INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center Street • Suite 600 • Berkeley, California 94704 • 1-510-642-4274 • FAX 1-510-643-7684

Recent Work in VLSI Elements for Digital Implementations of Artificial Neural Networks

Brian E. D. Kingsbury * †
Bertrand Irissou †
Krste Asanović * †
John Wawrzynek †
Nelson Morgan * †

TR-91-074

December 1991

Abstract

A family of high-performance, area-efficient VLSI elements is being developed to simplify the design of artificial neural network processors. The libraries are designed around the MOSIS Scalable CMOS design rules, giving users the option of fabricating designs in 2.0 μm or 1.2 μm n-well processes, and greatly simplifying migration of the libraries to new MOSIS technologies. To date, libraries and generators have been created for saturating and nonsaturating adders, a two's-complement multiplier, and a triple-ported register file. The SPERT processor currently being designed at ICSI will be based upon these libraries, and is expected to run at 50 MHz when realized in a 1.2 μm CMOS technology.

*International Computer Science Institute, 1947 Center Street, Berkeley, CA 94704
†Computer Science Division, EECS Department, University of California at Berkeley, Berkeley, CA 94720

1 Introduction

Many popular "connectionist" algorithms have features which make them amenable to implementation in digital VLSI, particularly the parallelism inherent in these algorithms and their low requirements for precision and dynamic range. Special-purpose processors designed to take advantage of these features promise significant improvements in cost/performance over both conventional computers and special-purpose machines built from commercially-available processors. For example, simulations predict that a system centered around the SPERT chip [ABK+91], a VLIW/SIMD processor currently being developed at ICSI, will sustain over 300×10^6 connections per second during pattern classification and about 100×10^6 connection updates per second during back-propagation training. This is a level of performance comparable to that available from a 40-node RAP multiprocessor [MBAB90]. However, the SPERT-based system will reside on a double-slot SBus card in a Sun SPARCstation, while the 40-node RAP system occupies ten 9U VME boards.

Although there are many benefits to the implementation of connectionist algorithms in digital VLSI, little research has been done in this direction. The design of a full-custom VLSI processor is a difficult undertaking, requiring a large investment of time and effort and "tall thin designers" who understand the application of the algorithm, the particulars of the algorithm itself, the architecture of the processor which will execute the algorithm, and the details of the VLSI circuits which make up the processor. The amount of time and effort required to produce a design can be greatly reduced, for a slight cost in area and/or speed, by using a semi-custom design. In a semi-custom design, a system is assembled from high-level building blocks (macrocells) such as PLA's, RAM's, adders, and multipliers. This greatly reduces, or even eliminates, the low-level circuit and layout design which consume much of the time and effort in a full-custom design. The Lager silicon compiler [JRB86, RPB85] is a notably successful example of a system for doing semi-custom VLSI designs based on the macrocell approach.

This report describes our work on the development of high-performance macrocell libraries for the design of "connectionist" processors. Section 2 outlines the design methods, and Section 3 describes the implementation strategy used in library development. Some details of the blocks developed are given in Section 4, and our future goals for the project are outlined in Section 5. Detailed schematics of the adders, multiplier, and register file are includes in Appendices B, C, and D respectively.

2 Design Methods

We have developed a coherent method for designing a macrocell which ensures that all criteria for the block are met. First, a timing specification based on the system clock speed, number of pipe stages allotted to the block, and the maximum size of the block is created. Next, an architectural and schematic design of the block is composed. This is often an iterative process in which several architectures are partially specified at the transistor level and then simulated to determine which ones are sufficiently fast. The SPICE-compatible circuit simulator CAzM[1] is typically used for detailed timing analyses. Those designs which are found to be fast enough are then evaluated on the basis of silicon area required, complexity of implementation, and power consumption, and a final design for the block is selected. A complete, transistor-level schematic is then constructed using the schematic entry tools in the Viewlogic[2] CAD system. Transistor sizes are adjusted and checked by simulation with CAzM, and then IRSIM, a switch-level simulator, is used to verify that the design is functionally correct. Once a correct transistor-level schematic is obtained, the leaf cells of the block are drawn using the Magic 6.0 layout editor. Leaf cells are checked by extracting SPICE-level representations from layout and simulating. The leaf cells are converted to Oct [HMSN86, Spi90] physical format and an Oct-based C++ program is written to generate layout of the block by tiling together leaf cells. In general, the program is able to generate many different versions of the block, based on the width (in bits) of the input(s) to the block. Finally, the complete block layout is checked for functional correctness and speed by simulating an extracted circuit model.

