VLSI DESIGN
1999, Vol. 10, No. 2, pp. 203–215
Reprints available directly from the publisher
Photocopying permitted by license only

Memory Chips with Adjustable Configurations*

LIZY KURIAN JOHN

Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712

(Received 10 June 1997; In final form 20 October 1998)

In this paper, we present the concept of Field Programmable Memory Cell Arrays (FPMCAs) as the memory counterpart to Field Programmable Gate Arrays which have proved their utility in design and rapid prototyping. Principles of dynamic reconfigurability using programmable logic and programmable interconnect are incorporated into random access memories to achieve this flexibility. We first present the design of a variable width RAM (VaWiRAM) which is a simple example of a Field Programmable Memory Cell Array. The configuration of VaWiRAMs can be adjusted by setting a few configuration pins on the memory chip. A VaWiRAM reconfigurable between widths 1 and $W_{\max}$ can be constructed with the extra cost of $W_{\max} - 1$ pass gates, $(W_{\max}/2)$ 2-to-1 multiplexers, and $\lceil \log_2[\log_2(k) + 1] \rceil$ mode pins. A novel scheme to overlap the address pins with mode control pins and achieve the mode control with only one extra pin is also presented. The paper discusses the architecture of the proposed VaWiRAMs in detail, analyzes the design tradeoffs and introduces the concept of FPMCAs.

Keywords: Dynamically alterable memory configurations, flexible memory, random access memory, reconfigurable memory

1. INTRODUCTION

Designing efficient memory systems for modern microprocessors is increasingly becoming a challenge [10, 12, 15, 18, 19, 21]. The width of DRAMs available in the market is lower than the bus-width of most modern high performance processors and hence most computers normally use single-in-line memory modules (SIMMs) which are small printed circuit boards stocked with multiple DRAMs arranged in parallel, to yield a width of 32 to 144 bits, matching the memory width to the computer's bus-width. *Granularity – the minimum increment of memory that may be added to a system* – [11] increases as the width of the memory chip becomes narrower with respect to the memory bus. For instance, on a 32-bit wide bus that uses 4-Mbit DRAMs, if the chips are organized as 1 M X 4 bits,

*This paper is based on "VaWiRAM: A Variable Width Random Access Memory Module", by Lizy Kurian John which appeared in the *Proceedings of the 9th International Conference on VLSI Design, January* 1996, page number 219–224, © 1995 IEEE. U.S. Patent 5, 867, 422 has been granted to the author on FPMCAs. Lizy John's research is supported by grants from the United States National Science Foundation, the State of Texas Advanced Technology Program, IBM, DELL, AMD, and Intel.

then it takes eight of them to fill out the 32-bit bus with a minimum granularity of 4 MBytes of memory. If, however, the memory chip is organized as 4M X 1 bit, then 32 of them are needed to fill out a 32-bit bus with a minimum granularity of 16 MBytes of memory. If the processor bus-width is 64 bits, the granularity increases even further. The organization of the chip thus affects the expandability of the memory system. It would be desirable for a user to be able to configure the memory chip to any desired width, without being forced to adopt a certain granularity. In this paper, we present the design of a variable width RAM (VaWiRAM) that can be programmed to configure itself to variable widths. In essence, if a memory chip of $N \times b$ bits is available, our design makes it possible to configure the chip to $2NXb/2$ bits, or $4NXb/4$ bits, or in general $kNXb/k$ bits, where k is a power of two.

Currently, each SIMM can be used only with processors that have exactly the same bus-width. Even a configuration with half the bus-width cannot efficiently use the same SIMM. With the availability of variable-width RAMs, programmable SIMMs can easily be constructed. By changing the values on a few control pins, the chips can be reprogrammed to different configurations.

Another fact motivating the research in this paper is that different processors need different interface logic in order to be able to use the same memory chip. Currently memory vendors manufacture separate cache modules for different processors. For instance, Motorola MCM72BA32 and MCM72BA 64 BurstRAMs are [17] designed to be secondary cache modules for the Intel Pentium processor. For the PowerPC and MC68040 processors, Motorola provides the 64 K X 18 Bit BurstRAM, MCM 67M618. It would be desirable to embed programmable logic on a memory chip so that it may be easily interfaced to various processors.

The primary objective of this paper is to present the design of a variable width RAM (VaWiRAM) which can be configured to different widths. A second objective of this paper is to present the design of a fully programmable memory cell array (FPMCA) with programmable logic on the chip, enabling to interface the memory module to various processors, with reduced parts count.

In Section 2, we describe the related research in the past. In Section 3, the design of the VaWiRAM is described. The design of the programmable output data control circuitry and the programmable column decoder unit are described in detail. Various trade-offs involved in the design are examined. In Section 4, extensions to the VaWiRAM, especially Field Programmable Memory Cell Arrays (FPMCAs), are explained. In Section 5, we offer concluding remarks.

2. RELATED RESEARCH

Some of the look-up table (LUT) based FPGAs from manufacturers such as Xilinx [22] and Altera [2] offer a method to realize reconfigurable RAM [20]. For instance, the look up tables on the Xilinx XC4000 series of FPGAs can be used to construct RAMs and it is possible to realize RAMs of different sizes using these FPGAs. However, since FPGAs are primarily intended for programmable logic, it is inefficient to use them as RAMs. For instance, Xilinx XC4020 (a 20,000 gate chip with more than 240 pins) [22] can yield only 28.8k RAM (not really surprising, because XC4020 is not intended to be a memory chip). What we present in this paper are dedicated memory resources with reconfigurability. Such FPMCAs were first discussed in [14]. Programmable memories have been discussed by Bertin *et al.* [4, 5], however the programmability had been for functionality rather than for configuration.

