# The Splash 2 Processor and Applications

Jeffrey M. Arnold, Duncan A. Buell, Dzung T. Hoang
Daniel V. Pryor, Nabeel Shirazi, Mark R. Thistle
IDA Supercomputing Research Center
17100 Science Drive
Bowie, MD 20715

**Abstract**

*Splash 2 is an attached parallel processor in which the computing elements are user programmable FPGA devices. The architecture of Splash 2 is designed to accelerate the solution of problems which exhibit at least modest amounts of temporal or data parallelism. Applications are developed by writing descriptions of algorithms in VHDL, which are then iteratively refined and debugged within a simulator. Once an application is determined to be functionally correct in simulation, it is compiled to a gate list and optimized by logic synthesis. The gate list is then mapped onto the FPGA architecture by automatic placement and routing tools to form a loadable FPGA object module. A C language library and a symbolic debugger comprise the execution environment. The Splash 2 system has been shown to be effective on a variety of applications, including text searching, sequence analysis, and image processing.*

## 1 Introduction

In recent years the emergence of reconfigurable Field Programmable Gate Arrays (FPGAs)[3][7] has given rise to a new form of computing in which the architecture of a computer may evolve over time, changing to fit the needs of each application it executes. With such a computer, the programmer may tailor the architecture to fit the problem solution, rather than vice versa. Several FPGA based architecture have been proposed and built recently. These machines all have very different programming models, but each relies on the use of reconfigurable hardware to customize the architecture to particular applications. (See[1] for a more extensive bibliography.)

Splash 2[2] is an attached parallel processor in which the computing engines are user programmable FPGA devices. The architecture of Splash 2 is designed to accelerate the solution of problems which exhibit at least modest amounts of temporal parallelism (pipelining) or data parallelism (single instruction multiple data stream). High I/O bandwidth makes Splash 2 particularly useful for problems with large data sets or continuous data streams. Finally, because the architecture may be shaped to fit the problem, Splash 2 is well suited to problems in which the size of the fundamental data objects do not match a conventional architecture (e.g. streams of character text).

## 2 Splash 2 Architecture

The Splash 2 system consists of a Sun Sparcstation host, an interface board, and from one to sixteen Splash array boards, as shown in figure 1. The interface board contains a programmable system clock and provides DMA access to the host memory through two user programmable FPGA devices, XL and XR, which are used to process incoming and outgoing data streams.

Each array board contains 16 processing elements (PEs), X1 through X16, arranged in a linear array and fully connected via a 16x16 crossbar switch which is regulated by the 17th control element, X0. Each of the 17 computing elements consists of a Xilinx XC4010 FPGA and 256K 16 bit words of memory, which is also directly addressable by the host. Each processing element has 36 bit bi-directional data paths to its left neighbor, to its right neighbor, and to the crossbar switch. The input to the array is provided by XL through the SIMD bus to X1 of the first board and to X0 of every board. Multiple array boards are linked together by extending the linear data path from the right side of one board to the left side of the next. The last board in the chain is connected to the RBus, and thence to XR on the interface board.

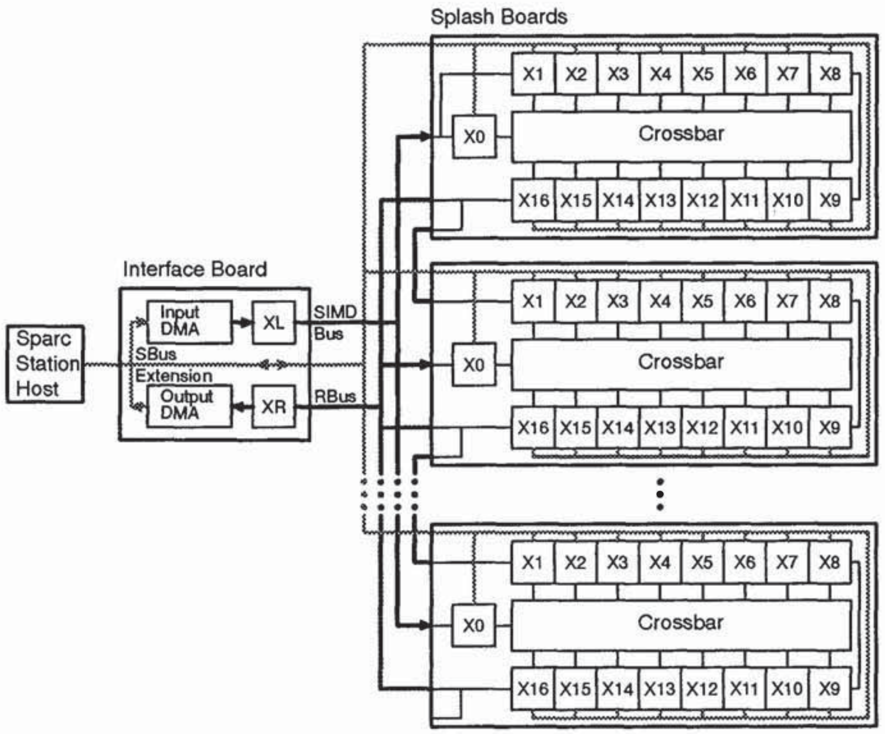

Figure 1. The architecture of the Splash 2 system.

