# The Virtual Wires Emulation System: A Gate-Efficient ASIC Prototyping Environment

Russell Tessier, Jonathan Babb, Matthew Dahl, Silvina Hanono, and Anant Agarwal
MIT Laboratory for Computer Science
Cambridge, MA 02139

## Abstract

FPGA-based ASIC development systems have become important tools in contemporary ASIC design. Existing systems exhibit low per-FPGA gate utilization (10 to 20 percent) due to limited inter-chip communication. Attempts at overcoming this limitation through the use of high dimensional interconnection topologies have met with limited success.

This paper focuses on the prototype hardware and software interfaces that have been developed for an FPGA-based ASIC emulation system based on a new technique for overcoming inter-chip communication limitations. This technique, referred to as *virtual wires*, intelligently multiplexes each physical FPGA wire among a number of logical wires. The Virtual Wires Emulation System exhibits high FPGA gate utilization while achieving system speeds comparable to existing logic emulators. A two-dimensional mesh interconnection topology of FPGAs is used to eliminate the cost of signal switching elements and to facilitate scalability.

A system capable of emulating 20,000 gates has been constructed for under $3000. This system includes both prototype emulation hardware and a Virtual Wires netlist compiler. Currently, this system is being used as both a simulation accelerator for the LSI Logic LSIM and Cadence Verilog simulators and as an in-circuit emulator. Results from mapping netlists, such as the 18K gate Sparcle microprocessor [2], to this system for simulation acceleration and in-circuit emulation indicate that virtual wires can substantially increase FPGA utilization without adversely affecting emulation speed.
*Keywords: FPGA, logic emulation, prototyping, mesh topology, static routing, virtual wires.*

## 1 Introduction

### 1.1 Virtual Wires

Field Programmable Gate Array (FPGA) based logic emulators are capable of emulating complex logic designs at clock speeds four to six orders of magnitude faster than even accelerated software simulators. This performance is achieved by partitioning a logic design across an interconnected array of FPGAs which is directly wired into a target system. The logic partition on each FPGA is then executed at hardware speeds.

Existing FPGA-based logic emulators suffer from limited inter-chip communication bandwidth, resulting in low gate utilization. This resource imbalance increases the number of chips needed to emulate a particular logic design and thereby decreases emulation speed, since signals must cross more chip boundaries. Current emulators only use a fraction of potential communication bandwidth because they dedicate each FPGA pin (physical wire) to a single emulated signal (logical wire). These logical wires are not active simultaneously and are switched at emulation clock speeds.

*Virtual wires* [4] overcome pin limitations by intelligently multiplexing each physical wire among multiple logical wires and pipelining these connections at the maximum clocking frequency of the FPGA. A virtual wire represents a connection from a logical output on one FPGA to a logical input on another FPGA established via a pipelined, statically-routed communication network. Virtual wires not only increase usable bandwidth, but also relax the absolute limits imposed on gate utilization. The resulting improvement in bandwidth reduces the need for global interconnect, allowing effective use of low dimension inter-chip connections (such as nearest-neighbor). Nearest-neighbor topologies, coupled with the ability of virtual wires to overlap communication with computation, can even improve emulation speeds.

### 1.2 The Virtual Wires Emulation System

A development system based on virtual wires, the Virtual Wires Emulation System, has been developed and is currently being tested. System software provides a path for the translation of an ASIC netlist into a set of FPGA partitions which communicate through the use of virtual wires. The target for these partitions is one or more FPGA-based circuit boards designed to emulate both ASIC logic and memory. Inter-chip FPGA I/O signals are connected in a nearest-neighbor, two-dimensional mesh leading to straightforward

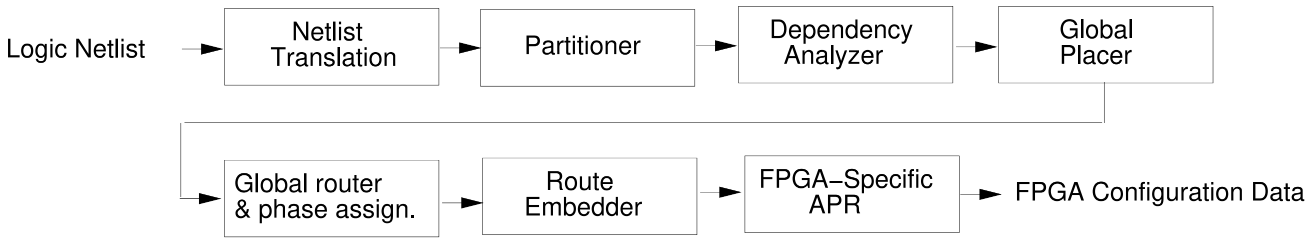


Figure 1: Virtual Wires Software System

software mapping and simplified system scalability.