Once a macrocell has been designed, fabricated, and tested to ensure that it does indeed meet all specifications, the cell libraries and layout generation routines are passed on to the Lager research group for incorporation into the Lager silicon compiler. Ulti-

[1] CAzM is a product of the Microelectronics Center of North Carolina (MCNC).

[2] Viewlogic is a registered trademark of Viewlogic Systems, Inc.

Figure 1: Floorplan for a single datapath cell

3.2 Floorplan

It is necessary to choose a consistent floorplan for macrocell components that are designed to be tiled together to form a datapath. Figure 1 illustrates the scheme we have chosen. All datapath cells are 56λ high; have inputs and outputs in metal1[3]; have Vdd, GND, clock, and control lines running in metal2 perpendicularly to data flow; and are designed so that additional data buses running in metal1 parallel to data flow can be placed between adjacent cells. Our selection of these parameters is based primarily upon experience—we have found this style to work well for the blocks we want to build. The 56λ pitch of our datapath cells is neither so great that simple cells like latches contain wasted space nor so small that more complex cells cannot be laid out easily. Our assignment of data buses and signal wires to metal1 and clock, power, and control wires to metal2 is based on the results of trial layouts using several different layout styles and layer assignments. We found that we could get compact layouts using the layer assignment described above. This layer assignment allows the use of wide power and ground wires which reduces IR drops, and it also minimizes coupling between successive stages in the pipeline through the power distribution network because successive stages usually do not share local power and ground lines.

[3] Metal layers are numbered in order of fabrication; thus, the metal1 layer is closer to the substrate than the metal2 layer. Our cells are targeted for processes with at least two metal layers.

Figure 3: Block diagram of simple 4-bit bitslice of the saturating adder.

Figure 4: Layout of two-phase 24-bit saturating adder

Figure 5: 5×5 Pezaris array. (x,y) denotes A_x AND B_y. To calculate $A \times B + C$, grounded inputs are connected to C. Outputs $P < 4:0 >$ are lower bits of the final product; outputs $N < 3:0 >$ and $O < 3:0 >$ need to be added together to get final result.

S Sum block.

SAT Saturation detection and buffering block.

SGN Sign-bit buffering block.

MX Multiplexer.

MH Multiplexer.

MT Multiplexer.

The BN, RN, BP, and RP blocks form the two-level binary tree used for carry lookahead within the slice. The CR blocks calculate the carries within the slice, while the MC block is used to calculate the carry for the next slice. The S block computes the sum, using the propagate and carry signals. The SAT block detects over/underflow by taking the XOR of the carry in and carry out of the most-significant bit: if the carry in and out are the same, the sum computed by the S blocks is valid; otherwise, saturation must take place. The saturation control signals are also buffered within the SAT block. The sign bit, needed for saturation, is buffered by the SGN block. The MX, MH, and MT blocks output either the sum from the S blocks or the sign bit (or its inverse, depending on the bit position), with the output controlled by the saturation control lines sat and $\overline{sat}$. A non-saturating adder may be constructed by tiling together simple slices without the multiplexers MX, MH, and MT.

Unlike many saturating adder designs, all of the saturation detection and control circuitry is internal to the main body of the adder. This allows datapaths containing saturating adders to be cleanly tiled together in SIMD arrays of datapaths.

A two-phase 24-bit saturating adder has been fabricated on a MOSIS Tiny Chip in 2.0 μm CMOS, and is fully functional at 25 MHz. The layout of this adder appears in Figure 4. A TSPC version of the saturating adder is almost ready for fabrication.

