VaWiRAM: A Variable Width Random Access Memory Module *

Lizy Kurian John
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
kurian@csee.usf.edu

Abstract

In this paper, we present the design of a variable width RAM (VaWiRAM) which will be extremely useful in building flexible memory systems. Principles of dynamic reconfigurability of programmable logic and programmable interconnect is incorporated into random access memories to achieve this flexibility. The chip can be reconfigured 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, $\frac{W_{max}}{2}$ 2-to-1 multiplexers, and $\lceil log_2[log_2(k) + 1]\rceil$ mode pins. The paper discusses the architecture of the proposed VaWiRAMs, and analyzes the trade-offs in their design.

Keywords: Dynamically alterable memory, Programmable hardware, Reconfigurable Memory, Random Access Memory.

1 Introduction

Processors now as always require more data per unit time than many standard memory chips have been designed to provide. 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 –* [8] 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, 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 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. Currently memory vendors manufacture separate cache modules for different processors. For instance, Motorola MCM72BA32 and MCM72BA64 BurstRAMs are 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, MCM67M618. If the memory chip contains programmable logic, it can easily be interfaced to different 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 develop a fully programmable memory array 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. In section 4, extensions to the VaWiRAM, especially the design of a Field Programmable Memory Cell Arrays (FPMCAs) is explained. In section 5, we offer concluding remarks.

2 Related Research

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. What this paper presents is the design of a *user-programmable* memory cell array which can be reconfigured any number of times.

Some of the look-up table (LUT) based FPGAs from manufacturer's such as Xilinx allow the realization of reconfigurable RAM. The look-up tables on the

*This work was supported in part by the Division of Sponsored Research, University of South Florida. A U. S. Patent application on VaWiRAMs and FPMCAs is pending.

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 would be inefficient to use them as RAMs. For instance, Xilinx XC4020 (a 20,000 gate chip with more than 240 pins) can yield only 28.8k RAM (not really surprising, because XC4020 is not meant to be a memory chip). What we present in this paper is dedicated memory resources with reconfigurability.

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. Fig. 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 in an ideal sense, 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 VaWiRAM contains memory cells and pass gates and multiplexers. The control voltage to control the pass gates and multiplexers is applied at special mode input pins (here, the MCU input pins).

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.

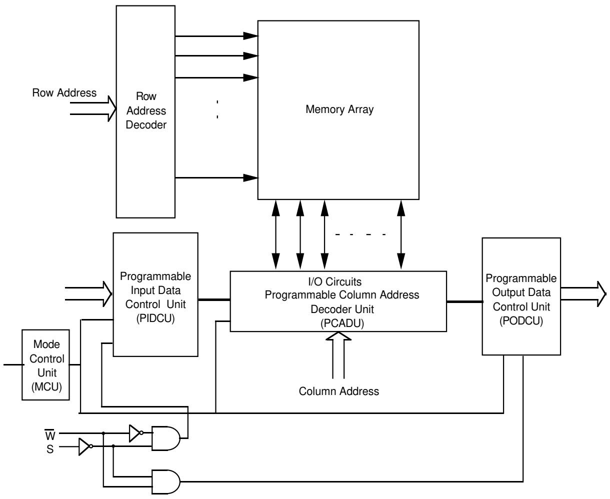

Figure 1: Architecture of VaWiRAM

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 1M X 1 chip which is reconfigurable to 512k X 2 bits, or 256k X 4 bits, or 128k X 8 bits, or 64k X 16 bits. Since we keep the row decoder to be fixed, the 1M X 1 chip can be viewed as one thousand 1k X 1 blocks. For this chip, maximum width, $W_{max} = 16$, minimum width, $W_{min} = 1$, 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 Fig. 2.a, and a layer of decoders in the second stage as in Fig. 2.b.

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-$2^{n-log(k)}$.