This emulation system may be used for either in-circuit ASIC emulation or ASIC simulation acceleration. In-circuit emulation involves the physical connection of a prototyping pod from the emulation system to the ASIC socket of the target system. ASIC simulation acceleration provides for the replacement of the ASIC simulation model in a simulator run by a remote call to the system emulating the ASIC. ASIC inputs from the simulator are passed to the emulation system and evaluated outputs are subsequently returned from the emulation system to the simulator.

An entire emulation hardware and software system has been developed and is operational. This system is currently being used as a simulation accelerator and an in-circuit emulator for sequential circuits of about 20,000 gates.

### 1.3 Background

The concept of reconfiguring digital hardware to accelerate computation has been present for a number of years [5]. Even before FPGAs existed, logic emulation based on *cellular arrays* was explored in academia [12]. This work could not be applied to logic emulation without the enabling technology of large scale integration found in today's FPGAs.

FPGA-based logic emulation systems have been developed for design complexity ranging from several thousand to several million gates. Quickturn Inc. [13] [11] has developed emulation systems which interconnect FPGAs in a two-dimensional mesh and, more recently, in a partial crossbar topology. The Quickturn Enterprise system uses a hierarchical approach to interconnection. The Virtual ASIC system by InCA [7] uses a combination of nearest neighbor and crossbar interconnect. Like Quickturn's systems, Virtual ASIC logic partitions are hardwired to FPGAs following partition placement. The AXB-AP4 by Aptix Corporation [3] uses field programmable interconnect chips (FPICs) for communication between small groups of FPGAs. This board has an area reserved for adding external logic modules not capable of fitting within the FPGAs.

### 1.4 Overview

The remainder of this paper is organized as follows: Section 2 gives an outline of the hardware and software components of the Virtual Wires Emulation System. Section 3 provides a description of the architecture of the hardware system and its interfaces to system software. In Section 4 we present the results of mapping several circuits to the emulation hardware. Finally, in Section 5 we provide a summary of our work and indicate future research directions.

## 2 System Overview

The software portion of the Virtual Wires Emulation System includes a number of tools to translate a design into a group of interconnected FPGA partitions that contain embedded routing circuitry. Input netlists are first partitioned using the Concept Silicon netlist partitioner by InCA [7]. Signal dependencies between partitions are then analyzed and logic partitions are assigned to physical FPGAs using a placement algorithm based on simulated annealing [8]. The Virtual Wires router is used to statically schedule and route inter-FPGA communication. Gates needed to perform this routing inside the FPGAs are added to the partition netlists. Each logic partition is subsequently translated into gates in the target FPGA technology and mapped to silicon by a vendor specific place and route tool. This generates a configuration bitstream which may be downloaded to the emulation hardware. For more information on the Virtual Wires software environment see [4].

The hardware component of a Virtual Wires emulation system consists of one or more FPGA-based emulation boards interfaced to a host workstation. Each FPGA-based board contains sixteen Xilinx XC4005 FPGAs [14] interconnected in a two-dimensional nearest-neighbor mesh. System size can be scaled by attaching additional boards to the existing system via connectors on any of the four sides of the current system boards thus extending the two-dimensional mesh either horizontally or vertically. Each emulation board can communicate with a host workstation through either a serial or parallel communications port. These interfaces may be used to observe emulation status and provide circuit inputs and outputs.

