SPLASH 2

Jeffrey M. Arnold
Duncan A. Buell
Walter J. Kleinfelder
Supercomputing Research Center
17100 Science Drive
Bowie, Maryland 20715-4300

ABSTRACT

Splash 2 is an attached processor system for Sun SPARC 2 workstations that uses Xilinx 4010 Field Programmable Gate Arrays (FPGAs) as its processing elements. The purpose of this paper is to describe Splash 2. The predecessor system, Splash 1, was designed to be used as a systolic processing system. Although it was very successful in that mode, there were many other applications that were not systolic, but which were successful, nonetheless, on Splash 1, or that were not implemented successfully due to one or more architectural limitations, most notably I/O bandwidth and interprocessor communication. Although other uses to increase computational performance have been found for the Xilinx FPGAs that are Splash's processing elements, Splash is unique in its goal to be programmable in a general sense.

INTRODUCTION

Splash 2 is an attached processor system for Sun SPARC 2 workstations that uses Xilinx 4010 Field Programmable Gate Arrays (FPGAs) as its processing elements. The purpose of this paper is to describe Splash 2.

The predecessor system, Splash 1 [1], was designed to be used as a systolic processing system [2] [3]. Although it was very successful in that mode, there were many other applications that were not systolic, but which were successful, nonetheless, on Splash 1, or that were not implemented successfully due to one or more architectural limitations, most notably I/O bandwidth and interprocessor communication. Although other uses to increase computational performance have been found for the Xilinx FPGAs that are Splash's processing elements (see, for example, [4] or [5]), Splash is unique in its goal to be programmable in a general sense.

THE HARDWARE

The architecture of Splash 2 is shown in Figures 1 and 2.

The Splash 2 System

The system-level view of Splash 2 is shown in Figure 1. (This shows a 3-board system; a system can contain 1 to 13 boards.) An interface board plugs into the backplane and an SBus adapter board plugs into a Sun SPARC 2 workstation to run the Splash 2 system via the interface board.

This paper is taken from a paper published in the Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures and copyrighted by the Association for Computing Machinery. It is published here by permission of the ACM.

The interface board extends the address and data buses from the Sun into the address/data buses in the backplane. The Sun can read from and write to memory and memory-mapped control registers on the Splash 2 boards via these buses. The Sun provides only 25 address bits (that we take to be 23 since we deal only with data on 32-bit-word boundaries), which is inadequate to address the 13 (boards) $\times$ 17 (memories) $\times$ $512K$ (bytes) of Splash 2 memory, so the interface board contains an address bank register that selects the Splash 2 board in the system.

There are three data paths into the Splash 2 system.

(1) On the memory bus, data can be read and written into memories attached to each Xilinx processing chip.

(2) A "linear data path" exists down the SIMD bus into the first Xilinx chip, X1, in the linear array of the first Splash 2 board in a daisy chain that can include as many as 13 Splash 2 boards. Output from the last Xilinx chip in the linear array, X16, of the first board passes as input to the X1 chip of the second board, and so on. Output from X16 of the last board in the daisy chain returns on the Rbus to the interface board.

(3) A SIMD path exists by using the SIMD bus for broadcast. The SIMD bus has a data path into Xilinx chip X0 on each board, which can then inject SIMD instructions or data into the crossbar and thus broadcast to the other Xilinx chips on that board.

There are three modes for sending data into the Splash 2 system.

(1) Splash 2 can communicate with the Sun via DMA transfers to and from the FIFOs of Figure 1. The two input FIFOs are $1K \times$ 36-bits; the two output FIFOs are $1K \times$ 32-bits. For these transfers, the interface board becomes a master on the Sun SBus and transfers bursts of data to or from the FIFOs. In typical operation the Sun programs and initializes the Splash 2 boards via memory-mapped transfers and then enables DMA for data transfer to/from Splash 2. In this mode, the 32 bits of data form the low 32 of the 36 bits in the FIFO. The high 4 bits are taken from a tag register. DMA data transfer can be sustained at about 38 Mbytes per second when the host workstation is CPU-loaded, or as fast as 54 Mbytes per second when the host is idle.

(2) The Sun host can also perform direct writes to the input FIFOs of Splash 2. In this mode the high 4 bits of the 36-bit FIFO word are bits 5-2 of the address.

(3) Splash 1 was and Splash 2 will be a useful processor for handling digital signals generated external to the Sun host. The external input accommodates input of such a signal directly to Splash 2. Further details of this are given below.

The Splash 2 Interface

The interface board is responsible for generating all the signals necessary in the backplane for running up to 13 Splash 2 boards.

