LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

A Comparison of Two Signal System Architectures for a Static Dataflow Machine

Computation Structures Group Memo 260
February 1986

David M. Marcovitz

Thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

A Comparison of Two Signal System Architectures for a Static Dataflow Machine

by

David M. Marcovitz

Submitted to the Department of Electrical Engineering and Computer Science on December 5, 1985 in partial fulfillment of the requirements for the Degree of Bachelor of Science.

Abstract

This thesis describes and compares two proposals for the architecture of the signal system of the MIT Static Dataflow Computer. The signal system regulates the flow of instructions to the processing element. The two systems are called the Distributed Control Model and the Central Control Model. The Distributed Control Model divides control into several separate units with one unit on each enable memory chip. The Central Control Model has one central controller that communicates to the enable memory chips across low-bandwidth lines. Since the signal system controller requires a bandwidth greater than the capacity of a single Very Large Scale Integration chip, I recommend the use of Distributed Control Model for the architecture of the signal system. This configuration not only allows greater bandwidth to the processing element, but also allows each signal system controller to be on the same chip as the enable memory. Overcoming the bandwidth constraints of the processing element and the enable memory, with respect to the signal system controller, will allow the MIT Static Dataflow Computer to run at maximum efficiency.

Acknowledgments

I would like to thank my mother, my father, my sister, all four of my grandparents, Eddie Gornish, David Brown, Al Magnani, Joyce Licini, Patti Lodi, Laura Kotovsky, and Andy Boughton for no apparent reason.

I would like to thank Jack Dennis, my thesis advisor, for providing me with the opportunity to do challenging work in the Computation Structures Group of the MIT Laboratory for Computer Science.

I would like to thank Lance Glasser, my undergraduate advisor, for all of his support and encouragement.

I would like to acknowledge the special support made available by the Undergraduate Research Opportunities Program. This program is what makes MIT the best. It provides the facility for undergraduates to do interesting and exciting research such as this thesis.

I would especially like to thank William Beekley Ackerman for all of his advice and support, and for knowing just about everything. Without him, this thesis never would have happened.

Table of Contents

Table of Figures

Table of Tables

Chapter One

The Signal System

1.1 Overview of the MIT Static Dataflow Computer

The Computation Structures Group of the MIT Laboratory for Computer Science is designing a static dataflow computer. This thesis discusses points of the design of this computer in the context of this academic research project.

The proposed MIT Static Dataflow Machine is a massively parallel system. It has many processing elements connected by a large routing network. Each processing element has two major components:

1. a processor
2. a signal system

The processor stores instructions and executes them. The signal system keeps track of the flow of data and tells the processor when a cell (an instruction) is ready to fire (execute).

1.2 Signals

In a dataflow machine, cells fire when their data is ready: execution follows the flow of data. For this to work, the machine must keep track of the flow of data. The MIT Static Dataflow Computer keeps track of data via signals. When a cell finishes firing, it sends signals to other cells to indicate that its data is ready. A cell also sends signals to the cells that produce its data indicating that it is finished with the data.

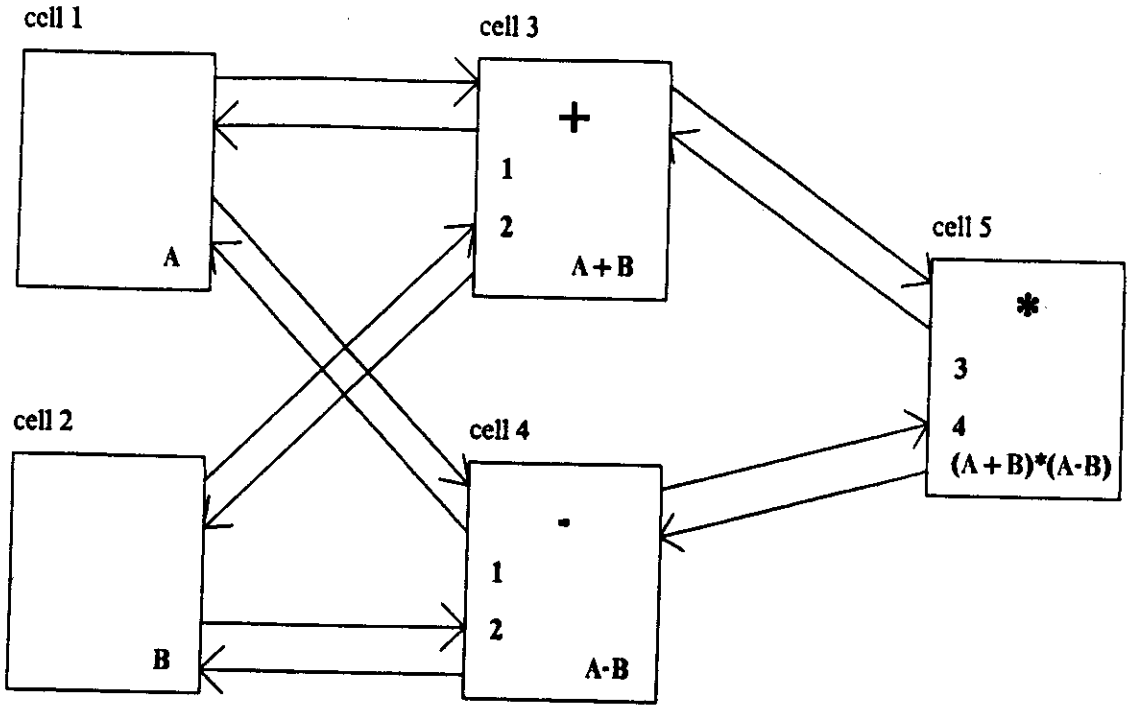


Figure 1-1: Typical Cell Signalling

Figure 1-1 shows a typical interaction of cells. In this figure, arrows represent signals and the numbered boxes represent cells. Within each box, the two numbers on the left represent the cell numbers from which that cell is getting data, and the expression on the right represents the result that the cell produces when it fires. The large symbol in the box represents the operation that the cell performs. For example, if cell 1 produces **A** and cell 2 produces **B**, then cell 5 would produce **(A + B) * (A - B)**. This is the result of the following course of events:

1. cells 1 and 2 fire producing **A** and **B** respectively

2. cell 1 signals cells 3 and 4; cell 2 signals cells 3 and 4

3. cells 3 and 4 each get data from cells 1 and 2 and then fire producing

(A + B) and (A - B) respectively

4. cell 3 signals cells 1, 2, and 5; cell 4 signals cells 1, 2, and 5

5. cell 5 gets data from cells 3 and 4 and then fires producing (A + B) * (A - B)

6. cell 5 signals cells 3 and 4

Signals can also handle conditional expressions. A cell can send signals based on the value of the result of its firing. For example, in an expression of the form:

```
if (a + b) > 0 then (c + d) else (e + f) endif
```

The cell that performs **(a + b)** sends a signal to the cell that performs **(c + d)** if **(a + b)** is greater than zero; it sends a signal to the cell that performs **(e + f)** if **(a + b)** is not greater than zero.[1]

Each cell can send many signals. The group of signals sent by a cell is called a signal list. The signal list is kept in the signal list memory (see Section 1.4.2).

Signals are sent to keep track of what data is ready so that the appropriate cells fire at the appropriate times.

1.3 Communication with the Processor

The signal system and the processor communicate using two buses:

- the done bus
- the fire bus

Figure 1-2 shows the configuration of the processor and the signal system.

---

[1] William B. Ackerman, *Program and Machine Structure for Incredibly Fast Computation*, Computation Structures Group Design Note 14, August 1984.

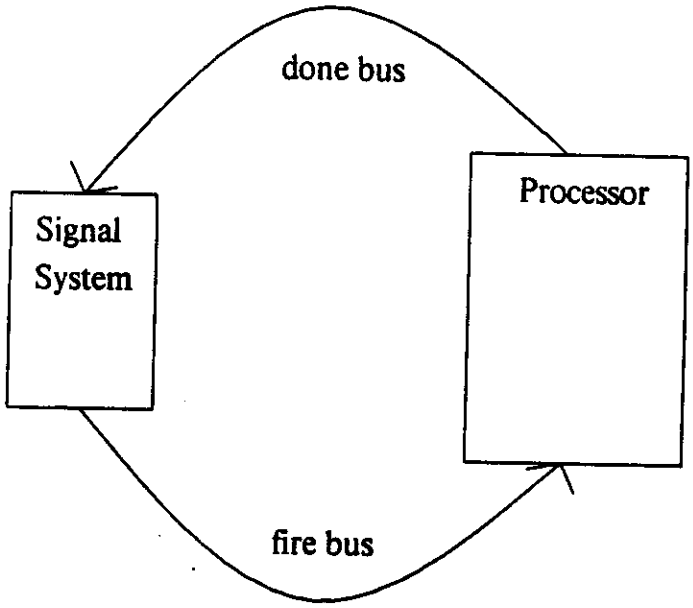


Figure 1-2: The Processor Communicating with the Signal System

When a cell finishes firing, it informs the signal system by sending a done message over the done bus. This message contains two parts:

1. the number of the cell

2. a completion code

The cell number tells the signal system which list of signals to send. The completion code tells the signal system information about the result of firing the cell, for example, the result was greater than zero. The signal system uses this information to send conditional signals as described in Section 1.2.

