983.

[37] W. D. Hillis. *The Connection Machine.* The MIT Press, Cambridge, Massachusetts, 1985.

[38] Bruce H. McCormick. The Illinois pattern recognition computer— ILLIAC III. *IEEE Transactions on Electronic Computers*, EC-12(6):791–813, December 1963.

[39] M. J. B. Duff. Review of the CLIP image processing system. In *Proceedings of the AFIPS National Computer Conference*, pages 1055–1060, 1978.

[40] Anthony P. Reeves. A systematically designed binary array processor. *IEEE Transactions on Computers*, C-29(4):278–287, April 1980.

[41] D. J. Hunt. The ICL DAP and its application to image processing. In M. J. B. Duff and S. Levialdi, editors, *Languages and Architectures for Image Processing*, chapter 22, pages 275–282. Academic Press, New York, 1981.

[42] R. M. Lea. ASP: a cost-effective parallel microcomputer. *IEEE Micro*, pages 10–29, October 1988.

[43] Hungwen Li and Massimo Maresca. Polymorphic-torus network. *IEEE Transactions on Computers*, 38(9):1345–51, September 1989.

[44] Massimo Maresca and Hungwen Li. Connection autonomy in SIMD computers: A VLSI implementation. *Journal of Parallel and Distributed Computing*, 7(2):302–320, October 1989.

[45] Charles C. Weems. Some example algorithms for the CAAPP and ICAP levels of the image understanding architecture. In Lana P. Kartashev and Steven I. Kartashev, editors, *International Conference on Supercomputing: Proceedings*, pages 42–52, St. Petersburg, Florida, 1988. International Supercomputing Institute. Volume III: Supercomputer Design: Hardware & Software.

[46] David B. Shu and J. Greg Nash. Minimum spanning tree algorithm on an image undersanding architecture. In David P. Casasent and Andrew G. Tescher, editors, *Proceedings of SPIE Vol. 939 Hybrid Image and Signal Processing*, pages 212–228, Orlando, Florida, 1988. SPIE — The International Society for Optical Engineering.

[47] David B. Shu, Greg Nash, and Charles Weems. Image understanding architecture and applications. In Jorge L. C. Sanz, editor, *Advances in Machine Vision*, chapter 9, pages 295–355. Springer-Verlag, New York, 1988.

[48] Martin C. Herbordt, Charles C. Weems, and Michael J. Scudder. Nonuniform region processing on SIMD arrays using the coterie network. *Machine Vision and Applications*, 5(2):105–125, Spring 1992.

[49] D. B. Shu, J. G. Nash, M. M. Eshaghian, and K. Kim. Implementation and application of a gated-connection network in image understanding. In Hungwen Li and Quentin F. Stout, editors, *Reconfigurable Massively Parallel Computers*, chapter 3, pages 64–87. Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[50] Martin C. Herbordt, Charles C. Weems, and James C. Corbett. Message-passing algorithms for a SIMD torus with coteries. In *SPAA '90: 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 11–20, New York, July 1990. Association for Computing Machinery.

[51] Martin C. Herbordt. private communication, March 1994.

[52] Martin C. Herbordt and Charles C. Weems. Multiassociative processing. unpublished proceedings, 2nd Associative Processing and Applications Workshop, July 1993.

[53] D. W. Blevins, E. W. Davis, R. A. Heaton, and J. H. Reif. BLITZEN: a highly integrated massively parallel machine. In Ronnie Mills, editor, *IEEE Second Symposium on the Frontiers of Massively Parallel Computation*, pages 399–406, October 1988.

[54] Caxton C. Foster and Fred D. Stockton. Counting responders in an associative memory. *IEEE Transactions on Computers*, C-20(12):1580–1583, December 1971.

[55] George A. Anderson. Multiple match resolvers: A new design method. *IEEE Transactions on Computers*, C-23:1317–1320, December 1974.

[56] Jon Patrick Wade. *An Integrated Content Addressable Memory System.* PhD thesis, Massachusetts Institute of Technology, May 1988.

[57] Kenneth E. Batcher. The multidimensional access memory in STARAN. *IEEE Transactions on Computers*, C-26(2):174–177, February 1977.

[58] Kenneth E. Batcher. Staging memory. In Jerry L. Potter, editor, *The Massively Parallel Processor*, pages 191–204. MIT Press, Cambridge, Massachusetts, 1985.

[59] Junji Ogawa, Yusuke Masuda, Kazuya Kobayashi, and Toshiya Nishi. A 4Mb block/vector addressable three-dimensional bit map memory. In *IEEE International Solid-State Circuits Conference*, pages 228–229,302, 1990.

[60] Steven L. Tanimoto. Memory systems for highly parallel computers. *Proceedings of the IEEE*, 79(4):403–415, April 1991.

[61] Daphne Yong-Hsu Shih. M.Eng. thesis, Massachusetts Institute of Technology, in preparation.

[62] Lawrence T. Hsu. A controller architecture for associative processors. Master's thesis, Massachusetts Institute of Technology, August 1993.

[63] Jeffrey C. Gealow, Frederick P. Herrmann, Lawrence T. Hsu, and Charles G. Sodini. System design for pixel-parallel image processing. In preparation.

[64] Wayne M. Loucks, Martin Snelgrove, and Safwat G. Zaky. A vector processor based on one-bit microprocessors. *IEEE Micro*, 2(1):53–62, February 1982.

[65] Kenneth E. Batcher. Array control unit. In Jerry L. Potter, editor, *The Massively Parallel Processor*, pages 170–190. MIT Press, Cambridge, Massachusetts, 1985.

[66] Frederick Paul Herrmann. A system architecture for associative memories and processors. Master's thesis, Massachusetts Institute of Technology, May 1989.

[67] Claus M. Habiger and R. Mike Lea. Hybrid-WSI: A massively parallel computing technology? *Computer*, 26(4):50–61, April 1993.

[68] A. E. Slade and H. O. McMahon. A cryotron catalog memory system. In *Proceedings Eastern Joint Computer Conference*, pages 115–119, December 1956.

[69] D. Crandall. A content addressable memory. Master's thesis, University of California, Berkeley, April 1981.

[70] S. Jones. Design, selection and implementation of a content-addressable memory for a VLSI CMOS chip architecture. *IEE Proceedings*, 135:165–172, May 1988.

[71] John V. Oldfield, Charles D. Stormon, and Mark Brule. The application of VLSI content-addressable memories to the acceleration of logic programming systems. In *COMPEURO 1987: VLSI and Computers*, pages 27–30, 1987.

[72] Charles D. Stormon, Nikos B. Troullinos, Edward M. Saleh, Abhijeet V. Chavan, Mark R. Brule, and John V. Oldfield. A general purpose CMOS associative processor IC and system. *IEEE Micro*, 12(6):68–78, December 1992.

[73] R. Igarashi, T. Kurosawa, and T. Yaita. A 150-nanosecond associative memory using integrated MOS transistors. In *IEEE International Solid-State Circuits Conference*, pages 104–105, 1966.

[74] Hiroshi Kadota, Jiro Miyake, Yoshito Nishimichi, Hitoshi Kudoh, and Keiichi Kagowa. An 8-kbit content-addressable and reentrant memory. *IEEE Journal of Solid-State Circuits*, SC-20(5):951–957, October 1985.

[75] James T. Koo. Integrated-circuit content-addressable memories. *IEEE Journal of Solid-State Circuits*, SC-5(5):208–215, October 1970.

[76] William N. Carr and Jack P. Mize. *MOS/LSI Design and Application.* McGraw-Hill, New York, 1972.

[77] Terry I. Chappell, Barbara A. Chappell, Stanley E. Schuster, James W. Allan, Stephen P. Klepner, Rajiv V. Joshi, and Robert L. Franch. A 2ns cycle, 4ns access 512kb CMOS ECL SRAM. In *IEEE International Solid-State Circuits Conference*, pages 50–51,288, 1991.

[78] Kazutaka Nogami, Takayasu Sakurai, Kazuhiro Sawada, Kenji Sakaue, Yuichi Miyuzawa, Shigeru Tanaka, Yuichi Hiruta, Katsuto Katoh, Toshinari Takayanagi, Tsukasa Shirotori, Yukiko Itoh, Masanori Uchida, and Tetsuya Iizuka. A 9-ns HIT-delay 32-kbyte cache macro for high-speed RISC. *IEEE Journal of Solid-State Circuits*, 25(1):100–108, February 1990.

[79] Yong-Chul Shin, Ramalingam Sridhar, Victor Demjanenko, Paul W. Palumbo, and Sargur N. Srihar. A special-purpose content-addressable memory chip for real-time image processing. *IEEE Journal of Solid-State Circuits*, 27(5):737–744, May 1992.

[80] Takeshi Ogura, Junzo Yamada, Shin-Ichiro Yamada, and Masa-Aki Tan-No. A 20-kbit associative memory LSI for artificial intelligence machines. *IEEE Journal of Solid-State Circuits*, 24(4):1014–1020, August 1989.

[81] B. Zehner. Associative memory with improved memory cell and method for operating same. U.S. Patent No. 4,538,243, August 1985.

[82] Reuben E. Joynson, Joseph L. Mundy, James F. Burgess, and Constantine Neugebauer. Eliminating threshold losses in MOS circuits by bootstrapping using varactor coupling. *IEEE Journal of Solid-State Circuits*, SC-7(3):217–224, June 1972.

[83] Leilani R. Tamura, Tsen-Shau Yang, Drew E. Wingard, Mark A. Horowitz, and Bruce A. Wooley. A 4-ns BiCMOS translation-lookaside buffer. *IEEE Journal of Solid-State Circuits*, 25(5):1093–1101, October 1990.

[84] Joseph L. Mundy. High density four-transistor MOS content addressed memory. U.S. Patent No. 3,701,980, October 1972.

[85] Joseph L. Mundy, James F. Burgess, Reuben E. Joynson, and Constantine Neugebauer. Low-cost associative memory. *IEEE Journal of Solid-State Circuits*, SC-7(5):364–369, October 1972.

[86] Richard S. Muller and Theodore I. Kamins. *Device Electronics for Integrated Circuits.* John Wiley & Sons, New York, second edition, 1986.

[87] Tadato Yamagata, Masaaki Mihara, Takeshi Hamamoto, Yasumitsu Murai, Toshifumi Kobayashi, Michihiro Yamada, and Hideyuki Ozaki. A 288-kb parallel content-addressable memory using a stacked-capacitor cell structure. *IEEE Journal of Solid-State Circuits*, 27(12):1927–1933, December 1992.

[88] Jon P. Wade and Charles G. Sodini. Dynamic cross-coupled bit-line content addressable memory cell. *IEEE Journal of Solid-State Circuits*, SC-21(1):119–121, February 1987.

[89] S. E. Schuster. Dynamic content addressable memory with refresh feature. *IBM Technical Disclosure Bulletin*, 26:5364–5366, March 1984.

[90] Anthony J. McAuley and Charles J. Cotton. A self-testing reconfigurable CAM. *IEEE Journal of Solid-State Circuits*, 26(3):257–261, March 1991.

[91] Takeshi Ogura, Shin-Ichiro Yamada, and Tadanobu Nikaido. A 4-kbit associative memory LSI. *IEEE Journal of Solid-State Circuits*, SC-20(6):1277–1282, December 1985.

[92] Takeshi Ogura, Shin-Ichiro Yamada, and Junzo Yamada. A 20Kb CMOS assosiative memory LSI for artificial intelligence machines. In *Proceedings of the IEEE International Conference on Computer Design*, pages 574–577, October 1986.

[93] Carlo H. Séquin and Michael F. Tompsett. *Charge Transfer Devices.* Academic Press, New York, 1975.

[94] Frederick P. Herrmann, Craig L. Keast, Keisuke Ishio, Jon P. Wade, and Charles G. Sodini. A dynamic three-state memory cell for high-density associative processors. *IEEE Journal of Solid-State Circuits*, 26(4):537–541, April 1991.

[95] C. L. Keast and C. G. Sodini. A CCD/CMOS process for integrated image acquisition and early vision signal processing. In *Proceedings of SPIE Vol. 1242 Charge-Coupled Devices and Solid State Optical Sensors*, pages 152–161, Bellingham, Washington, 1990. Society of Photo-Optical Instrumentation Engineers.

[96] Jon P. Wade and Charles G. Sodini. A ternary content addressable search engine. *IEEE Journal of Solid-State Circuits*, 24(4):1003–1013, August 1989.

[97] Jeffrey Carl Gealow. *An Integrated Computing Structure for Pixel-Parallel Image Processing.* PhD thesis, Massachusetts Institute of Technology, in preparation.

[98] Carver A. Mead and Lynn A. Conway. *Introduction to VLSI Systems*, page 152. Addison-Wesley Publishing Company, Reading, Massachusetts, 1980.

[99] Chris Rowen, Mark Johnson, and Paul Ries. The MIPS R3010 floating-point coprocessor. *IEEE Micro*, 8(3):53–62, June 1988.

[100] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*, page 49. Addison-Wesley, Reading, Massachusetts, 1985.

[101] Walter S. Scott, Robert N. Mayo, Gordon Hamachi, and John K. Ousterhout. 1986 VLSI tools: Still more works by the original artists. Technical Report UCB/CSD 86/272, Computer Science Division, EECS Department, University of California at Berkeley, Berkeley, California, December 1985.

[102] Robert N. Mayo, Michael H. Arnold, Walter S. Scott, Don Stark, and Gordon T. Hamachi. 1990 DECWRL/Livermore Magic release. Technical Report 90/7, Digital Equipment Corporation Western Research Laboratory, Palo Alto, California, September 1990.

[103] Orbit Semiconductor, Inc., Sunnyvale, California. *Foresight User's Manual,* Revision 1.5, July 1992.

[104] Daphne Yong-Hsu Shih. A data path for an associative processor. 6.199 advanced undergraduate project report, Massachusetts Institute of Technology, 1994.

[105] Guido Groeseneken, Herman E. Maes, Nicolas Beltrán, and Roger F. De Keersmaecker. A reliable approach to charge-pumping measurements in MOS transistors. *IEEE Transactions on Electronic Devices*, ED-31(1):42–53, February 1984.

[106] Kathleen S. Krisch. *Mechanisms of Improved Reliability in Reoxidized Nitrided Oxide MOS Gate Dielectrics.* PhD thesis, Massachusetts Institute of Technology, February 1993.

[107] Craig L. Keast. *An Integrated Image Acquisition, Smoothing and Segmentation Focal Plane Processor.* PhD thesis, Massachusetts Institute of Technology, 1992.

[108] Charles G. Sodini, S. Simon Wong, and Ping-Keung Ko. A framework to evaluate technology and device design enhancements for MOS integrated circuits. *IEEE Journal of Solid-State Circuits*, 24(1):118–127, February 1989.

[109] Craig L. Keast. A CCD/CMOS process for integrated image acquisition and early vision signal processing. Master's thesis, Massachusetts Institute of Technology, September 1989.

[110] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, Reading, Massachusetts, 1985.

[111] Paul Horowitz and Winfield Hill. *The Art of Electronics.* Cambridge University Press, Cambridge, England, second edition, 1989.

Appendix A

Chip Data

A.1 Pins and Signals

Figure A.1 is a pin placement diagram for the associative processor chip. This appendix documents all of the chip's external signals.

Supplies

Vbb	Substrate bias. Six pins are used for this signal to reduce the total supply inductance; all are connected internally. Should be tied to Vss externally during normal operation.
Vss	Ground supply. Six pins.
Vdd	Positive (5–6 V) supply. Six pins.
Vsp	Negative supply for output pads. Should be tied to Vss externally. Eight pins.
Vdp	Positive supply for output pads. Should be tied to Vdd externally. Eight pins.

Clocks

clk0 clk1	Quadrature clock signals (Fig. A.2). Instructions begin on the rising edge of clk0.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	Vdp 144	tw 140	t27 136	t25 135	t23 133	ein1 129	Vbb 128	Vdd 125	ein2 123	sum2 122	t17 119	t11 116	t7 114	eout3 111	Vdp 108
B	rsp- 3	eout0 143	tc 139	t29 137	sum0 134	sum1 131	eout1 130	eout2 124	t19 120	t15 118	t9 115	t3 112	ein3 110	t1 107	rsp0 104
C	Vss 6	rspout 2	ein0 142	te 141	t31 138	Vsp 132	Vsp 127	Vss 126	t21 121	t13 117	t5 113	Vdp 109	f13 106	Vss 103	rsp1 100
D	Vbb 8	cin- 4	Vdp 1	nc									cout- 105	Vbb 101	rsp2 99
E	nin3 11	Vdd 7	fi 5										Vdd 102	rsp3 98	rr 97
F	sout2 14	sout3 10	sin3 9										ur 96	lr 95	ue 93
G	nin2 15	nout3 12	sin2 13										Vsp 91	re 94	ab- 92
H	nck0 17	nout2 16	Vsp 18										Vsp 90	i0 88	i1 89
J	nck1 20	sout1 22	Vsp 19										f02 85	t0 84	clk1 87
K	sin1 21	nin1 23	nout1 24										t6 81	t4 82	clk0 86
L	sin0 25	sout0 26	Vdd 30										t8 77	Vdd 79	t2 83
M	nin0 27	Vbb 29	fhd 33										Vdp 73	t10 76	Vbb 80
N	nout0 28	Vss 31	fld 34	Vdp 37	tn 41	arin0 45	Vsp 49	Vsp 54	Vss 55	t26 60	t22 66	t16 69	win3 70	t14 74	Vss 78
P	fo 32	fsh 35	win0 38	th 40	t30 43	arin1 46	t28 48	wout1 52	wout2 58	arr 59	arout2 62	arout3 65	t20 67	wout3 71	t12 75
Q	Vdp 36	wout0 39	ts 42	arout0 44	arout1 47	rspin 50	win1 51	Vbb 53	Vdd 56	win2 57	t24 61	arin2 63	arin3 64	t18 68	Vdp 72

Figure A.1: Pin placement diagram (top view). Pin D4 is not connected.

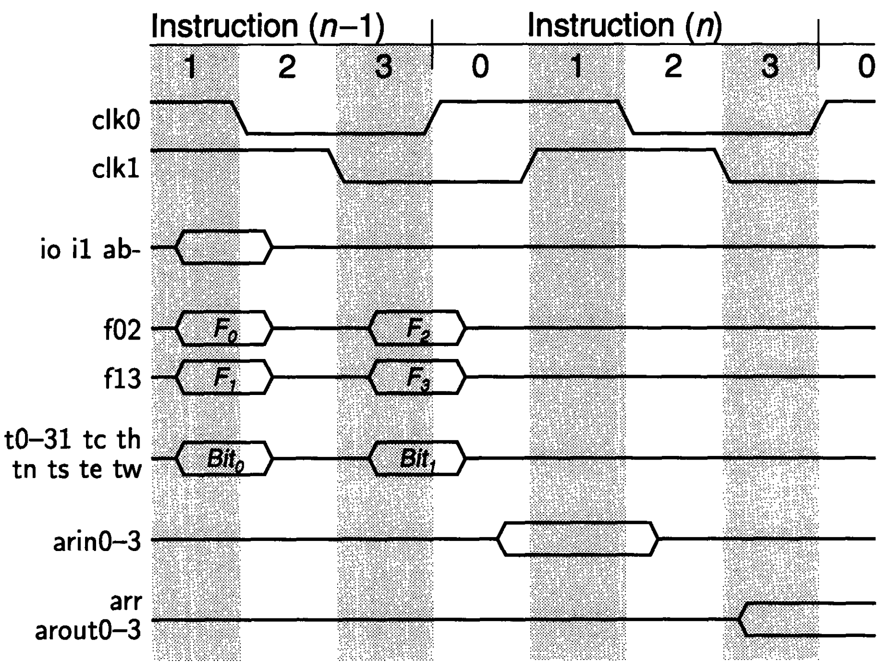


Figure A.2: Instruction timing diagram indicating when inputs are read and when outputs are produced for instruction *n*.

Table A.1
Operation Codes

i1	i0	ab-	Chip Instruction	System Instruction
0	0	0	Nop	Nop
0	0	1	Nop	Board command
0	1	0	Match B	Match B
0	1	1	Match A	Match A
1	0	0	Write B	Write B
1	0	1	Write A	Write A
1	1	0	Shift	Shift 0
1	1	1	Shift	Shift 1

Instruction

i0 i1 ab- Operation code, as described in Table A.1. Signal ab- is not decoded by the chip during Nop and Shift instructions. The demonstration system uses ab- to control arin, so that Shift 0 and Shift 1 become separate system instructions. Similarly, a chip Nop with ab- set is decoded by the system as a test board command. Sampled on the falling edge of clk0.

Table A.2
Trit Encoding

Cells	Trit	Match Bit_0	Match Bit_1	Write Bit_0	Write Bit_1
All trit cells & H cell pair	0	1	0	0	1
	1	0	1	1	0
	X	1	1	0	0
	–	0	0	1	1
NEWS cells	0	1	0	0	1
	1	0	0	1	1
	X	0	0	0	1
	–	1	0	1	0

f02 f13	Multiplexed function inputs, specifying one of the sixteen functions of Table 2.3. As shown in Fig. A.2, four bits F_{0-3} are obtained by sampling these two signals on both edges of clk0.
t0 – t31 tn ts te tw tc th	Multiplexed trit pattern inputs, encoded with two bits per trit. Bit_0 is sampled on the falling edge of clk0, Bit_1 on the rising edge. As shown in Table A.2, The NEWS cells' trit encoding is different from that of the home and trit cells.

Activity Register

arr	Activity register some/none responder. This is the OR of all PEs' AR bits. The output changes after the falling edge of clk1.
arin0–3	Inputs for the test shift registers running through each of the four subarrays. Must be set up before the rising edge of clk1 and held past the falling edge of clk0.
arout0–3	Outputs for the four test shift registers. Data becomes available after the falling edge of clk1.

Network

nck0 nck1	Quadrature clocks for network multiplexers. Tied to clk0 and clk1 during normal operation, but may be driven separately during testing.
nin0–3 sin0–3 ein0–3 win0–3	Multiplexed network inputs for the four compass directions. 4 bits per pin $\times$ 4 pins per edge $\times$ 4 edges = perimeter 64.

nout0–3 sout0–3 eout0–3 wout0–3	Multiplexed network outputs, to be tied to neighbors' network inputs.

Response Resolver

rspout	Some/none responder output, the OR of all PEs' HB cells.
rsp-	Open-drain responder output, the NOR of all PEs' HB cells.
rsp0–3	Subarray responder outputs. The OR of each subarray's PEs' HB cells.
rspin	Input for responder feedback. A high level indicates that responders are present in the system.
cin-	Responder chain input. A high level indicates that there are no responders in higher-priority chips.
cout-	Daisy-chain responder output, for systems not using a tree for multi-chip response resolution. May be tied to next-lowest priority chip's cin- input. Unused in demonstration system.

Responder Tree

Each chip has the necessary logic to implement one node of a multi-chip responder tree (Fig. 2.13). These five pins have no internal connection to any other chip signals.

ue	Up enable input from higher node in tree.
lr rr	Left and right responder inputs.
ur	Responder output to higher node. $\mathsf{ur} = \mathsf{lr} \vee \mathsf{rr}$.
re	Right enable output. $\mathsf{re} = \mathsf{ue} \wedge \overline{\mathsf{lr}}$.

Image I/O Shift Register

fhd fsh fld	Hold, shift, and load control signals. At most one of these should be asserted at a time, and fhd must be asserted after every fsh and fld pulse.
fi	I/O shift register input. Sampled on falling edge of fsh.
fo	I/O shift register output. Valid after rising edge of fhd.