The Sun data bus is latched and buffered to drive the backplane data bus for memory-mapped reads and writes. The Sun address lines are latched and buffered to feed the backplane, and the Sun can load an address bank register with a 7-bit address extension to obtain 30 bits of 32-bit-word addresses.

A clock generator provides the clock signal to the Splash 2 boards, can be programmed by the Sun to various frequencies, and can be programmed to single-step, N-step, or to stop on an interrupt.

Interrupts can be requested by any Splash 2 board, and the DMA controller can request an interrupt when transfers are completed. An interrupt register permits the Sun interrupt program to enable or disable interrupts and to read which interrupt source generated an interrupt. FIFO full/empty determination is under the control of Xilinx chips XL and XR.

The inclusion of Xilinx chips XL and XR was to provide for control of data transfer, clock (even a clock supplied by the external input), and tag bits independent of the Splash 2 boards. In Splash 1 such control was usually done in the first array chip, leading to asymmetry and crowded designs. With proper programming of XL and XR, the asynchronies of DMA transfer and external input and clock should not be seen by the Splash 2 boards themselves, and the XL and XR programs should function much like a system I/O library. A size register indicates the number of Splash 2 boards in a system, providing a signal to the Splash 2 boards so that one board is enabled to deliver data to the Rbus. A DMA controller performs SBus-compatible burst DMA transfers to and from the FIFOs in 16-word bursts.

To accommodate variable modes of data entry into a Splash 2 system, provision for an external signal input exists in the form of a daughterboard attached to the interface board. In this way, small changes in input signal conditioning can be made without requiring the entire board to be re-engineered. The daughter board can be configured to provide an external clock, thus allowing the Splash 2 system to be run synchronously with external data.

The Splash 2 Processing Boards

The Splash 2 board is detailed in Figure 2. Each board contains 17 Xilinx 4010 chips. Sixteen of these, X1-X16, form the processor array, connected both linearly and via the crossbar by 36-bit-wide data paths. The 17th chip, X0, has several uses to be mentioned later. Each of chips X1-X16 is connected via a 36-bit-wide path (18 address, 16 data, 2 control) to the $256K \times$ 16-bit memories. The memories can be read from or written to directly by the Sun on a 32-bit data path.

The linear data path brings data from either the previous Splash 2 board or from the SIMD bus into X1, through the linear array, and out from X16 to either the next Splash 2 board or to the Rbus, and from there to the interface board.

Among the many control lines on Splash 2 is a single interrupt line from each Xilinx chip back through the interrupt latch and mask to the host. This is useful for applications such as searches in which a Xilinx chip that found the solution can signal that fact back to the host and interrupt the processing. In addition, a global AND/OR and a global VALID line (GOR, GORV) extend from each Xilinx chip to the control chip X0, and a system global AND/OR runs from each Splash 2 board to the interface board.

A final feature of the Splash 2 board is the ability to load or store a configuration state into the Xilinx chips. Readout of the state was possible in Splash 1 and was invaluable for debugging and program optimization; the new ability also to load the Xilinx chips with a starting state configuration will greatly enhance the ability to monitor program behavior.

The 17th Xilinx chip X0 serves several functions. Its primary purpose is to control the crossbar. The crossbar itself is bit-sliced from nine TI SN74ACT8841 4-bit crossbar chips. Up to eight different configurations can be chosen; X0 is used to select which configuration is in effect at any given cycle, and the crossbar control determines the direction in which data is transferred. Using multiple configurations can, for example, allow the 16 chips to be viewed as a two-dimensional mesh, or a 4-dimensional binary cube, provided that only one data path per Xilinx chip is used in any given cycle (since only one path exists). In this way, the realization of common communication patterns is relatively straightforward. For example, a 4-dimensional binary cube is realized as follows: View the linear array as a hamiltonian path through the 4-cube. Properly chosen, and with an appropriate coordinate labelling, this path provides all the connections in the x-dimension, four of eight in the y-dimension, and two and one, respectively, in the z- and w-dimensions. Six crossbar configurations, one for each direction for each of the y-, z-, and w-dimensions, now provide the additional connections to realize a 4-cube. Although arbitrary communication is not possible—only three and not four ports exist per chip—it is possible to communicate one dimension at a time, and many cube algorithms exhibit this characteristic pattern. In an analogous way one can realize a 4×4 mesh, although in one of the two dimensions only half the needed communication paths are available at a time.