When the signal system determines that a cell has received enough signals, it sends a fire message over the fire bus telling the processor to fire that cell. This message consists only of a cell number.

When a dataflow program executes, the following processes continually occur:

- the processor fires a cell
- the processor sends a done message to the signal system over the done bus
- the signal system sends signals to cells
- the signal system determines that a cell has received enough signals
- the signal system sends a fire message to the processor over the fire bus

1.4 The Hardware

The signal system has three major parts:

1. the signal system controller [SSC]
2. the signal list memory [SLM]
3. the enable memory [EM]

The SSC controls the signal system, the SLM stores the signal lists for each cell, and the EM stores stores the signal counts for each cell.

1.4.1 Signal System Controller

The SSC receives information from the processor about the cell that has just fired, and from this information, it figures out which cells need to be signalled and tells the EM to signal them. The SSC is a special-purpose sequential computer; it has several programs (signal lists) stored in the SLM. The processor tells it which of these programs to run by telling it: (1) which cell has just fired and (2) some information about the result of firing the cell. SSC programs consist of two basic types of statements:

- jump statements
    1. unconditional
    2. conditional
- signal statements
    1. local
    2. non-local

Jump statements say jump to a different part of the program. Signal statements say tell the EM to signal a cell or tell another EM to signal a cell. When a program is finished, the SSC asks the processor for a new program to run.

The SSC is the heart of the signal unit. The rest of the signal unit is mostly memory. The SSC manipulates that memory to tell the processor which cells to fire.

1.4.2 Signal List Memory

The SLM is the instruction memory for the SSC. It contains the programs that the SSC executes to determine which cells to signal.

1.4.3 Enable Memory

The EM is the memory that keeps track of signal counts. The EM is a grid of memory locations consisting of N rows and M columns. Organizing memory as an N by M array is standard practice in the design of large memories; this organization helps to minimize access time to each memory location. Each EM location stores the current count and the reset count for a cell. Each location in the EM corresponds to one cell in processor memory.

The SSC signals a cell by giving the EM a cell number in the form of row number,

column number. When a cell is signalled:

1. the EM decrements the current count
2. if decrementing causes the current count to become zero, the EM:
    a. tells the processor to fire that cell
    b. resets the current count to the value stored in the reset count

The SSC tells the EM which cells to signal. The EM then determines which cells are ready to fire and tells the processor to fire these cells.

1.4.4 Summary of the Signal System Hardware

The signal system contains three hardware components: SSC, SLM, and EM. These components can be organized in many different ways; each arrangement affects the performance of the machine. The effects of various organizational considerations will be explored in Chapter 3.

1.5 Conclusion

The signal system of the dataflow machine keeps track of and controls the flow of data. The efficiency of the entire machine depends upon the signal system. The processor speed is limited by the speed of the signal system; the processor can only execute instructions as fast as the signal system can tell it that there are instructions to execute. If the signal system is slow, the entire machine is limited by this slowness.

Chapter Two

The Two Models of the Signal System

There are currently two proposals for the architecture of the signal system:

- the Distributed Control Model
- the Central Control Model

The first proposal, the Distributed Control Model, contains several signal units. Each signal unit has a local enable memory, a signal system controller, and a signal list memory. The signal units communicate with each other via a non-local signal bus and communicate with the processor via the fire bus and the done bus. The second proposal, the Central Control Model, contains one central signal system controller. This signal system controller is attached to several enable memories and several signal list memories. The controller communicates with the memories via private lines for each memory. It also communicates with the processor via the fire bus and done bus.

The Central Control Model proposes an interleaved system in which the controller has two control units, subSSCs (see Section 2.2.1), for each signal list memory. Each subSSC accesses the signal list memory on alternate cycles. The use or non-use of interleaving is an issue separate from other issues in the proposals. Either proposal could just as easily use interleaving. Therefore, for simplicity of discussion, I will compare the two proposals without interleaving.

The two signal system proposals are described in detail in this chapter; the two proposals are compared in Chapter 3.

2.1 The Distributed Control Model

The Distributed Control Model for the signal system (see Figure 2-1) allows for several signal units for each processing element. Each signal unit has three major parts:

1. a signal system controller [SSC]
2. a signal list memory [SLM]
3. an enable memory [EM]

The EM and the SSC are on one custom chip, and the SLM is on one or more commercial chips.

Aside from the done bus and the fire bus (see Section 1.3), the only bus associated with the signal system is the non-local signal bus. Signal units talk to each other using this bus.

2.1.1 Signal System Controller

The function of the SSC is described in Section 1.4.1. The SSC is on the same fully custom designed chip as the EM to allow a high-bandwidth path between them. This high-bandwidth path allows the SSC to execute signal statements that tell the EM to signal more than one cell at a time.

2.1.2 Signal List Memory

The SLM is the instruction memory for the SSC. It consists of one or more commercial chips arranged to minimize access time. Each signal unit has its own SLM which talks directly and exclusively to the SSC. A detailed description of the programs stored in the Distributed Control Model of the SLM can be found in Appendix A.

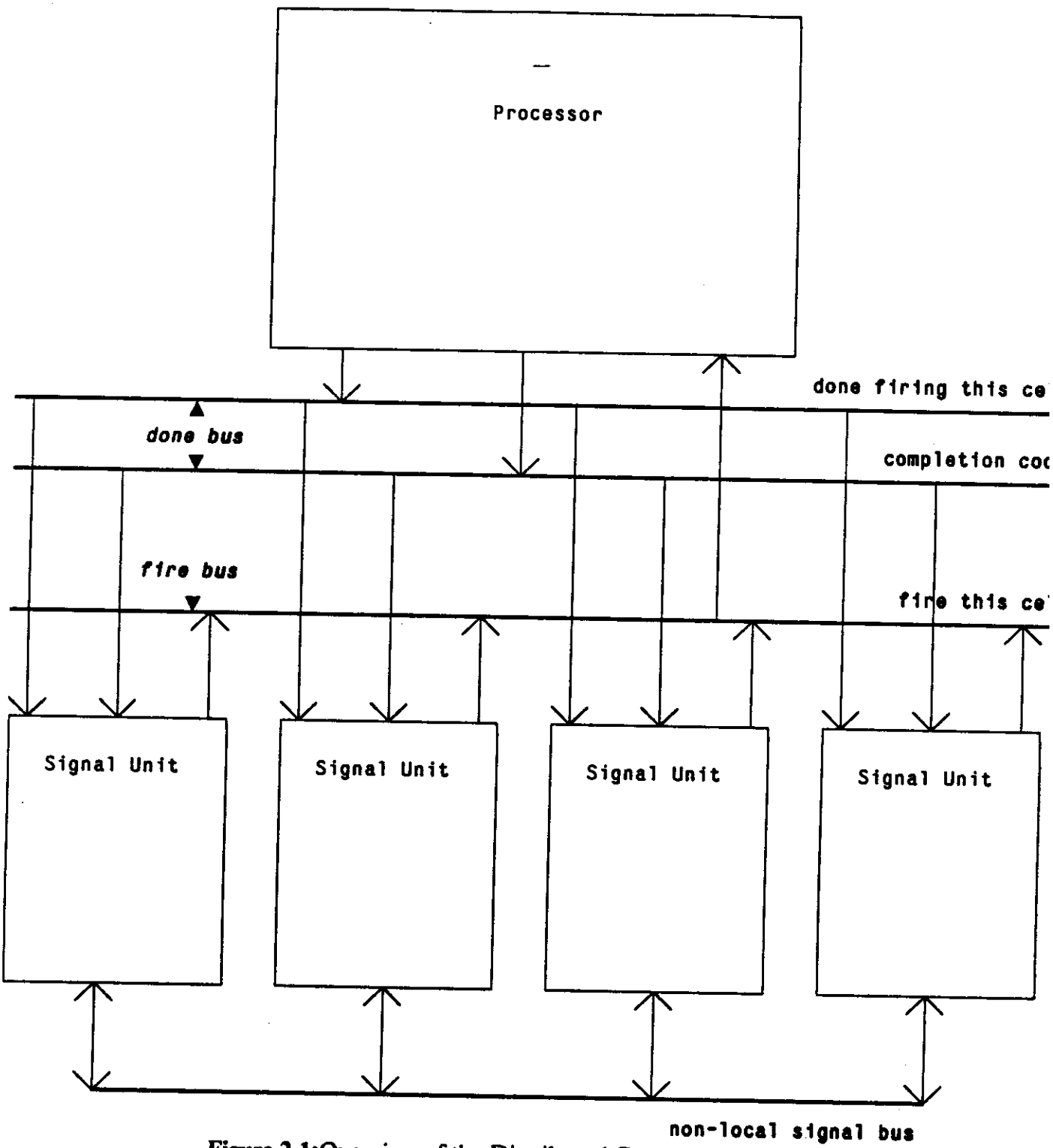


Figure 2-1: Overview of the Distributed Control Model

2.1.3 Enable Memory

The function of the EM is described in Section 1.4.3. A special feature of the Distributed Control Model is that at one time, the SSC can signal one, two, or three cells in the same row by formatting signals in the form row number, column number, [column number], [column number]. This added feature is possible because the EM and the SSC are on one custom chip allowing the path between them to have high bandwidth.