A.2 Internal Timing

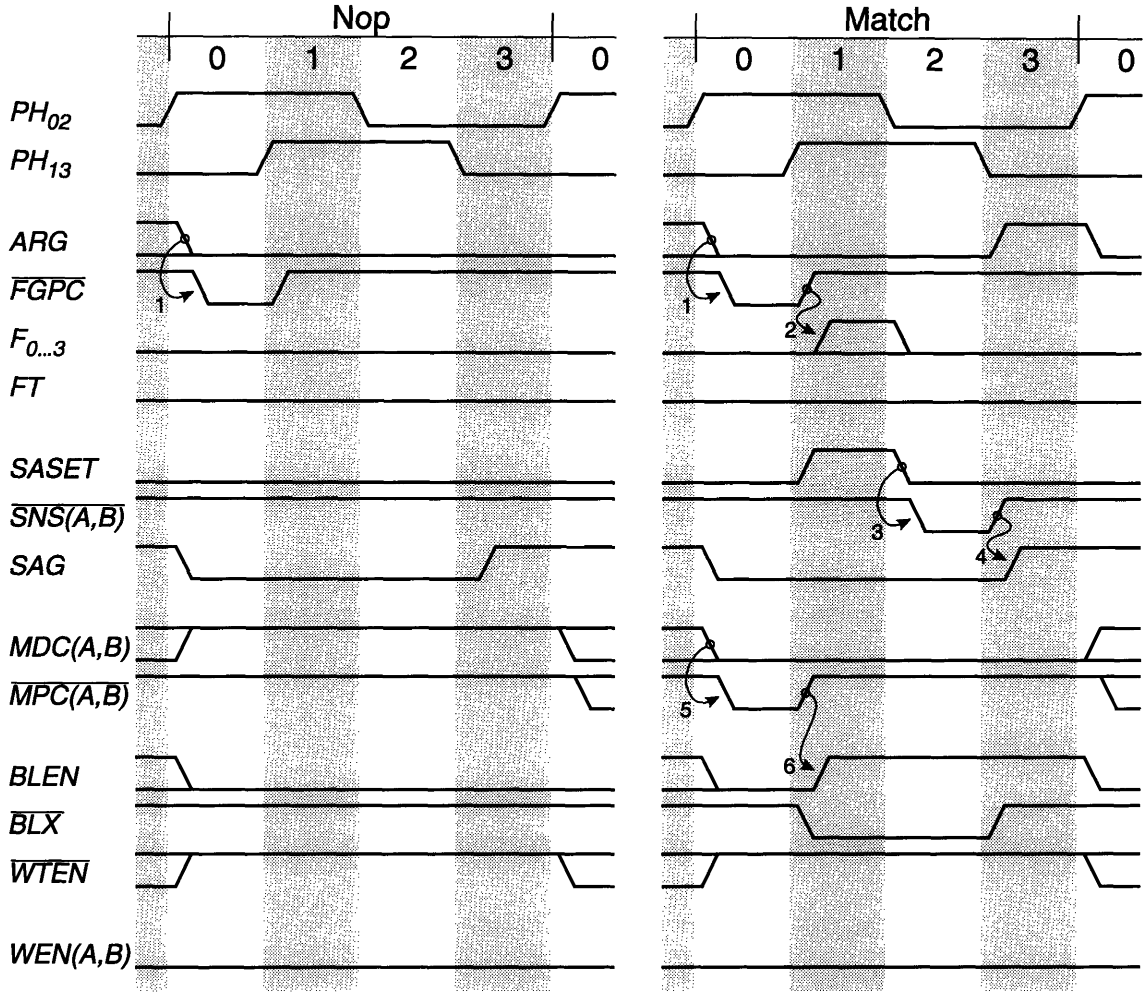


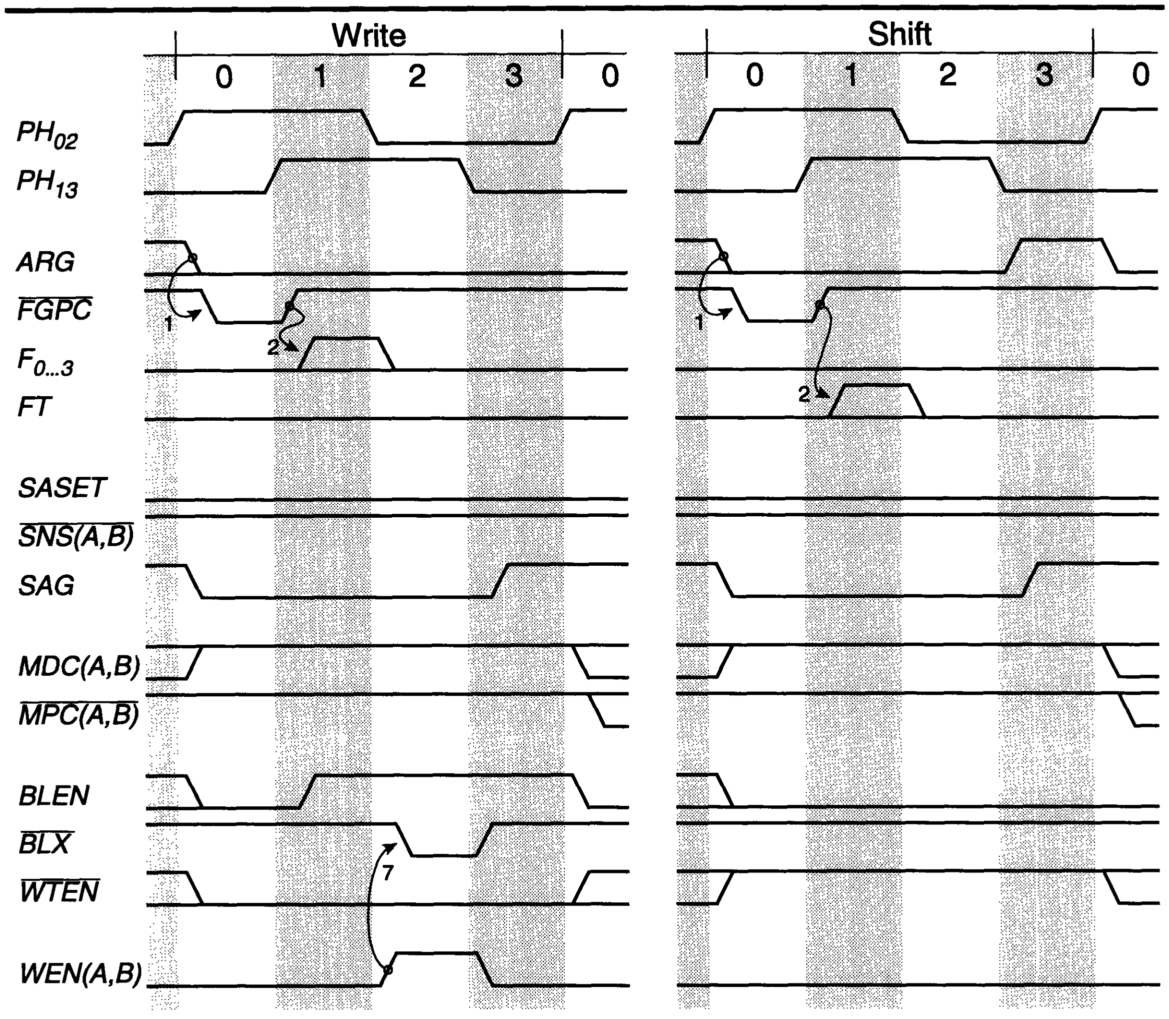
Figure A.3: Timing of internal control signals. *Above:* Nop and Match instructions. *Page 113:* Write and Shift instructions.

The timing diagram in Fig. A.3 shows how the most important array control signals behave during the four chip instructions. The control logic enforces certain signal dependencies, as indicated in the figure and described below.

1. $ARG \rightarrow \overline{FGPC}$. Wait until the AR input is disabled before precharging the function generator, in order to avoid reseting the AR erroneously.

2. $\overline{FGPC} \rightarrow F_{0...3}$, $\overline{FGPC} \rightarrow FT$. Turn the function generator precharge device off before enabling any of the NAND stacks, in order to prevent excess current flow.

3. $SASET \rightarrow \overline{SNS(A,B)}$. As above, make sure the sense amplifier precharge device is off before beginning the sense phase.



4. $\overline{SNS(A,B)} \rightarrow SAG$. Complete the sense phase before turning on the output pass transistors, to prevent possible charge sharing problems with function generator input nodes.

5. $MDC(A,B) \rightarrow \overline{MPC(A,B)}$. Turn off the discharge device before precharging the match line, to prevent excess current flow.

6. $\overline{MPC(A,B)} \rightarrow BLEN$. Make sure the precharge device is off before beginning the signal development phase.

7. $WEN(A,B) \rightarrow \overline{BLX}$. Reduce switching noise by preventing the simultaneous switching of the write-word drivers and the large bit line drivers.

Appendix B

MTL Process Data

This appendix provides a summary of MTL process parameters for use in hand calculations. In some cases, the tables give values for both layers of polysilicon. Transistors made with poly 0 gates are not optimized. All values are taken either from [109] or from extracted HSPICE models.

Table B.1
Device Parameters

Parameter		Poly	n	p	Units
Threshold voltage	V_T	0	1.00	−1.55	V
		1	0.63	−0.72	V
Oxide capacitance	C_{ox}	0	1.5	1.5	fF/μm^2
		1	1.4	1.4	fF/μm^2
μC_{ox} product	μC_{ox}		94.2	32.5	μA/V^2

Table B.2
Junction Capacitance

Junction		n	p	Units
Bottom	C_{j0}	0.34	0.21	fF/μm^2
	φ_0	464	585	mV
	m	0.693	0.497	
Sidewall	C_{j0}	0.36	0.37	fF/μm
	φ_0	281	389	mV
	m	0.334	0.273	

Table B.3
Interconnect Resistance

Material		Sheet ($\Omega/\square$)	Contact (Ω)
Diffusion	n	36	28
	p	68	55
Polysilicon	0	21	12
	1	17	9.9
Metal	1	0.05	
Metal	2	0.05	

Table B.4
Interconnect Capacitance

	Metal 2	Metal 1	Poly 1	Poly 0
Field	.016	.035	.086	.086
Diffusion			C_{ox}	C_{ox}
Poly 0	.020	.058	.82	
Poly 1				
Metal 1	.031			

(fF/μm^2)

Interconnect perimeter capacitance can be conveniently estimated by extending the top plate's dimensions by distance X in all directions. A cylindrical approximation [110, p. 136] is used to find

$$X \approx \frac{\pi H}{\ln\left(1 + 2\frac{H}{T}\left(1 + \sqrt{1 + T/H}\right)\right)} - \frac{T}{4}. \tag{B.1}$$

A film of thickness T and height H over a ground plane will have perimeter capacitance approximately equal to an ideal parallel plate of width X. The extensions in Table B.5 below are all in microns, with approximate values rounded to the nearest half micron.

Table B.5
Extensions for Perimeter Capacitance

Film	Plane	T	H	X	$\sim X$
Polysilicon	Field	0.5	0.4	0.7	0.5
	Metal 1	0.5	0.6	0.9	1.0
	Metal 2	0.5	1.6	1.7	1.5
Metal 1	Field	1.0	1.0	1.5	1.5
	Poly or Diff	1.0	0.6	1.1	1.0
	Metal 2	1.0	1.0	1.5	1.5
Metal 2	Field	1.0	2.0	2.5	2.5
	Poly or Diff	1.0	1.6	2.1	2.0
	Metal 1	1.0	1.0	1.5	1.5

Appendix C

Layout Design Rules

This appendix documents the geometric rules used to lay out the associative processor chip. They are suitable for use with both the MTL CCD-CMOS [109] and Orbit 1.5 μm CMOS [103] processes. Note that although both processes have two layers of polysilicon, MTL optimizes transistors made with second-poly gates while Orbit optimizes the first-poly devices. In the rules below, *polysilicon* and *poly* always refer to the optimized layer, and the non-optimized layer is called the *electrode*. All dimensions are in microns.

		MTL	Orbit	Common
1. N-Well				
1.1	Width	3.0	2.5	3.0
1.2	Space	12.0	8.0	12.0
2. Active Area				
2.1	Width			
2.1.2	Interconnect	3.0	2.5	3.0
2.1.2	Channel	3.0	3.0	3.0
2.2	Space	2.0	2.0	2.0
2.3	*n*-active to outside *n*-well edge	6.0	6.0	6.0
2.4	*n*-active to inside *n*-well edge	3.0	0.0	3.0
2.5	*p*-active to outside *n*-well edge	3.0	3.0	3.0
2.6	*p*-active to inside *n*-well edge	6.0	2.0	6.0
3. Polysilicon				
3.1	Width	1.5	1.5	1.5
3.2	Space	1.5	1.5	1.5
3.3	Poly extension beyond gate	1.5	1.5	1.5
3.4	Active extension beyond gate	2.0	2.5	2.5
3.5	Space to unrelated active	1.0	0.5	1.0

		MTL	Orbit	Common
4. Electrode				
4.1	Edges not coincident with poly			
4.2	Width of interconnect	1.5	1.5	1.5
4.3	Channel length	2.0	2.0	2.0
4.4	Space	1.5	2.5	2.5
4.5	Space to poly over field		1.5	1.5
4.6	Space to poly over active		2.0	2.0
4.7	Active extension beyond gate	2.0	2.5	2.5
4.8	Electrode extension beyond gate	1.5	1.5	1.5
4.9	Space to unrelated active	1.0	1.0	1.0
4.10	Minimum overlap of poly	0.5	1.0	1.0
4.11	Space to inside poly edge	0.5	1.0	1.0
4.12	Capacitor with large poly			
4.12.1	Space inside poly	0.5	1.0	1.0
4.12.2	Poly space to active	1.0	0.5	1.0
4.13	Capacitor with large electrode			
4.13.1	Surround of poly	0.5	1.0	1.0
4.13.2	Electrode space to active	1.0	1.0	1.0
5. Contact				
5.1	Size (exact)	1.5×1.5	1.5×1.5	1.5×1.5
5.2	Spacing	1.5	2.0	2.0
5.3	Active surround	1.0	1.0	1.0
5.4	Active contact space to poly gate	2.0	1.5	2.0
5.5	Active contact space to electrode gate	2.0	2.0	2.0
5.6	Poly contact space to active	2.0	1.5	2.0
5.7	Electrode contact space to active	2.0	2.0	2.0
5.8	Poly surround	1.0	1.0	1.0
5.9	Electrode surround	1.0	1.0	1.0
5.10	Poly contact space to electrode		2.0	2.0
5.11	Electrode contact space to poly		2.0	2.0
6. Metal 1				
6.1	Width	3.0	2.0	3.0
6.2	Space	2.0	2.0	2.0
6.3	Contact surround	1.5	0.75	1.5

		MTL	Orbit	Common
7. Via				
7.1	No stacked vias			
7.2	Size[1]	2.0×2.0	1.5×1.5	
7.3	Spacing	2.0	2.0	2.0
7.4	Space to outside active edge	2.0	1.5	2.0
7.5	Space to inside active edge	2.0	2.0	2.0
7.6	Space to poly edge	2.0	1.5	2.0
7.7	Space to electrode edge	2.0	1.5	2.0
7.8	Space to contact	2.0	1.5	2.0
7.9	Metal 1 surround	1.0	1.0	1.0
8. Metal 2				
8.1	Width	4.0	2.5	4.0
8.2	Space	3.0	2.0	3.0
8.3	Via surround	1.5	1.0	1.5
9. Pad				
9.1	Opening size		90×90	90×90
9.2	Via surround of opening		2.0	2.0
9.3	Metal 1 and 2 surround of opening		5.0	5.0
9.4	Metal 1 and 2 pad size		100×100	100×100
9.5	Pad–pad space		75	75
9.6	Pad metal space to circuits		20	20
9.7	Pad metal space to scribe		20	20
9.8	Pad metal space to metal 1			30
9.9	Pad metal space to metal 2			30

[1]There is no common rule for via size, but the Magic layout editor can generate mask data with the correct size for either technology.

Appendix D

Cell Design

Figure D.1 is a schematic diagram of the cell as it is actually implemented. Cells are laid out in pairs, as shown in figure D.2, with bit line contacts shared between vertically adjacent cells. The figure omits overlapping geometries from neighboring cells for clarity. The unit size is $32\,\mu\text{m} \times 33.5\,\mu\text{m}$.

Mismatch current is generally limited by the storage transistors $M_{S0,1}$, which must be wide enough to pull down the match line when the stored charge is reduced. In contrast, the non-limiting diodes can be drawn at minimum size. The smaller diodes contribute less capacitance to the match line, allowing a faster match operation while reducing the necessary mismatch current. The cell uses two diode devices $M_{D0,1}$, as this topology was found to use less area than a single-diode layout.

The parasitic capacitance at the source of the diode has very little effect on match performance. In matching columns, the node is isolated from the match line by the diode. In mismatching columns it contributes only a small fraction of the total match line capacitance. It must be included, however, when computing the match line charging capacitance.

The W_{WA} and W_{WB} lines run horizontally in polysilicon and are strapped by second metal (contacts every eight cells not shown). Both cells in the pair share a common W_T line, which runs vertically in metal. Separate contacts are used so that the two first-poly gates are connected only by metal. This safety precaution prevents the possibility that second-poly stringers could short the W_{WA} and W_{WB} lines. Such stringers were observed during the development of the MTL CCD-CMOS process, but they have since been corrected [109].

Two second metal pitches are used for the network signals NET_A and NET_B. They are shielded from the match line by the W_W lines, which are driven low during match operations.

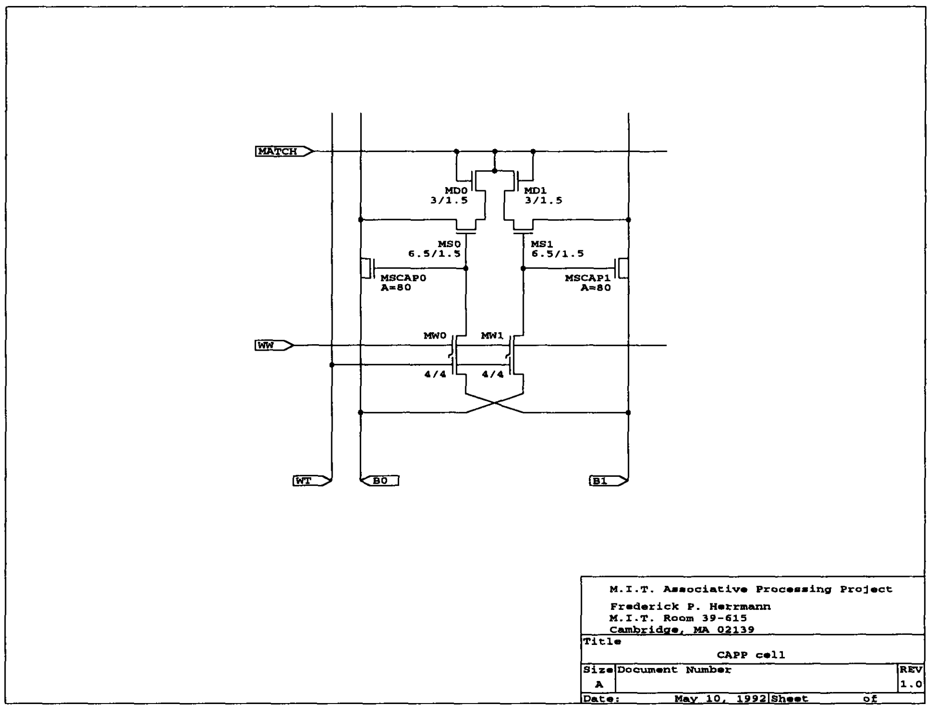


Figure D.1: Dynamic associative processor cell schematic.

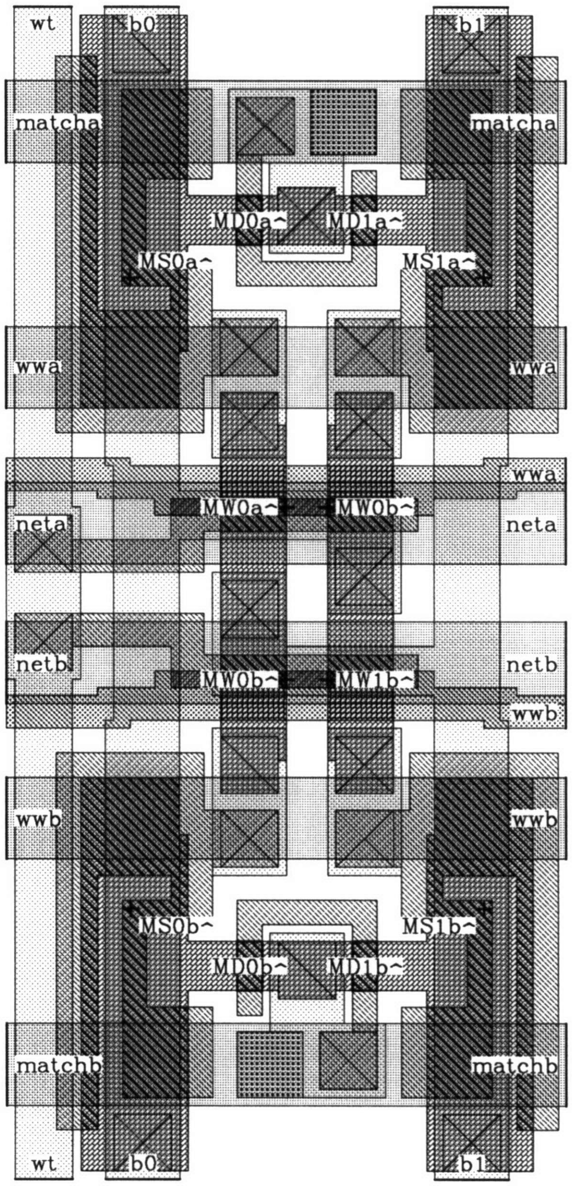


Figure D.2: Dynamic associative processor cell layout. A mirrored pair of cells is shown, omiting some overlapping geometries. The unit size is $32\,\mu\text{m} \times 33.5\,\mu\text{m}$.

Appendix E

Schematic Diagrams

This appendix provides complete schematic diagrams for all circuits on the associative processor chip except for the cell itself, which is described in Appendix D. The appendix is organized into seven sections corresponding to basic chip subsystems:

E.1 Trit column drivers

E.2 Word logic

E.3 Specialized cells

E.4 Word logic control

E.5 Specialized cell control

E.6 Pad circuits and perimeter control logic

E.7 Image I/O shift registers and control

The schematics are organized hierarchically within each section, matching the structure of the HSPICE simulation decks. Notes on the schematics indicate where this organization differs from that of the layout.

E.1 Trit Column Drivers

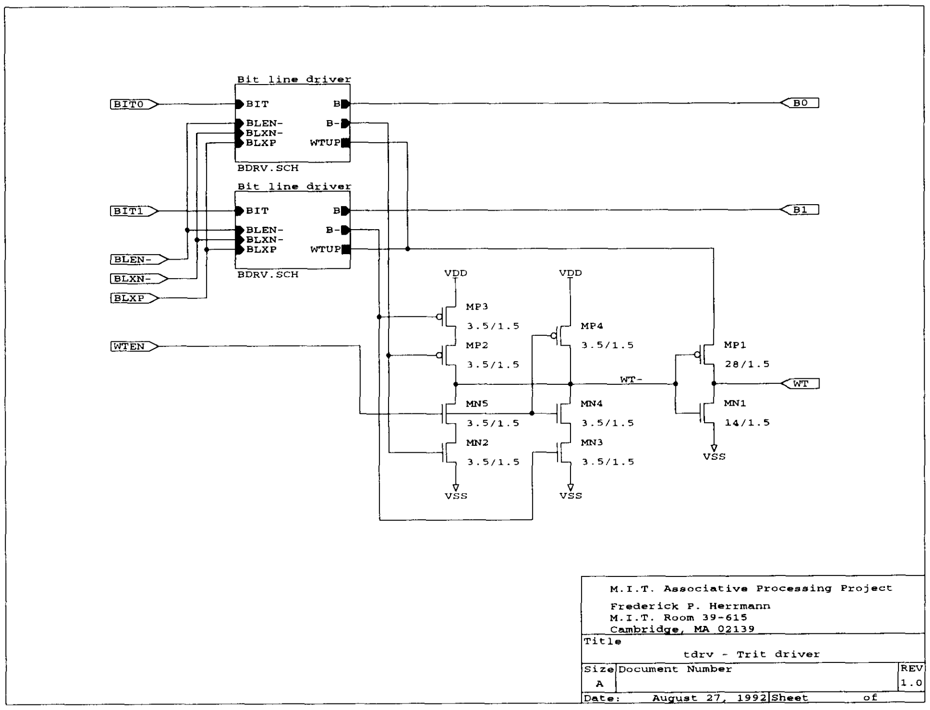


Figure E.1: Trit column driver.

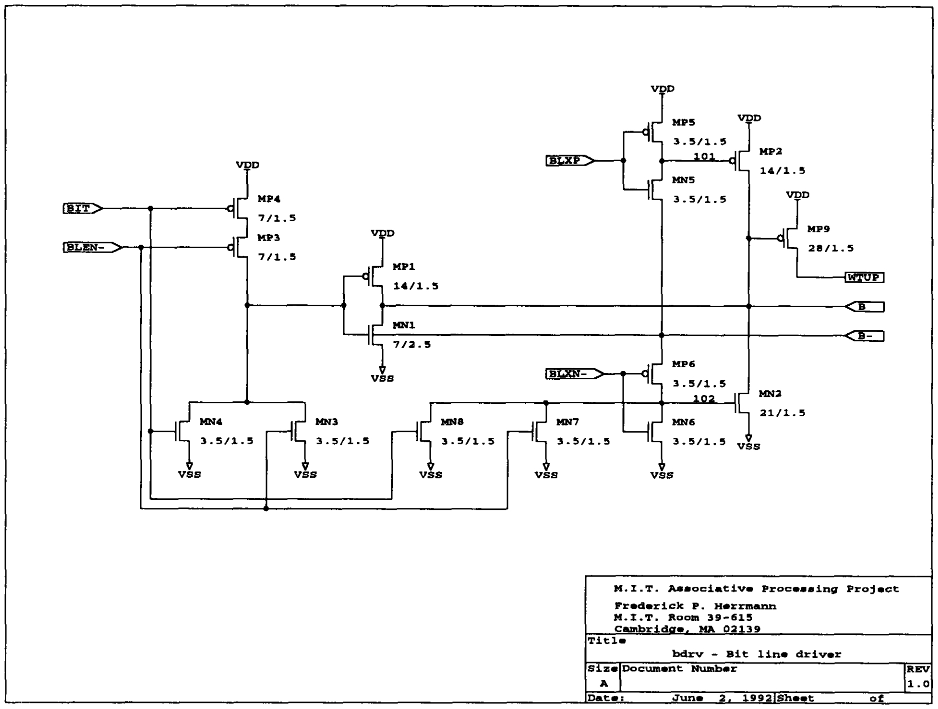


Figure E.2: Bit line driver. Transistors $M_{N7,8}$ are vestiges of an earlier design. They were once needed to keep the circuit static when the clock was stopped, but changes in the $\overline{BLXN}$ timing have made them unnecessary.