A second function of X0 is to provide a broadcast capability into the crossbar. Splash 2 can be used as a SIMD computing engine, as will be discussed below, and the connection from the linear data path through X0 into the crossbar allows for a broadcast of instruction and immediate data to all chips on a board at a time, using the lines into the crossbar shared by X0 and X16.

To allow X0 to be sent "subroutine calls" in SIMD mode and to execute stored subroutines, and to allow for the lookup tables that can be expected to be heavily used, X0 possesses its own local memory.

One complication exists in that the memories are 16 and not 32 bits wide. To allow for both the host and the Splash 2 board to view the normal data width as 32 bits, the memories on the Splash 2 board are double-cycled; the host and the interface board pass 32 bit data to/from the Splash 2 board, and the board reads/writes 32 bits on word boundaries by using two cycles for every data transfer to/from the interface board. This design decision was based on the I/O pin count of the Xilinx 4010 chips. Many designs were considered, but it proved impossible to retain the linear array data path (2×36 bits), add a crossbar connection (1×36 bits), add a direct connection to memory (18 bits address, 32 data), and have any of the 160 I/O pins left over for control.

SIMD COMPUTING MODE

A Splash 2 board allows a 256-bit load/store in parallel to 16 Xilinx processing chips. The combination of crossbar and the linear array provides a powerful parallel data transfer capability similar to a network. With this view of the Splash 2 board, its use for SIMD computing is quite natural. To effect this mode of computing it is necessary to support broadcast of instructions and/or immediate data. This is made possible by lines running down the SIMD bus into Xilinx chip X0 of every board and from there directly to Xilinx chips X1 through X16 over the crossbar. In this mode, chip X0 could be programmed explicitly to serve as an instruction decode module and possibly also to convert from a vertical to a horizontal encoding of the instructions.

PROGRAMMING

There are three levels at which the Splash 2 system must be programmed: the Splash board, the interface, and the host. At the Splash board level the programmable components consist of the Xilinx processing chips, X1 through X16; the control chip, X0; and the crossbar. At the interface level the Xilinx chips XL and XR are user programmable. The host interface must provide input data streams and control the operation of the Splash system. A library of common control functions is provided for the interface board chips, XL and XR, and for the Splash board control chip, X0. Many linear and SIMD applications use only a single crossbar configuration, which can also be provided in a library. The host interface can be driven from either a C program that makes calls to a package of control routines or through an interactive graphical debugger. Therefore, the minimal Splash 2 program consists of a single replicated Xilinx program for X1 through X16 and a selection of library components for the rest of the system.

The programming environment for Splash 2 is based upon the VHSIC Hardware Description Language (VHDL). VHDL is a hardware specification language with many modern programming language features such as block structured control; user defined data types; and overloaded procedures, functions, and operators. VHDL programs can freely mix behavioral specifications with more traditional structural descriptions. The VHDL programming model includes the concept of time, so VHDL specifications can be simulated directly.

The Splash 2 programming methodology relies heavily upon simulation and logic synthesis. Users develop applications by writing VHDL behavioral models of their algorithms, which are then simulated and debugged within the Splash 2 simulator. Once an algorithm is determined to be functionally correct, it is compiled into a set of Xilinx chip configurations and the timing analyzed and optimized.

The Splash 2 simulator is a hierarchical model of the Splash 2 system comprising a set of VHDL models for each of the components of the system. When an application program is simulated, it is able to interact with the system exactly as it would with the physical hardware. The system models also verify that the application program meets any hardware constraints such as memory sequencing and setup and hold times. Because the simulator is based upon commercial tools, a full source level debugging interface is available to the user.

A mix of logic synthesis and standard compilation techniques are used to compile VHDL programs into Xilinx configurations. A commercial logic synthesis tool is used to map the VHDL code into a gate list, where a peephole optimizer is used to perform a variety of Xilinx- and Splash-specific optimizations. The resulting gate list is then mapped into the CLBs and placed and routed using the Xilinx tool package. The Xilinx tools are used also to extract the detailed timing information from the placed and routed design. This information is used to construct a new VHDL model for each chip, which is then fed back to the Splash 2 simulator for timing analysis.

ACKNOWLEDGEMENTS

We acknowledge those who have contributed to Splash 2, including at least Neil Coletti, Steve Cuccaro, Elaine Keith, Brad Fross, Maya Gokhale, William Gromen, William Holmes, Daniel Kopetzky, Andrew Kopser, James Kuehn, Sara Lucas, Ronald Minnich, Michael Mascagni, John McHenry, Fred More, Louis Podrazik, Daniel Pryor, Craig Reese, Judith Schlesinger, Nabeel Shirazi, David Smitley, Douglas Sweely, Mark Thistle, Chris Tscharner, Paul Schneck, and Ken Wallgren.