2.1.4 Non-local Signal Bus

The non-local signal bus connects the signal units in a signal system. One signal unit sends signals to another signal unit, i.e., non-locally, over this bus. Some of the signal instructions are non-local and say signal a cell in a different signal unit. The SSC executes these instructions by sending the signal through the non-local signal bus. This bus is the means of communication used between signal units.

2.1.5 Conclusion

The main features of the Distributed Control Model are:

- several separate and identical signal units
- one EM/SSC custom chip per signal unit
- one SLM per signal unit made of one or more fast commercial chips
- a high-bandwidth path between the SSC and the EM
- a non-local signal bus carrying information between signal units

When the processor sends information over the done bus, one of the signal units performs the following operations:

- picks up the information with its SSC

- runs the correct program contained in the SLM
- sends signals
    - to its EM through a high-bandwidth path
    - to other EMs through the non-local signal bus
- sends fire messages to the processor over the fire bus

This cycle is repeated continuously throughout the execution of a dataflow program.

2.2 The Central Control Model

The Central Control Model for the signal system (see Figure 2-2) has three major parts:

1. one central signal system controller [SSC]
2. several signal list memories [SLM]
3. several enable memories [EM]

The SSC is a single chip using gate array technology. Each EM is a single custom chip attached to the SSC. The SLMs are each one or more fast commercial chips.

2.2.1 Signal System Controller

The SSC acts like several SSCs (as described in Section 1.4.1) put together on one chip. Each of these subSSCs is dedicated to one EM and one SLM. Just as in the Distributed Control Model, each subSSC runs a program from the SLM.

The paths between the subSSCs and the EMs are narrow because they are on different chips. The number of EMs per processor is limited by the number of pins on the SSC chip. Pin count is also a limiting factor in the amount of information that can pass from the SSC to the EMs. This limits the feasibility of multiple signal

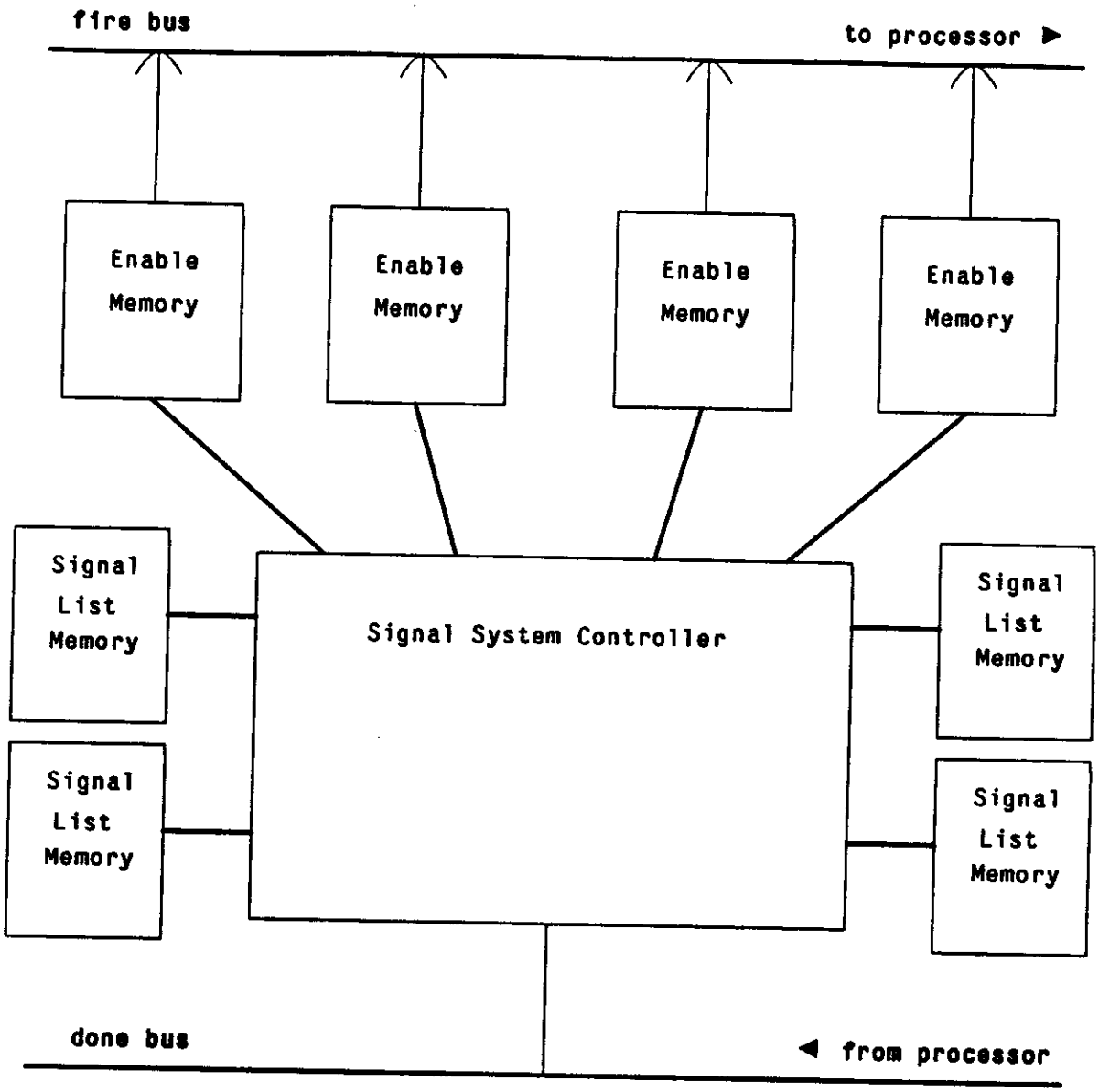


Figure 2-2: Overview of the Central Control Model

statements (see Section 2.1.3), i.e. single statements that send more than one signal to an EM.

2.2.2 Signal List Memory

There are several SLMs associated with the SSC. Each subSSC has its programs stored in one SLM. As in the Distributed Control Model (see Section 2.1.2), each SLM consists of one or more commercial chips arranged to minimize access time.

2.2.3 Enable Memory

The Central Control Model of the EM is almost identical to that of the Distributed Control Model (see Section 2.1.3). There are two major differences in the Central Control Model:

1. the EM is on a separate chip from the SSC
2. the EM cannot signal more than one cell in a single cycle

The SSC and the EM cannot be on the same chip because they use different technologies: the SSC uses gate array technology and the EM is a fully custom design. Because of pin limitations on the SSC chip, the path between the SSC and each EM cannot have high bandwidth. The bandwidth of this path is lower than the bandwidth necessary to signal more than one cell per cycle.

2.2.4 Conclusion

The main features of the Central Control Model are:

- one central SSC chip made of gate array technology
- several custom EM chips attached to the SSC via low-bandwidth paths
- several SLMs per SSC; each SLM made of one or more fast commercial chips

When the processor sends information over the done bus, the SSC picks up the information and sends it to the correct subSSC. The subSSC:

- runs the correct program contained in the SLM
- sends signals
    - through a low-bandwidth path to its EM
    - through a high-bandwidth path to another subSSC which in turn sends the signals to its EM through a low-bandwidth path

When a cell in an EM has received enough signals, the EM sends a fire message to the processor over the fire bus. This cycle is continuously repeated throughout the execution of a dataflow program.

Chapter Three

Comparison of the Signal System Proposals

In Chapter 2, I described both the Distributed Control Model and the Central Control Model of the signal system. In this chapter I will discuss the merits and drawbacks of each proposal in the context of the MIT Static Dataflow Project.

3.1 Considerations

There are several criteria I have used to evaluate the suitability of each proposal for use in the dataflow machine. The major criteria are:

- complexity
- design time
- operating speed
- technological feasibility
- ability to be improved with time

All of these criteria are important in designing a signal system architecture. However, for the purposes of an academic research project, some are more important than others.

It is difficult to quantify many of the tradeoffs in the signal system design. Complexity, for example, is difficult to define, and design time cannot be accurately determined until the system is designed. Furthermore, it is difficult to assess the relative importance of each criterion to the overall design.

3.1.1 Complexity

There are a number of reasons why simplicity is a goal. Many of these reasons are closely related to the other criteria of suitability. A simple system is generally easier to design, making the design time shorter. As the needs of the computer change over time, a simpler system can adapt more easily.