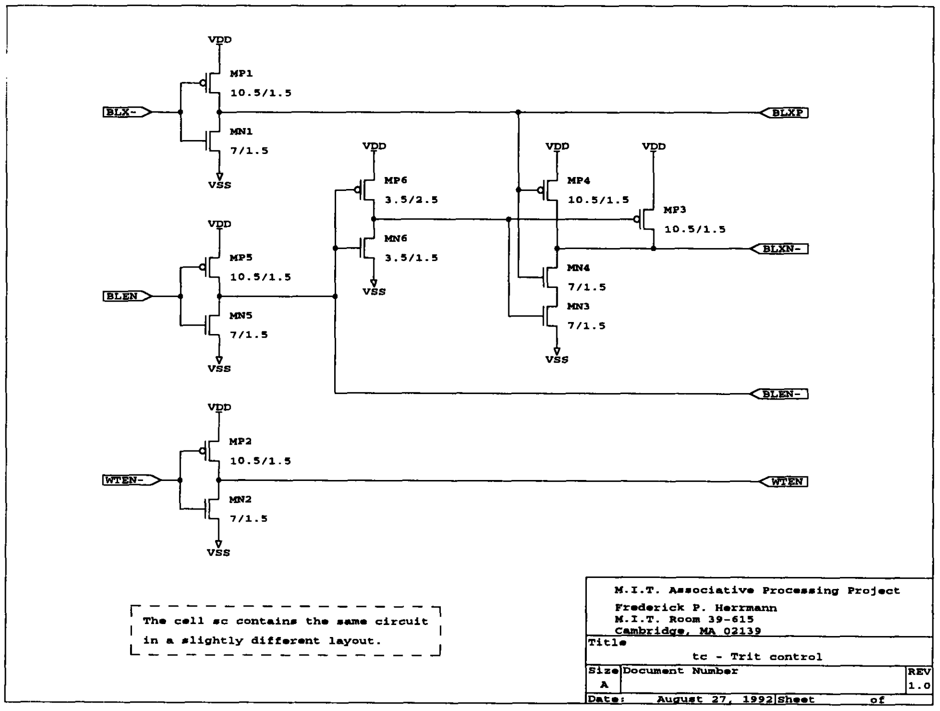


Figure E.3: Trit control. Each instance of this cell serves four trit columns.

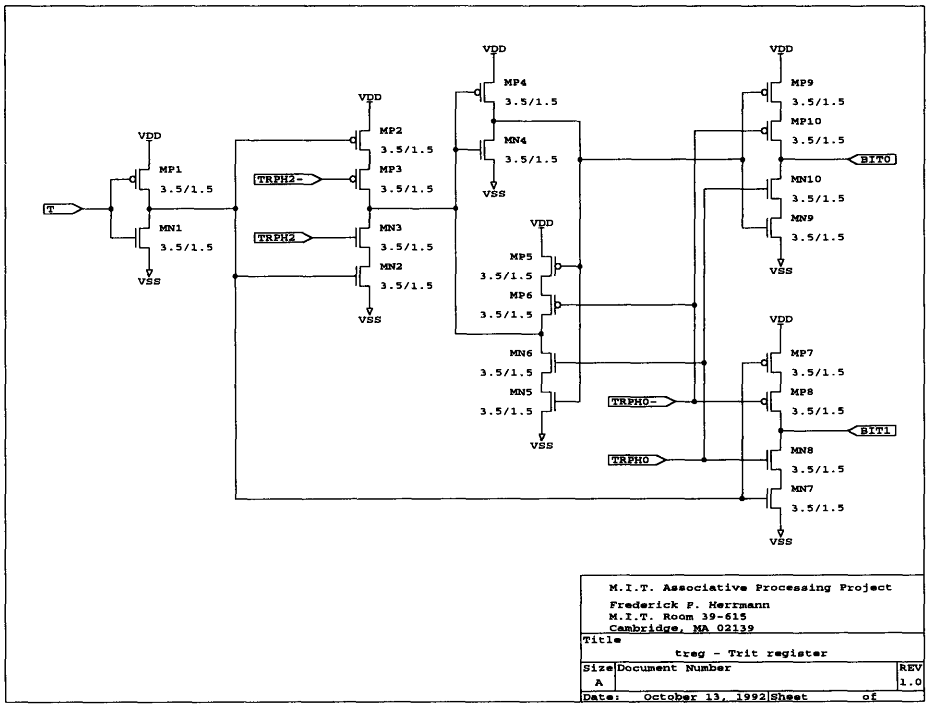


Figure E.4: Trit register. The two bit line values are time-multiplexed on the T input signal and are sampled on phases 0 and 2.

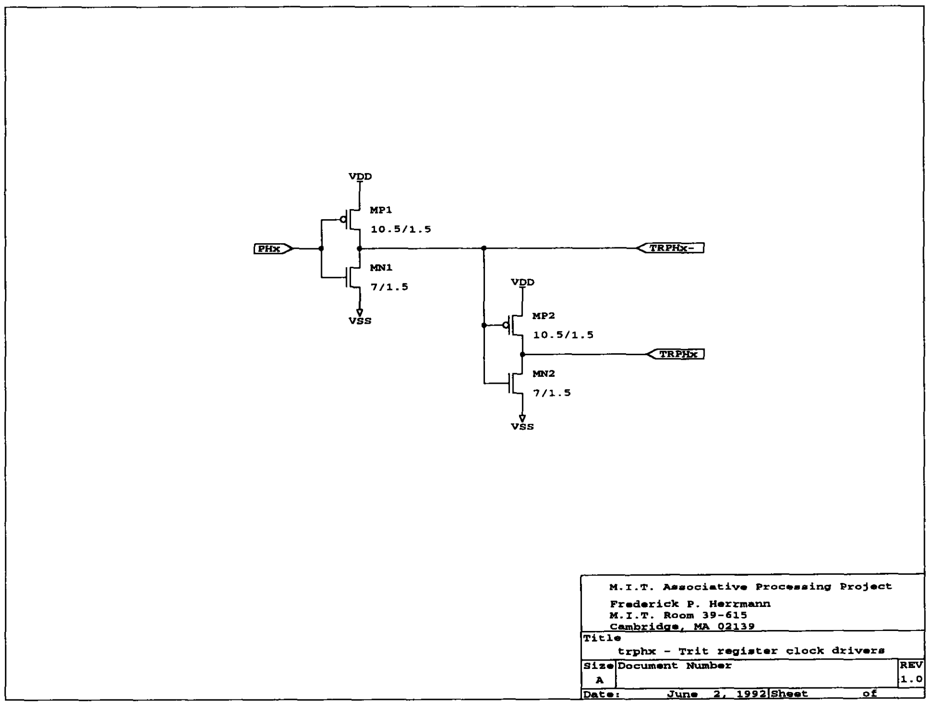


Figure E.5: Trit register clock drivers. Each instance of this cell serves four trit column registers.

E.2 Word Logic

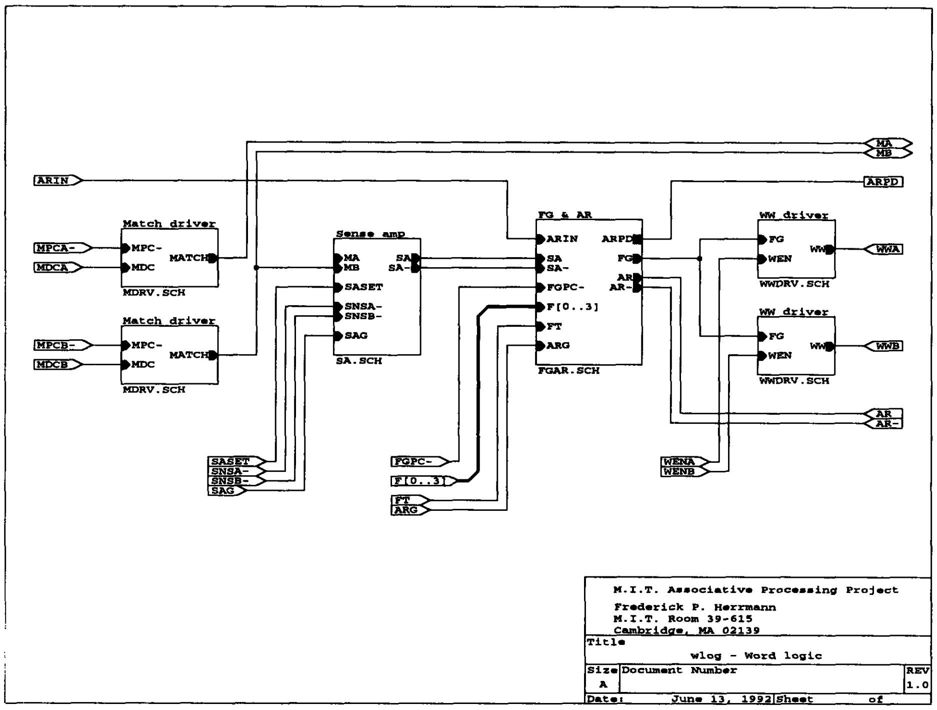


Figure E.6: Word logic.

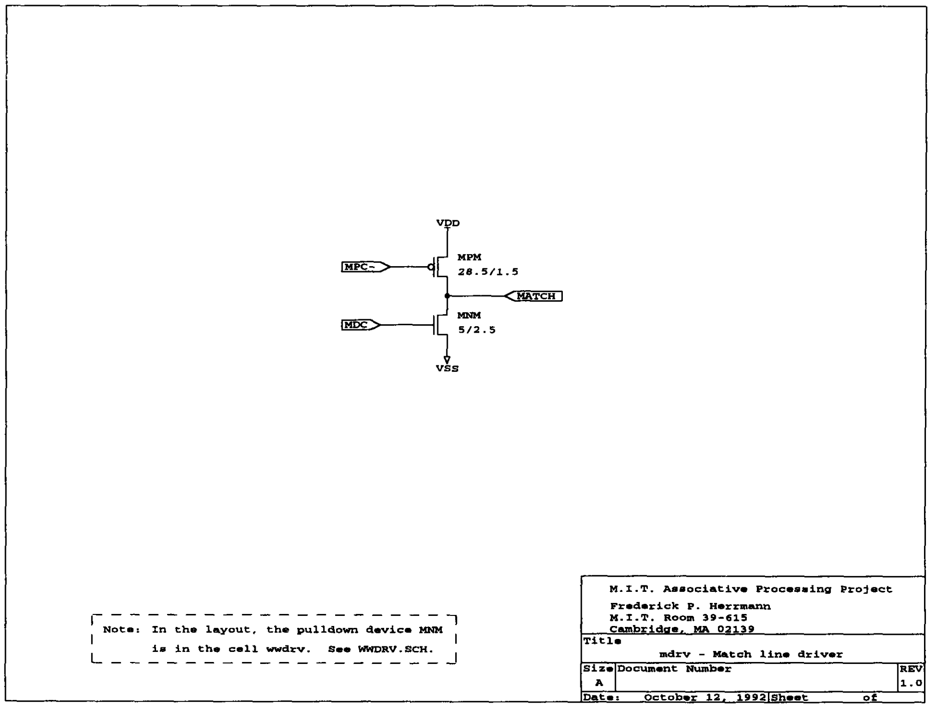


Figure E.7: Match line driver.

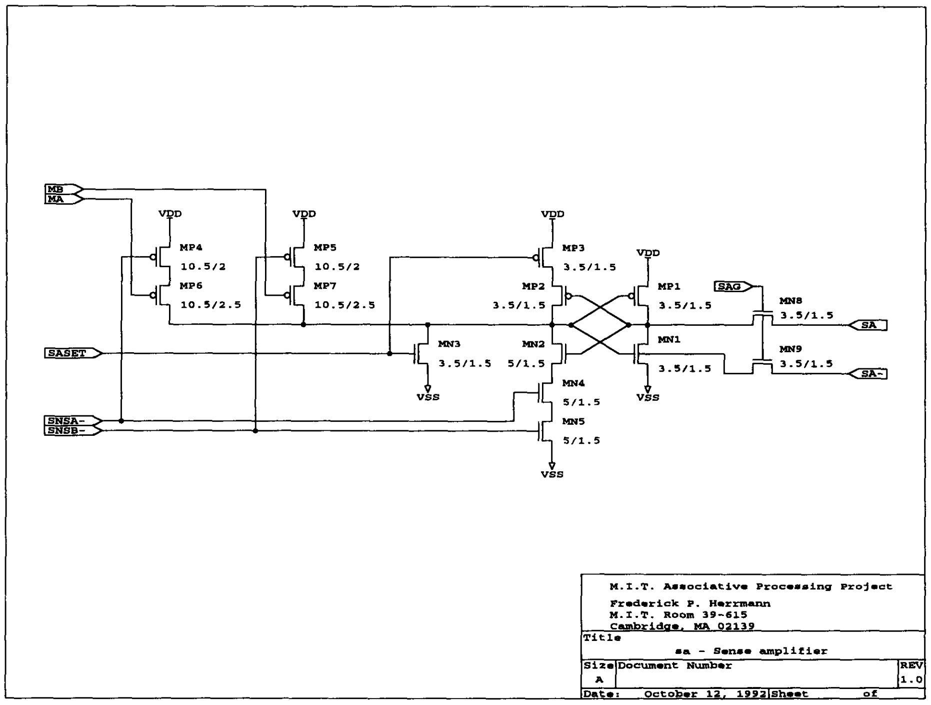


Figure E.8: Sense amplifier.

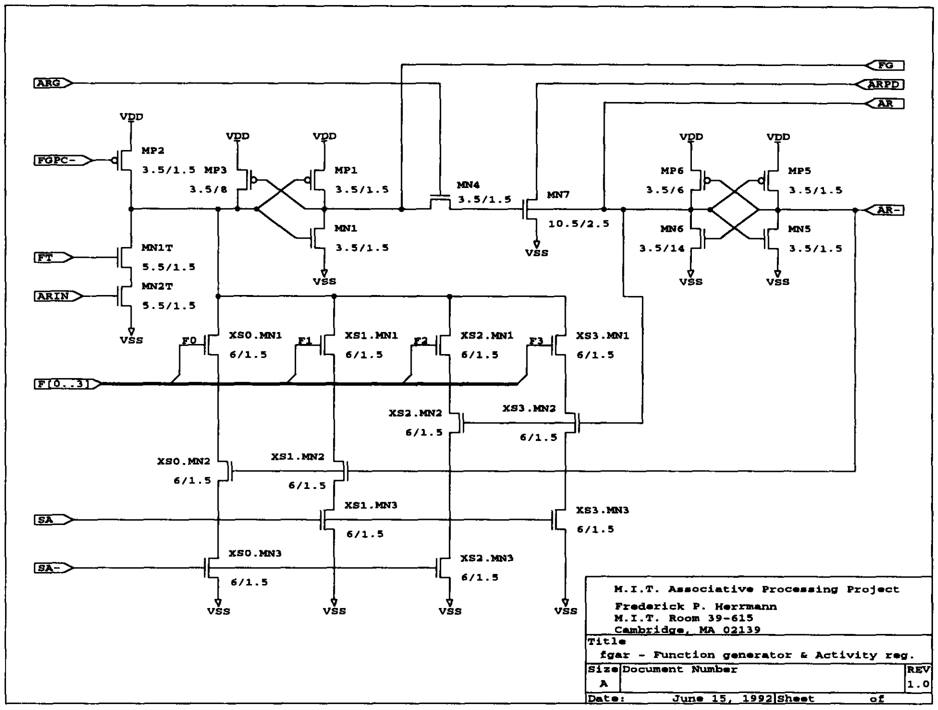


Figure E.9: Function generator and activity register.

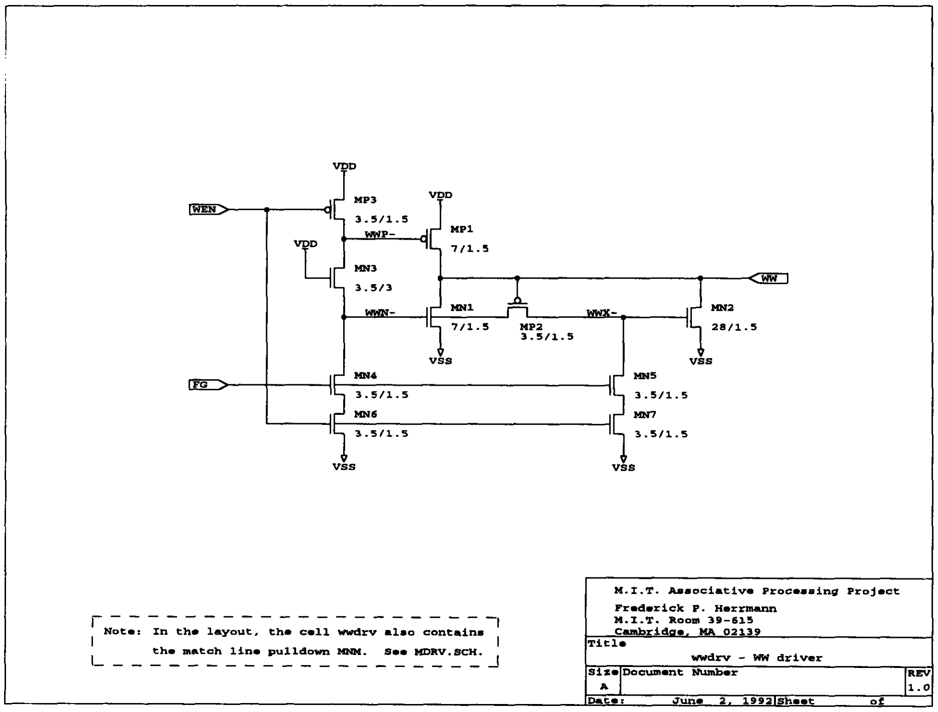


Figure E.10: Write-word driver.

E.3 Specialized Cells

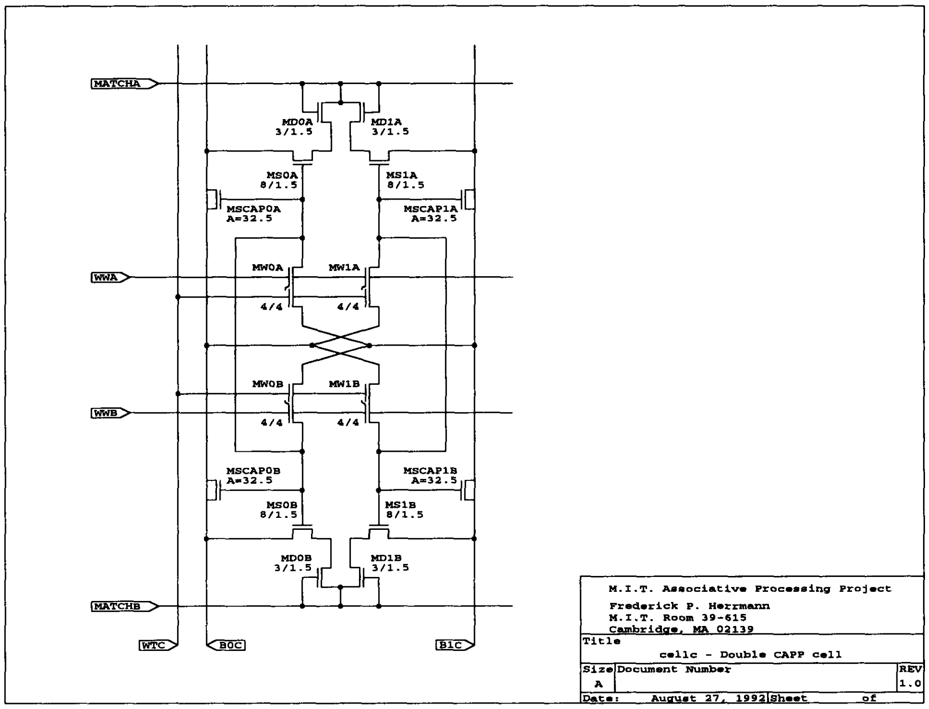


Figure E.11: Carry cell, constructed of two associative processor cells with their storage nodes tied together.

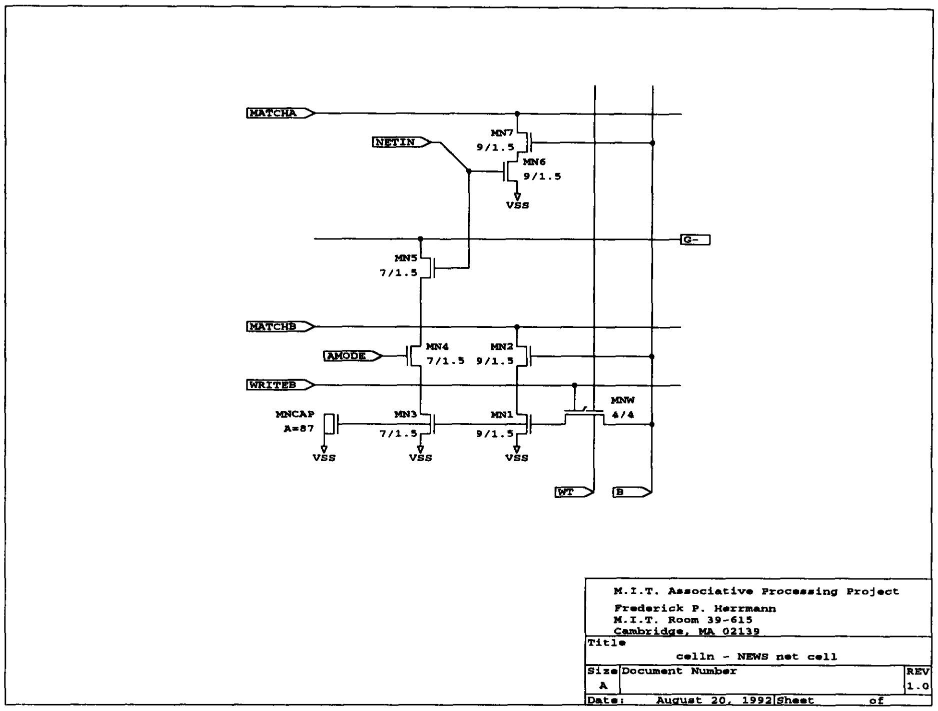


Figure E.12: Network direction cell pair. Each PE has four of these, corresponding to the compass directions. The match-only A half of the cell pair is used in synchronous mode, while the B half is used to establish ARM connections.