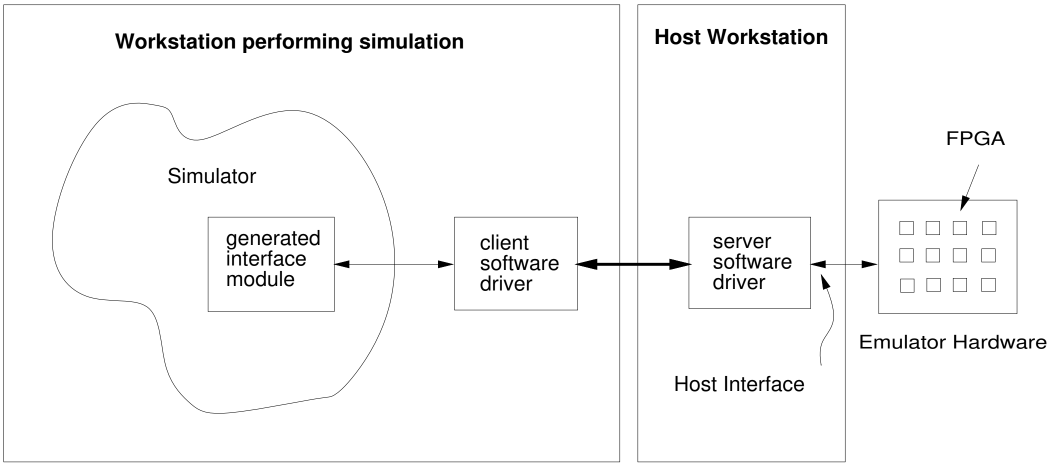


Figure 2: Simulator Interface

The hardware system may be utilized differently at distinct points of ASIC development. Initially, it may be configured to perform accelerated simulation of an ASIC in a target system simulation. Alternatively, it may be configured as a logic emulator to take the place of an ASIC in a target circuit.

### 2.1 Simulation Acceleration

Prior to performing in-circuit ASIC emulation in a target system it is desirable to simulate the logical behavior of an ASIC either individually or as a part of a target system simulation. Current simulation technology limits comprehensive testing of this type due to the slow gate evaluation speed of contemporary simulators. Our prototype system addresses this limitation by allowing the logical behavior of the ASIC to be *emulated* while the rest of the system is *simulated*.

Simulation is accelerated by replacing the ASIC simulation model in a standard simulator run with a remote call to the emulation system. Data inputs to the ASIC which would otherwise be evaluated by the simulation model are instead passed to the FPGA-based hardware through a host workstation and emulated. Results of this evaluation are subsequently returned to the simulator for graphical display or for use by other portions of the system simulation. This technique of remote access to the emulation system works particularly well for synchronous logic since evaluation times may be limited to transitions of the emulated system clock.

A simulation interface has been implemented for both the LSI Logic LSIM and Cadence Verilog simulators. At a given simulated clock edge, data representing ASIC inputs are transferred by the simulator to the host workstation via a set of software drivers. This data is subsequently forwarded to the emulation system from the host workstation via a parallel or serial host interface. Results are generated and then returned to the host workstation through the same data link. These results are then passed back to the simulator via the drivers for display or further simulation.

### 2.2 Emulation Interface

The ultimate goal of this project is to perform in-circuit emulation of the Alewife Communications and Cache Controller [9], an 86K gate ASIC under development at MIT, using prototype emulation hardware. Following netlist compilation by Virtual Wires software, a partitioned and translated ASIC design may be downloaded to the emulation system and set into operation.

A target system interfaces with the emulation system via a prototyping pod which plugs into the ASIC chip socket in the target system. After FPGA configuration, the emulation system exchanges data with the target system at each emulation clock while performing evaluation of logic and internally transferring results at FPGA device speeds. Prototyping pods interface to the emulation system through the same connectors along the side of the board that may be used to add additional boards to the system. Each connector provides a direct interface to the Xilinx array for that board.

Results of ASIC emulation may be observed by using the state readback feature of the FPGAs. At any point during emulation an FPGA may be triggered to return its current state to the host computer for evaluation. Software analysis of current state can be performed using this data.

## 3 Hardware System Architecture

### 3.1 Overview

The hardware building block of the Virtual Wires system is an FPGA-based emulation board which has been designed to be inexpensive to manufacture and easy to build. Each