Some reconfigurability at the factory is available in most RAMs for fault-tolerant purposes. For instance, many RAMs incorporate spare rows and columns to replace faulty cells. In many cases, the programming technology is laser-interconnect and reconfigurability is limited to one time. This paper presents the design of a *user-programmable* memory cell array which can be adjusted to different configurations multiple times.

3. ARCHITECTURE

Conventional memory chips contain a matrix of memory cells, row and column address decoders, input and output buffers and control circuitry. In dynamic RAMs, the I/O circuits will include sense amplifiers *etc.* Figure 1 illustrates the architecture of the proposed VaWiRAM chip, which is similar to conventional memory chips except that programmability has been incorporated into the various blocks. For instance, the column address decoder (PCADU), output data control unit (PADCU), and input data control unit are programmable. The row decoder and the memory cells are not shown to be reconfigurable, but they also could be made reconfigurable. We have restricted the programmability to a few selected units in order to reduce the cost of reconfigurability. (We chose the column decoder to be programmable, but one could have chosen the row decoder instead.) The architecture also includes an extra block, the Mode Control Unit (MCU) which receives mode information from the user and delivers appropriate control signals to the various programmable units.

In order to incorporate programmability into the VaWiRAM, we use pass gates, and multiplexers. A pass transistor can be used to realize a programmable interconnection between two wires. The pass transistor turns on or off depending on the control value. When off, the pass gate presents a very high resistance and thereby an open connection between the two wires. When the pass gate is turned on, it forms a relatively low resistance and thereby a closed connection between the two wires. Thus the connection between the two wires can be turned on or off by changing the control values. Similarly, a bidirectional pass gate can be constructed with two pass transistors and an inverter. A multiplexer may be used if there are several input wires. The control signals to control the pass gates and multiplexers is applied at special mode input pins (here, the MCU input pins).

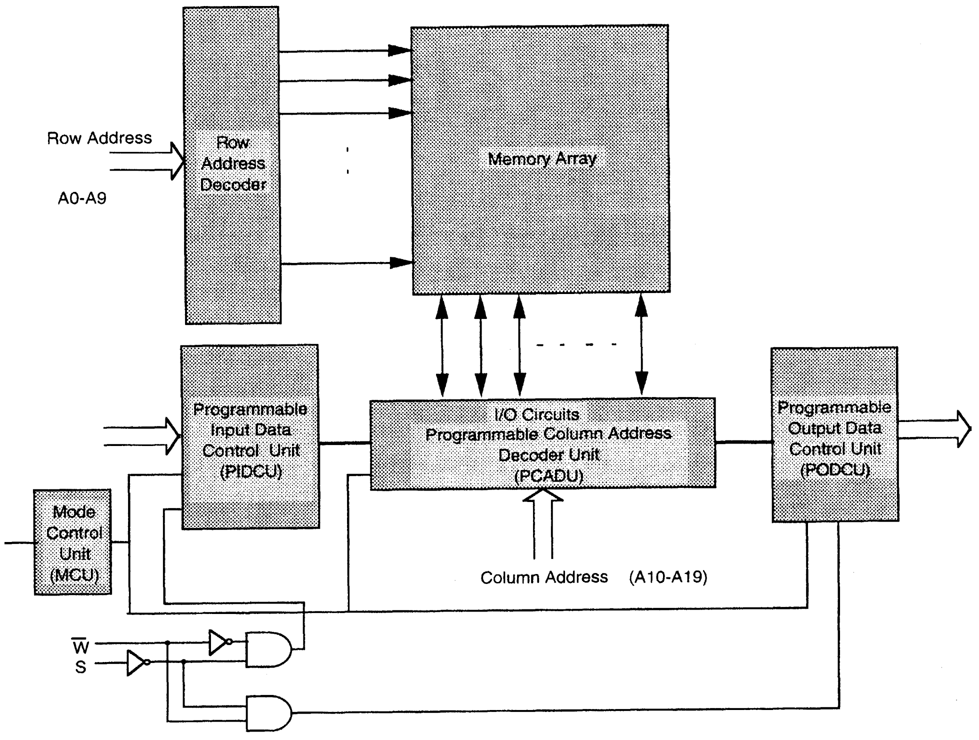
Row Address
A0-A9
Row Address Decoder
Memory Array
Programmable Input Data Control Unit (PIDCU)
I/O Circuits Programmable Column Address Decoder Unit (PCADU)
Programmable Output Data Control Unit (PODCU)
Mode Control Unit (MCU)
Column Address (A10-A19)
W̄
S

FIGURE 1 Architecture of VaWiRAM.

The details of the various blocks inside the chip are presented below. The design uses a hardwired row decoder and a programmable column decoder. The job of the programmable circuitry is to configure the data width to the desired width and correspondingly configure the column decoder, column address pins and chip data pins. At this juncture, we define a few terms.

DEFINITIONS

$W_{\max}$: The maximum allowed width of the chip.
$W_{\max}$ configuration: A VaWiRAM in its maximum width.
$W_{\min}$: The minimum allowed width of the chip.
$W_{\min}$ configuration: A VaWiRAM in its minimum width.
Width Variability factor (k): The ratio of the highest desired width to the lowest desired width.