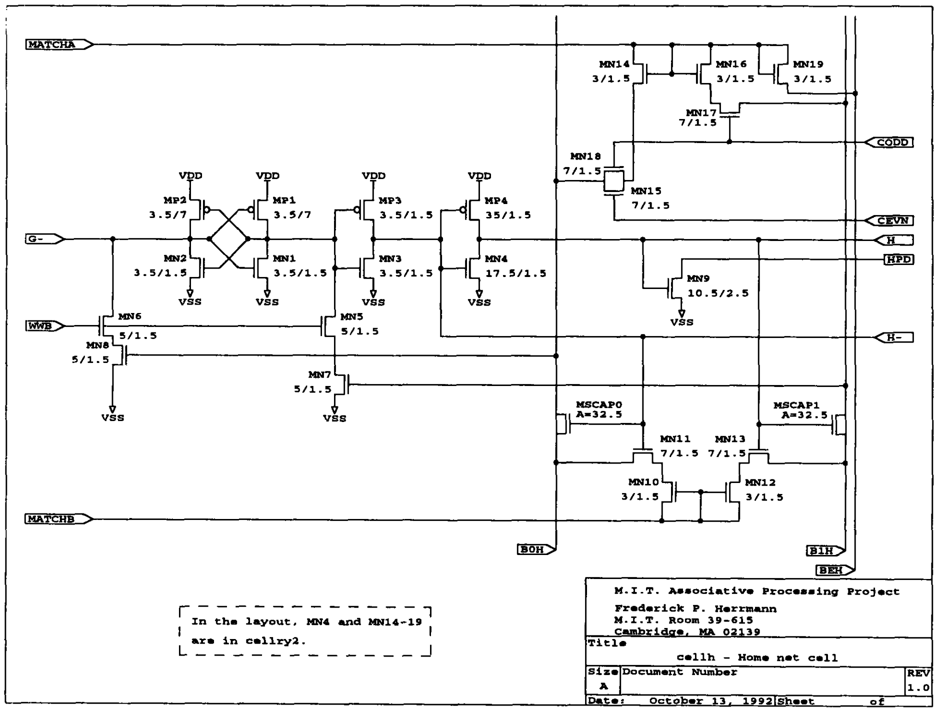
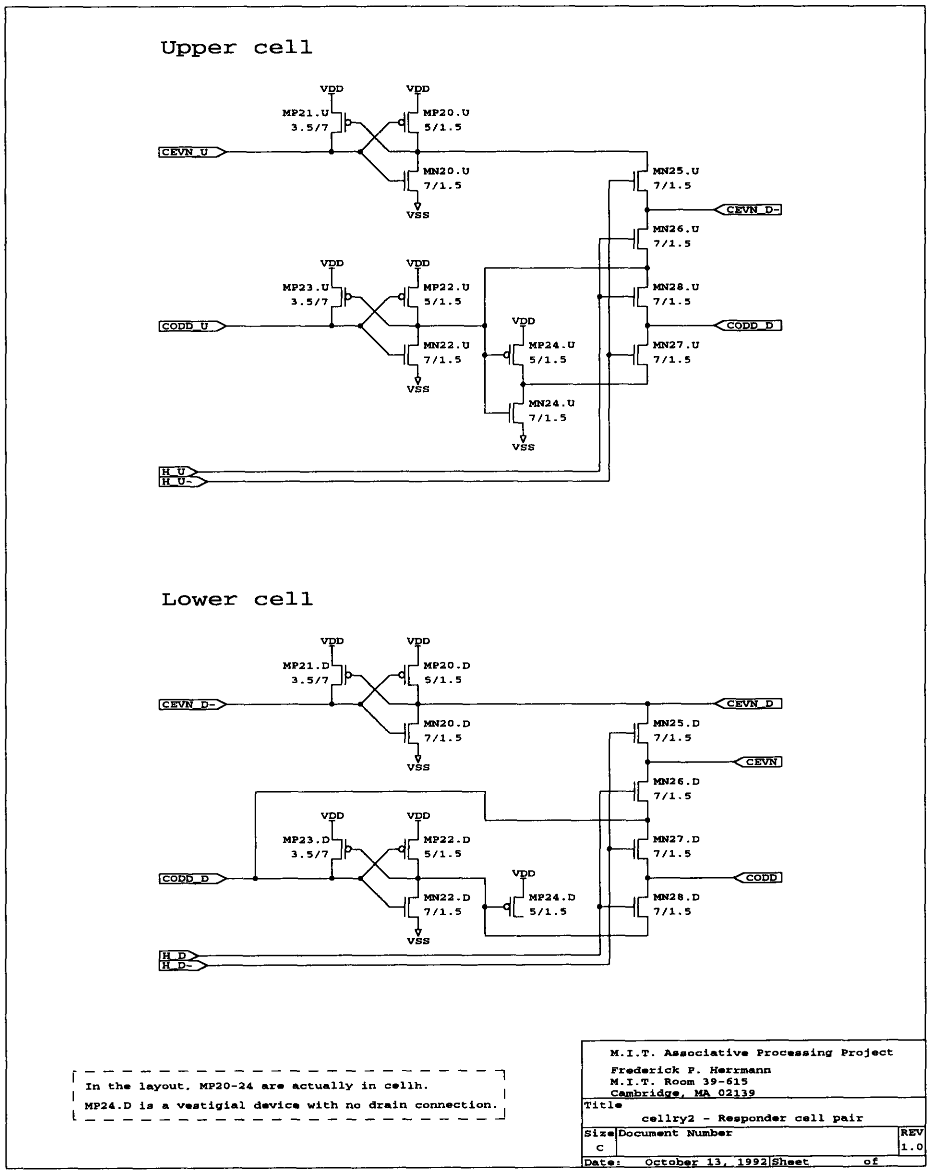


Figure E.13: Home cell pair. The HA cell is match-only and is used to examine the responder chain. The HB cell serves as both network driver and responder flag.



In the layout, MP20-24 are actually in cellh.
MP24.D is a vestigial device with no drain connection.

M.I.T. Associative Processing Project
Frederick P. Herrmann
M.I.T. Room 39-615
Cambridge, MA 02139

Title	cellry2 - Responder cell pair	
Size C	Document Number	REV 1.0
Date: October 13, 1992	Sheet of	

Figure E.14: Responder chain. The upper and lower links alternate in the chain, with the result that signals are restored after every other processing element.

E.4 Word Logic Control

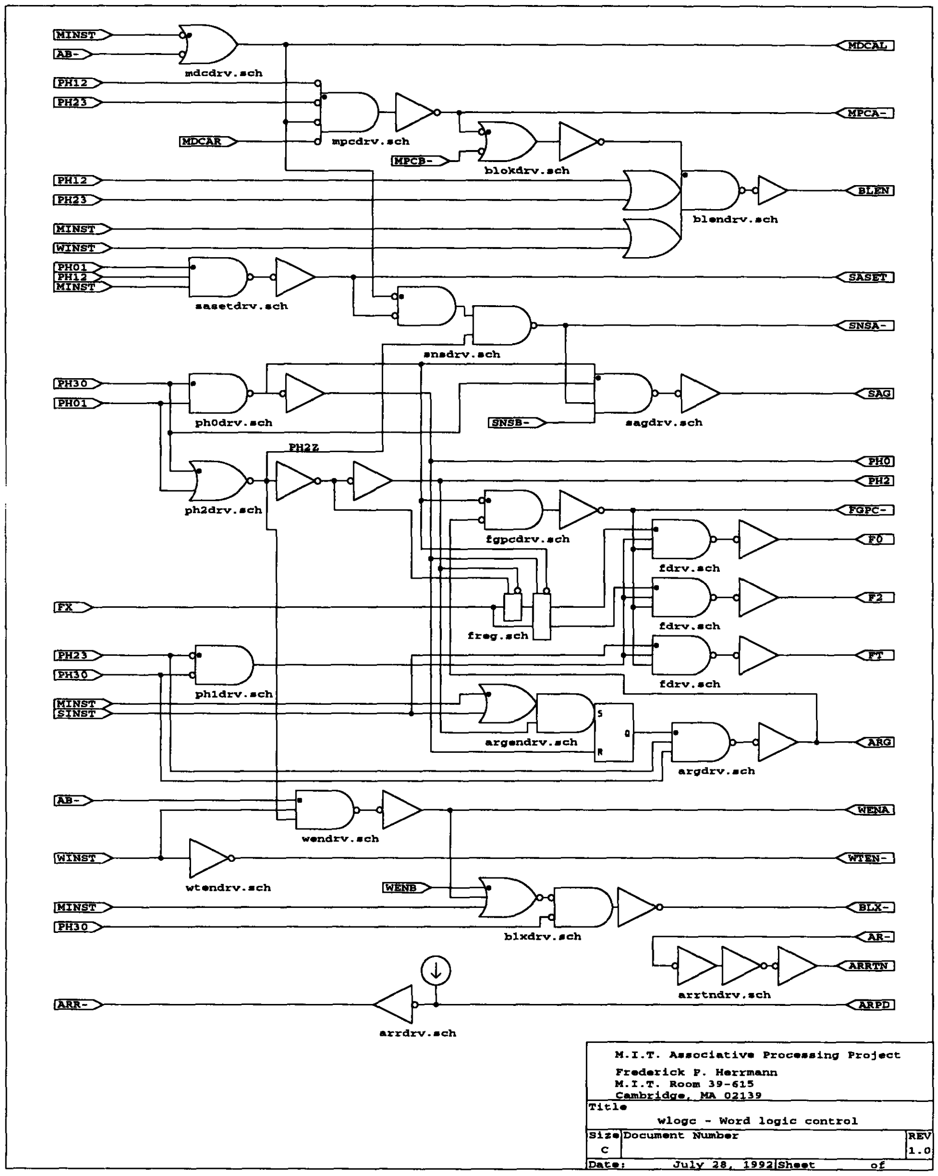


Figure E.15: Word logic control.

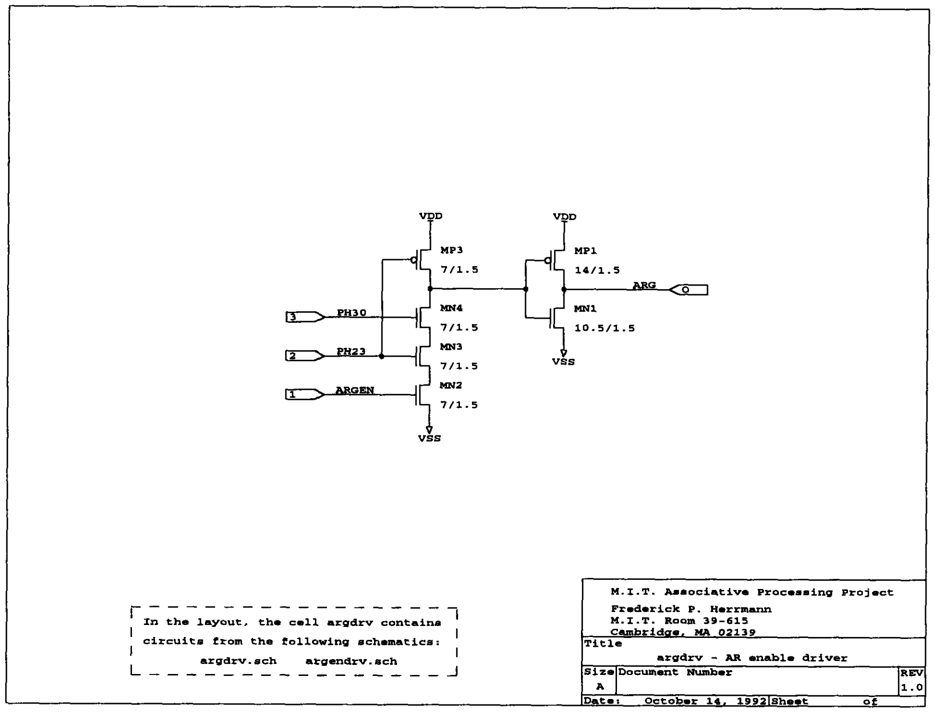


Figure E.16: Activity register enable driver.

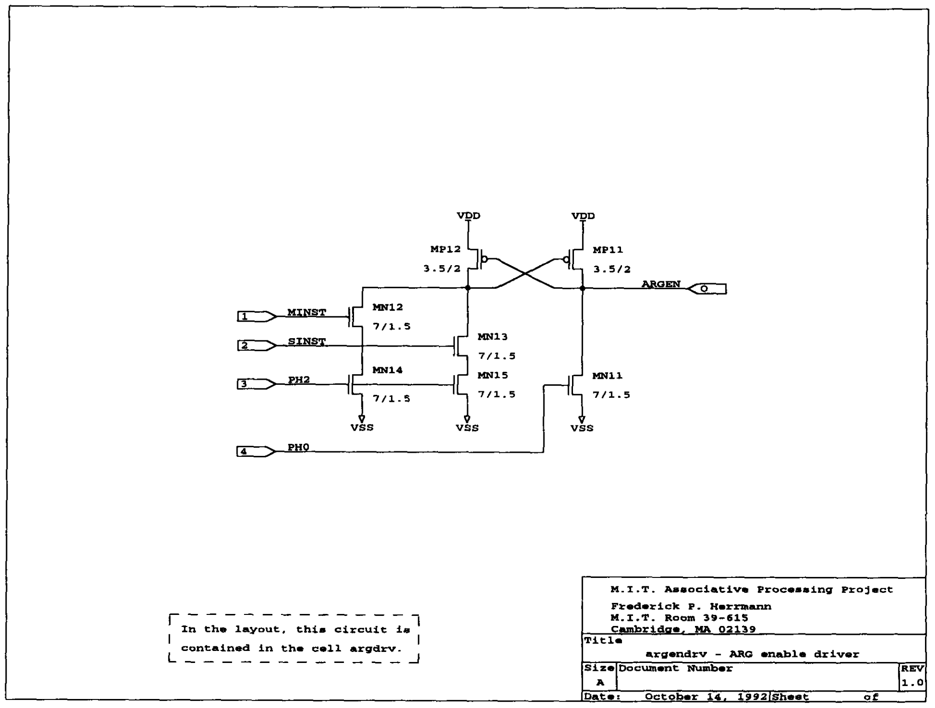


Figure E.17: *ARG* enable driver.

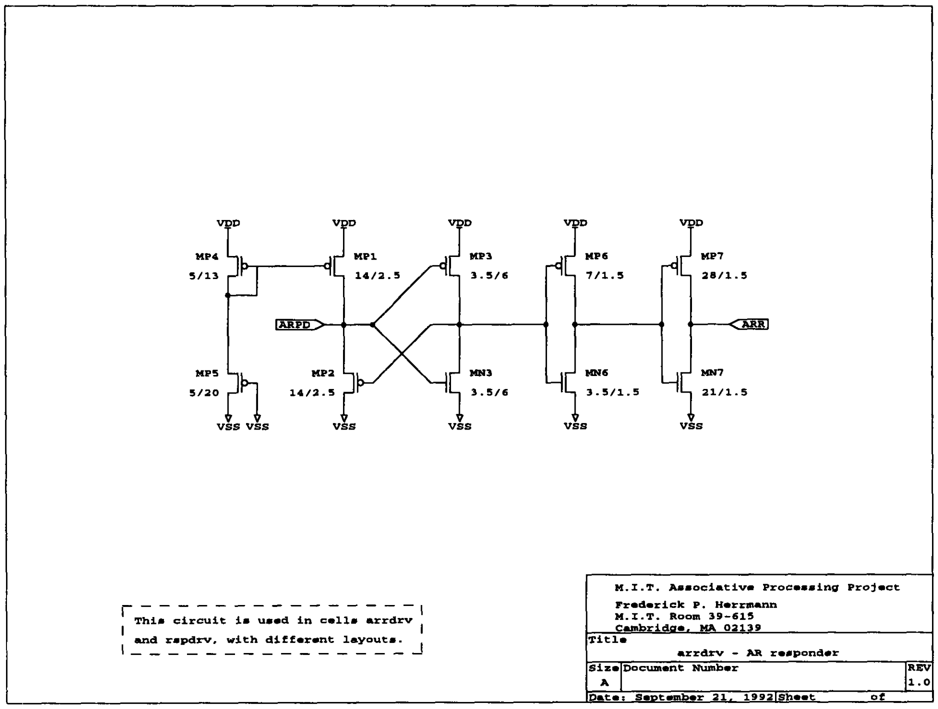


Figure E.18: Activity register wired-OR responder.

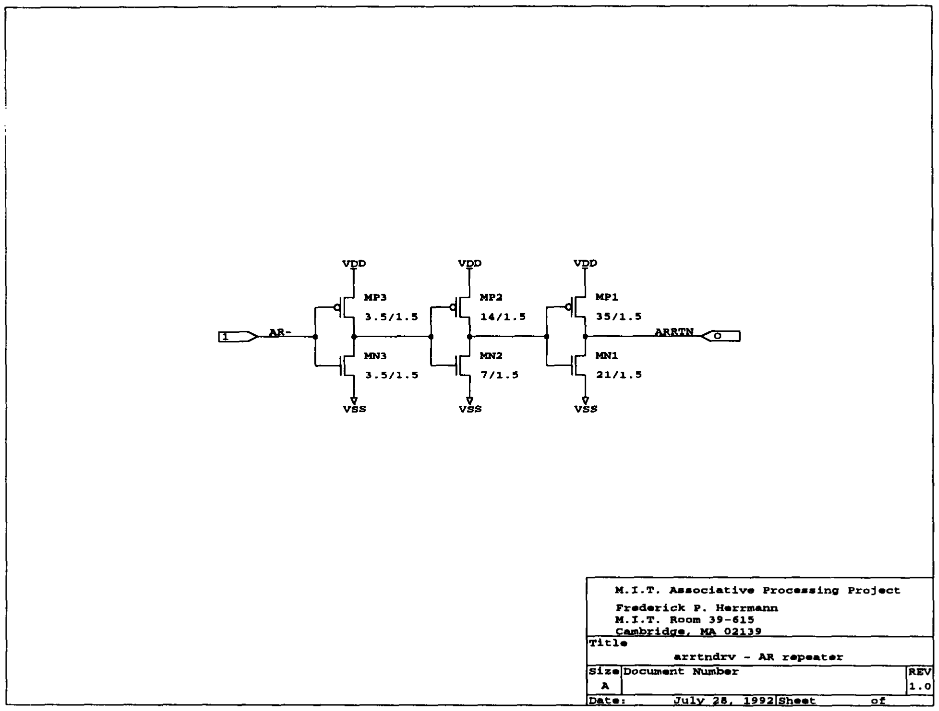


Figure E.19: AR return driver, to buffer signal before sending it to the top of the next subarray.

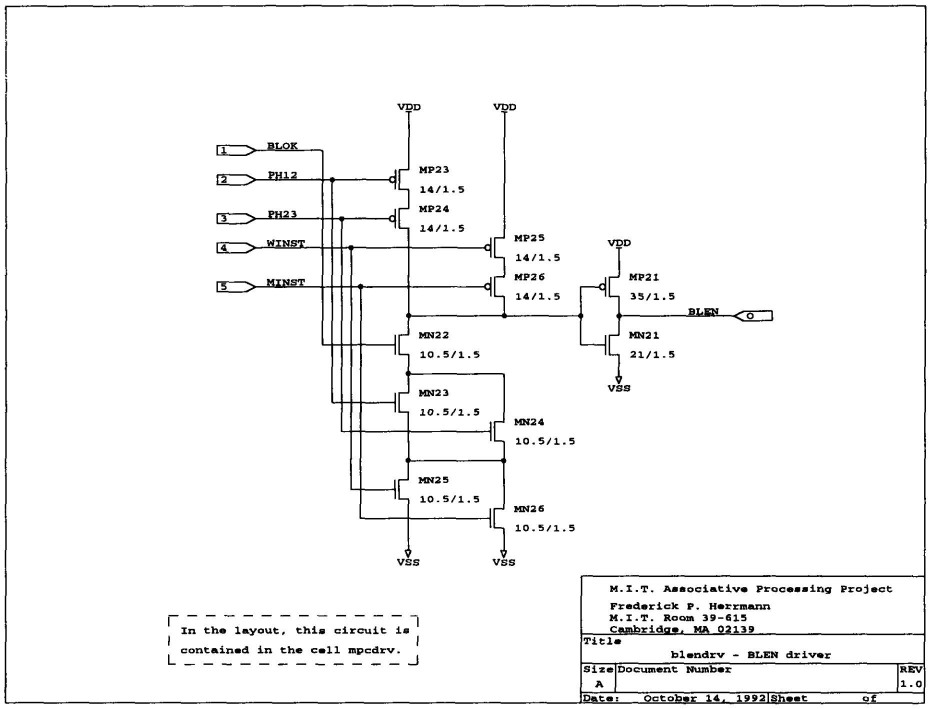


Figure E.20: Bit line enable (*BLEN*) driver.

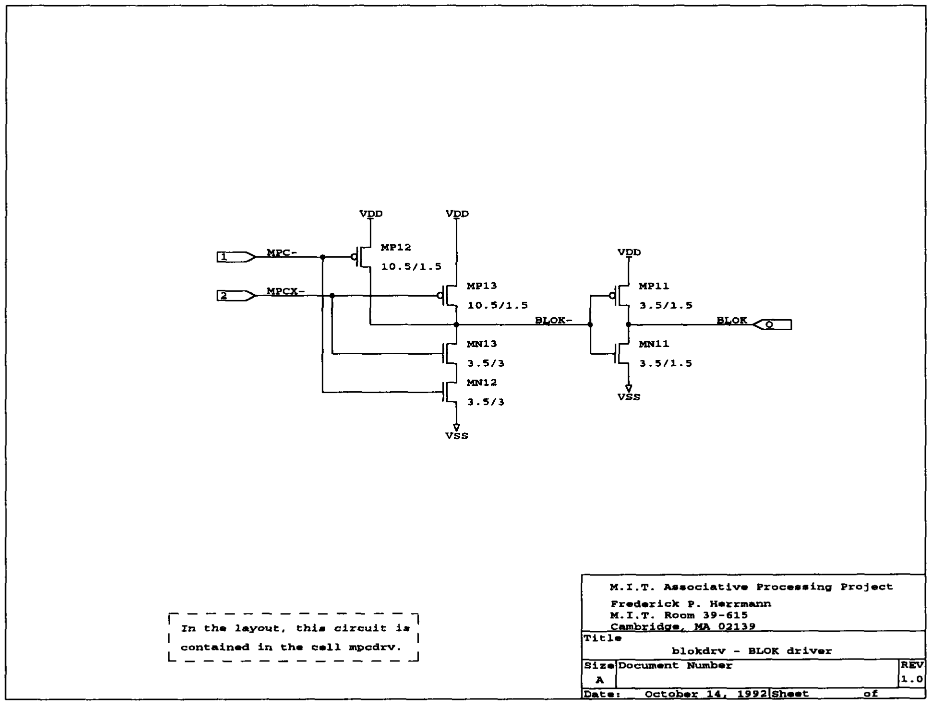


Figure E.21: Bit line okay (*BLOK*) driver. The ratios of the input NAND gate ensure that the *BLOK* signal will not be asserted until the match line precharge drivers are fully off.

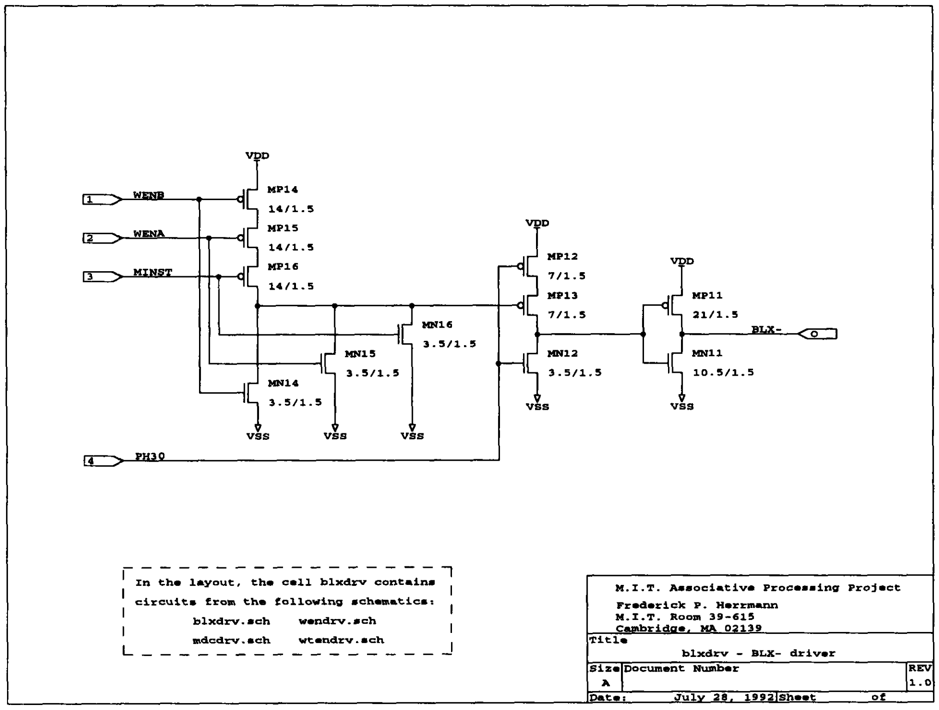


Figure E.22: $\overline{BLX}$ driver.

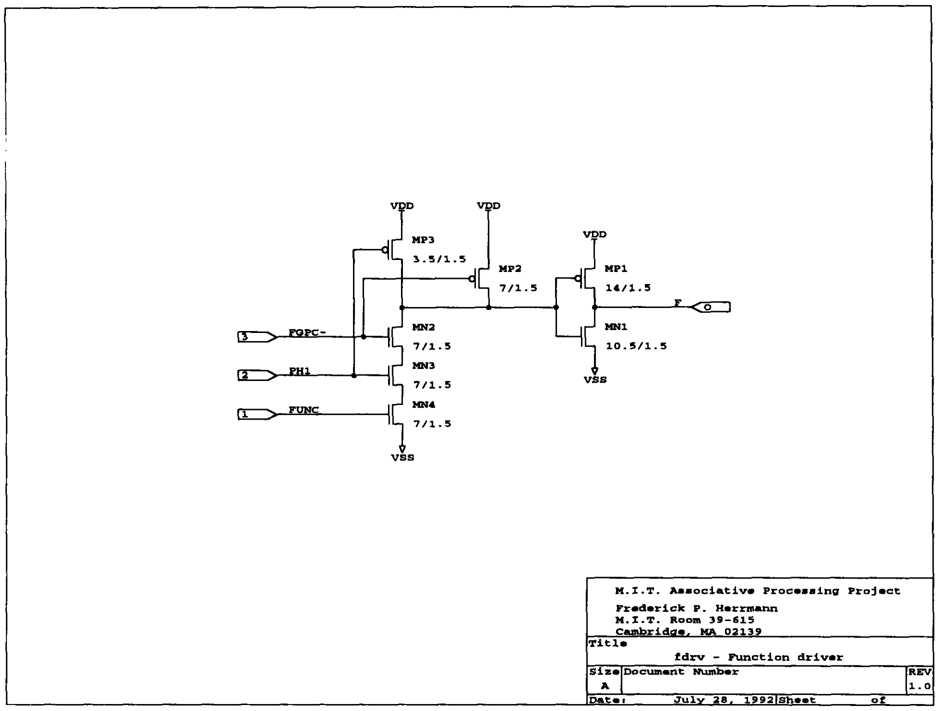


Figure E.23: Function driver. Each subarray uses five of these, for signals $F_{0...3}$ and F_T.