4.2 Multiplier

A two's-complement multiplier has also been developed. A 24×8 version of this multiplier will operate at 50 MHz when implemented in a 1.2 μm process. The multiplier requires two cycles to complete a multiply, but has a throughput of one multiply per cycle. It uses a Pezaris tri-section array [Hwa79] to generate and sum partial products; the architecture of this array is illustrated in Figure 5. The non-saturating

Figure 6: Layout of 4×4 multiplier array

adder described above is used to do the final addition. The partial product array is split in half, separated by a latch, so that the first 24×5 partial products are calculated and summed in one half-cycle, and the remaining 24×3 partial products are calculated and summed on the next half-cycle. A pipeline latch is also placed between the array and final adder. The array is fully static logic, to save on area and power consumption (a precharged implementation would require fully complementary logic). The full adder circuits in the array are based on a transmission gate full adder described in [WE85]. This design was used because the delays through the carry and add paths are almost identical, and because the two variants of full adder used in the Pezaris array have almost identical layouts using transmission gates. Appendix C contains full schematics for the multiplier.

The layout of the multiplier is somewhat unusual because it had to fit into an array of SIMD datapaths; thus, both operands flow into the multiplier from one side, and the complete result is output at the other. A more common arrangement has the operands entering the multiplier on adjacent sides of the block and the result appearing at the other two sides.

A 4×4 version of the array is out for fabrication on a MOSIS Tiny Chip.

4.3 Register File

Two versions of a register file, a 16-register×128-bit array and 16-register×32-bit array are required for SPERT. Both are designed to operate at 50 MHz when fabricated with a 1.2 μm CMOS process. With appropriate transistor sizing changes, the same basic circuits and cell layouts can be used for both versions of the register files.

The register files can sustain two reads and one write per cycle, with a one cycle latency for the read. The value of a read is undefined if the same register is written in the same cycle. The register file is split in two pipelined stages. The first stage comprises the read and write address decoders and the row drivers. The read and write addresses have to be asserted during the high phase of the clock and are latched during the low phase. The circuitry precharges during the high phase of the clock and evaluates during the low phase. By the rising edge of the clock, the row select line are stable and the select lines are latched. The second stage of the register file consists of the sense circuitry and the write circuitry. During the low phase of the clock, the bitlines are precharged high. Then, on the high phase of the clock, the bitcell pulls the rbit and rbitb bitlines that are read by the sense amplifiers. The data is then latched on the falling edge of the clock. For writing the cell, the write data is latched during the high phase of the clock and the write lines of the cell wbit and wbitb are pulled differentially. Figure 7 shows a detailed block diagram for the register file with one bitcell shown. The functions of the blocks are:

NLATCH Transparent latch on the high phase of the clock

PLATCH Transparent latch on the low phase of the clock

ADRDRV Differential address drivers

DECODER 16-bit address decoder

ROWDRV Row select line drivers

BITCELL 1-bit memory cell

SENSEAMP Single-sided sense amplifier

WDRV Differential write bitlines driver.

The basic memory cell consists of a static cross-coupled CMOS inverter pair with four access transistors. The write circuitry uses two ports to differentially drive the cell, while the sense amplifiers of the read ports are single-sided. Both the write and the sense circuits are dynamic. A high-threshold circuit in the form of cascaded precharged inverters are used as the read sense amplifier. This circuit senses small voltage changes on the bit-lines, allowing the use of reduced voltage swings, thus speeding operation and

Figure 7: Block diagram of register file

Figure 8: Floorplan of register file

reducing power consumption. The three register file decoders also use precharged circuits and are arranged in a "nand-tree" structure. The floorplan for the register files is shown in Figure 8. More details are available in [Waw92]. Schematics for the register file are included in Appendix D.

An 8-register×8-bit version of the register file has been fabricated on a MOSIS Tiny Chip in a 2.0 μm n-well process. It is fully functional at 20 MHz.