For illustration purposes, we use a 1MX1 chip which is reconfigurable to 512kX2 bits, or 256kX4 bits, or 128kX8 bits, or 64kX16 bits. Since we keep the row decoder to be fixed, the 1MX1 chip can be viewed as one thousand 1kX1 blocks as in Figure 2. For this chip, maximum width,

$$W_{\max} = 16$$

minimum width,

$$W_{\min} = 1$$

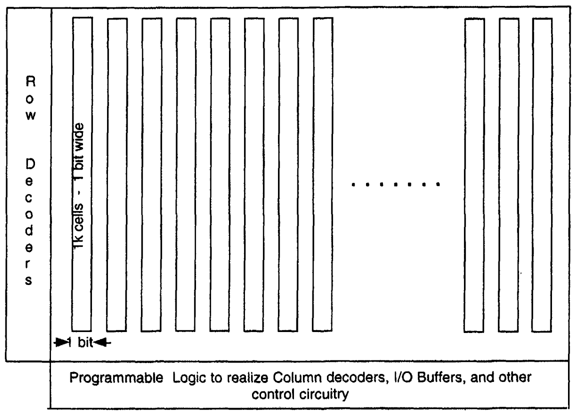
R
o
w
D
e
c
o
d
e
r
s
1K cells - 1 bit wide
bit
Programmable Logic to realize Column decoders, I/O Buffers, and other control circuitry

FIGURE 2 A 1M-bit chip can be viewed as 1000 blocks of $1k \times 1$ bit size.

and width-variability factor,

$$k = 16.$$

3.1. Programmable Column Address Decoder Unit (PCADU)

When data-width of a chip increases by factor k, the number of address lines get reduced by $\log_2(k)$ and hence if there are n address bits in the column address in the minimum width case there will be $n - \log_2(k)$ address bits in the maximum width case. For instance, the chip under design will need $20, 19, 18, 17$, or 16 address pins and $1, 2, 4, 8, 16$ data pins respectively in the five different configurations.

The PCADU is set up in two stages with a layer of multiplexers constituting the first stage as in Figure 3a, and a layer of decoders in the second stage as in Figure 3b.

THEOREM 1 *If there are n bits in the column address in the minimum width case and $n - \log_2(k)$ bits in the maximum width case, the programmable column decoder can be set up as $k/2$ multiplexers, each of size 2-to-1, and k decoders each of size $n - \log_2(k)$ to $2^{n-\log_2(k)}$.*

In this example, we set up the first stage as eight 2-to-1 muxes as in Figure 3a, and the second stage as sixteen 6-to-64 decoders as illustrated in Figure 3b. Let us assume that $A0, A1, A2, \ldots, A19$ denote the twenty address lines. In the $1M \times 1$ configuration, the lower 10 address pins $A0, A1, \ldots, A9$ will be connected to the row decoder, and the higher 10 address bits, $A10, A11, \ldots, A19$ are connected to the column decoder. The column decoder will use $10, 9, 8, 7$, and 6 bits when the width of the chip is $1, 2, 4, 8$, and 16 bits respectively. The 6 input lines to all of the sixteen 6-to-64 decoders in Figure 3b are connected to address lines $A10, A11, \ldots, A15$. Only one of the sixteen 6-to-64 decoders will be enabled concurrently if chip width is 1. Two, four, eight or all sixteen of the decoders will be enabled simultaneously if desired chip-width is $2, 4, 8$, or 16 respectively. The circuitry that will accomplish the

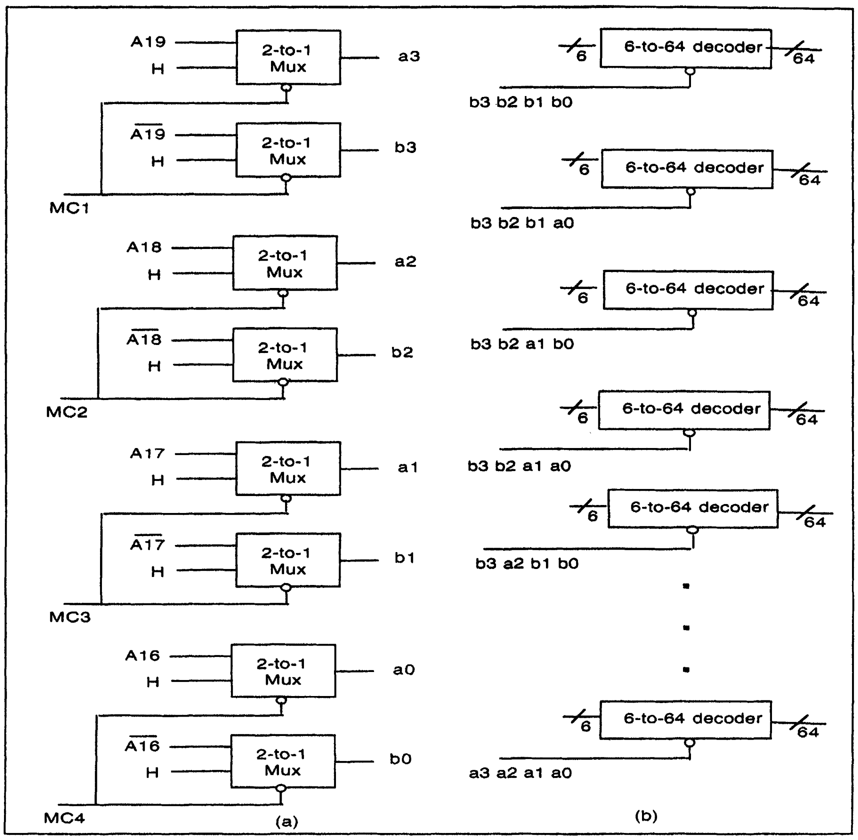

FIGURE 3 Programmable column address decoders.

enabling of different numbers of decoders is shown in Figure 3a. The inputs and outputs of the circuitry in Figure 3a are illustrated in Table I.