As an academic project, too much complexity is bad because less time is available for investigating important issues. Complicated additions increase the speed of the machine, but often speed is not the most important factor. The research time should be spent exploring issues about which less information is available.

I will discuss complexity only in the context of the other criteria of suitability and not as an independent criterion.

3.1.2 Design Time

As a commercial venture, design time is important. Every day spent on the design is another day the company is spending money without any income. The Dataflow Computer must be completed within approximately two years for it to be a successful commercial venture.

As an academic or research oriented project, design time is far less important. There is not intense competition in academia to be the first to produce a static dataflow computer. As a research project, more time can be spent considering tradeoffs, such as operating speed versus complexity; more time can also be spent carefully designing prototypes that will later develop into a marketable computer.

3.1.3 Operating Speed

The purpose of this static dataflow project is to design a fast scientific computer, faster than currently available supercomputers. The speed of the entire computer is bounded by the speed of the signal system. No cells can execute until the signal system determines that it is time for them to execute. If the signal system is slow in its job, the computer will be slow.

The Computation Structures Group has determined target goals for the operating speed of the machine. We would like all components to run on a common fifty nanosecond clock. For the machine to be operating at maximum capacity, in each processor, one cell should begin firing every clock period. The average number of signals that each cell needs to fire is approximately four. This means that if the signal unit (Distributed Control Model) or subSSC (Central Control Model) can produce an average of one signal every cycle, the signal system must consist of approximately four signal units/subSSCs per processor.

3.1.4 Technological Feasibility

A working prototype is an important goal of this project. Any proposal for the signal system must be feasible given currently available technology. The main technological considerations that could present a problem are:

Pin Count

Currently, the best packaging technology available to us allows for chips with up to approximately 185 pins. As of May 1983, Fujitsu offered gate array chips with a maximum of 160 pins. As of April 1984, LSI Logic offered gate array chips with a maximum of 180 pins. As packaging technology improves, the number of available pins increases, but these increases are not large and cannot be used in designing a chip today.

Custom Chip Speed

Presently, the fastest custom chips that we can design have clock speeds of 5 megaHertz (200 nanoseconds). It

	is possible for us to make faster chips, but that will take several years of tedious optimization for each chip.
Custom Chip Reliability	It is difficult to make working custom chips. This problem is compounded by the lack of high quality design and simulation tools at MIT.
On Chip Density	It is hard to quantify how much functionality can be placed on a single chip. There are limits, for example, in a 1.2 micron process, which will be available from MOSIS soon, it should be possible to fit 2K of enable memory and some added control circuitry on a single chip.

It is important to understand these considerations before undertaking a design. A design can be fantastic, except that it uses 250 pins on a chip, making it useless to us until we have access to such technology.

3.1.5 Ability To Be Improved with Time

The first design of the signal system will not be perfect. Most of the "absolute" numbers on which it will be based are guesses. For example, no one really knows the average number of signals per cell firing or the number of words of signal list memory each cell will need. Because of this, it is best to design a system that can easily be adapted to the changing ideas of what the needs of the machine are.

3.2 Criteria of Comparison for the Distributed Control Model

3.2.1 Design Time

The Distributed Control Model uses custom chips which take a long time to design. It should be possible to design a working prototype within one year. Speed improvements could take much longer.

3.2.2 Operating Speed

In order to keep the processor busy, the signal system must send one fire message every cycle. If N is the number of signal units and K is the average number of signals per cell, each signal unit must average K/N signals per cycle to keep the processor busy.

The Distributed Control Model reads one word from signal list memory every cycle. A word of signal list memory can signal zero, one, two, or three cells. (Using the dispatch command, a word of signal list memory can signal up to sixteen cells, but the use of this command is not expected to be common enough to affect this discussion.). Since most cells will be signalling other cells which are close, many cells will require just one word of signal list memory to do their signalling. This means that the number of cells signalled each cycle can be greater than one. This is important because the best available figures from the compiler[2] show that on the average, each cell needs to send slightly more than four signals per firing, i.e. $K = 4 + \varepsilon$. With four signal units and $4 + \varepsilon$ signals per cell firing, each signal unit must produce an average of $1 + \varepsilon/4$ signals per cycle to keep the processor busy.

These figures indicate two possible methods of using the Distributed Control Model of the signal system:

1. Four signal units which rely on the compiler to produce code in which cells come in blocks of nearby cells that signal each other. By locality of signalling the average number of signals per cycle can be greater than $1 + \varepsilon/4$.

2. Eliminate the need for each signal unit to rely heavily on locality by using more than four signal units.

Both of these configurations meet the operating speed specifications.

---

[2]Charles Goldman and William B. Ackerman, Private Communication, August 1985.

Most programs come with a great deal of locality. The entire computer depends on varying degrees of locality to divide the program:

- into chunks of 8K cells among processors
- into chunks of 2K cells among EMs
- into chunks of 32 cells among rows of the EM

The signal system can use this last form of locality to become faster.

Since each signal unit is identical, adding more signal units is easy. The only change required in the design is in the bus arbitration.

3.2.3 Technological Feasibility

There are no pin problems in the Distributed Control Model. The Distributed Control Model (see Figure 3-1) requires:

- 15 pins to the fire bus
- 29 pins to the SLM: 13 pins for the address and 16 pins for the data
- 20 pins to the done bus: 13 pins for the cell number, 5 pins for the completion code, and 2 pins for handshaking
- 15 pins to the non-local signal bus
- 2 pins for the two clock phases
- 2 pins for power and ground

This adds up to 83 pins which is possible using current technology.

Many people doubt whether it is feasible for a university to design custom chips. The Computation Structures Group has proven that it is not only feasible, but also relatively easy to design custom chips. A prototype dataflow processor is presently

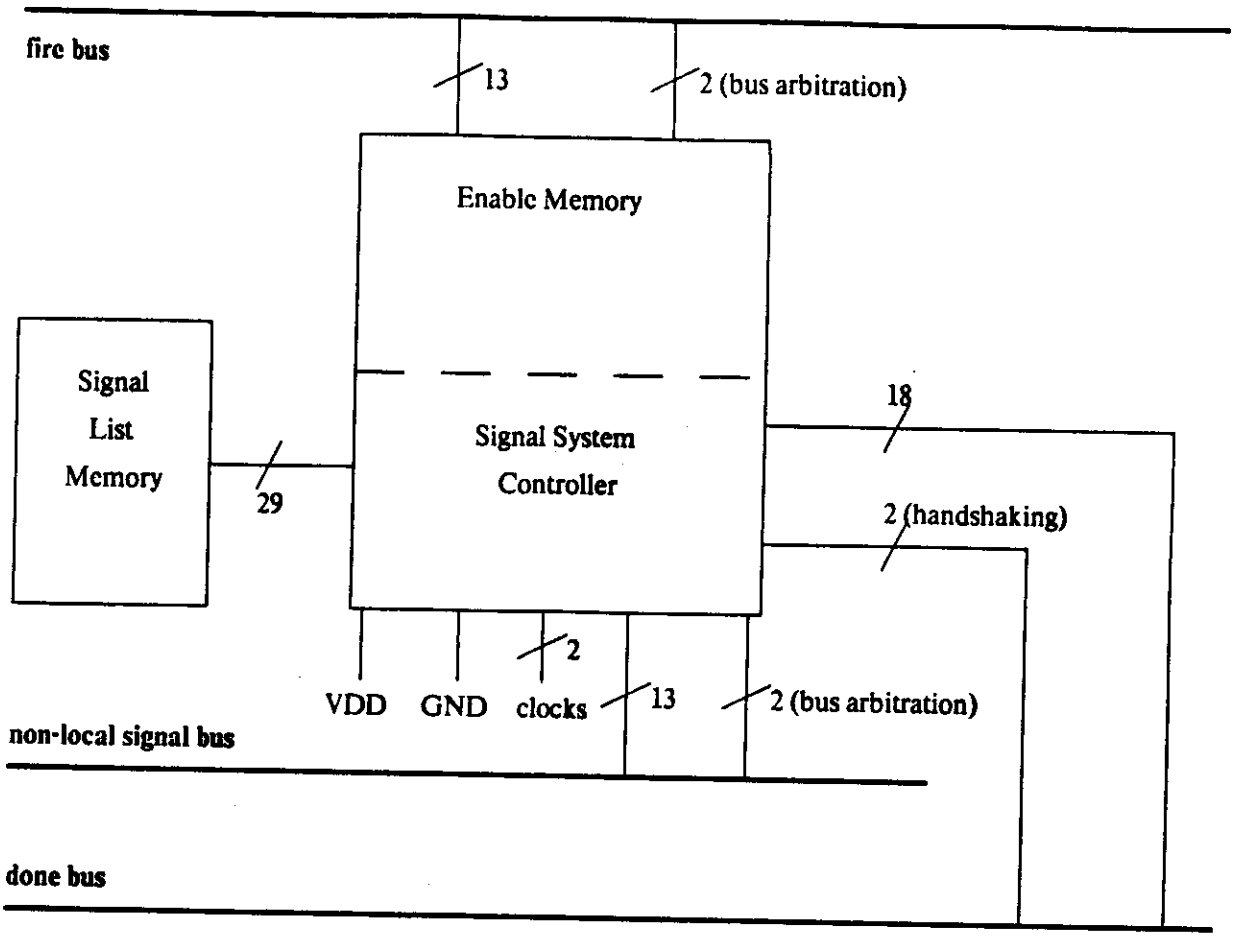


Figure 3-1: Pin Counts for the Distributed Control Model SSC

running in Dr. William Ackerman's office. This entire custom design was completed by Dr. Ackerman on the MIT campus. A 2 X 2 Router chip for the routing network was also designed by Tam-Anh Chu. A first prototype of the fully custom Enable Memory chip was designed at MIT by Gao Guang Rong and Kevin Theobald. At least one EM chip was fabricated, tested, and shown to work.[3] It is clear that the design of fully custom chips is technologically feasible, even in an academic environment.