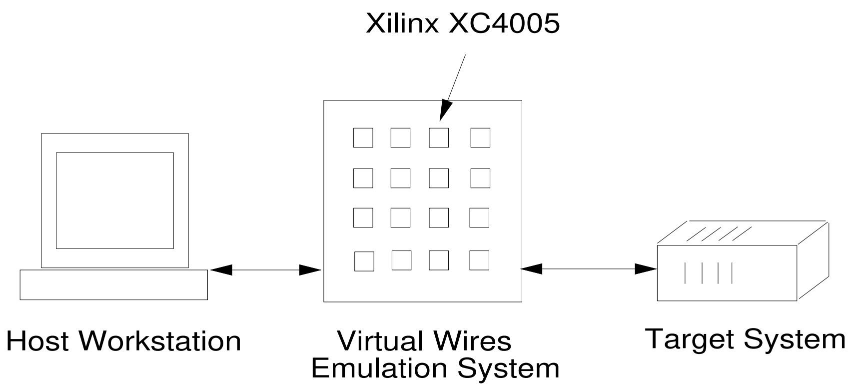


Figure 3: Virtual Wires Emulation System

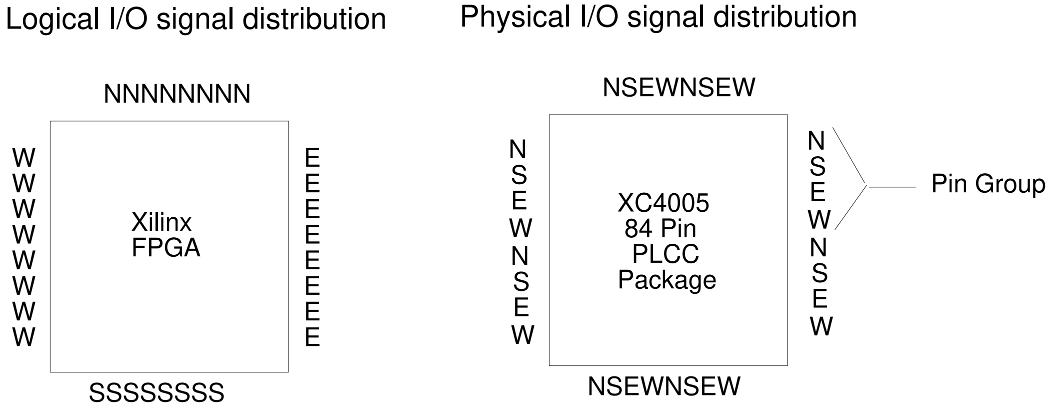


Figure 4: FPGA Pin Permutation

emulation board contains six layers, uses only through-hole devices, and is ten inches square in size. At the present time we use a SparcStation 10 as the host interface for this system although the emulation board may be reconfigured to interface to virtually any host computer.

## 3.2 FPGA Array

Each prototype board contains sixteen Xilinx XC4005 FPGAs in 84 pin PLCC packages connected in a two-dimensional grid. Each FPGA communicates with its four nearest neighbors (logically North, South, East, and West) through eight bidirectional I/O signals. These groups of I/O signals are referred to as chip I/O ports.

While Virtual Wires software is optimized to reduce the amount of communication necessary between non-nearest neighbor FPGAs, some multiple-hop signal routing is still necessary. Data signals in this case must pass through intermediate FPGAs in the two-dimensional mesh when travelling from source to destination. Since pass-through signals exit intermediate FPGAs without modification, it is desirable to use minimal routing resources within these FPGAs to form a path from signal input to output. In our system this is accomplished by distributing the signals of the logical I/O ports along the physical chip package periphery in a set pattern. As seen in Figure 4, I/O port signals are physically allocated so that adjacent I/O pins are assigned to signals from differing ports. A regular pin assignment grouping of signals from logical ports North, South, East, and West along adjacent I/O pins is used. With this permutation a signal passing through the chip need only be routed a length of several pins rather than across the body of the entire chip. Note that two pin groups are located along each side of the package.

One of the benefits of this emulation system is its simplicity. Costs are reduced by eliminating the need for expensive interconnect chips such as FPICs or custom crossbars. All statically-routed inter-chip communication is performed by inter-FPGA logic which has been synthesized at compile time.

Low pin-count FPGAs in the form of 84 pin PLCCs are used in this system to emphasize the utility of virtual wires as a communication technique and also to reduce the cost of the resulting system. These FPGAs may be clocked at speeds approaching 40 MHz thus resulting in small interchip delays and high emulation throughput.