3.2. Programmable Output Data Control Unit (PODCU)

The PODCU in VaWiRAM incorporates pass gates in order to realize width variability. Pass gates are placed between the data lines as in Figure 4, so that the lines can be connected together or not, depending on the desired width.

THEOREM 2 *To achieve a width-variability factor of k, $\log_2(k)$ levels of pass gates are required.*

Proof By connecting together pairs of data lines in the $W_{\max}$ configuration, a width of $W_{\max}/2$ can

TABLE I Inputs and outputs of circuitry in Figure 3a

Inputs				Outputs								Configuration
MC1	MC2	MC3	MC4	$a3$	$a2$	$a1$	$a0$	$b3$	$b2$	$b1$	$b0$	
1	1	1	1	A19	A18	A17	A16	$\overline{A19}$	$\overline{A18}$	$\overline{A17}$	$\overline{A16}$	1 bit wide
1	1	1	0	H	A18	A17	A16	H	$\overline{A18}$	$\overline{A17}$	$\overline{A16}$	2 bit wide
1	1	0	0	H	H	A17	A16	H	H	$\overline{A17}$	$\overline{A16}$	4 bit wide
1	0	0	0	H	H	H	A16	H	H	H	$\overline{A16}$	8 bit wide
0	0	0	0	H	H	H	H	H	H	H	H	16 bit wide

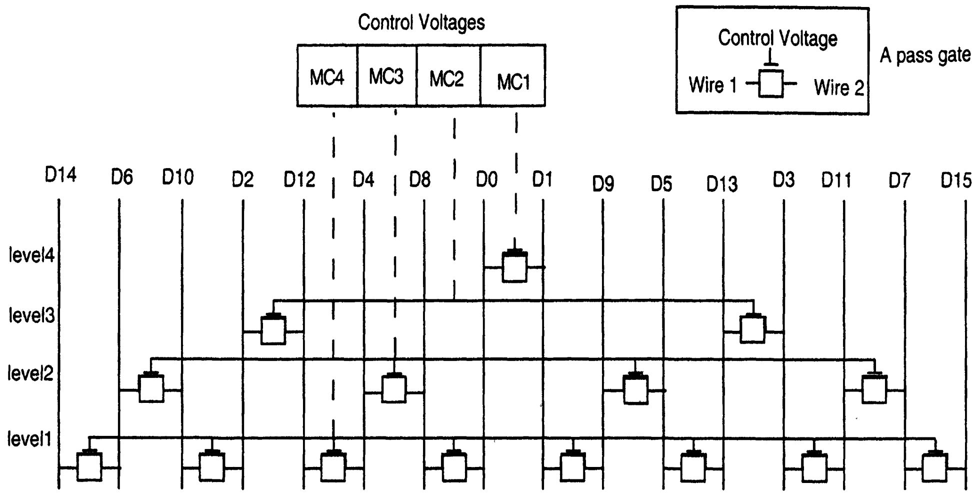

FIGURE 4 Circuitry for reconfiguring data width. A simplified symbol is used to represent the pass gate.

be realized. Since width-variability factor is k, $W_{\max}/W_{\min} = k$. The width gets divided by two for each level of pass gates; so $\log_2(k)$ levels of pass gates will be required to achieve a width variability factor of k.

THEOREM 3 *The total number of pass gates required to achieve a width-variability factor of k is*

$$\sum_{i=1}^{log_2(k)} \frac{W_{max}}{2^i}$$

Proof In the $W_{\max}$ configuration, ($W_{\max}/2$) pass gates are required to connect together pairs of data lines and obtain a width of ($W_{\max}/2$). ($W_{\max}/4$) pass gates will be required for the next level, ($W_{\max}/8$) pass gates for the next level and so on. Hence the total number of pass gates will be

$$W_{\max}/2 + W_{\max}/4 + W_{\max}/8 + \cdots + W_{\max}/k.$$

THEOREM 4 *If $W_{min} = 1$, the total number of pass gates required to achieve a width-variability factor of k is*

$$W_{max} - 1.$$

Proof If $W_{\min} = 1, k = W_{\max}$ and

$$W_{\max}/2 + W_{\max}/4 + W_{\max}/8 + \cdots + W_{\max}/k = W_{\max} - 1.$$

THEOREM 5 *To achieve a width-variability factor of k, $log_2(k)$ control values are sufficient.*

Proof To achieve width-variability factor of k, $\log_2(k)$ levels of pass gates are sufficient. Each pass gate in the same level has identical values. Hence $\log_2(k)$ control values are sufficient. Actually we can further reduce the number of control signals, which is explained in the subsection on Mode Control Unit (MCU).

For illustration, from Theorem 2, four levels of pass gates are required to change the width between 1 and 16. Figure 4 illustrates how 16 data lines can be controlled by 4 mode select lines to yield any width from 16 to 1, which are powers of two. From Theorem 3, the number of gates required can be verified to be 15. Figure 5 illustrates the values required to control the pass gates.

A similar circuit has to be incorporated into the input circuitry (PIDCU) also. Because of the similarity, the design of PIDCU is not described here. The mode select lines controlling the pass

Control Values				Resulting Configuration
0	0	0	0	16 bit wide
1	0	0	0	8 bit wide
1	1	0	0	4 bit wide
1	1	1	0	2 bit wide
1	1	1	1	1 bit wide
MC4	MC3	MC2	MC1	

FIGURE 5 Values for the select signals for the 5 different configurations.

transistors in the input data lines can be the same as those in the output circuitry.