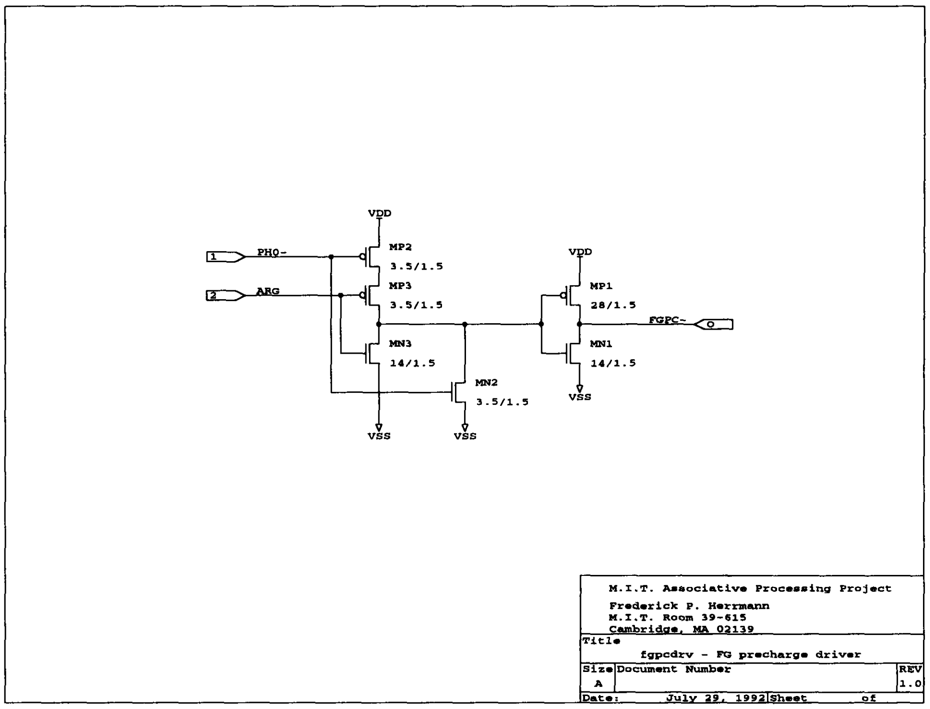


Figure E.24: Function generator precharge ($\overline{FGPC}$) driver.

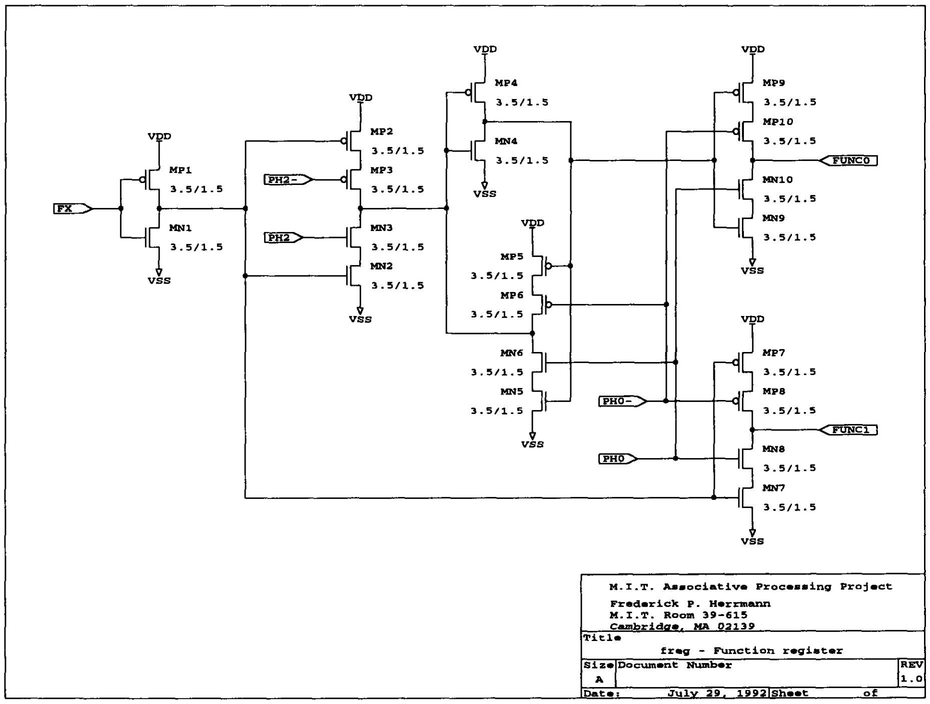


Figure E.25: Function register. Two function bits are time-multiplexed on the *FX* input signal and are sampled on phases 0 and 2.

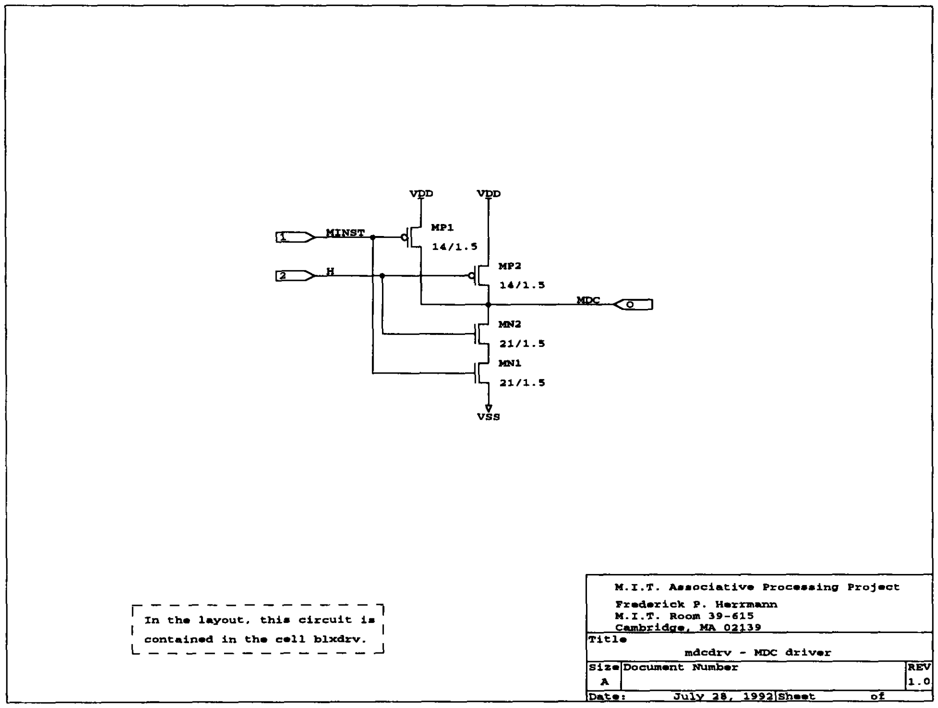


Figure E.26: *MDC* driver.

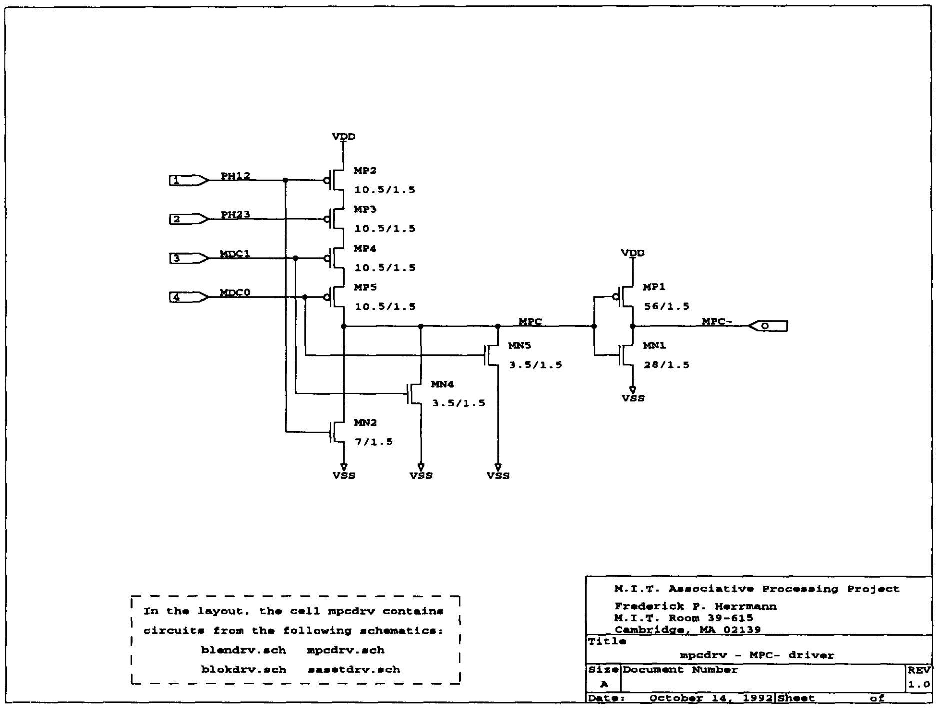


Figure E.27: $\overline{MPC}$ driver.

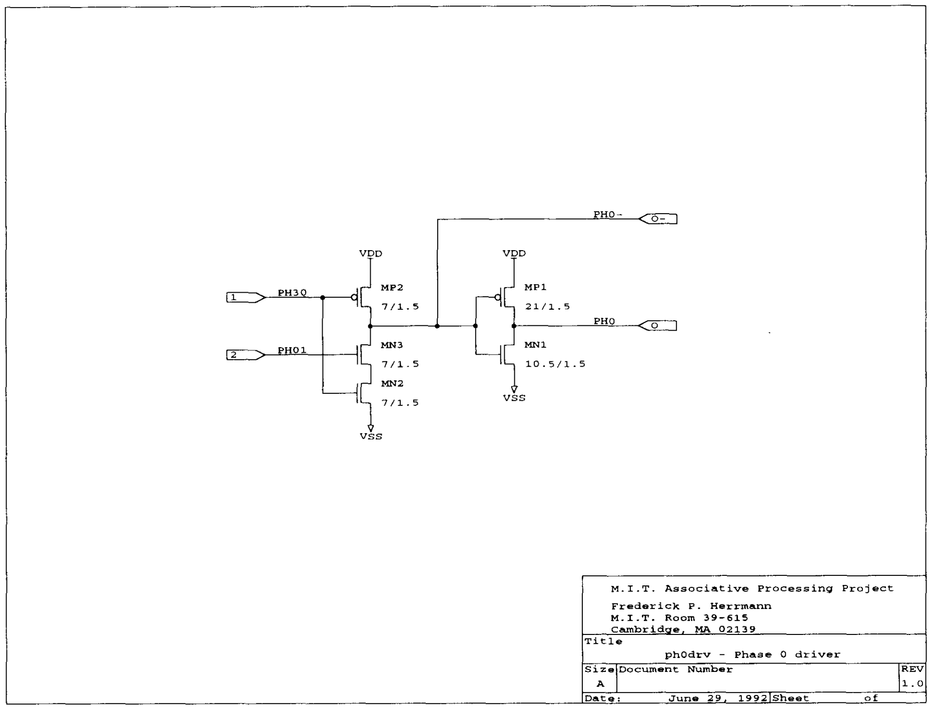


Figure E.28: Phase 0 clock driver.

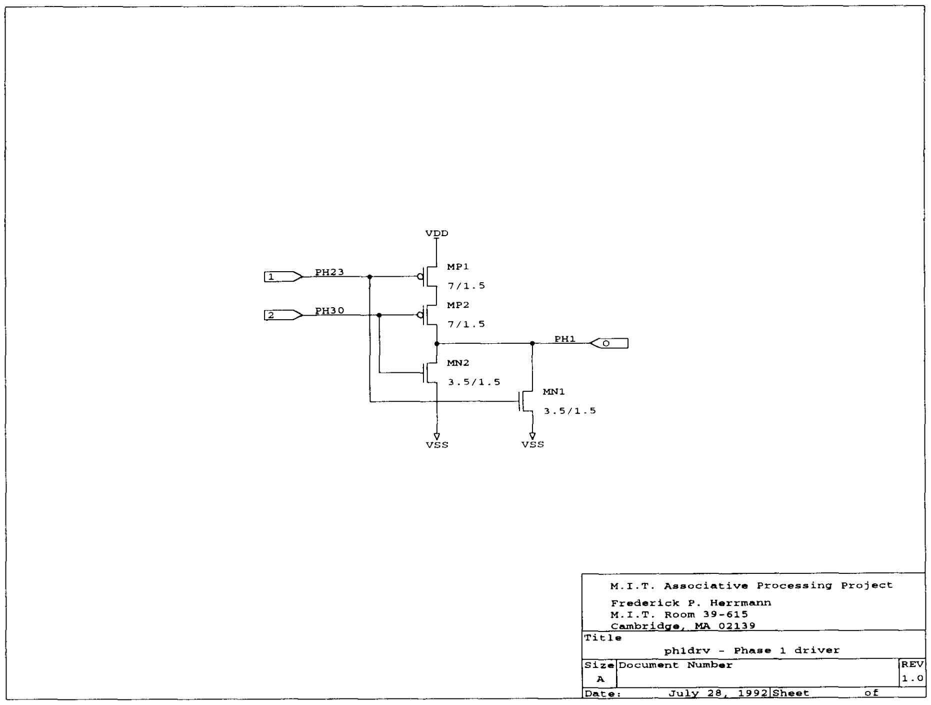


Figure E.29: Phase 1 clock driver.

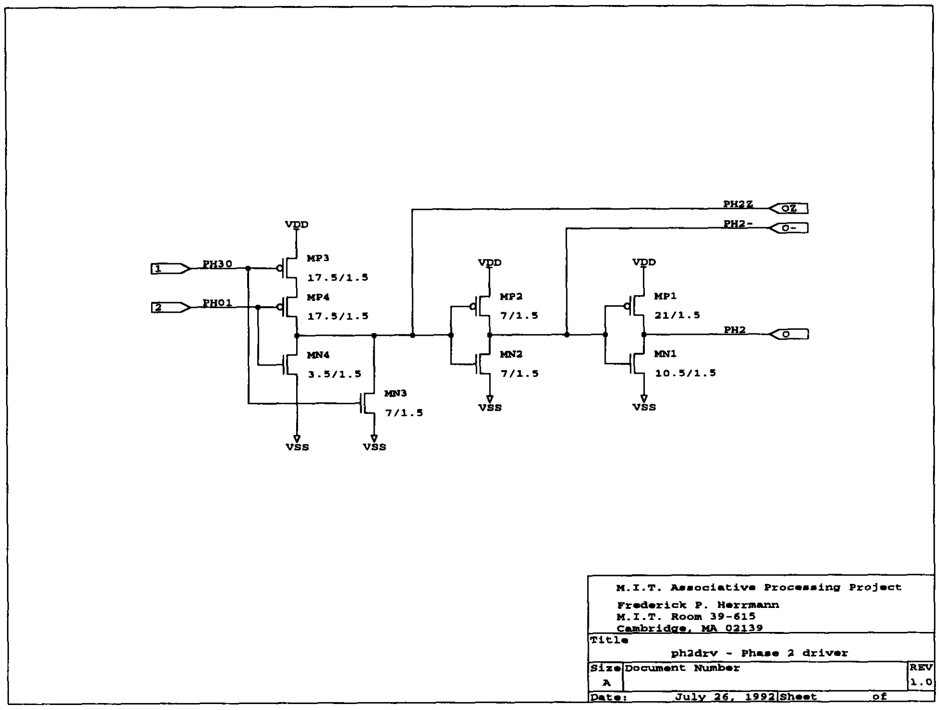


Figure E.30: Phase 2 clock driver.

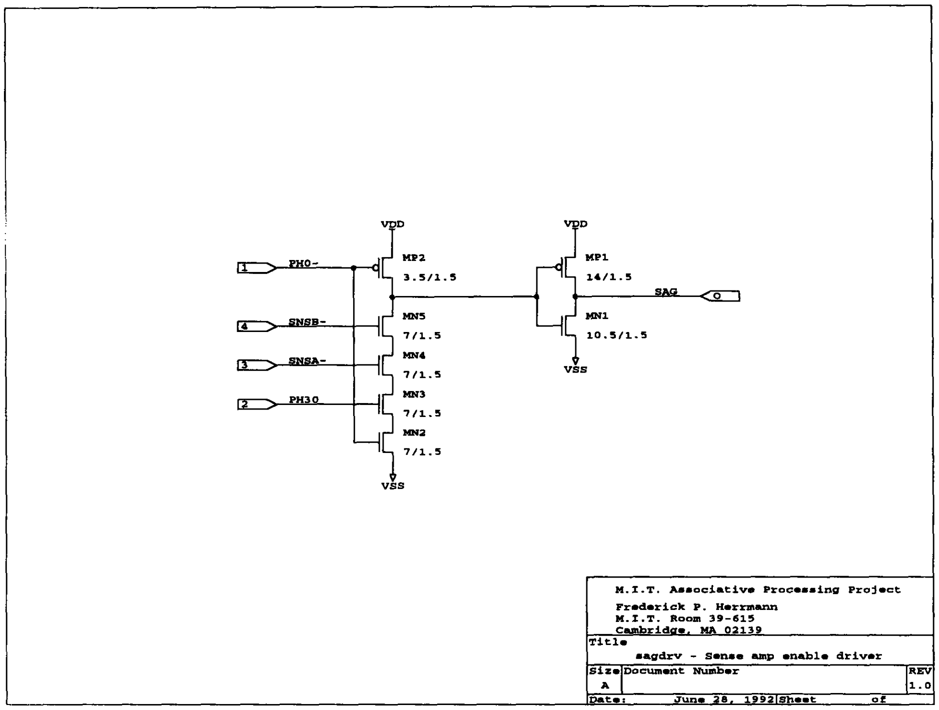


Figure E.31: Sense amplifier enable (SAG) driver.

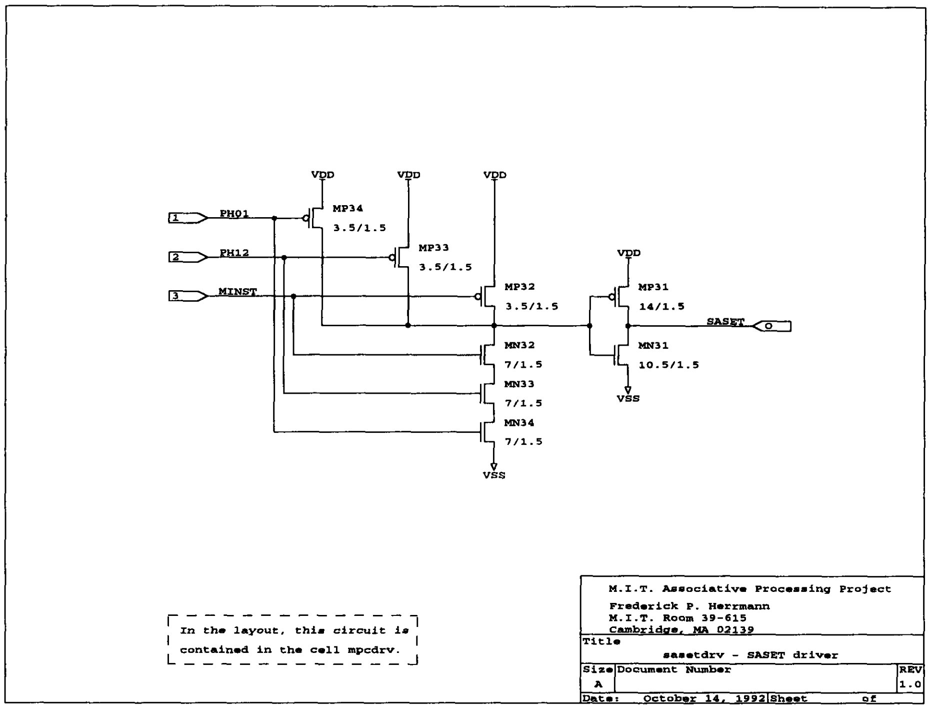


Figure E.32: Sense amplifier set (*SASET*) driver.

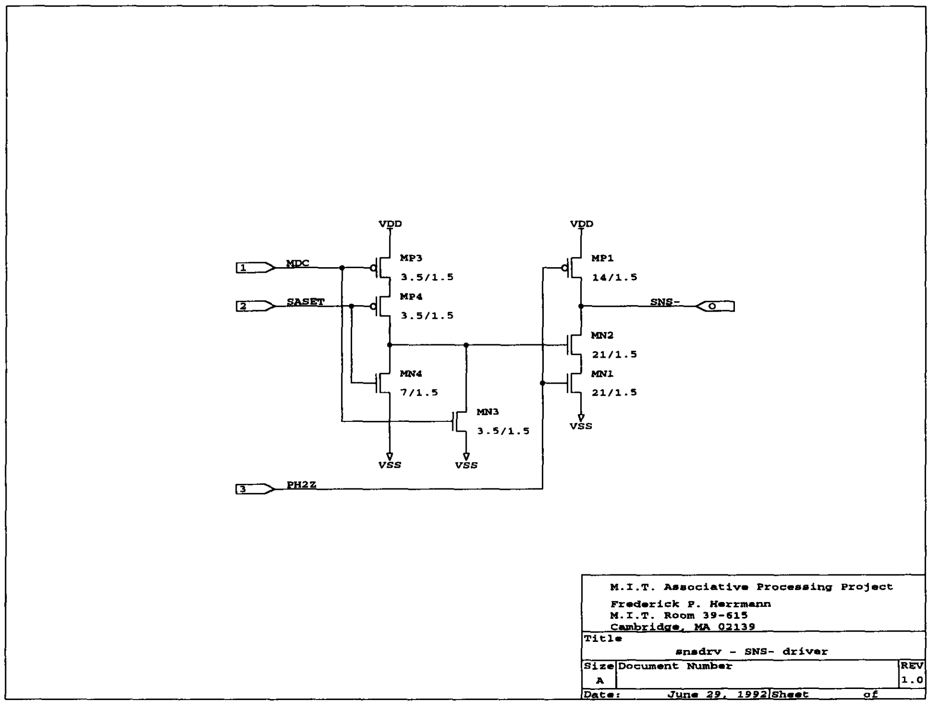


Figure E.33: Sense amplifier sense ($\overline{SNS}$) driver.

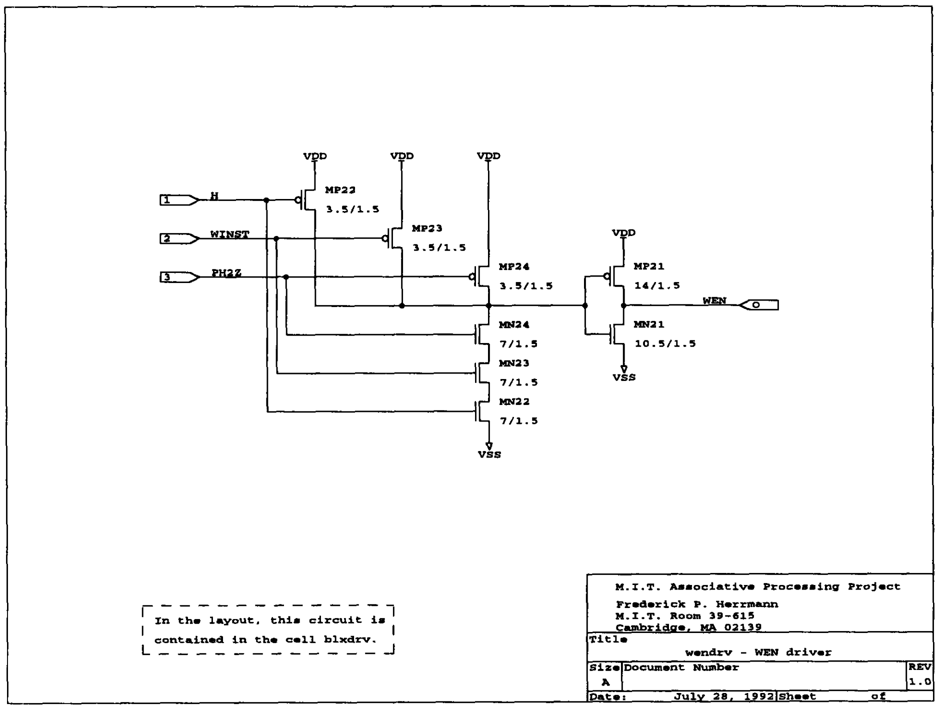


Figure E.34: *WEN* driver. The signal is used by the word logic to control W_W lines.