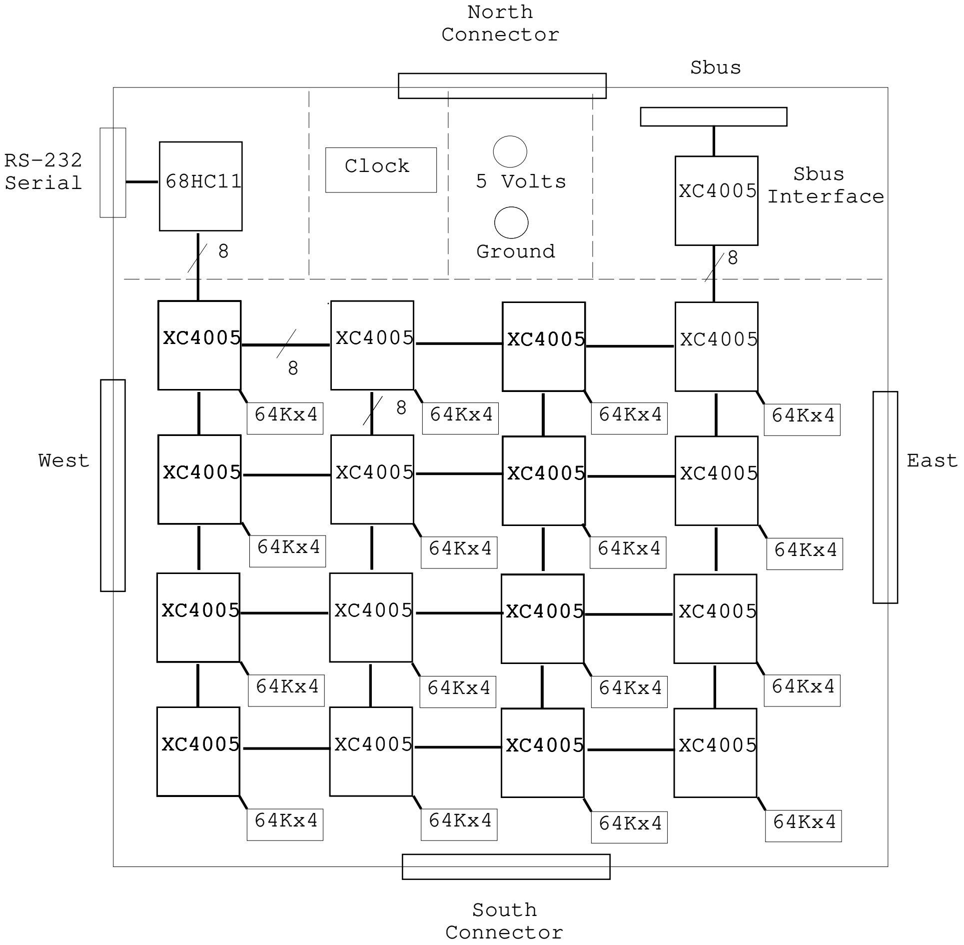


Figure 5: Virtual Wires Emulation Board

## 3.3 Scaling to Multiple-Board Systems

Multiple-board systems may be constructed by connecting individual boards together to form a two-dimensional mesh. FPGA configuration information is transferred in a serial chain to all FPGAs in the system starting at the board connected to the download cable in the upper rightmost corner of the mesh. All FPGAs in the system are forced to remain in configuration mode until the last FPGA has been configured. At this time all FPGAs are allowed to assume their programmed functionality. Currently, configuration data is downloaded to the emulation system through a download cable purchased from Xilinx.

Bidirectional FPGA I/O signals along the periphery of each board extend across connectors in each of the four logical directions thus allowing system scalability. Figure 6 shows how FPGAs are connected logically for a system consisting of four emulation boards.

System size is constrained only by the size of the configuration bitstream for the Xilinx chips. Since up to 160 chips may be configured serially per bitstream, a total of ten

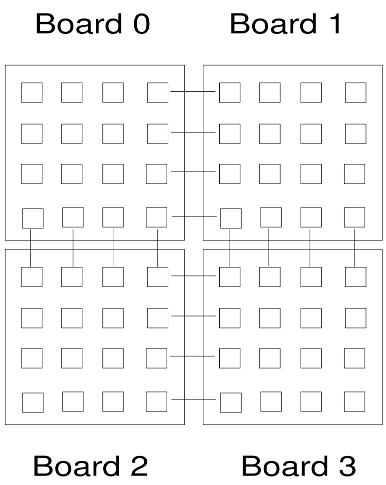


Figure 6: Four Board Emulation System

boards may be connected together for this system.

Each Virtual Wires prototype board can function as either a standalone system or as part of a larger group of boards. A clock driver chip and remote leads for clock cables may be populated to allow one board to serve as the global clock source for the other boards. All boards in the system operate on the same system clock. Clock signals are fanned out to local logic at the destination boards using a clock distribution chip.

### 3.4 Memory Interface

Each FPGA in the two-dimensional mesh has twenty-two dedicated I/O lines which interface to a 64Kx4 SRAM chip. These chips can be used to emulate sections of ASIC on-chip memory. SRAM chips may be populated as needed or left off the board entirely if not used. Virtual Wires software is used to multiplex address, data, and control signals for the SRAM so that a number of individual SRAM accesses to the same SRAM chip may take place during each emulation clock cycle. Twenty nanosecond SRAMs are used in the current prototype. SRAM interface signals have been allocated to dedicated FPGA pins to reduce capacitive loading on inter-FPGA signal lines and to simplify system software.

### 3.5 Host Interfaces

Low bandwidth communication between the host computer and the emulation system occurs through a serial interface implemented using a MC68HC11 embedded controller. This interface provides low bandwidth access to the array for data transfer, configuration, and FPGA state readback. Data signals from the MC68HC11 interface directly to the North port of the Xilinx chip in the upper left corner of the array. Although not yet implemented, the embedded controller has the capability to download configuration information to the array thus eliminating the need for an additional download cable from Xilinx.

One of the features of the serial interface is the capability to perform readback from FPGAs on a chip by chip basis. The MC68HC11 can be programmed to trigger a readback bitstream from any FPGA in the system and subsequently transfer the values back to user software on the host workstation.