3.3. Mode Control Unit (MCU)

The control values required for the PCADU, PODCU and PIDCU are identical and are generated by the MCU. Out of the $\log_2(k)$ control values, not all combinations are used and hence we could further minimize the number of control signals required for the operation of the chip.

THEOREM 6 *If the width-variability factor is k, $\lceil log(log_2 k + 1)\rceil$ bits of information are sufficient for the mode information.*

Proof If the width-variability factor is k, there are $\log_2 k + 1$ distinct widths considering the restriction that only powers of two are allowed. If there are $\log_2 k + 1$ distinct widths, $\lceil \log(\log_2 k + 1)\rceil$ bits of information will be sufficient to indicate the desired width.

Illustration In our example, the width-variability factor is 16. Hence there will be $\log_2(16) + 1$ or 5 distinct widths that may be desired (actually $1, 2, 4$, 8, and 16). Since there are 5 different modes to be supported, $\lceil \log_2(5)\rceil$ or 3 bits are required to represent the mode information. Even in a case for programming from 1 bit to 1024 bit width, only four lines are sufficient.

For our example design, the mode control unit has three input bits which are used to generate the select signals for the programmable column decoder and the input and output data control units, according to Figure 6. The input combinations to correspond to each output combination is rather arbitrary and different choices can lead to different number of gates in an implementation. The input combinations shown in Figure 6 is the trivial choice but not the optimal choice. There are ways to overlap the functions of some of the mode control pins with the address pins, which is discussed later.

Inputs			Outputs			
M3	M2	M1	MC4	MC3	MC2	MC1
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	1	0	0
0	1	1	1	1	1	0
1	0	0	1	1	1	1

FIGURE 6 MCU inputs and outputs.

3.4. Tradeoffs

3.4.1. Pin Limitations

The most serious problem associated with VaWiRAMs is the requirement for a large number of pins for the chip. VaWiRAM chips should be able to support the number of data pins corresponding to the largest width to be supported, the number of address pins corresponding to the smallest width, and at least $\log_2[\log_2(k) + 1]$ mode pins. Wider configurations naturally require larger number of data pins and hence the increase in the number of data pins should not be considered as a disadvantage. The need to have mode control pins is a disadvantage, however, we can overlap some of the mode control pins with the address pins as described below.

While increasing the data width by k times, the number of address lines required reduces by $\log(k)$ and those lines can be used to deliver the mode information to the chip. Figure 7 illustrates the

Inputs					Outputs				
M	A19	A18	A17	A16	MC4	MC3	MC2	MC	Configuration
0	X	X	X	X	1	1	1	1	1 bit wide
1	0	X	X	X	1	1	1	0	2 bit wide
1	1	0	X	X	1	1	0	0	4 bit wide
1	1	1	0	X	1	0	0	0	8 bit wide
1	1	1	1	X	0	0	0	0	16 bit wide

FIGURE 7 Combining address pins with mode pins.

manner in which the address lines and the mode control bits can be shared. All mode control pins except one can be time-shared with the address pins. That line if 0 commands the chip to be in the narrowest possible configuration. If this mode control line is 1, the chip is in a wider mode. Hence at least the highest address line (here, A20) is not required for addressing. So the highest address line can be used for mode information. If that line is also 0, the chip is twice as wide as the narrowest configuration. If A20 is a 1, the chip is even wider. So A19 is not required for addressing and will be used for mode information. If A19 is 0, the chip is 4 times as wide as the narrowest configuration. If A19 is 1, the chip is wider than 4 times the original width. So A18 will not be required for addressing and can be used for mode information. If A18 is a 0, the chip is 8 times the original width and if A18 is a 1, the chip is 16 times the original width. Thus mode control can be accomplished by just one extra pin.

3.4.2. *Area Overhead*

The VaWiRAM requires more hardware than a conventional RAM. The input and output data control unit will need ($W_{\max} - 1$) extra pass gates each. Similarly the programmable address decoder unit requires $k/2$ extra multiplexers each of size 2-to-1. The mode control unit also consumes a few gates.

Table II illustrates the area overhead for achieving the programmability for a 64k bit memory array. The overhead is computed as the ratio of

TABLE II Area overhead for VaWiRAMs

Size of RAM (cells)	Width (bits)	Overhead area (transistor count)	% Overhead
64k	1–16	208	0.053%
	1–32	394	0.100%
	1–64	756	0.192%
	1–128	1470	0.372%
	1–256	2888	0.729%

the number of transistors utilized for reconfigurability to the number of transistors utilized for the actual memory cells. Static RAM cells with 6-transistors are assumed for the RAM. A pass gate is assumed to contain 4 transistors and a 2-input mux is assumed to contain 6 transistors including the required inverters. The mode control unit is assumed to be built as a compound logic function implemented at transistor level. Figure 8 illustrates the percentage overhead for programmability upto a width variability factor of 256, for 16k, 64k and 1M RAMs.

Compared to the flexibility yielded by the design, the incurred overhead is negligible. For example, for a 1M RAM, the area expended in achieving a programmable width from 1 bit to 256 bits is only 0.046%. For smaller RAMs, the percentage overhead is higher. Overall, the area overhead increases with the width variability factor k and reduces with increasing array sizes.