[3]Elmer Lyons, Computation Structures Group Design Meeting, August, 1985.

3.2.4 Ability To Be Improved with Time

The numbers used for operating speed and compiler output are approximations. It is possible that the figures quoted in Section 3.2.2 are wrong by a factor of two in either direction. This presents no problem for the Distributed Control Model. A factor of two increase in the number of signals required per cell simply means a factor of two increase in the number of signal units required. This expandability is important because of the lack of confidence in the figures on which design decisions are based.

Since the Distributed Control Model involves a custom chip for the SSC, speed improvements can be obtained by extra work. To design a slow working custom-chip with the complexity of the SSC requires less than a year's work. Once it is finished, several tricks can be played to increase the speed. For example, sense amplifiers are being added to the EM address decode logic which will result in approximately a factor of two increase in the speed of the EM.[4] Implementing these tricks takes a great deal of time and probably is not worthwhile for a first prototype. With a lot of time and effort, however, these tricks can greatly improve the speed of the signal system.

3.3 Criteria of Comparison for the Central Control Model.

3.3.1 Design Time

Design Time is minimized in the Central Control Model by using a gate array chip for the SSC. Gate arrays are a semi-custom technology requiring minimum design time. Turnaround time for a gate-array chip of about the complexity of the SSC is

---

[4]Kevin Theobald and Gao Guang Rong, Computation Structures Group Design Meeting, August 1985.

approximately six months.[5] This will allow a working prototype of the Central Control Model to be produced within six months. This time includes producing a logic diagram and putting it into a format from which a company such as LSI Logic could produce a gate array chip. It also allows five weeks for laying out and producing the chip.[6]

3.3.2 Operating Speed

Because of pin limitations, the SSC can only signal one cell in each EM per cycle. This means that the maximum capacity of the signal system is N signals per cycle, where N is the number of EMs. According to preliminary figures for the compiler (see Section 3.2.2), the signal system needs to signal more than four cells per cycle to keep the processor busy. If these figures are accurate, the Central Control Model will require five EMs per signal system to keep the processor at maximum capacity.

3.3.3 Technological Feasibility

The major problem of the Central Control Model is pin count (see Section 3.1.4). Taking Figure 2-2 and adding pin counts around the SSC produces Figure 3-2. Using this version of the Central Control Model we get the following pin count:

- 44 pins to the EMs: 11 pins x 4 EMs
- 116 pins to the SLMs: 29 pins x 4 SLMs
- 20 pins to the done bus: 13 pins for the cell number, 5 pins for the completion code, and 2 pins for handshaking

---

[5]This figure is taken from gate array chips of similar complexity designed for the MIT Multiprocessor Emulation Facility.

[6]Bernard Conrad Cole, *How Gate Arrays Are Keeping Ahead*, Electronics, September 23, 1985, p. 52.

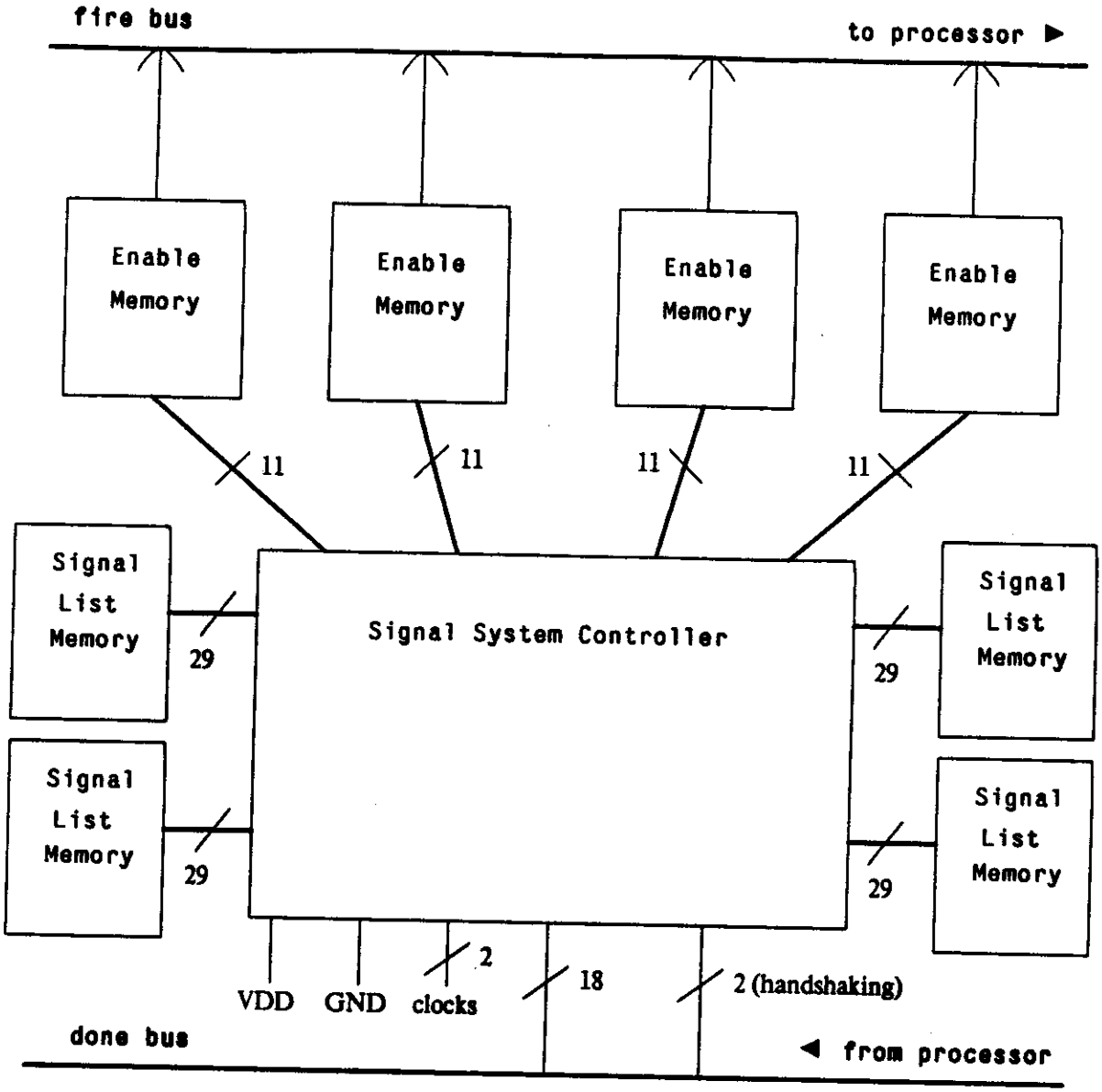


Figure 3-2: Pin Counts for the Central Control Model SSC

- 2 pins for the two clock phases
- 2 pins for power and ground

This adds up to 184 pins. This is close to the limit of currently available packaging technology.

This model only contains four EMs. According to the current figures, this model will need at least 5 EMs to keep the processor busy. Adding another 40 pins for an extra EM and SLM brings the pin count to 222. This number is outside the ability of the current technology.

If the SSC can only operate at half the speed of the SLM, then it is feasible to interleave the SLM, i.e. use one SLM for every two subSSCs. With four subSSCs per signal system, 58 pins on the SSC could be eliminated. If the clock speed of the machine is the same as the speed of the SSC, then the savings is substantial. If, however, the clock speed is the same as the SLM access time, then the signal system needs twice as many subSSCs to keep pace with the processor. In this case, we may need more pins to support twice as many EMs. The advantages that interleaving offers are not clear and require further investigation, but it appears to me that interleaving can only be advantageous if the processor and signal system are not as fast as is projected.