To provide a higher bandwidth interface to the host an additional Xilinx XC4005 chip serves as an intermediary between an Sbus interface card located in the host Sparcstation and the Xilinx array. This chip is configured to transfer words of data between the Sbus card [6] and the eight bit North port of the Xilinx chip in the upper right position of the array. Estimated transfer rate is about one Mbyte per second. This interface chip may be reconfigured to interface to other types of buses as well. In addition to serving as a bus interface, this chip can be used to trigger global signals for the array such as a global system reset.

| Statistic | Palindrome | Sparcle |
|---|---|---|
| LSI Gate Count | 14,241 | 17,252 |
| Element Count | 4,623 | 4,802 |
| Element Complexity | 3.1 | 3.6 |
| Memory Bit Count | 0 | 4,352 |
| Net Count | 4,626 | 5,094 |
| Pins per Net | 2.900 | 3.484 |
| FPGAs | 16 | 20 |
| Avg. Gates / partition | 890 | 868 |
| Avg. IO / partition | 45 | 126 |
| Mesh Emulation Speed (MHz) | 1.00 | 0.12 |

Table 1: Hardware Statistics for Simulation Acceleration

## 4 Results and Status

We have implemented the complete emulation system described, including the Virtual Wires compiler, simulator interfaces, and hardware prototype.

With our initial software system we have successfully compiled designs up to 18K gates for this system, and expect to be able to pack at least 20K gate designs into a single emulation board. Our current compile times are roughly 3 to 4 hours on a Sparcstation 10. However, 90 percent of this time is consumed in the vendor specific FPGA compile, which must be processed for each FPGA. These FPGA compiles are independent processes and can be run in parallel on a network to speed up total compilation time.

In this section we present results from the successful compilation of two medium size designs, Palindrome and Sparcle. Palindrome is a 15K gate systolic circuit for detecting strings that read the same both forward and backwards - for example 'AHA'. Sparcle is an 18K Sparc microprocessor with modifications to enhance its usefulness in a multiprocessor environment. Table 1 presents some basic statistics for these two designs when mapped to the emulation system for simulation acceleration. We have successfully used the system to accelerate circuit simulation of these two designs. It should be noted that mesh emulation speed refers to the evaluation rate of the emulation hardware rather than the simulation rate. The latter rate is typically limited by the speed of the simulation software or the bandwidth across the host interface. Simulation acceleration of Sparcle is driven by a behavioral model test shell running in the LSI Logic simulator [10]. The 136-word Sparcle register file is emulated by the on-board SRAM available on the Virtual Wires prototype board.

The emulation system also has been used in place of a

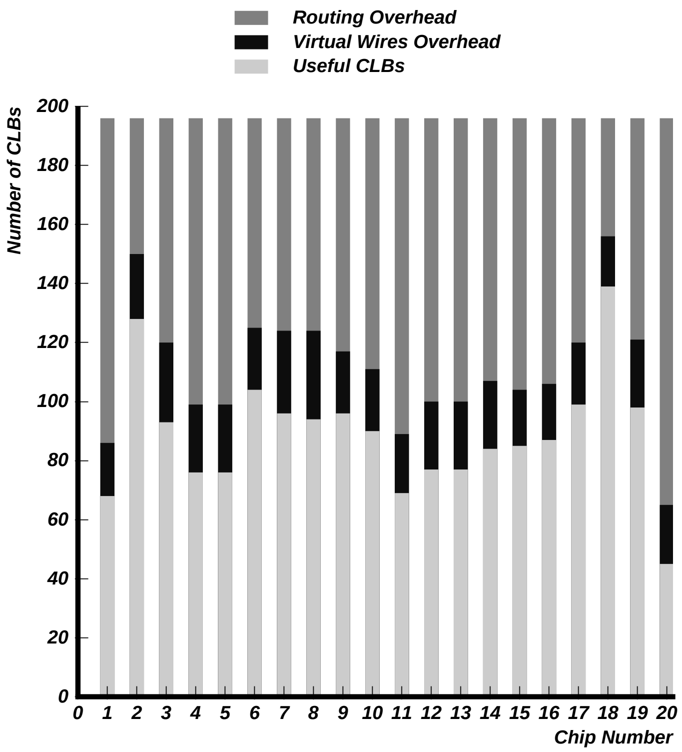


Figure 7: FPGA Resource Allocation for Sparcle

| Statistic | Sparcle |
|---|---|
| FPGAs | 24 |
| Avg. Gates / partition | 718 |
| Avg. IO / partition | 119 |
| Mesh Emulation Speed (MHz) | 0.18 |

Table 2: Hardware Statistics for In-circuit Emulation

Sparcle chip in a testbed board developed for the Alewife multiprocessor project. The emulator plugs directly into the Sparcle chip PGA socket using a commercially-built interface pod attached to the emulator board edge connectors. The emulator synchronizes automatically to the Sparcle system clock and control signals. Therefore, no modifications to the target system are needed other than slowing the system clock. Table 2 shows some basic statistics for Sparcle when mapped to the system for in-circuit emulation. The emulated Sparcle has executed Alewife system test programs successfully at speed, including booting of the prototype Alewife operating system.