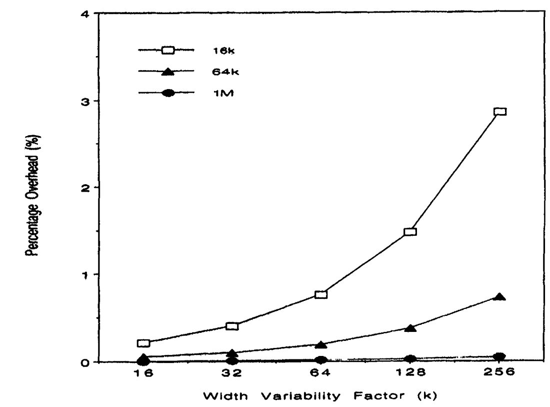

FIGURE 8 Area overhead of VaWiRAMs of sizes 16k, 64k and 1M bits.

3.4.3. *Time Considerations*

The circuitry incorporated for the reconfigurability of VaWiRAMs can contribute to longer line delays and slower switching. The various programmable units, namely the PCADU, PODCU, and MCU will add a few gate delays to the RAM access process unless carefully implemented. A naive design will slow down the chip, however, on careful observation, it was clear that many of these delays could be eliminated by appropriate circuit design. The delays can be reduced or eliminated by overlapping them with operations in the conventional RAM. It is difficult to obtain precise estimate of the delays without detailed simulations of the whole circuit and such an analysis is beyond the scope of this paper. A preliminary evaluation shows that the PODCU will contribute one transistor delay. Similarly, the MCU will lead to adding two transistor delays.

4. EXTENSIONS TO THE VaWiRAM ARCHITECTURE

In VaWiRAM, we restricted reconfigurability to the column decoder and the input-output data control. We could remove this restriction to yield a chip with programmable memory array as well as programmable row and column decoders. The most general view of such a programmable memory cell array would be as in Figure 9. There will be a sea of memory blocks (individual memory cells or large groups of cells) which can be interconnected in ways desired by the user, using programmable interconnect present in various locations within the chip. The row and column decoders, I/O circuits and interconnections between the memory blocks will be configured by the user from the programmable logic present in the chip. The chip will also have I/O blocks which can be configured as input or output buffers. The pins on the chip are not hard-wired to any address or data lines and will be connected by the user to yield the desired configuration. These programmable logic blocks may be similar to the configurable logic blocks in commercial FPGAs or they can be tailored to constitute elements specific to interface logic needed by memory chips. The programmable logic can be used to dynamically alter the data width of the RAM, and to configure any control logic required to interface the memory to the processor.

The voltages controlling the various programmable points in the chip reside in static RAM cells constituting the ‘configuration RAM’. The chip configuration can be altered by loading a different set of bits into the configuration RAM. Smaller memory blocks will yield increased flexibility during reconfiguration but will increase the overhead associated with reprogramming. Higher amount of routing resources will help easy routing but will increase the overhead. Partitioning, placement and routing algorithms developed in connection with FPGAs may be used for FPMCAs as well.

Every programmable point consumes silicon and reduces the real estate that can be used for memory cells. Fine-grain programmability in which every memory block is a single cell will be very inefficient due to this reason. A whole range of devices from less efficient and highly flexible to highly efficient and less flexible, as in Figure 10 can be designed. The key to designing a successful programmable device is striking the right balance between the overhead associated with reprogrammability and the flexibility gained by it.

The FPMCA concept could be implemented in several ways. Depending on the amount of programmability provided by the chip, we categorize FPMCAs to various levels as indicated in Table III.

(i) **Level 0 is conventional RAM** with no reconfigurability of organization. There is no extra flexibility or overhead. Level 0 is defined only to form a baseline configuration for comparison purposes. It may be noticed that level 0 FPMCAs have an overhead of 0%.

(ii) **Level 1 or Coarse Grain Reconfigurable** FPMCA is in fact **VaWiRAM**; *i.e.*, RAM with very minimal programmability. A chip