In this example, we set up the first stage as eight 2-to-1 muxes as in Fig. 2.a, and the second stage as sixteen 6-to-64 decoders as illustrated in Fig. 2.b. For illustration purposes, let A0, A1, A2, ...A19 denote the twenty address lines. In the $1M \times 1$ config-

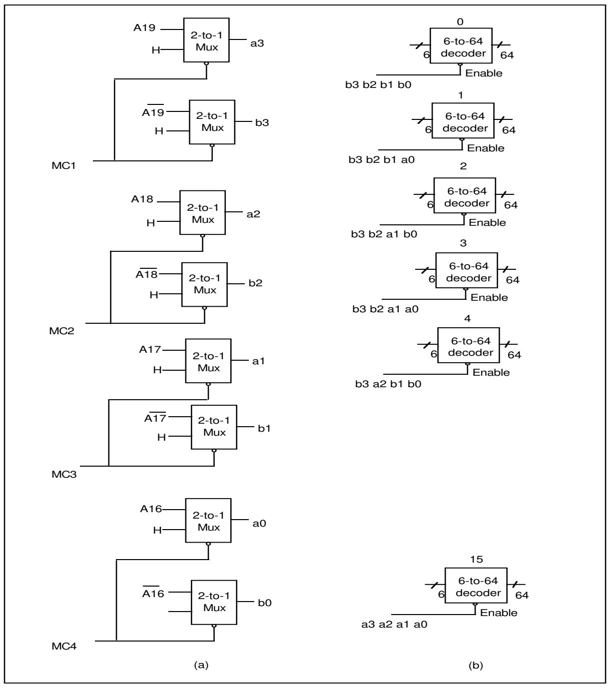

Figure 2: Programmable Address Decoders

uration, the lower 10 address pins A0, A1, ...A9 will be connected to the row decoder, and the higher 10 address bits, A10, A11, ... 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 inputs to all of these sixteen 6-to-64 decoders are connected to address lines A10, A11, ...A15. Only one of the sixteen 6-to-64 decoders will be enabled simultaneously 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 enabling of different numbers of decoders is shown in Fig. 2.a. The inputs and outputs of the circuitry in Fig. 2.a. are illustrated in Fig. 3.

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

Figure 3: Input and outputs of circuitry in Fig. 2.a

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 Fig. 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 for varying the width to k times.

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

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

$$\sum_{i=2}^{k} \frac{W_{max}}{i}.$$

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

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

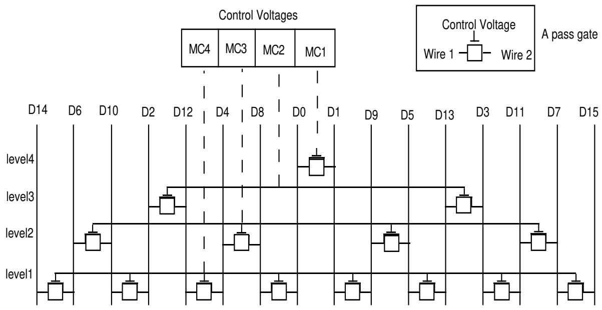

Figure 4: Circuitry for reconfiguring data width

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

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

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

Control Values				Resulting Configuration
MC4	MC3	MC2	MC1	
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

Figure 5: Values for the select signals for the 5 different configurations

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 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 Fig. 6. There are ways to overlap the functions of some of the mode control pins with the address pins, but we postpone the discussion till 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. If the capability to be used in a wide configuration is required, it is imperative to have the data pins. 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. Fig. 7 illustrates the 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, A19) is not required for addressing. So the highest address line can be used for mode information. If that line is 0 (while mode line is 1), the chip is twice as wide as the narrowest configuration. If A19 is a 1, the chip is even wider. So A18 is not required for addressing and will be used for mode information. If A18 is 0, the chip is 4 times as wide as the narrowest configuration. If A18 is 1, the chip is wider than 4 times the original width. So A17 will not be required for addressing and can be used for mode information. If A17 is a 0, the chip is 8 times the original width and if A17 is a 1, the chip is 16 times the original width. Thus mode control can be accomplished by just one extra pin. The large number of pins may not be a serious problem in embedded RAMs and multichip modules, and VaWiRAMs are certainly a very advantageous option under those circumstances.