Another way in which the pin count can be reduced in the Central Control Model is through linearization. This is a compilation technique which can greatly reduce the number of signals required to fire each cell (K of Section 3.2.2). The premise of linearization is that the order of evaluation of many cells within a single processing element is known at compile time. With this knowledge, the compiler minimizes the number of signals to ensure the order of execution. Some believe that with extensive use of this technique, K can be held below four. This is not certain, and I

have assumed in this paper that it is not true. If, however, K can be held below four, then the version of the Central Control Model with four subSSCs is feasible given current packaging technology.

3.3.4 Ability To Be Improved with Time

Most of the numbers are not known to be accurate (see Section 3.1.5). Only half the number of EMs represented in Figure 3-2 may be required, or that number could be doubled. If the same or fewer EMs are required, the Central Control Model works fine. If more are needed, the technology will probably not be able to handle the increased pin requirements.

3.4 Conclusion

There are three factors which are important in evaluating the two signal system proposals:

1. design time
2. technological feasibility: ability to design a custom chip
3. technological feasibility: pin count
4. ability to be improved with time: flexibility in light of changing statistics associated with the dataflow machine

3.4.1 Design Time

The Central Control Model is the clear winner in design time. The expected turnaround time for a gate array chip is approximately six months. The Distributed Control Model, being a fully custom design, will take at least a year for the first prototype and probably an additional one to two years for a fast chip.

3.4.2 Technological Feasibility: Ability To Design a Custom Chip

At MIT we have the ability to design, layout, and simulate working custom chips. This concern has been completely answered by the working chips at MIT.

3.4.3 Technological Feasibility: Pin Count

The Distributed Control Model has no problems with pin count. With the current figures, the Distributed Control Model SSC has about 83 pins. The Central Control Model, however, has a serious problem with pins. Assuming the current figures are correct, it will need 222 pins to satisfy the speed requirements. This is one reason that the Central Control Model is not reasonable with current technology.

3.4.4 Ability To Be Improved With Time: Flexibility in Light of Changing Statistics

As is stated in Section 3.1.5, all of the important numbers are guesses. The signal system must be able to adapt and remain feasible as the numbers change.

If the Central Control Model is not fast enough, its speed can be increased only by adding more pins. It has no pins to spare. This is the main reason why I believe that the Central Control Model is the wrong architecture for the signal system.

The Distributed Control Model can be easily adapted to the changing statistics. If more speed is needed, more signal units can be added. For this reason, I believe that the Distributed Control Model is the right choice for the architecture of the signal system.

3.4.5 Conclusion

Although the Central Control Model takes less time to design, it cannot meet the operating specifications using currently available technology. It also cannot adapt as these specifications change. The Distributed Control Model is more difficult to

design, but it adapts well to the changing requirements of the MIT Static Dataflow Computer. Therefore, I conclude that the Distributed Control Model should be used for the signal system of the MIT Static Dataflow Computer.

Appendix A

A Proposal for the Structure of Signal List Memory for the Distributed Control Model

This appendix describes the proposed instruction set for the Distributed Control Model of the signal system.[7] Certain typeface conventions are used in this appendix:

- A **bold** typeface is used for the name of an instruction.
- An *italic* typeface is used for the name of a specific field in an instruction.

A.1 Operation of the Signal System

Types of Statements

The SSC receives messages from the processor indicating which cell's signal list should be executed. Each cycle the SSC gets a word from the SLM. This word is either a signal statement (see Section A.2), a jump statement (see Section A.3), or a miscellaneous control statement (see Section A.4). A signal statement indicates what cells to signal. A jump statement indicates the address of the next word to be read from the SLM. In the case of the **conditional jump** instruction, the jump statement indicates the address of the SLM word to be read in two cycles, and not the next SLM word. The address of each SLM word is one greater than that of the previous SLM word unless otherwise specified by a jump statement or an end of list condition (see Section A.1).

---

[7] An earlier version of this appendix was printed as Computation Structures Group Design Note 14 by David Marcovitz, February 27, 1985.

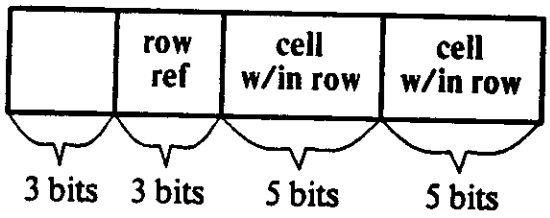
End of List

When an end of list condition is encountered, the signal unit has finished executing the signal list of a cell, and it gets information from the processor about the next cell's signal list to execute. An end of list condition can occur in three ways:

1. a *row ref* of -4 in a short signal (see Section A.2, Page 38)
2. a 1 in the *end* field of a long signal (see Section A.2, Page 39)
3. any type of signal statement in header space (see Section A.5, Page 43)

A.2 Signals

Short Signals



The Enable Memory consists of a grid of 64 rows by 32 columns. The **short signal** specifies one row number and two column numbers causing two cells in the same row to be signalled at the same time.

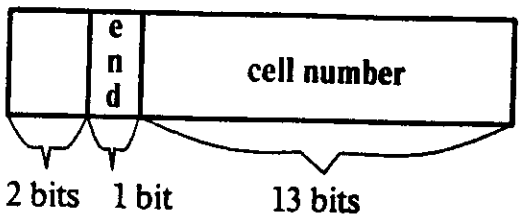
The row is specified by the *row ref* field which is a three-bit two's-complement relative offset. It specifies a row up to ± 3 away from the row of the current cell. (The current cell usually refers to the cell that is doing the signalling, but that is not always the case. See the **change cell** command, Section A.4.)

The *cell w/in row* fields specify the column numbers of the cells to be signalled.

A *row ref* of -4 (`1 0 0`), indicates the end of list condition (see Section A.1). The row that is used for the signals is the row of the current cell with no offset.

This command has a prefix space of tow bits because it is expected to be the most common command. Two bits is the maximum reasonable size for this command's prefix space because any larger prefix space would reduce the range of cells that this cell could signal.

Long Signals

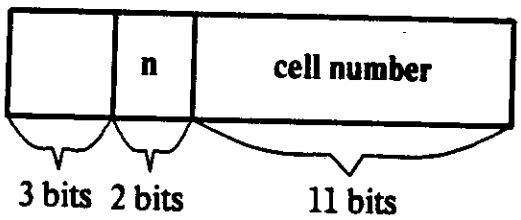


The **long signal** can signal a cell within a signal system. Given a structure with four signal units (of 2K cells each) per processing element, a **long signal** can signal any cell in any of the four signal units. The 13 bit *cell number* field is the absolute address of a cell within the same block.

Setting the *end bit* causes an end of list condition to occur (See Section A.1). A signal is sent to the cell indicated in the *cell number* field, but that is the last cell for this signal list.

The prefix space for a long signal is two bits. With the assumption that a block will contain 8K cells, the cell number space must be thirteen bits allowing the prefix space to be as wide as three bits. An extra bit, however, is taken up by the *end bit*, leaving only two bits for the prefix space.

Dispatch

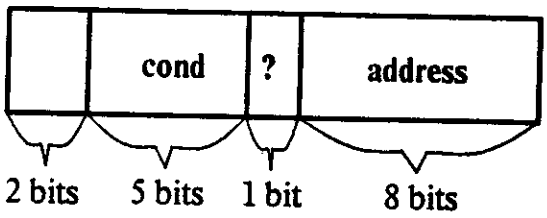


The **dispatch** command signals one of two, four, eight, or sixteen consecutive cells. The *n* field tells whether to dispatch on the low one, two, three, or four bits of the completion code. The cell number field contains an absolute cell number within the signal unit. This is a base number; the low *n* bits of the completion code are then added to this number to obtain the number of the first cell to be signalled.

The prefix space is three bits. This allows a cell to signal any cell within its own signal unit.

A.3 Jumps

Conditional Jumps



The *cond* field is a five-bit condition field representing thirty-two different possible conditions. (See Table A-1.)

The *?* field is a one bit error propagate flag.[8] If this bit is set, and the condition is true, the controller sets its error propagate flag until the next **conditional jump** is encountered.

The *address* field is an eight-bit field indicating the relative address to jump to. It can jump up to +127 and -128 from the address of the current word: the current word is the word containing this command.

---

[8] Ackerman, Program and Machine Structure for Incredibly Fast Computation, p. 23.