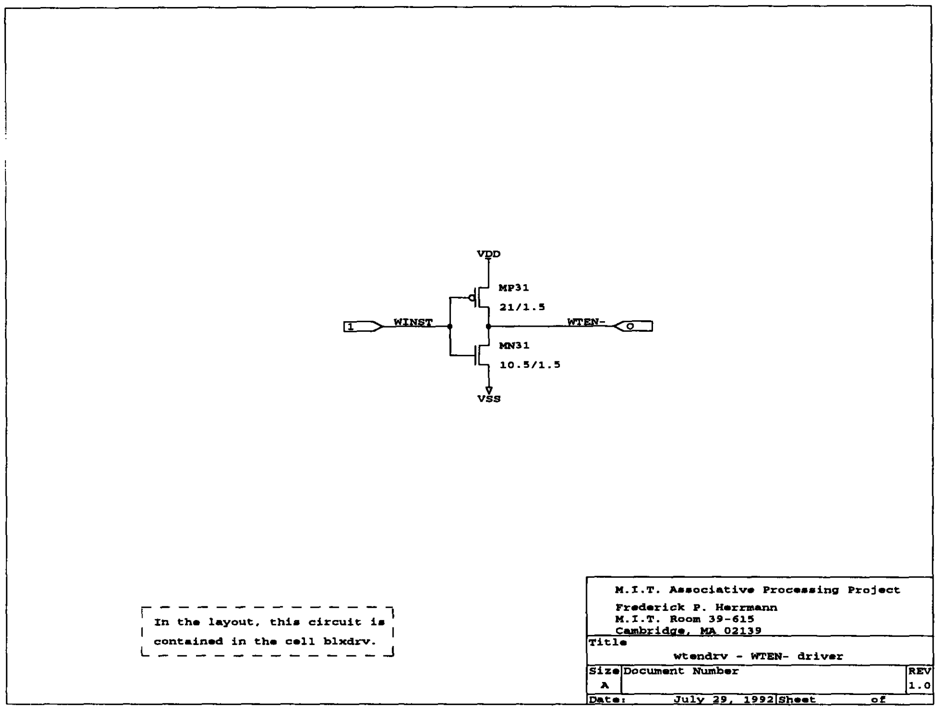


Figure E.35: $\overline{WTEN}$ driver. The signal is used by the trit drivers to control W_T lines.

E.5 Specialized Cell Control

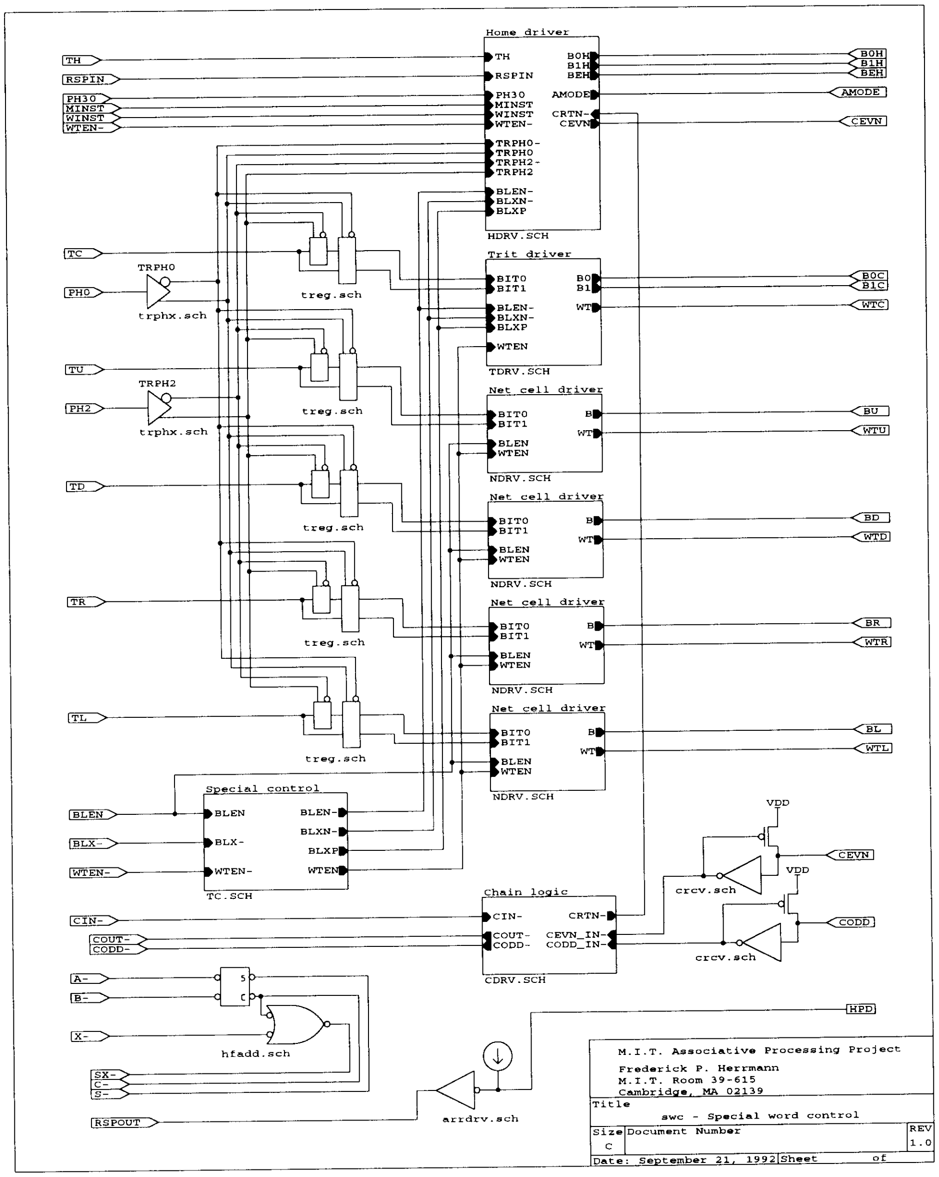


Figure E.36: Specialized cell control.

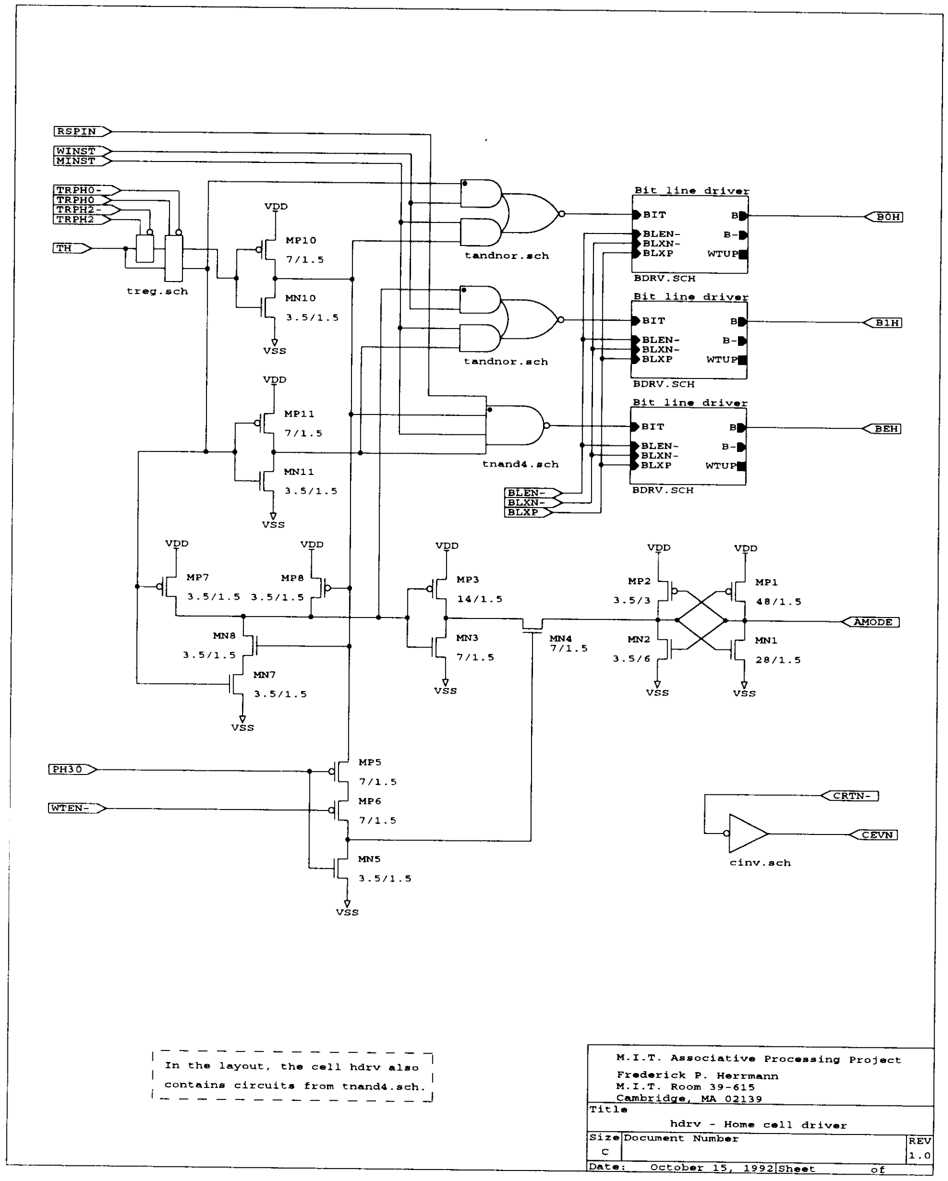


Figure E.37: Home cell driver.

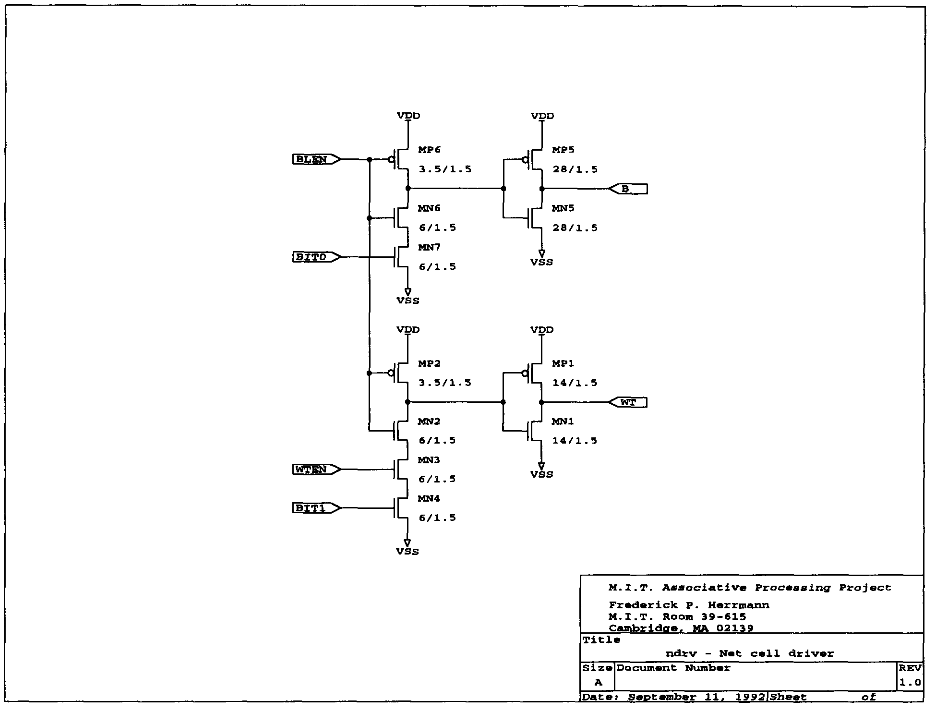


Figure E.38: Network cell driver.

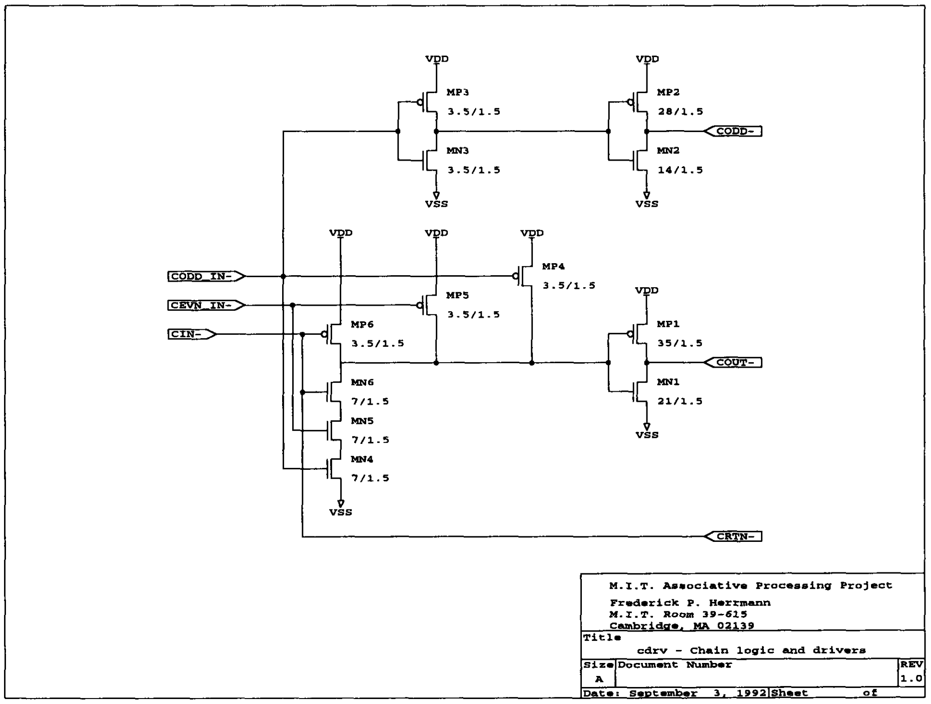


Figure E.39: Responder chain logic and drivers.

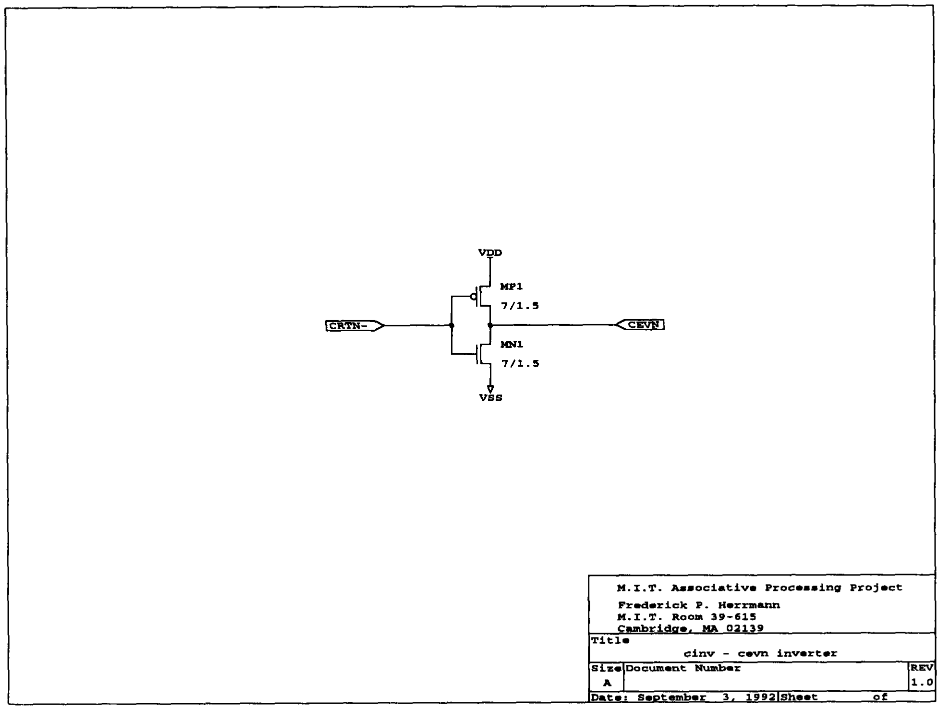


Figure E.40: *CEVN* inverter.

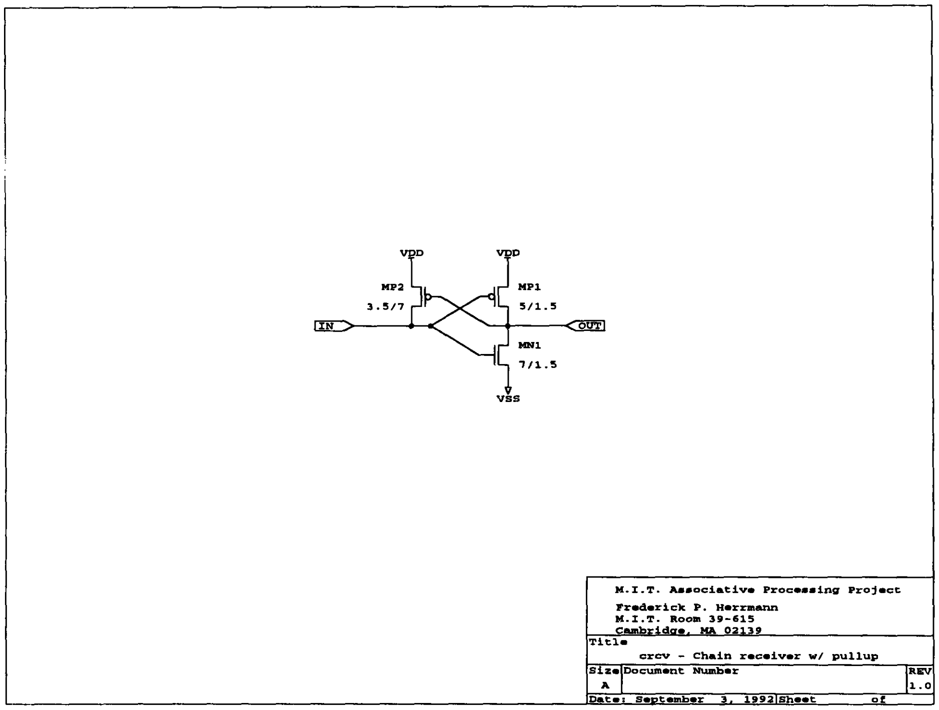


Figure E.41: Chain receiver with pull-up.

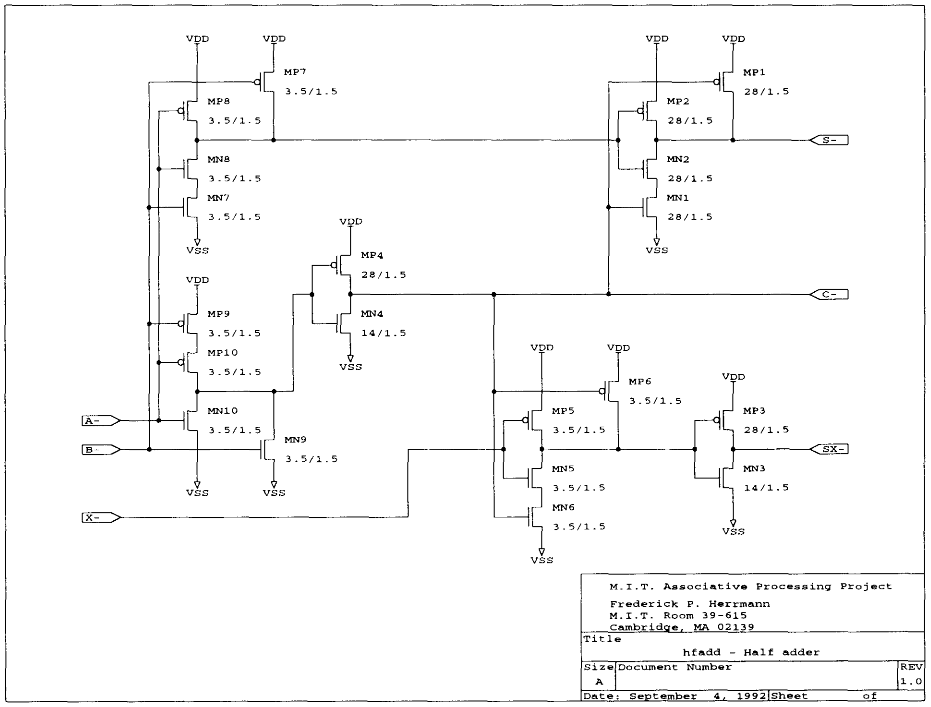


Figure E.42: Half adder, with AND gate. Four of these circuits are used to combine responder count results from the four subarrays.

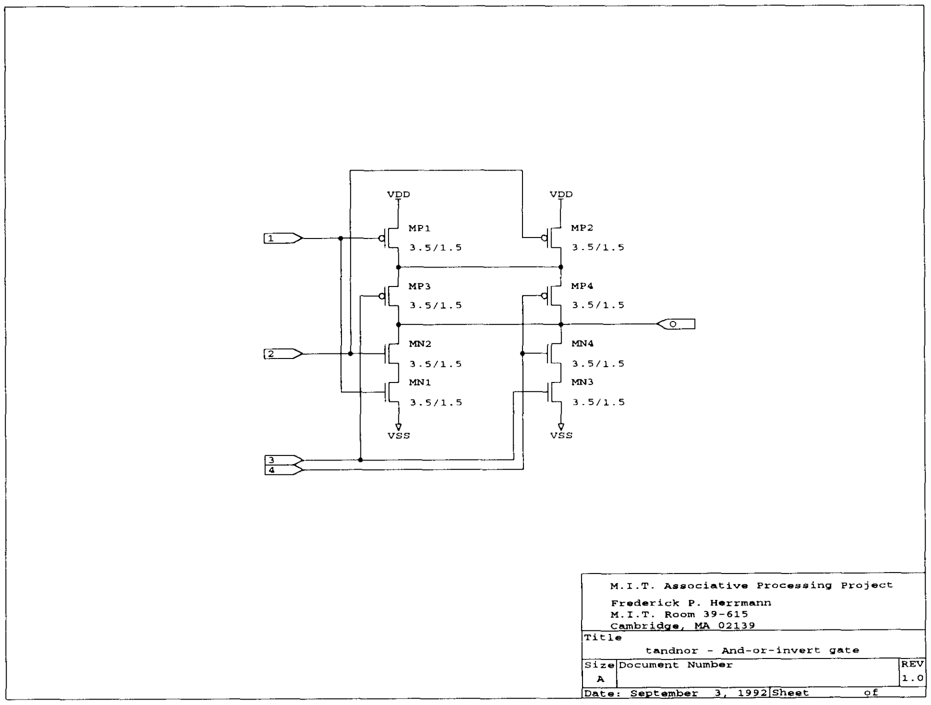


Figure E.43: Two-wide two-input AND-OR-INVERT gate.

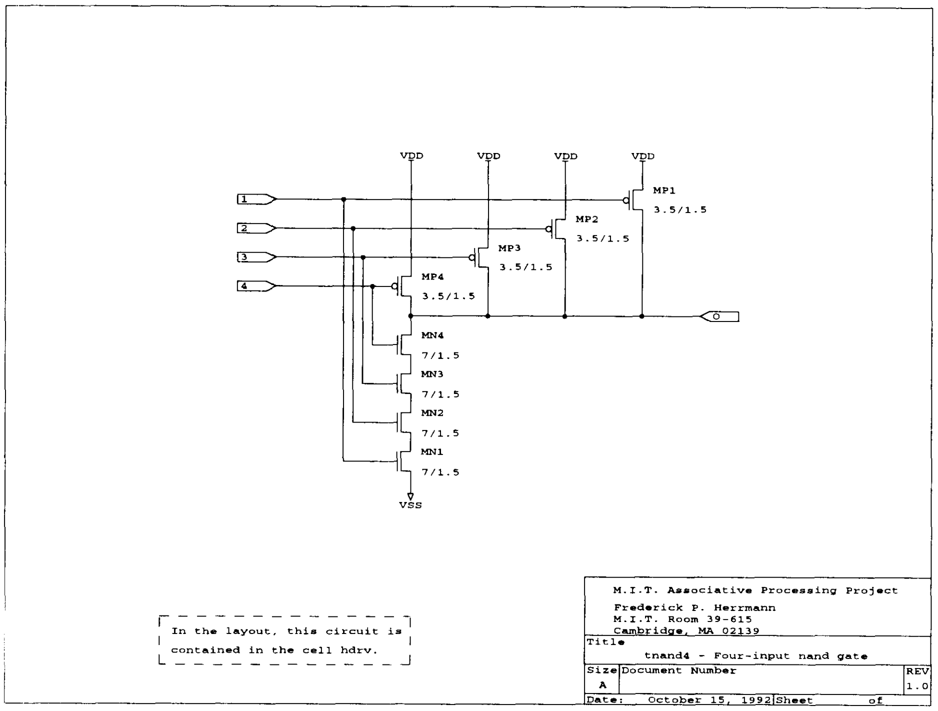


Figure E.44: Four-input NAND gate.