3.4.2 Area

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. Compared to the flexibility yielded by the design, the incurred overhead is negligible.

Inputs					Outputs				
M	A19	A18	A17	A16	MC4	MC3	MC2	MC1	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

3.4.3 Time

The pass gates and the multiplexers incorporated for the reconfigurability 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. For instance, the first stage in the PCADU will result in increasing the decoding delays by two gate delays. The PODCU also contributes one gate delay. Similarly, the MCU will contribute two gate delays, and if address pins are shared with the mode control pins, as many gate delays as the number of pins shared will further increase.

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 Fig. 8. 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 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.

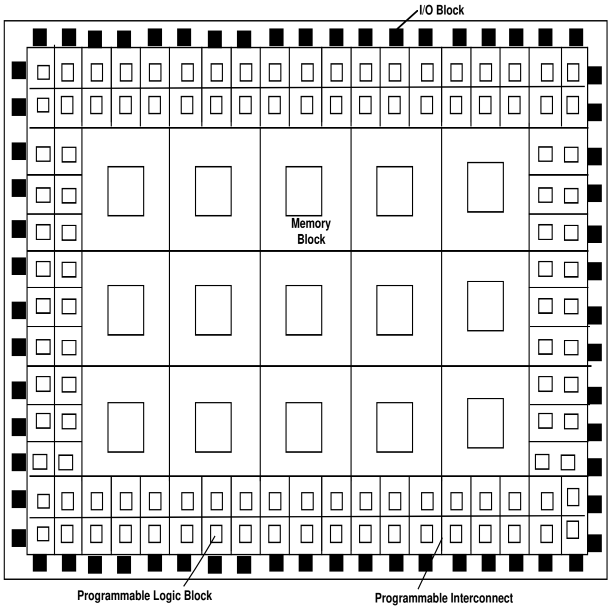

Figure 8: A Field Programmable Memory Cell Array (FPMCA) with a Sea of Memory Blocks

Partitioning, placement and routing algorithms developed in connection with FPGAs may be used for programmable memory arrays 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 Fig. 9, can be designed.

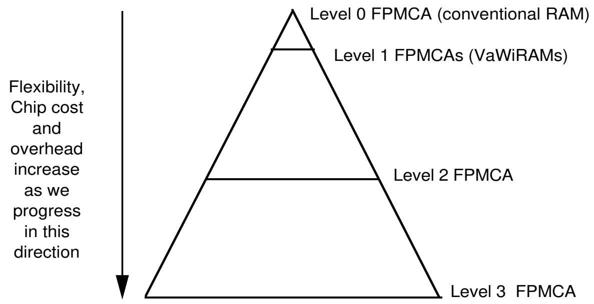

Figure 9: The spectrum of programmable memory cell arrays

Depending on the amount of programmable interconnect present within the chip, we categorize FPMCAs to various levels as follows.

(i) Level 0 is the 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.

(ii) Level 1 FPMCA is RAM with very minimal programmability. A chip reconfigurable between widths

1 and W_{max} can be constructed using $W_{max} - 1$ pass gates, $\frac{W_{max}}{2}$ 2-to-1 multiplexers, and $\lceil log_2 \lceil log_2(k) + 1 \rceil \rceil$ mode pins. In fact, Level 1 is the same as VaWiRAM.
(iii) Level 2 FPMCAs contain a programmable memory array in addition to the width as in VaWiRAMs. But each of the memory blocks is not a single cell, it is a macrocell consisting of a two-dimensional array of individual cells. Overhead decreases as the size of the block increases.
(iv) Level 3 FPMCAs contain a highly programmable memory array in which each memory block is an individual cell. As expected, the overhead is going to be extremely high, and may possibly be 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. It may be noticed that this classification does not consider the programmable logic on the chip.