Table A-1: Condition Codes for the Conditional Jump Instruction

negation [1]	bad error	Z < 0	error	under/zero	
0	0	0	0	0	positive
0	0	0	0	1	zero
0	0	0	1	0	pos-over
0	0	0	1	1	pos-under
0	0	1	0	0	negative
0	0	1	0	1	VAL ">0" [2 3]
0	0	1	1	0	neg-over
0	0	1	1	1	neg-under
0	1	0	0	0	VAL "<0" [2 3]
0	1	0	0	1	bad error [2 3]
0	1	0	1	0	UK [3]
0	1	0	1	1	UD [3]
0	1	1	0	0	overflow [2 3]
0	1	1	0	1	underflow [2 3]
0	1	1	1	0	ZD [3]
0	1	1	1	1	error [2 3]

negation [1]	bad error	Z < 0	error	under/zero	
1	0	0	0	0	not positive and not bad error
1	0	0	0	1	not zero and not bad error
1	0	0	1	0	not pos-over and not bad error
1	0	0	1	1	not pos-under and not bad error
1	0	1	0	0	not negative and not bad error
1	0	1	0	1	not VAL ">0"
1	0	1	1	0	not neg-over and not bad error
1	0	1	1	1	not neg-under and not bad error
1	1	0	0	0	not VAL "<0"
1	1	0	0	1	not bad error
1	1	0	1	0	not UK
1	1	0	1	1	not UD
1	1	1	0	0	not overflow
1	1	1	0	1	not underflow
1	1	1	1	0	not ZD
1	1	1	1	1	not error

[1] This is loosely the negation bit. If is is on, it means "not" COND. In certain cases the "not" excludes bad errors.

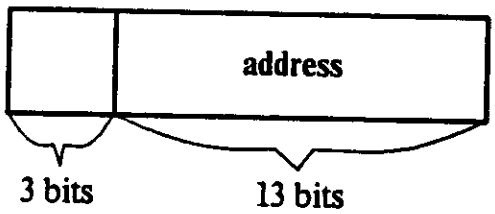
[2] These are combinations of standard conditions; e.g., overflow means either pos-over or neg-over.

[3] The assignment of bit values to these conditions was somewhat random. The other non-negated cases correspond to the "Dennis completion code" bit format.

In order to speed up next address calculation in the signal unit, the instruction immediately following a conditional jump statement will always be executed regardless of whether or not the jump is taken. The jump takes effect after the execution of that instruction. One restriction placed on the instruction after a conditional jump is that it cannot be a jump instruction. This would lead to an ambiguity of what the following instruction should be.

The prefix space is two bits since this is one of the most common commands. The maximum reasonable prefix space is three bits, but this would force the address to have a more limited range.

Long Jumps

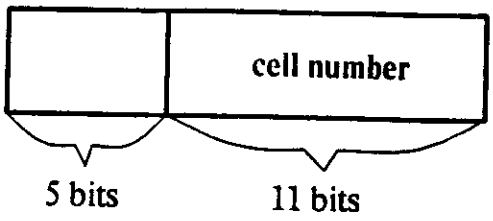


The **long jump** statement changes the program counter to the address listed in the *address* field. Assuming each SLM has 8K of memory, it can jump to any address within the SLM with its thirteen-bit address field.

The prefix space for this instruction is three bits wide. With an 8K memory, any other size would not be feasible.

A.4 Miscellaneous

Change Cell



The **change cell** command is used to allow cells with **short signals** to share signal list space. The signal system controller keeps track of the current cell number. Originally, this cell number is the signalling cell. When a change cell command is encountered, this cell number becomes the cell number listed in the *cell number* field. **Short signals** whose *row refs* are relative will now have their *row refs* relative to this new cell number.

The prefix space for this instruction is five bits. With 2K cells per signal unit, eleven bits are needed to specify the cell number. This leaves five bits for prefix space.

No-op

In case it is impossible to make effective use of the instruction a **no-op** can be placed after a conditional jump.

The prefix space for this instruction is four bits and could be up to sixteen bits.

A.5 Header Space

The header space contains one word for each cell. The low eleven bits of the cell number contain the address of the word in the signal unit header space to be executed first for that cell. The original concept of header space was that each cell would have one fixed "known" location that could be accessed. This location contains a pointer into the "unknown" realm of extension space allowing variable

length signal lists.

Unfortunately, this model has some inherent inefficiencies. For example, a cell that only needs to signal one other cell, has to execute two signal list instructions. Thus, the concept of header space just being a list of pointers was abolished.

Three types of instructions can be included in header space:

1. Long Jumps
2. Short Signals
3. Short3 Signals

The first two instructions are basically the same as in extension space, and the last one is specific to header space.

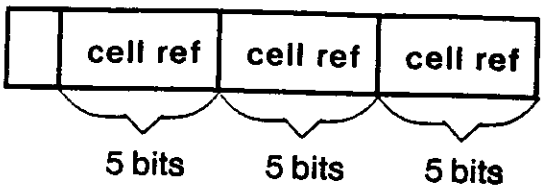
Long Jumps

This is the only instruction from the original concept of header space. It is a **long jump** just like in extension space. It contains a three-bit prefix space and a thirteen-bit address space containing the absolute address of the next instruction. (For more details see Section A.2, page 39.)

Short Signals

If a cell is only signalling one or two close cells, it can use the **short signal** format to signal the cells. In header space, the use of this format means signal these two cells and stop as opposed to the usual meaning: signal these two cells and continue with the next instruction. The second interpretation is meaningless because the next instruction is the header of some other cell. (For more details, see Section A.2, page 38.)

Short3 Signals



Since it should be common for a cell to signal three cells:

- the two cells supplying the operands and
- the one cell which is using the result

it should be inexpensive to signal three cells. This can be done with the **short3 signal** instruction. This instruction can only be used in header space; it is impractical in extension space because of the one-bit prefix space requirement.

The five-bit *cell ref* fields are column numbers in the enable memory. The row number used is the same as the current cell. The three cells listed in these fields are signalled and that is all; no more instructions are executed for this signal list.

The prefix space for this cell is one bit. This allows for three five-bit *cell ref* fields. Since the column number in the enable memory is five bits, no other prefix space size is practical.

Appendix B

Details of the Architecture of the Distributed Control Model

This appendix contains detailed drawings of the hardware of the SSC of the Distributed Control Model.

- Figure B-1 contains an overview of the SSC
- Figure B-2 contains details of the Next Address Logic
- Figure B-3 contains details of the Signal Sending Logic

In Figures B-1 and B-3 the *p* bit associated with the row, column 2, and column 3 registers stands for present. If the row p-bit is set, then the data in the row and column 1 registers is valid. If the column 2 p-bit is set, then the data in the column 2 register is valid. If the column 3 p-bit is set, then the data in the column 3 register is valid. If any signals are being sent, the row p-bit is set. If two signals are being sent (a short signal command), then the row and column 2 p-bits are set. If three signals are being sent (short3 signal command), then all three of the p-bits will be set.