5 Future Directions

In the process of completing the SPERT implementation, libraries and generators for static and dynamic latches, a 32b shifter, a simple logic unit, comparator, and a 32b limiter will be created. Because we anticipate that future processors will require more on-chip storage than is needed in SPERT, development of a library for static RAM is underway.

We are also working with the Lager research group at UC Berkeley to make our macrocells available as Lager libraries. A version of the triple-ported register file has already been moved into Lager.

6 Summary

We have described our work on macrocell libraries for the implementation of artificial neural network algorithms. For word widths of up to thirty-two bits, the resulting macrocells will operate at 50 MHz when fabricated in a 1.2 μm process. A single-phase clocking strategy has been used to reduce the skew problems which often occur in high-speed CMOS designs, as well as to obtain greater efficiency in power, area, and speed. As a reasonable compromise between design effort and performance, custom design has been used for for critical leaf cells that are then tiled together by a C++ based layout generator. These blocks will ultimately be used for neurocomputing systems that we are designing, and will also be integrated into a higher level semi-custom design system: Lager.

7 Acknowledgements

The National Science Foundation provided support for the VLSI building blocks and design tools through Grant No. MIP-8922354, and also with Graduate Fellowship support for Brian Kingsbury. John Wawrzynek received support from the National Science Foundation through the Presidential Young Investigator (PYI) award, MIP-8958568. The larger project continues to be supported by the International Computer Science Institute.

References

[ABK+91] Krste Asanović, James Beck, Brian E. D. Kingsbury, Philip Kohn, Nelson Morgan, and John Wawrzynek. SPERT: A

VLIW/SIMD Microprocessor for Artificial Neural Network Computations. Technical Report TR-91-072, International Computer Science Institute, 1991.

[AS90] Morteza Afghahi and Christer Svensson. A Unified Single-Phase Clocking Scheme for VLSI systems. *IEEE JSSC*, 25(1):225–233, February 1990.

[HMSN86] David S. Harrison, Peter Moore, Rick L. Spickelmier, and A. Richard Newton. Data Management and Graphics Editing in the Berkeley Design Environment. In *Proc. of the International Conference on Computer Aided Design*, pages 20–24, November 1986.

[Hwa79] Kai Hwang. *Computer Arithmetic: Principles, Architecture, and Design.* John Wiley and Sons, 1979.

[JRB86] R. Jain, P. Ruetz, and R. Brodersen. Architectural Strategies for Digital Signal Processing Circuits. In S.Y. Kung, R.E. Owen, and J.G. Nash, editors, *VLSI Signal Processing II.* IEEE Press, 1986.

[MBAB90] N. Morgan, J. Beck, E. Allman, and J. Beer. RAP: A Ring Array Processor for Multilayer Perceptron Applications. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, & Signal Processing*, pages 1005–1008, Albuquerque, New Mexico, USA, 1990.

[RPB85] J.M. Rabaey, S.P. Pope, and R.W. Brodersen. An Integrated Automated Layout Generation System for DSP Circuits. *IEEE Transactions on CAD*, CAD-4(3):285–296, July 1985.

[Spi90] Rick L. Spickelmier. Oct Tools Distribution 4.0, August 1990.

[Waw92] J. Wawrzynek. A 250MHz 64-bit Datapath in 1.2 μm CMOS. Technical Report In Preparation, Computer Science Division (EECS), University of California, Berkeley, 1992.

[WE85] Neil Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley, Reading, MA, 1985.

[YS89] Jiren Yuan and Christer Svensson. High-Speed CMOS Circuit Technique. *IEEE JSSC*, 24(1):62–70, February 1989.

Figure 9: Symbols used in schematics for adder and multiplier

Figure 10: Generate/propagate unit (GP)

Figure 12: Reduction unit—N-type (RN)

Figure 14: Reduction unit—P-type (RP)

Figure 17: Sum unit (S)

Figure 21: Multiplexer unit (MH)

Figure 22: Multiplexer unit (MX)

Figure 24: And/nand unit

Figure 25: Latch

Figure 26: Decoder logic

Figure 27: Sense and write logic