The FPMCAs will be extremely useful for prototyping large memory systems. Field Programmable Interconnect Devices (FPIDs) can be combined with FPGAs and memory to realize field programmable boards. A set of programmable boards could be plugged into a programmable backplane to let product developers easily customize or change functionality at the system level. 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 itself to desired widths. A VaWiRAM chip reconfigurable between widths 1 and W_{max} can be constructed using $W_{max} - 1$ pass gates, $\frac{W_{max}}{2}$ 2-to-1 multiplexers, and $\lceil log_2 \lceil log_2(k) + 1 \rceil \rceil$ mode pins. This paper discussed the architecture of the proposed Variable Width RAMs (VaWiRAMs), and analyzed the trade-offs in their design. We also discussed a more powerful extension to the VaWiRAM, the Field Programmable Memory Cell Arrays (FPMCAs) which are reconfigurable memory arrays with additional embedded logic to realize the interface with other system components using 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.

References

[1] Aptix, The Programmable Interconnect Company Data Book, February 1993.

[2] P.M. Athanas, H.F. Silverman, "Processor Reconfiguration Through Instruction Set Metamorphosis", Computer, March 1993, p. 11–18.

[3] P.Bertin, D.Roncin, J.Vuillemin, "Introduction to Programmable Active Memories", PRL Report 3, Digital Equipment Corporation, Paris Research Lab, 1989.

[4] P.Bertin, D.Roncin and J.Vuillemin, "Programmable Active Memories: Performance Measurements" ACM/SIGDA Workshop on Field Programmable Gate Arrays, 1992, p. 57 – 59.

[5] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, Field Programmable Gate Arrays, Kluwer Academic Publishers, 1992.

[6] P.K.Chan, M.D.F. Schlag, and M.Martin, " BORG: A Reconfigurable Prototyping Board using Field-Programmable Gate Arrays", ACM/SIGDA Workshop on Field Programmable Gate Arrays, 1992, p. 47 – 51.

[7] I. Dobbelaere, A.E. Gamal, D. How, and B.Kleveland, "Field Programmable MCM Systems — Design of an Interconnection Frame", ACM/SIGDA Workshop on Field Programmable Gate Arrays, 1992, p. 52 – 56.

[8] B. Prince, "Memory in the fast lane", IEEE Spectrum, Feb. 1994, pp. 38-41.

[9] B. Prince, Semiconductor Memories, John Wiley and Sons, 1991.

[10] M.Gokhale, W.Holmes, A.Kosper, S.Lucas, R.Minnich, D.Sweely, and D.Lopresti, "Building and Using a Highly Parallel Programmable Logic Array", IEEE Computer, January 1991, p. 81–89.

[11] D. A. Hodges, Semiconductor Memories

[12] Special Report on Memory Systems, IEEE Spectrum, October 1992, pp. 34-57.

[13] W. J. Hsieh, "Reconfigurable Hardware", Computer Design, March 1993, pp. 27-30.

[14] Motorola Fast SRAM data book, 1993.

[15] Oooshi, Asakura, Hidaka, Arimoto and Fujishima, "Stretchable memory-matrix DRAMs", IEICE Transactions on Electronics, vol. E75-C, No. 11, November 1992, pp. 1323-1332.

[16] S. Przybylski, Cache and Memory Hierarchy Design: A Performance-Directed Approach, Morgan Kaufman, 1990.

[17] R. A. Quinnell, High Speed Bus Interfaces, Electronic Design News, Sept. 30, 1993, p. 42

[18] D. Tuite, Cache architectures under pressure to match CPU performance, Computer Design, March 1993, pp. 91-97.

[19] B. Williams, "Synchronous DRAMs: Designing to the JEDEC Standard", Micron Design Line, Vol. 2, Issue 2, 1993.