E.6 Pad Circuits

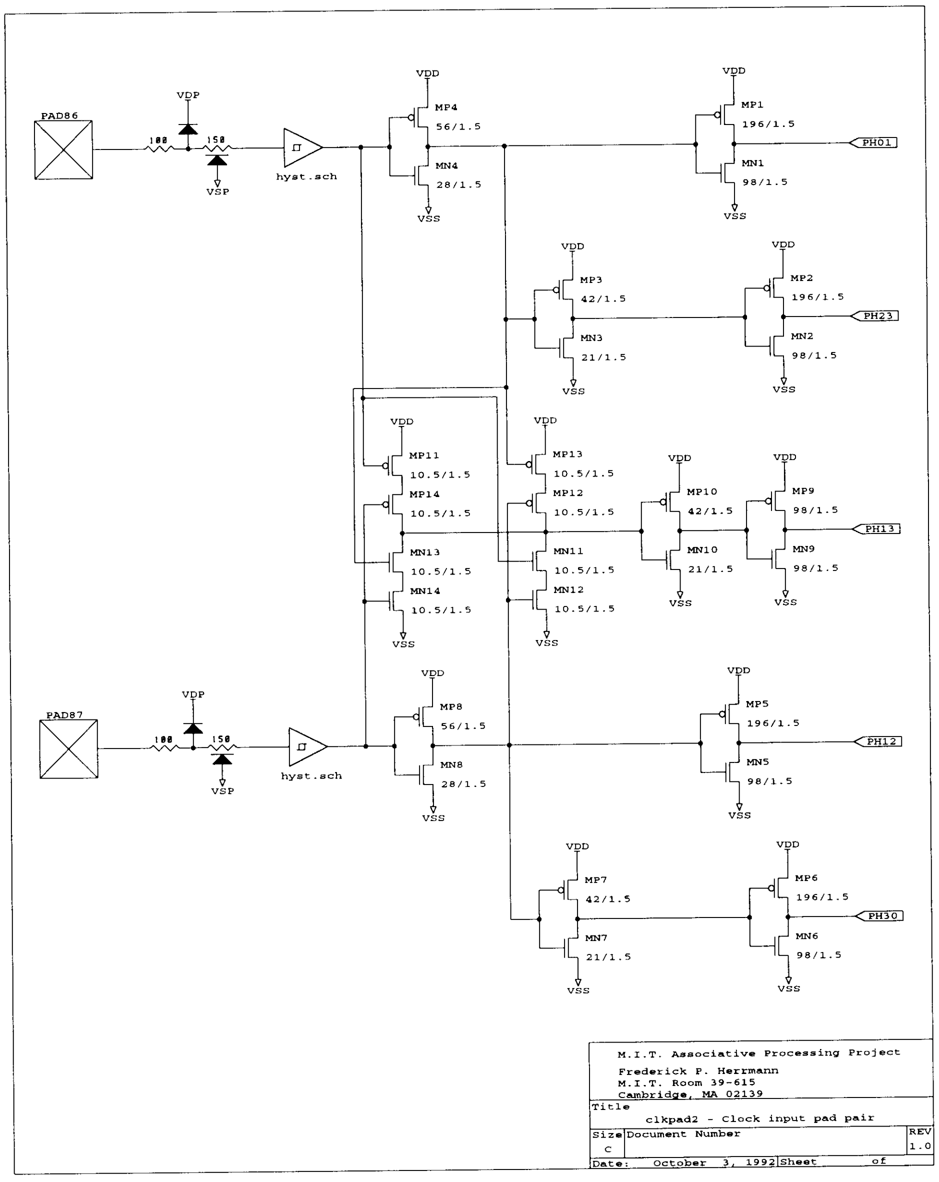


Figure E.45: Clock input pad pair.

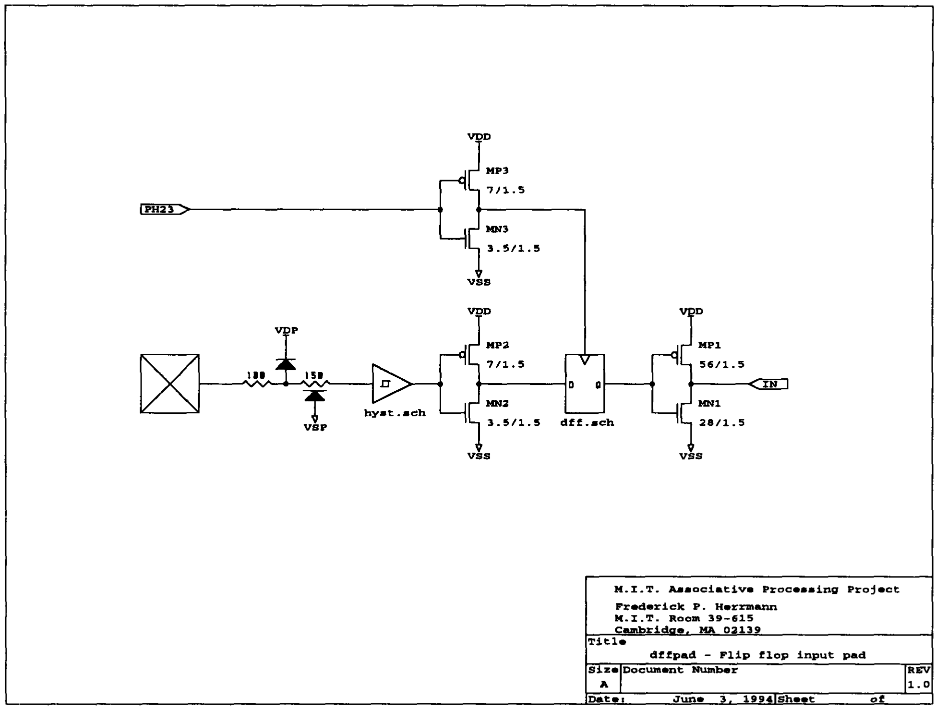


Figure E.46: Flip-flop input pad.

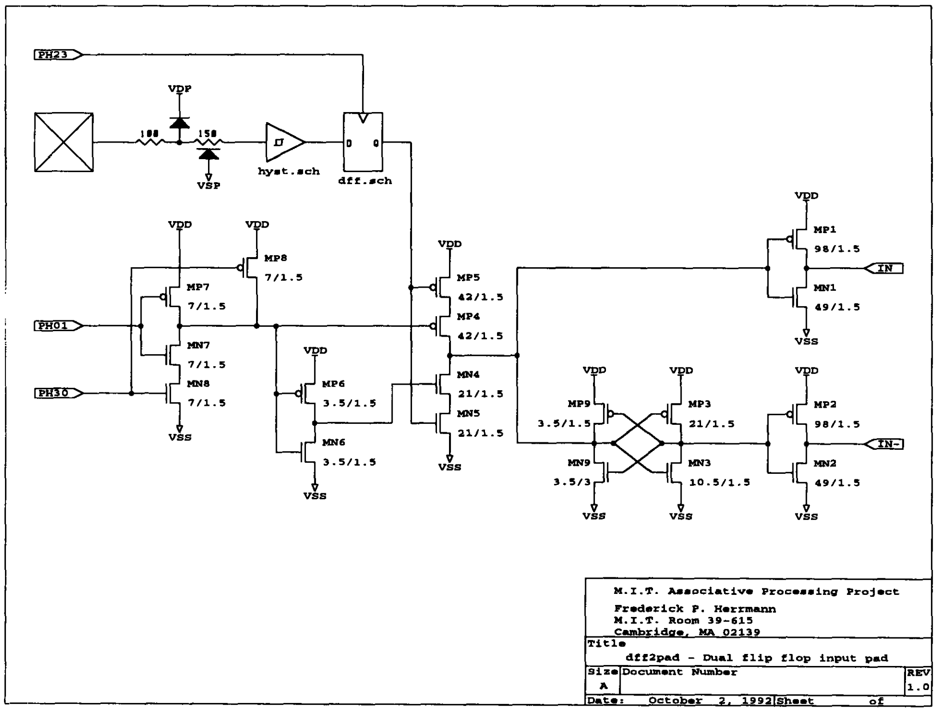


Figure E.47: Dual flip-flop input pad.

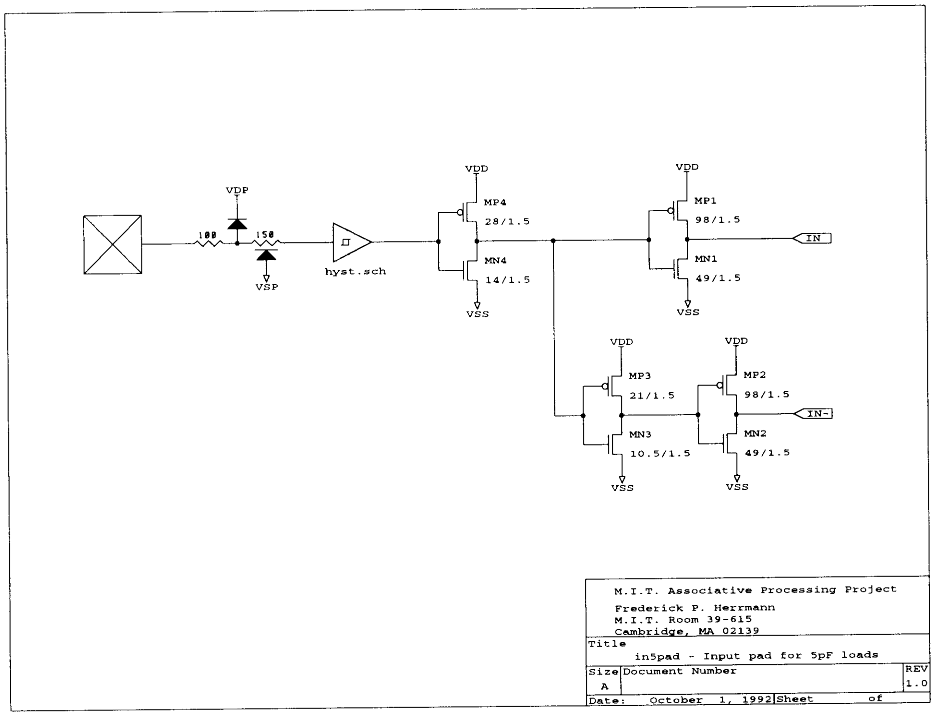


Figure E.48: Input pad for 5-pF load.

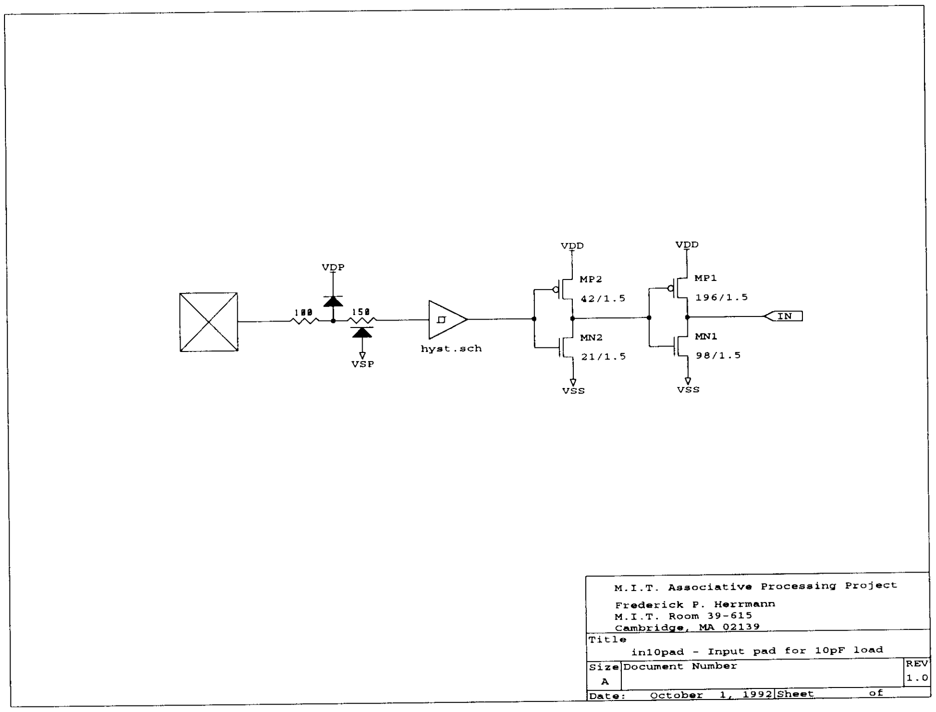


Figure E.49: Input pad for 10-pF load.

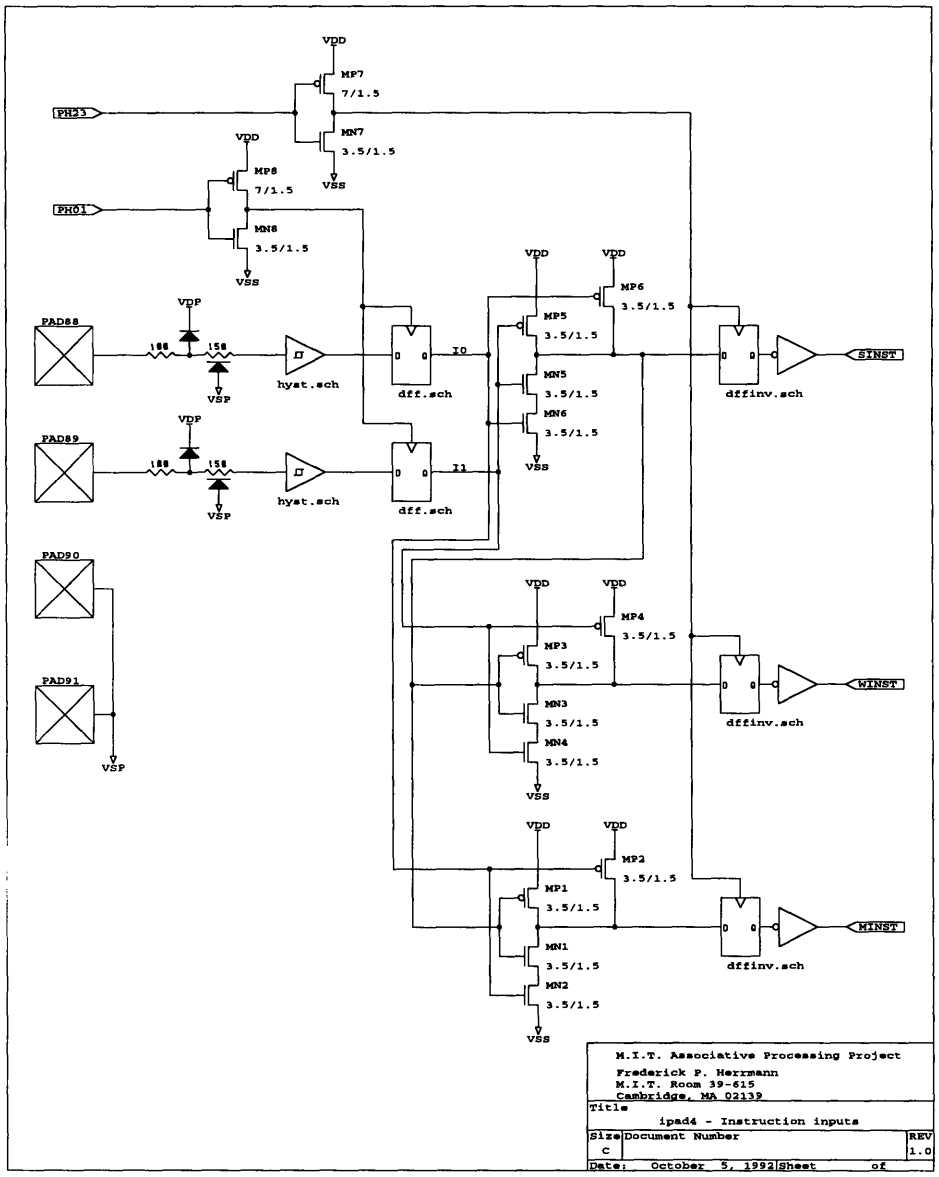
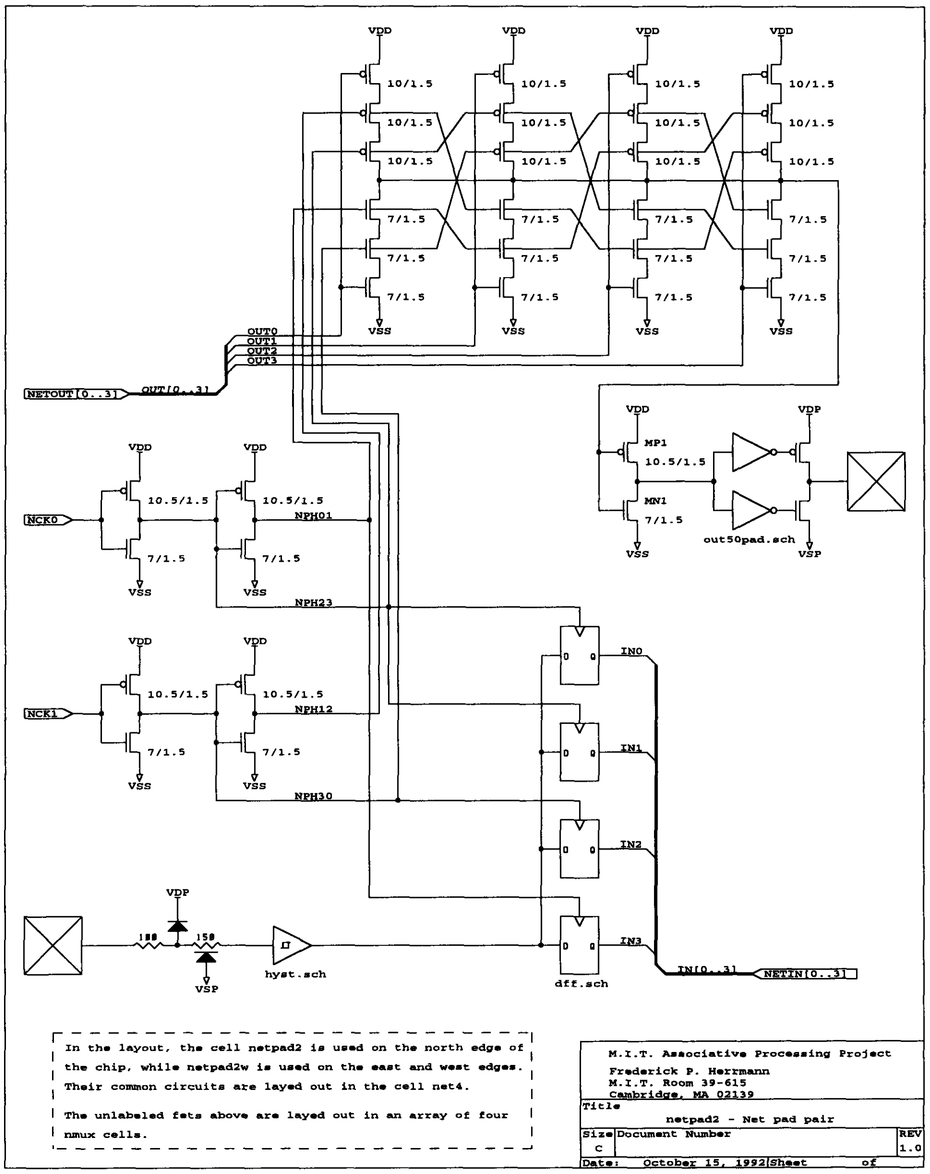


Figure E.50: Instruction input pads. This cell also includes two supply pads, which afford some extra room to accomodate the instruction decode logic.



In the layout, the cell netpad2 is used on the north edge of the chip, while netpad2w is used on the east and west edges. Their common circuits are layed out in the cell net4.

The unlabeled fets above are layed out in an array of four nmux cells.

M.I.T. Associative Processing Project
Frederick P. Herrmann
M.I.T. Room 39-615
Cambridge, MA 02139

Title	netpad2 - Net pad pair	
Size C	Document Number	REV 1.0
Date: October 15, 1992	Sheet of	

Figure E.51: Multiplexed network input and output pads.

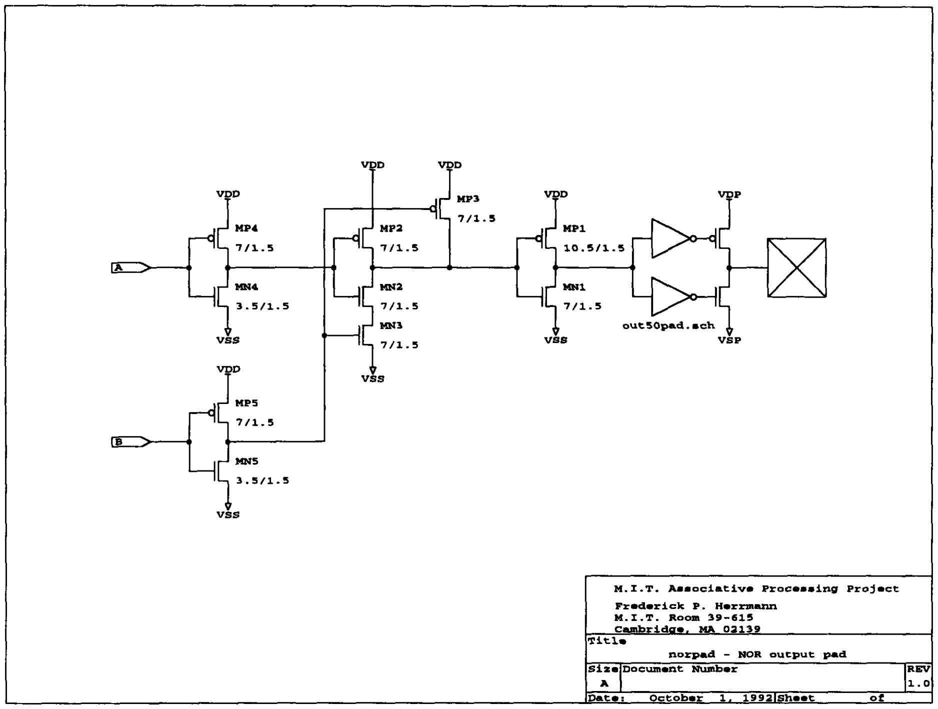


Figure E.52: NOR output pad.

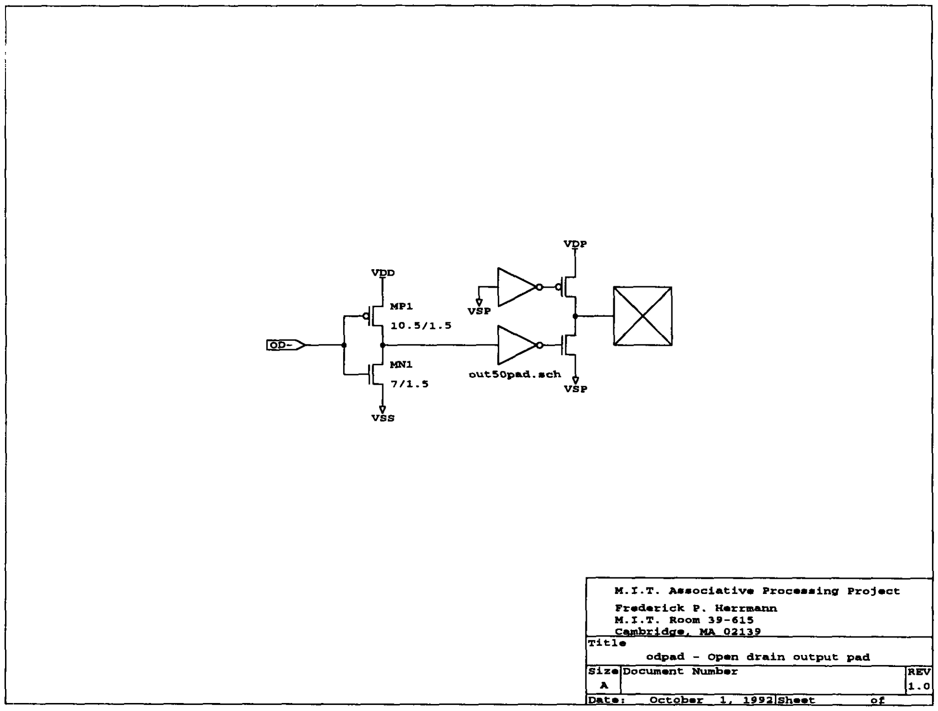


Figure E.53: Open drain output pad.

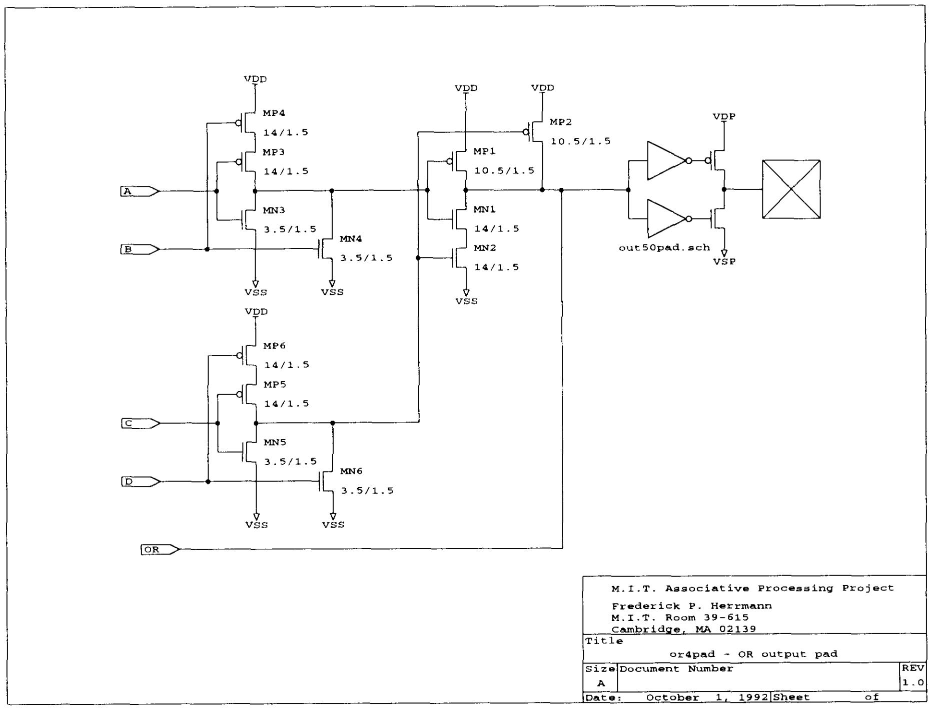


Figure E.54: OR output pad.