Figure 7 shows the allocation of resources inside each FPGA for the Sparcle design. There is a fixed overhead of roughly 12 percent of the CLB's for virtual wires, as well as a small decrease in chip routability.

## 4.1 Pin Multiplication Factor

In [4] it was shown that logic designs, partitioned for given FPGA gate counts, exhibit partition pin counts which are typically greater than those found in FPGA packages. As a measure of system performance we define the *Pin Multiplication Factor (PMF)* as the effective pin multiplier for FPGA packages as a result of using virtual wires. By multiplexing package pins, FPGAs appear to have PMF times as many physical pins, thus allowing greater utilization of FPGA gates. The PMF, assuming a full crossbar interconnect, is formally defined as follows:

$$PMF_{crossbar} = V_{io}/P_{io} \tag{1}$$

where $V_{io}$ is the effective I/O of a logical partition, and $P_{io}$ is the physical pin limit. When the physical topology is a mesh, the PMF also includes the increase in pin multiplexing due to through wires, and is:

$$PMF_{mesh} = (V_{io} + 2 * H_{io})/P_{io} \tag{2}$$

where $V_{io}$ is the effective I/O of a logical partition, $H_{io}$ is the number of through wires in a partition, and $P_{io}$ is the physical pin limit. The factor of two is due to the fact that each through wire must consume one pin to enter the FPGA and a second pin to exit. Figure 8 shows the PMFs for Sparcle for our current mesh interconnect in comparison with a theoretical full crossbar interconnect. The averages for

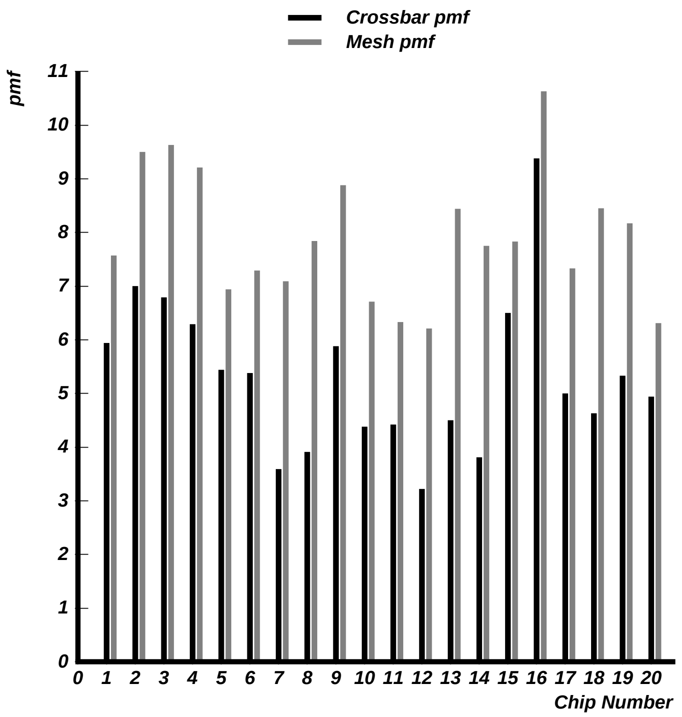


Figure 8: Pin Multiplication Factor for Sparcle

Sparcle are $PMF_{crossbar} = 5.32$ and $PMF_{mesh} = 7.91$. The mesh PMF must be higher than the crossbar PMF to allow for wires stretching across multiple FPGA hops.

# 5 Conclusions and Future Work

This paper describes the hardware portion of a project at MIT to produce a scalable, low-cost FPGA-based emulation system which maximizes FPGA resource utilization. While this paper has focused on the use of this system for simulation acceleration and logic emulation, we also intend to use this system to explore other areas of FPGA computing.

We have implemented the complete emulation system described, including the Virtual Wires compiler, simulator interfaces, and hardware prototype. By utilizing the pin multiplication effect of virtual wires in combination with a low dimensional interconnect, we have produced a logic emulation system capable of simulating/emulating 20K gates for less than $3000.