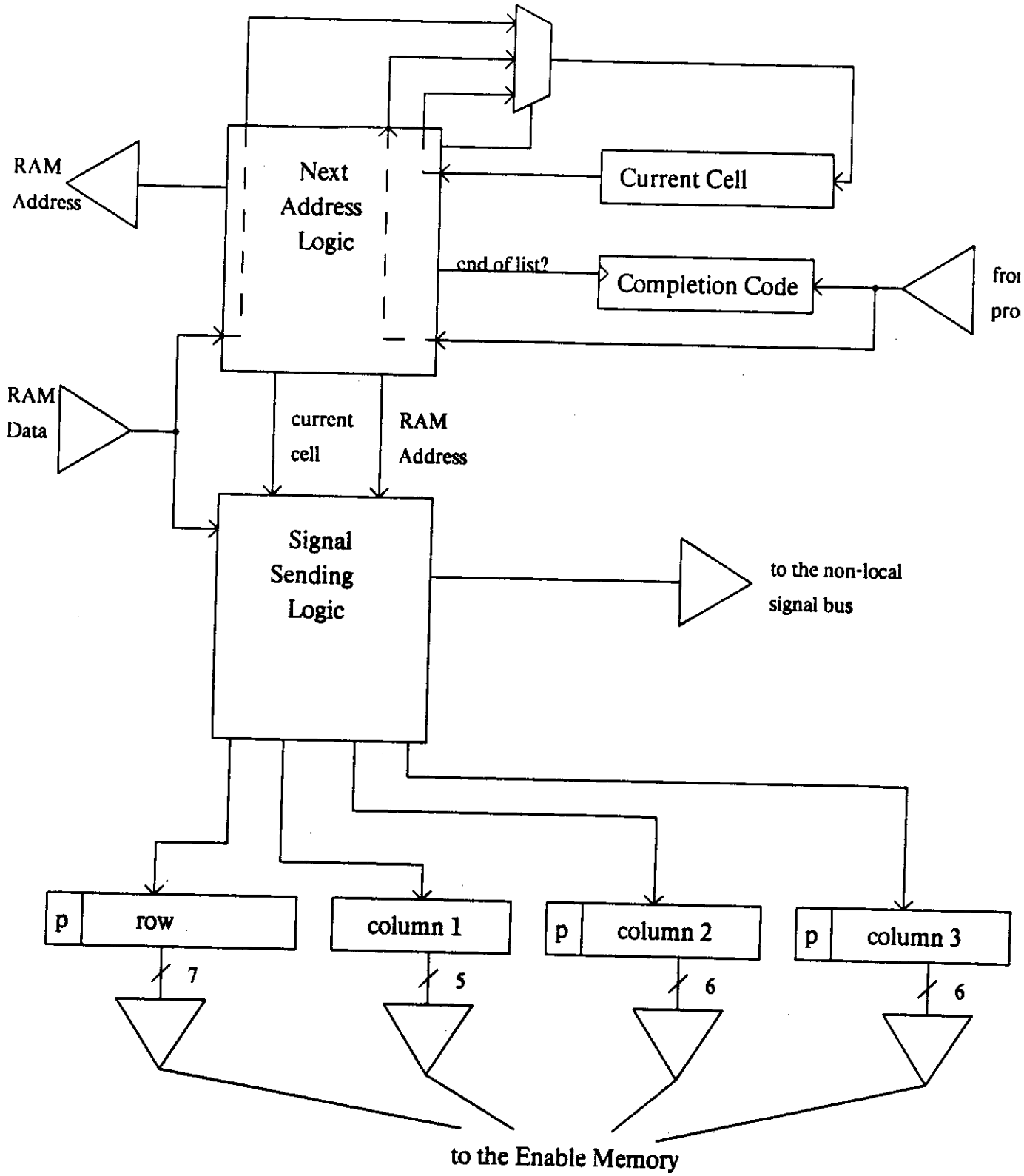


Figure B-1: Overview of the Signal System Controller

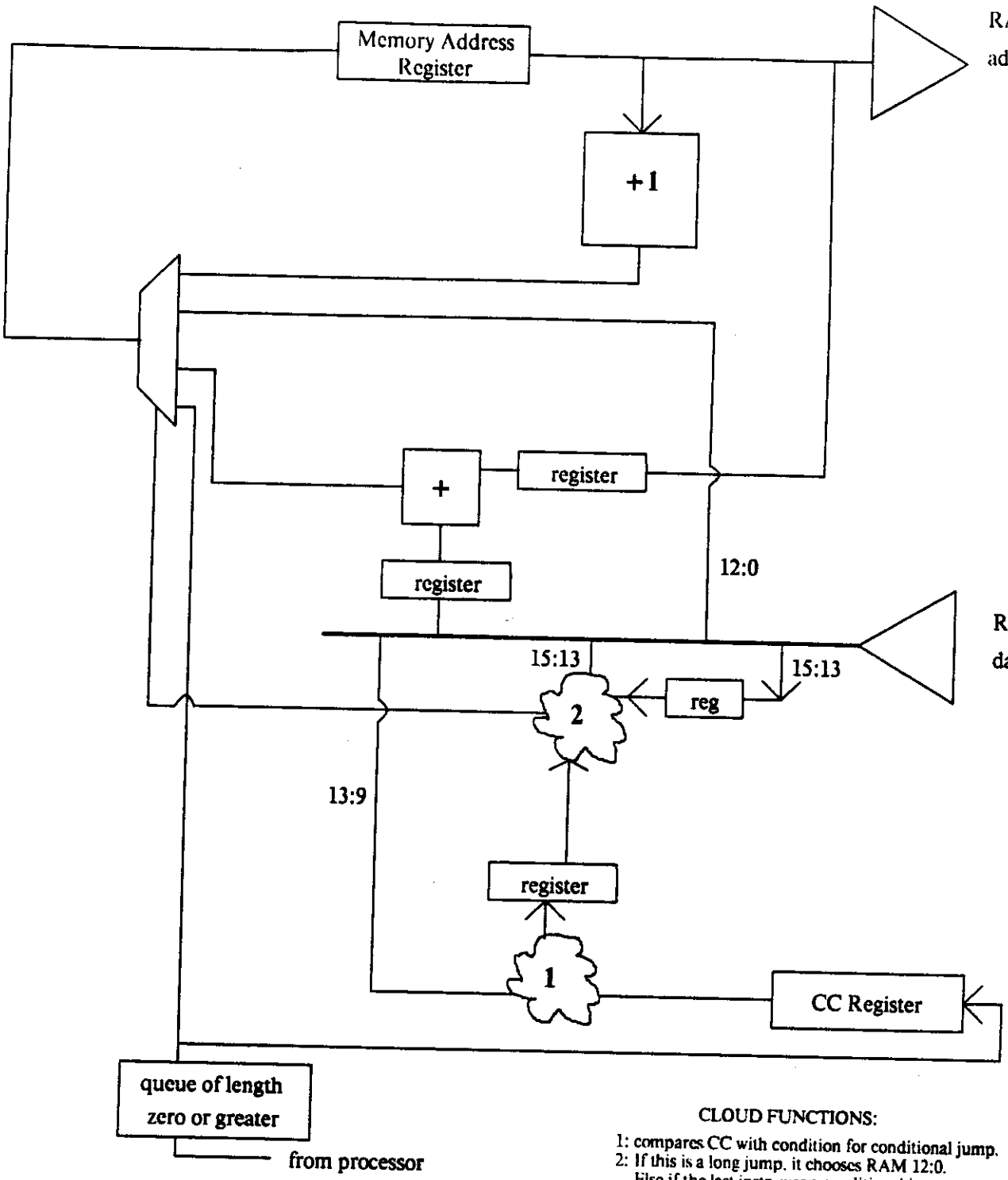


Figure B-2: Next Address Logic

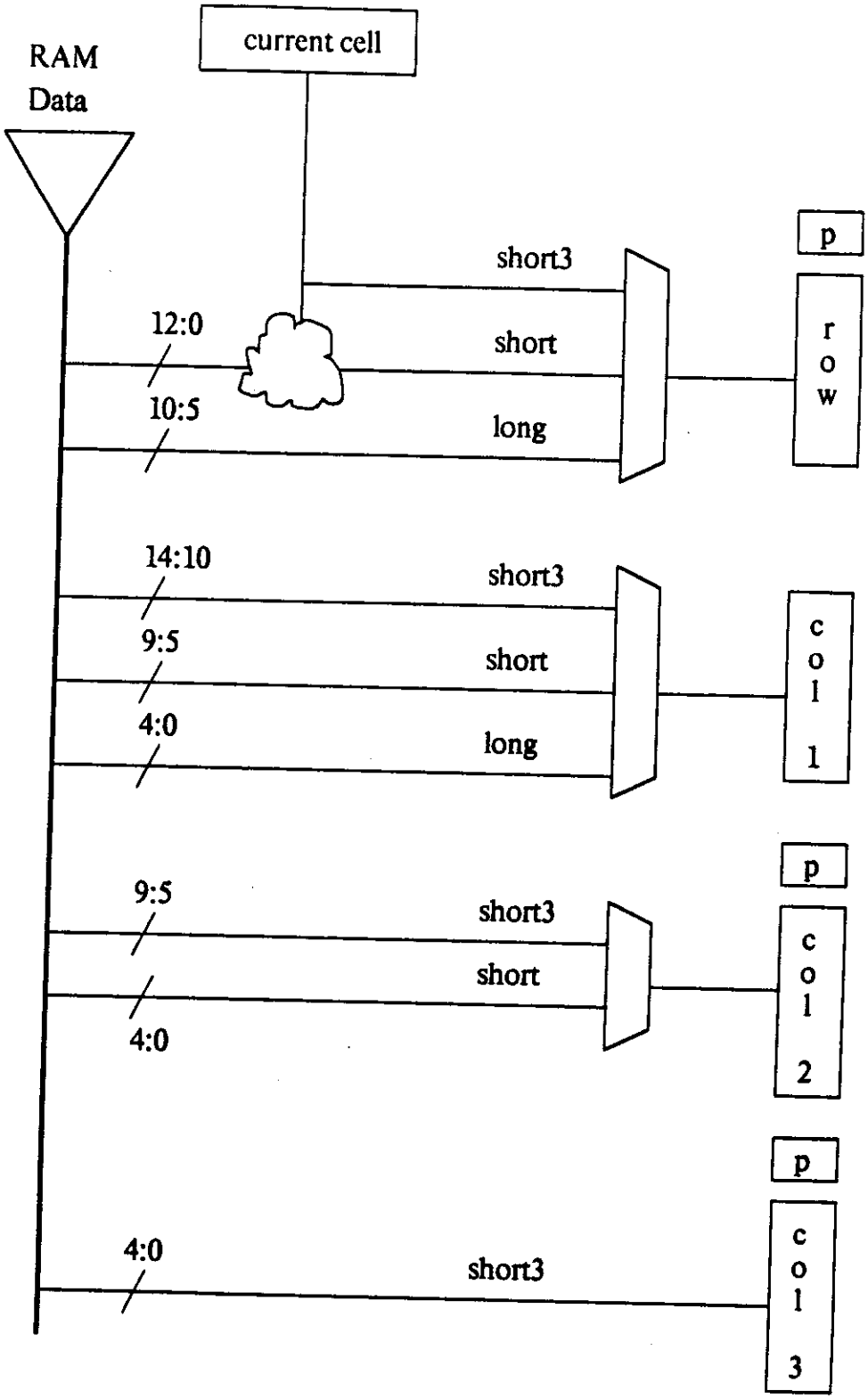


Figure B-3:Signal Sending Logic