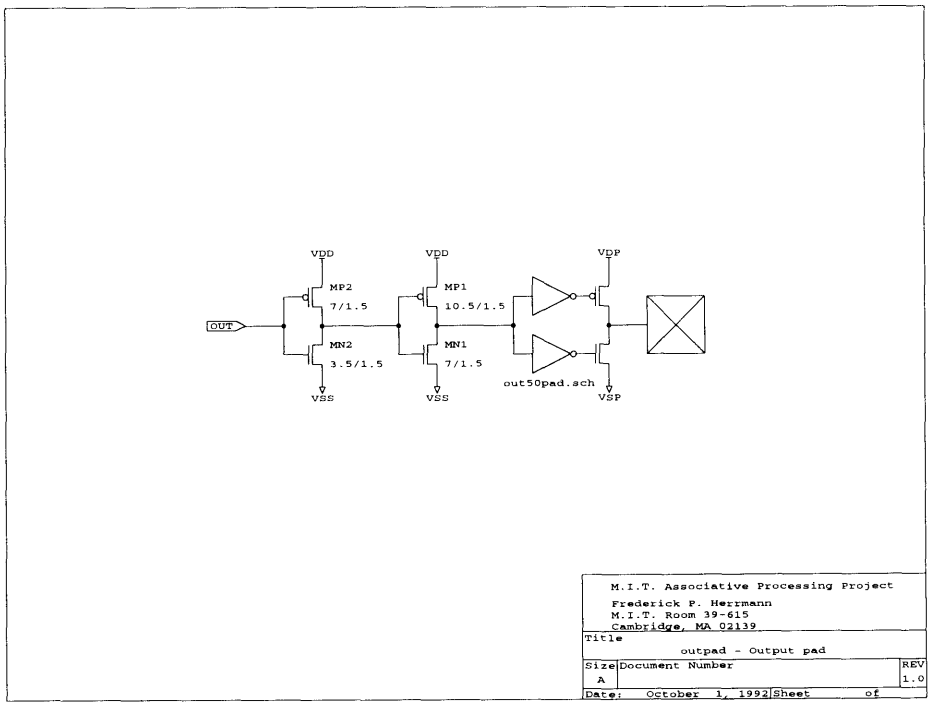


Figure E.55: Output pad.

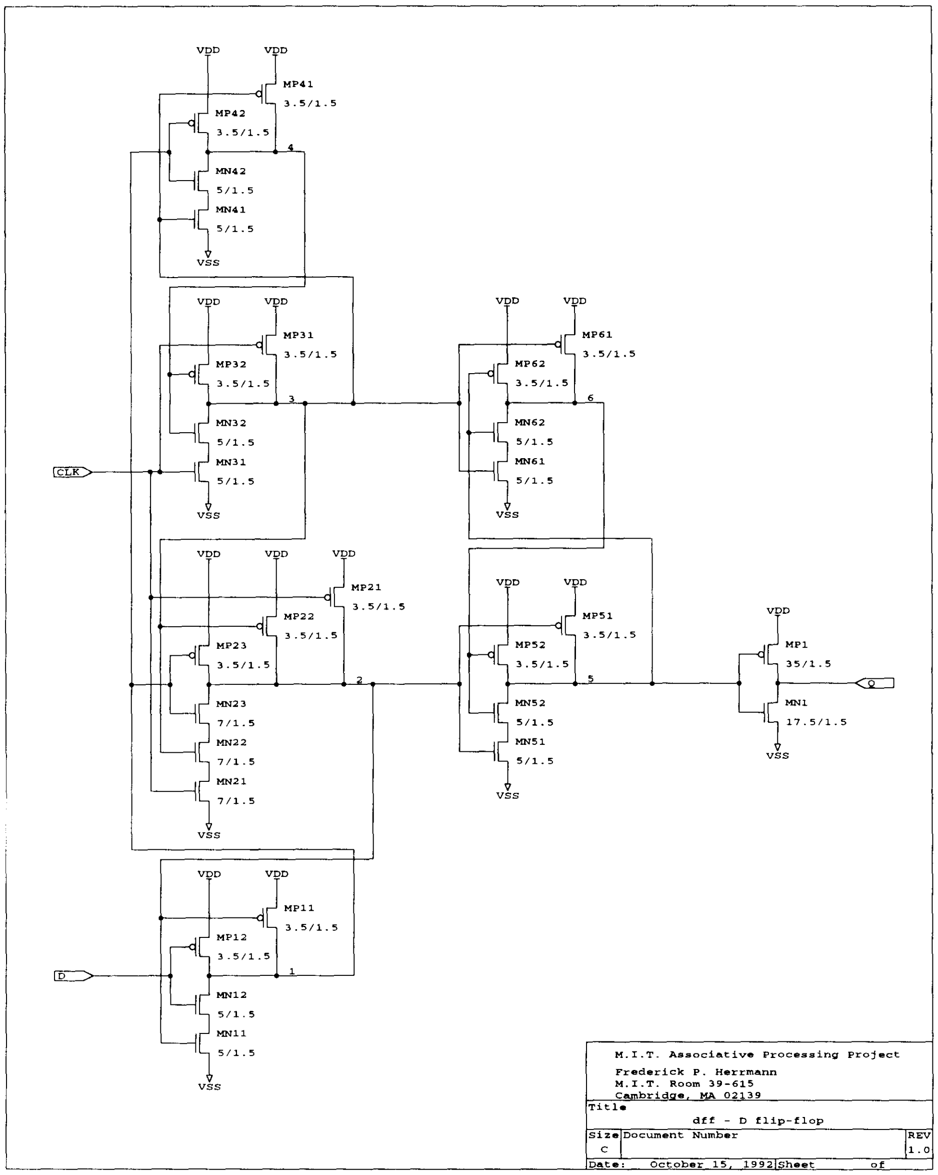


Figure E.56: D flip-flop [111].

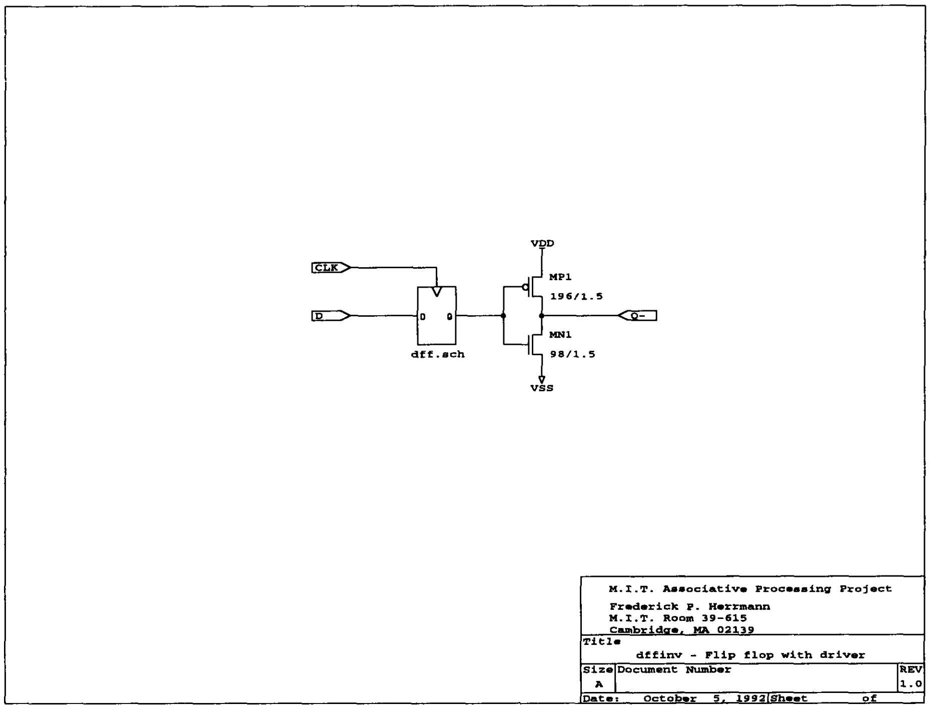


Figure E.57: Flip-flop with inverting driver.

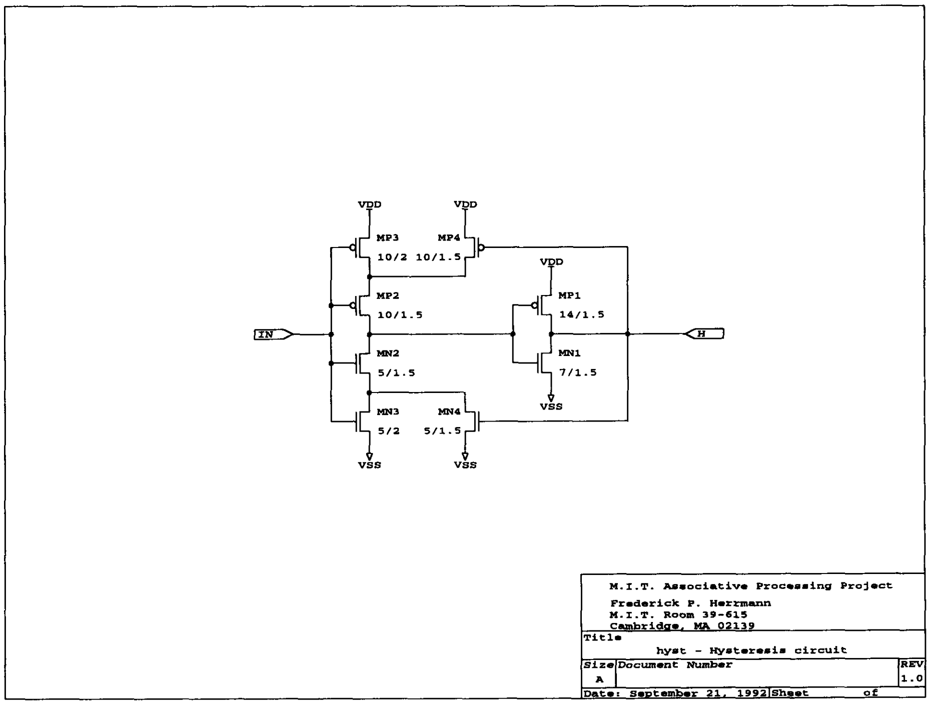


Figure E.58: Input circuit with hysteresis.

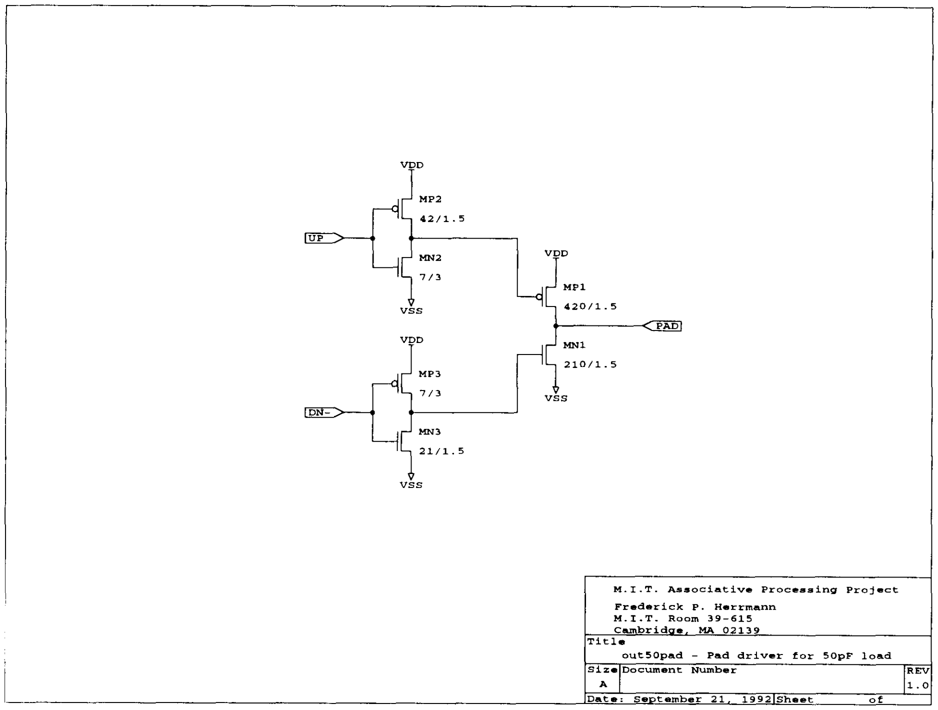


Figure E.59: Pad driver for 50-pF load.

E.7 Image I/O

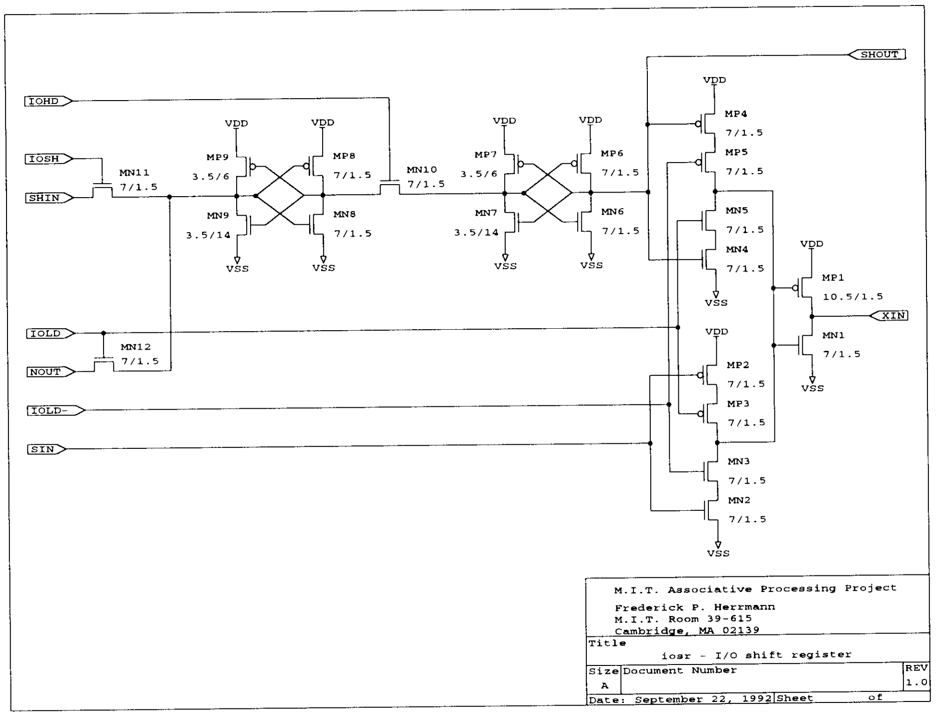


Figure E.60: Image I/O shift register.

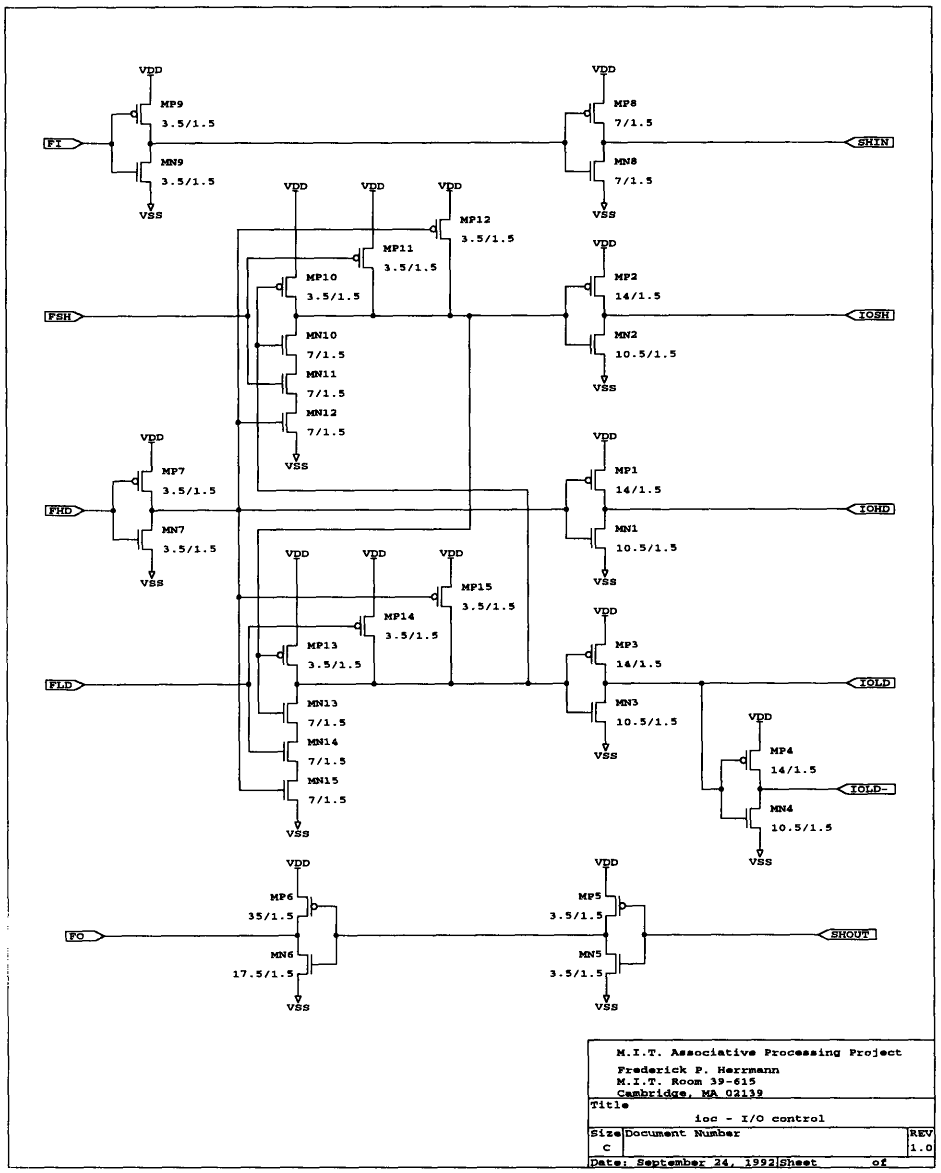


Figure E.61: Image I/O control logic, which ensures that at most one of the signals *IOSH*, *IOHD*, and *IOLD* will be asserted at the same time.

Appendix F

Basic Algorithms

F.1 Refresh

Conventional dynamic RAMs refresh a row at a time, reading a row of data into bit line sense amplifiers and restoring it with a subsequent write operation. The associative processor can not use this method, however, because it does not have bit line sense amplifiers. A column refresh procedure is used instead, as shown in Table F.1. One pair of Match and Write operations is used to refresh each of the three possible states in the second column. The procedure is repeated for every trit cell in the word. Note that all three Matches preserve the contents of the activity register. This property makes it easier to implement an interrupt-driven refresh mechanism, since only the sense amplifier can be modified. The system software automatically determines when each sequence can be interrupted safely.

It is sometimes the case that a column is known to contain only binary data, so that it is not necessary to refresh the X state. Table F.2 gives a three-step procedure, which is also appropriate for the NEWS B network cells. The binary HB cell is static and does not need to be refreshed.

Table F.1
Ternary Column Refresh Procedure

Step	Instruction	Function	Pattern	Comments
1	Match	(AR)	X 1 X X	
2	Write	($\overline{\text{SA}}$)	– 0 – –	Refresh state 0
3	Match	(AR)	X 0 X X	
4	Write	($\overline{\text{SA}}$)	– 1 – –	Refresh state 1
5	Match	(AR)	X – X X	
6	Write	(SA)	– X – –	Refresh state X

Table F.2
Binary Column Refresh Procedure

Step	Instruction	Function	Pattern	Comments
1	Match	(AR)	X X 0 X	
2	Write	(SA)	– – 0 –	Refresh state 0
3	Write	($\overline{\text{SA}}$)	– – 1 –	Refresh state 1

F.2 Arithmetic

The algorithms discussed in this section assume that all of a processing element's memory cells are part of a single word. It is not difficult to change the algorithms to work with the actual chip's half-word organization, but doing so would require handling numerous special cases that would make the presentation here more cumbersome.

Table F.3 presents a procedure for computing the sum $A + B \rightarrow \Sigma$, where A and B are three-bit addends and Σ is the four-bit field receiving the sum. The carry bit is kept in the activity register. Step 1 initializes the sum field with zeros. Steps 2–4 handle the least significant bit, which is the easiest because there is no carry input. Five steps are performed for each of the remaining bits, and the final Write computes the last carry in the function generator and writes it to the most significant bit of the sum.

Table F.3
Non-Destructive Addition Procedure

Step	Instruction	Function	Pattern			SA	AR
			A_2–A_0	B_2–B_0	Σ_3 – Σ_0	Contents	Contents
1	Write	(1)	– – –	– – –	0 0 0 0		
2	Match	—	X X 1	X X X	X X X X	A_0	
3	Match	(SA)	X X X	X X 1	X X X X	B_0	A_0
4	Write	(SA ⊕ AR)	– – –	– – –	– – – 1		
5	Match	(SA ∧ AR)	X 1 X	X X X	X X X X	A_1	C_1
6	Match	(SA ⊕ AR)	X X X	X 1 X	X X X X	B_1	$A_1 \oplus C_1$
7	Write	(SA ⊕ AR)	– – –	– – –	– – 1 –		
8	Match	(SA ∧ AR)	X 1 X	X 0 X	X X 0 X	$A_1 \overline{B_1}\,\overline{\Sigma_1}$	$B_1(A_1 \oplus C_1)$
9	Match	(SA ∨ AR)	X 1 X	X 1 X	X X X X	$A_1 B_1$	T_1
10	Match	(SA ∧ AR)	1 X X	X X X	X X X X	A_2	C_2
11	Match	(SA ⊕ AR)	X X X	1 X X	X X X X	B_2	$A_2 \oplus C_2$
12	Write	(SA ⊕ AR)	– – –	– – –	– 1 – –		
13	Match	(SA ∧ AR)	1 X X	0 X X	X X 0 X	$A_2 \overline{B_2}\,\overline{\Sigma_2}$	$B_2(A_2 \oplus C_2)$
14	Match	(SA ∨ AR)	1 X X	1 X X	X X X X	$A_2 B_2$	T_2
15	Write	(SA ∨ AR)	– – –	– – –	1 – – –		

$T_n = A_n B_n \overline{C_n} \vee A_n \overline{B_n} C_n \vee \overline{A_n} B_n C_n$ $\quad C_0 = 0 \quad$ $C_{n+1} = T_n \vee A_n B_n C_n = T_n \vee A_n B_n$

Table F.4
Scalar Addition Procedure

	First			*Increment*			*Carry*		
Step	Inst.	Function	A_i	Inst.	Function	A_i	Inst.	Function	A_i
1	Match	—	1	Match	( )	1	Match	( )	1
2	Write	(SA)	1	Write	$(\mathrm{SA} \wedge \overline{\mathrm{AR}})$	1	Write	$(\overline{\mathrm{SA}} \wedge \mathrm{AR})$	1
3	Write	$(\overline{\mathrm{SA}})$	0	Write	$(\overline{\mathrm{SA}} \wedge \overline{\mathrm{AR}})$	0	Write	$(\mathrm{SA} \wedge \mathrm{AR})$	0
next		(SA)			$(\mathrm{SA} \vee \mathrm{AR})$			$(\mathrm{SA} \wedge \mathrm{AR})$	

The sequence of instructions needed to perform scalar addition ($A + s \rightarrow A$) depends on the value of the scalar s. If s is a multiple of two, then no work needs to be done on the least significant bit of A. The procedure begins with the least significant 1 bit of s, with the *First* sequence of Table F.4 inverting the corresponding bit of A. Either the *Increment* or *Carry* procedure should be executed for each remaining bit of s. For example, if $s = 26 = 11010_2$, one skips A_0 and performs *First* on A_1, then *Carry* on A_2, *Increment* on A_3, and finally *Increment* again on A_4. The activity register maintains the carry bit, and the last line of the table gives the function that should be used in the next bit's first Match instruction.

Table F.5
Absolute Value Procedure

Step	Instruction	Function	Pattern $A_3 - A_0$	SA Contents	AR Contents
1	Match	—	1 X X 1	A_3A_0	
2	Match	(SA)	1 X 1 X	A_3A_1	A_3A_0
3	Write	$(\mathrm{SA} \wedge \overline{\mathrm{AR}})$	– – 1 –		
4	Write	$(\mathrm{SA} \wedge \mathrm{AR})$	– – 0 –		
5	Match	$(\mathrm{SA} \vee \mathrm{AR})$	1 1 X X	A_3A_2	$A_3(A_1 \vee A_0)$
6	Write	$(\mathrm{SA} \wedge \overline{\mathrm{AR}})$	– 1 – –		
7	Write	$(\mathrm{SA} \wedge \mathrm{AR})$	– 0 – –		
8	Write	$(\mathrm{SA} \vee \mathrm{AR})$	0 – – –		

Table F.5 presents an algorithm for computing the absolute value, $|A| \rightarrow A$. The example procedure operates on a four-bit field. Step 1 identifies odd negative values, with the result A_3A_0 indicating that the more significant bits should be inverted. The rest of the procedure checks the remaining terms, so that bit A_i will be inverted if $A_{\mathrm{MSB}}A_j$ is true for some $j < i$.