While our initial system is capable of emulating up to 1 Megabit of SRAM, the user must still hand-partition ASIC memory into emulation system SRAM. We are working on an automatic memory partitioner and mapper - our goal is a complete turnkey system for the end user. By using this compiler and the scalability of our current system, we plan to emulate circuits in the 100K+ gate range.

# 6 Acknowledgements

The research reported in this paper is funded by ARPA contract # N00014-91-J-1698 and NSF grant # MIP-9012773.

# References

[1] A. Agarwal et al. The MIT Alewife machine: A large-scale distributed memory multiprocessor. In *Scalable Shared Memory Multiprocessors*. Kluwer Academic Press, 1991.

[2] A. Agarwal, J. Kubiatowicz, D. Kranz, B.-H. Lim, D. Yeung, G. D'Souza, and M. Parkin. Sparcle: An Evolutionary Processor Design for Multiprocessors. *IEEE Micro*, June 1993.

[3] Aptix, Inc. *Aptix AXB-AP4 Data Sheet*, Oct. 1993.

[4] J. Babb, R. Tessier, and A. Agarwal. Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators In *IEEE Workshop on FPGAs for Custom Computing Machines*, Napa, California, April 1993.

[5] J. Cocke and R. E Miller. Configurable Computer System. *IBM Technical Disclosure Bulletin*, 15:9, February 1973.

[6] A. DeHon, S. Perentz. Transit Sbus Interface. Transit Note No. 67, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June 1992.

[7] InCA Inc. *Concept Silicon Reference Manual*, Nov. 1992. Version 1.1.

[8] S. Kirkpatrick, C. D. Gellatt, and M. P. Vecchi. Simulated annealing. *Science*, 220, 1983.

[9] J. Kubiatowicz. User's Manual for the A-1000 Communications and Memory Management Unit. ALEWIFE Memo No.

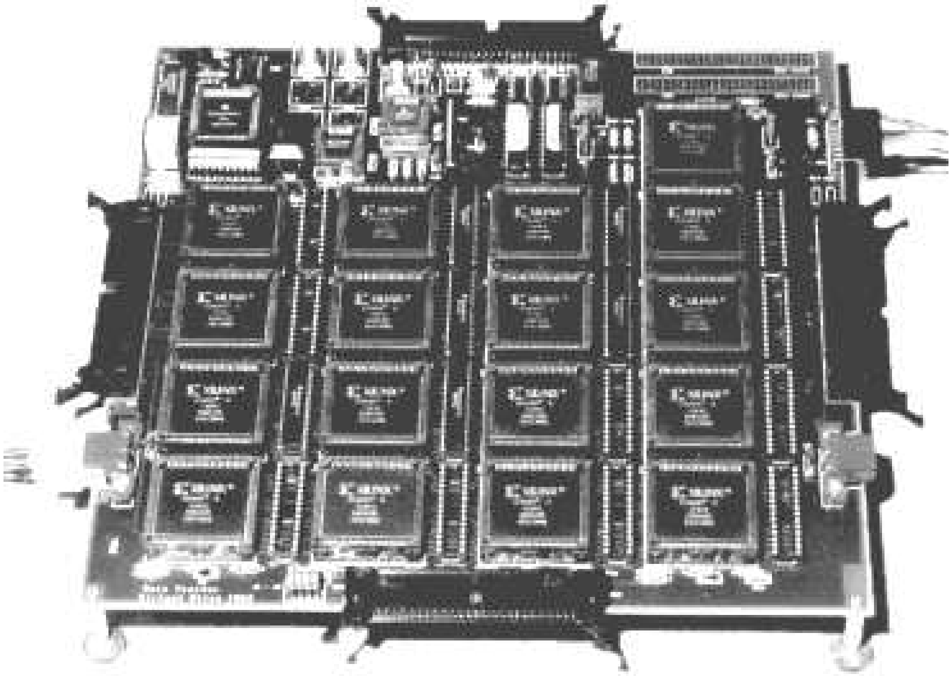

Figure 9: Virtual Wires Emulation Board

19, Laboratory for Computer Science, Massachusetts Institute of Technology, January 1991.

[10] LSI Logic Corporation *User's Guide for LDS Simulation and Verification Tools*, Oct. 1989. Release 10.0.

[11] L. Maliniak. Multiplexing enhances hardware emulation. *Electronic Design*, Nov. 1992.

[12] F. Manning. Automatic Test, Configuration, and Repair of Cellular Arrays *MIT/LCS Technical Report-151*, June 1975.

[13] S. Walters. Computer-aided prototyping for ASIC-Based systems. *IEEE Design and Test of Computers*, June 1992.

[14] Xilinx, Inc. *The XC4000 Data Book*, Aug. 1992.