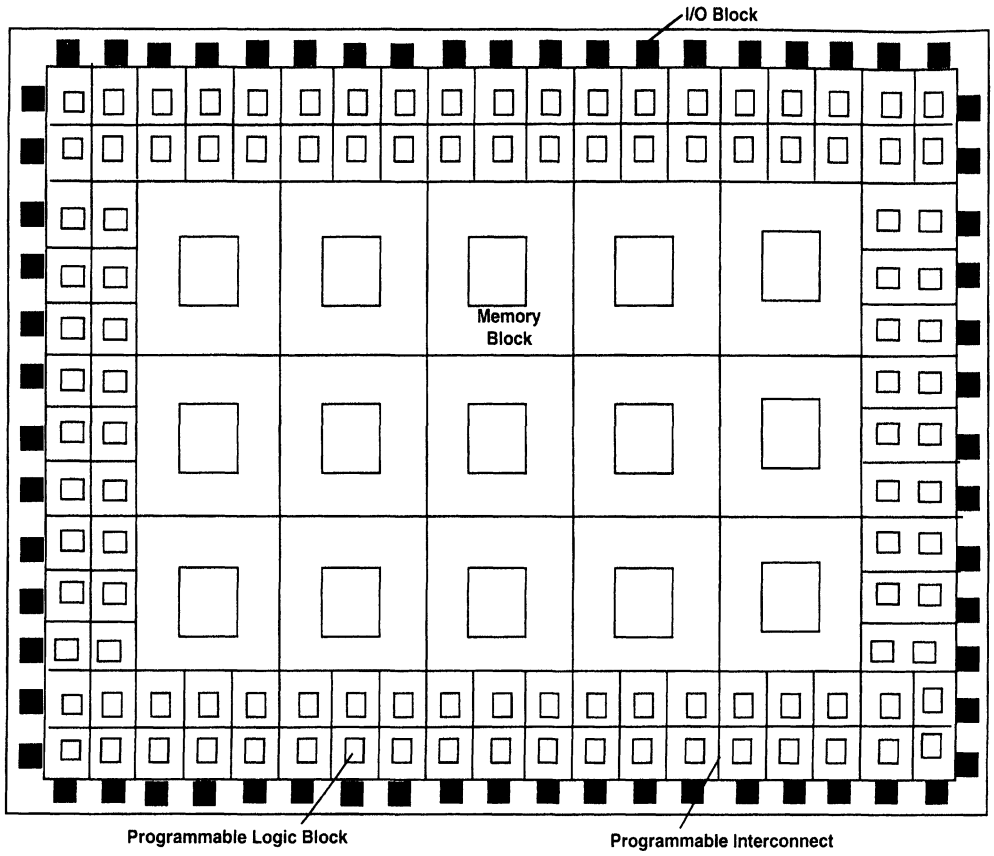

FIGURE 9 A field programmable memory cell array (FPMCA) with a sea of memory blocks.

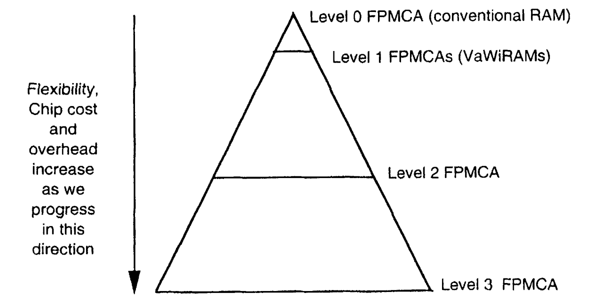
Flexibility, Chip cost and overhead increase as we progress in this direction
Level 0 FPMCA (conventional RAM)
Level 1 FPMCAs (VaWiRAMs)
Level 2 FPMCA
Level 3 FPMCA

FIGURE 10 The spectrum of programmable memory cell arrays.

reconfigurable between widths 1 and $W_{\max}$ can be constructed using $W_{\max} - 1$ pass gates, $(W_{\max}/2)$ 2-to-1 multiplexers, and $\lceil \log_2[\log_2(k) + 1] \rceil$ mode pins. Level 1 FPMCAs have an overhead typically smaller than 10%. The area overhead of a few FPMCAs with programmability equal to the square root of the total array size is presented in Table IV.

The area overhead is seen to be less than 6.5% for RAMs of size 1k to 64k bits. As the size of the RAM reduces, the overhead can go beyond

TABLE III Classification of FPMCAs

Classification	Reconfigurability	Basic memory grain	Typical overhead
Level 0 (Conventional RAM)	None	The whole chip	none
Level 1 (VaWiRAM)	Coarse grain	1 or more columns	Less than 10%
Level 2	Medium grain	Several cells	10–50%
Level 3	Fine-grain	1 cell	Greater than 50%

TABLE IV Area overhead for VaWiRAMs with a width variability factor equal to the square root of the size of the array

Size of RAM (cells)	Width variability factor (k)	% Overhead
256	16	11.894%
1024	32	6.029%
4096	64	2.984%
16384	128	1.473%
65536	256	0.729%

TABLE V Area overhead for level 3 FPMCAs

Size of RAM (cells)	Width variability factor (k)	% Overhead
256	256	65.3%
512	512	65.0%
1024	1024	64.9%
2048	2048	64.8%

10% (for instance, in the case of a 256 bit array with a width variability factor of 16, the overhead is 11%). However, it should be remembered that the classification is being used only to illustrate the general trend that reconfigurability at a fine-grain costs more than coarse-grain reconfigurability.

(iii) **Level 2 or Medium Grain Reconfigurable** FPMCAs contain two-dimensional programmability rather than only programmability of width as in VaWiRAMs. Each of the memory blocks is not a single cell, but it is a macrocell consisting of a two-dimensional array of individual cells. Level 2 FPMCAs have overhead typically ranging from 10% to 50%. It may be noticed that according to our definition, 50% overhead means that the overhead is equal to the usable memory area. Overhead decreases as the size of the macro-block increases.

(iv) **Level 3 or Fine Grain Reconfigurable** FPMCAs contain a highly programmable memory array in which each memory block is an individual cell. As expected, the area overhead is extremely high, and is higher than the real estate spent on memory. (Not too surprising – in many FPGAs, more silicon is spent on interconnections rather than on programmable logic). Level 3 FPMCAs have an overhead higher than 50%. An evaluation of the overhead for level 3 FPMCAs of sizes 256 bits to 2k bits is illustrated in Table V. It may be noticed that this computation considers only the hardware required to yield a programmable array; it does not consider any programmable interface logic on the chip intended to interface the chip to processors or other devices.

The distinction between different classes of devices is not rigid. As mentioned before, the classification is intended only to illustrate the general trends. FPMCAs is a new concept and the treatment provided here is rather preliminary. However, it is an important step towards understanding the tradeoffs in their design.

The dynamic reconfigurability of FPGAs [2,. 6, 22] have been employed in several interesting applications [3, 8, 9, 13]. Availability of FPMCAs can lead to interesting applications requiring large amounts of flexible memory. It will be useful for developing adaptive cache architectures or branch predictors that dynamically reconfigure itself to suit the application being run on the processor. As in FPGA based systems where hardware upgrades become a software procedure that can be accomplished by sending a different configuration pattern on a diskette or *via* an electronic transmission [16], systems with FPMCAs can easily be upgraded or reconfigured. The proposed FPMCA will add more flexibility to all users of memory chips. Memory manufacturers need not make X1, X4, X16 and X32 chips. The proposed generic memory array can be programmed to the desired configuration easily.