REFERENCES

[1] Maya Gokhale, William Holmes, Andrew Kopser, Sara Lucas, Ronald Minnich, Douglas Sweely, and Daniel Lopresti, *Building and using a highly parallel programmable logic array*, IEEE Computer **24** (1991), 81-89.

[2] H. T. Kung, *Why systolic architectures?*, IEEE Computer **15** (1982), 37-46.

[3] H. T. Kung and C. E. Leiserson, *Systolic arrays for VLSI*, Introduction to VLSI Systems, by C. A. Mead and L. C. Conway, Addison-Wesley, Reading, Massachusetts, 1980, pp. 271-292.

[4] Will B. Moore and Wayne Luk (eds.), *FPGAs*, Abingdon EE & CS Books, Abingdon, England, 1991.

[5] M. Shand, P. Bertin, and J. Vuillemin, *Hardware speedups for long integer multiplication*, Proceedings, ACM Symposium on Parallel Algorithms and Architectures (1990), 138-145.

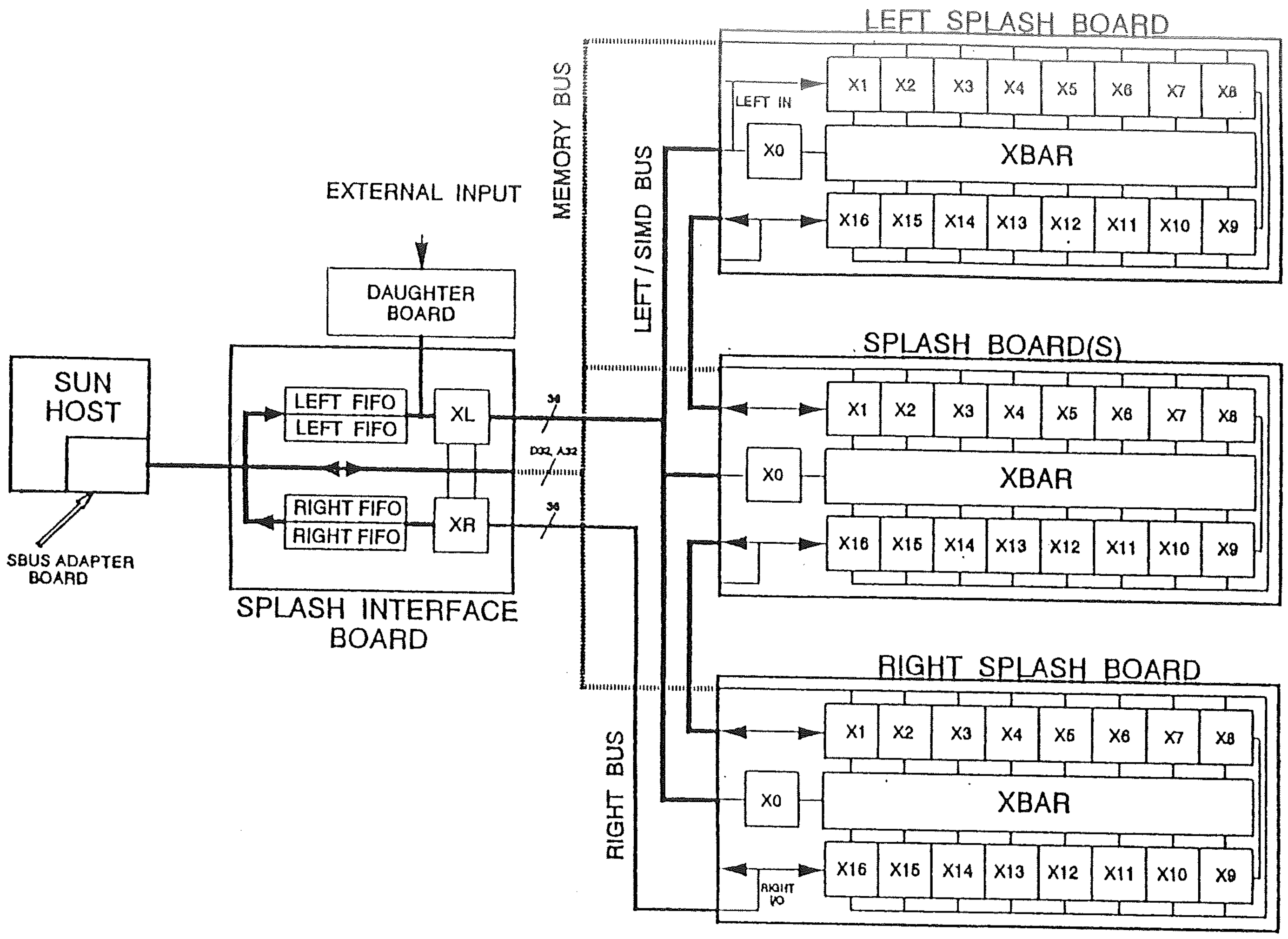

Figure 1: The SPLASH II System

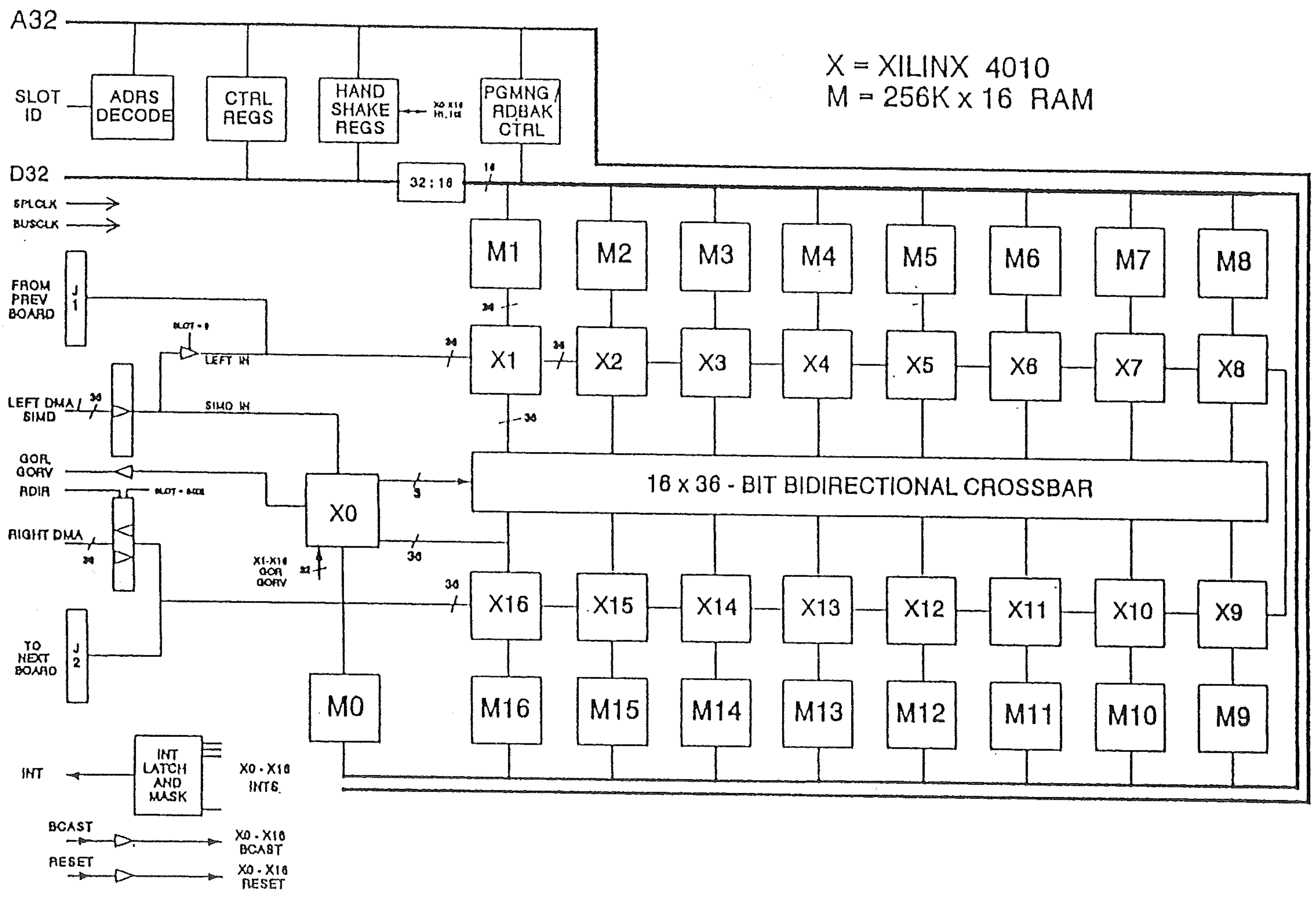

Figure 2: The SPLASH II Board