The FPMCAs will also be extremely useful for prototyping large memory systems. Field Programmable Interconnect Devices (FPIDs) [1] are often combined with FPGAs to realize field programmable boards [7]. The FPMCA will add more flexibility to such field programmable boards and programmable backplanes. Currently different memory chips have to be interconnected through

FPIDs in order to realize flexible memory organizations. With the availability of FPMCAs, the programmable boards and backplanes will become more efficient. There's no doubt that FPMCAs will be a powerful addition to the suite of programmable hardware consisting of FPGAs, FPIDs, and field programmable boards and backplanes.

5. SUMMARY AND CONCLUDING REMARKS

In this paper, we presented the design of a variable width RAM called VaWiRAM which allows the user to configure the RAM to the desirable width. A VaWiRAM chip reconfigurable between widths 1 and $W_{\max}$ can be constructed at an additional cost of $W_{\max} - 1$ pass gates, ($W_{\max}/2$) 2-to-1 multiplexers, and $\lceil \log_2[\log_2(k) + 1] \rceil$ mode pins. This paper discussed the architecture of the proposed Variable Width RAMs (VaWiRAMs), and analyzed the trade-offs in their design. Although a naive implementation requires $\lceil \log_2[\log_2(k) + 1] \rceil$ mode pins, a novel scheme requiring only a single mode pin was presented. We also discussed a powerful extension to the VaWiRAM, the Field Programmable Memory Cell Arrays (FPMCAs) which are reconfigurable memory chips with embedded logic to realize the interface logic to other system components with reduced parts count. The FPMCAs illustrate the potential for a fully reconfigurable memory component, whereas the VaWiRAM illustrates that limited versions of such systems can be constructed with extremely low overhead. Although the concept can be applied to static and dynamic RAMs, design of adjustable dynamic RAMs is more complex (compared to SRAMs) due to the need for refreshing.

References

[1] Aptix, The Programmable Interconnect Company Data Book, February 1993.
[2] http:// www.altera.com
[3] Athanas, P. M. and Silverman, H. F., "Processor Reconfiguration Through Instruction Set Metamorphosis", *Computer*, March 1993, pp. 11–18.
[4] Bertin, P., Roncin, D. and Vuillemin, J. (1989). "Introduction to Programmable Active Memories", PRL Report 3, Digital Equipment Corporation, Paris Research Lab.
[5] Bertin, P., Roncin, D. and Vuillemin, J. (1992). "Programmable Active Memories: Performance Measurements", *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 57–59.
[6] Brown, S. D., Francis, R. J., Rose, J. and Vranesic, Z. G. (1992). Field Programmable Gate Arrays, Kluwer Academic Publishers.
[7] Chan, P. K., Schlag, M. D. F. and Martin, M. (1992). "BORG: A Reconfigurable Prototyping Board using Field-Programmable Gate Arrays", *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 47–51.
[8] Cox, C. E. and Blanz, W. E., "GANGLION – A Fast Field- Programmable Gate Array Implementation of a Connectionist Classifier", *IEEE Journal of Solid State Circuits*, **27**(3), March 1992.
[9] Dobbelaere, I., Gamal, A. E., How, D. and Kleveland, B. (1992). "Field Programmable MCM Systems – Design of an Interconnection Frame", *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 52–56.
[10] Katayama, Y.,"Trends in Semiconductor Memories", *IEEE Micro*, November/December 1997.
[11] Prince, B.,"Memory in the fast lane", *IEEE Spectrum*, Feb. 1994, pp. 38–41.
[12] Prince, B. (1991). Semiconductor Memories, John Wiley & Sons.
[13] Gokhale, M., Holmes, W., Kosper, A., Lucas, S., Minnich, R., Sweely, D. and Lopresti, D., "Building and Using a Highly Parallel Programmable Logic Array", *IEEE Computer*, January 1991, pp. 81–89.
[14] John, L. K., "VaWiRAM: A Variable Width Random Access Memory Module", *Proceedings of the Ninth International Conference on VLSI Design*, 1996 January.
[15] Special Report on Memory Systems, *IEEE Spectrum*, October 1992, pp. 34–57.
[16] Hsieh, W. J. (1993). "Reconfigurable Hardware", Computer Design, March 1993, pp. 27–30.
[17] Motorola Fast SRAM data book, 1993.
[18] Przybylski, S. (1990). Cache and Memory Hierarchy Design: A Performance-Directed Approach, Morgan Kaufman.
[19] Quinnell, R. A., High Speed Bus Interfaces, Electronic Design News, Sept. 30, 1993. p. 42.
[20] Salcic, Z. and Smailagic, A. (1997). "Digital Systems Design and Prototyping Using Field Programmable Logic", Kluwer Academic Publishers, p. 75.
[21] Tuite, D., Cache architectures under pressure to match CPU performance, Computer Design, March 1993, pp. 91–97.
[22] http://www.xilinx.com/

Author's Biography

Lizy Kurian John is an assistant professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin. She received her Ph. D in Computer Engineering from Penn State in August 1993. Her research interests include high performance processor and memory architectures, program behavior studies, compiler optimization for high performance processors, rapid prototyping, *etc*. Currently she is supported by grants from the National Science Foundation including a CAREER Award, and the State of Texas Advanced Technology Program. She received a Junior Faculty Enhancement Award from Oak Ridge Associated Universities in 1996 – 1997. She is senior member of IEEE and a member of ACM and ACM SIGARCH. She is also a member of Eta Kappa Nu, Tau Beta Pi and Phi Kappa Phi.