The crossbar switch is capable of storing up to eight separate, dynamically selectable configurations, with each configuration specifying a different set of connections among the 16 ports. The configuration in use in any given clock cycle is selected by the control element, X0, which shares the 36 bit connection to the crossbar with X16. X0 may therefore receive broadcast data from the host on the SIMD bus and rebroadcast it through the crossbar to each of the 16 PEs.

The Sparcstation host performs a variety of control functions for Splash 2. The host is responsible for downloading the configuration information to the FPGA computing elements, X0–X16, the interface elements, XL and XR, and the crossbar switch. There are a variety of both synchronous and asynchronous mechanisms with which the host may communicate with the Splash system, including direct access to the computing element memories and the ability to start and stop the system clock.

As a research vehicle Splash 2 was designed to support a variety of programming models, including a single-instruction multiple-data stream (SIMD) model, a one dimensional pipelined systolic model, and several higher dimensional systolic models. SIMD applications may be programmed on Splash 2 by using the X0 element and the crossbar switch on each board to broadcast instructions and data to all processing elements simultaneously. The instruction stream is sent from the host to the X0 chip on each board via the SIMD Bus. The X0 elements may in turn decode this instruction stream, possibly by fetching microcode from their external memories. This decoded instruction is then broadcast through the crossbar switch to each of the 16 processing elements on the board. Each processing element is programmed with one or more identical SIMD computing engines. These SIMD engines simultaneously receive and execute the decoded instruction, performing nearest neighbor communication through the linear data path. Global synchronization is supported through an AND/OR reduction network.

One dimensional systolic applications may be mapped onto Splash 2 by using the linear data path to form a continuous pipeline from the host, through the array, and back to the host. The crossbar switch may be configured to bypass certain PEs during some or all of the processing time. Many higher dimensional applications can be mapped onto Splash by using both the linear data path and the dynamic selection capability of the crossbar switch to build higher dimensional topologies, such as trees, hypercubes, and toroids.

## 3 Programming Splash 2

The programming environment for Splash 2 is based upon the VHSIC Hardware Description Language (VHDL)[5], simulation, and logic synthesis. Applications for Splash are developed by writing behavioral descriptions of algorithms in VHDL which are then iteratively refined and debugged within the Splash 2 simulator. During the course of this iteration, the VHDL implementation is manually partitioned into a set of Splash computing element programs. Once the implementation is determined to be functionally correct in simulation it is compiled into a gate list and optimized by logic synthesis techniques. The gate list is then mapped onto the FPGA architecture by automatic placement and routing tools to form a loadable FPGA object module. Static timing analysis tools are applied to the object module to determine the maximum operating frequency and the set of critical paths. This information may be used to manually optimize the design. Finally, a C language interface library and a symbolic debugger form the execution environment.

## 4 Applications

### 4.1 Text Searching

Splash 2 has been programmed by the last three authors to detect key words from a dictionary in a stream of text [6]. In a preprocessing step, a dictionary is streamed past a hash function. The hash function computes an address from each word in the dictionary and sets a presence bit in a memory table. During the processing, the text is streamed through the hash function, an address computed from words, and the corresponding memory location examined. A set presence bit indicates a potential key word hit. The individual probabilities of false hits multiply if multiple (assumed to be) independent hash functions are used, so the overall probability of a false hit can be made as small as one likes.

The interface board chip XL unpacks four-character (32-bit) input words in two two-character "superbytes" which are broadcast on the SIMD bus to X0 on the Splash 2 array boards. X0 uses its attached memory as a translation table to convert upper case letters to lower case, and white space characters to a simple end-of-word marker. This translation table can be used to accommodate multiple alphabets or other special processing requirements.

X0 sends the translated superbyte to X1 over the crossbar. Xilinx chips X1-X16 run the same program with different hash functions receiving data from the linear data path, computing an updated hash function (different for each chip), and performing a lookup into memory if the processed superbyte contained an end-of-word marker. If the lookup is done, the presence bit is ANDed to the incoming accumulated presence bits and passed onto the next chip in the linear array. The final determination of a hit is made by XR on the interface board, which reports the existence and position in the file of the hit.

The memories attached to X1-X16 each are $2^{22}$ bits in size. The English language has about $2^{18}$ words (ignoring case-sensitivity). In the simplistic scheme adopted here, the set of keywords is doubled in size to accommodate end-of-word markers in either the odd or even byte positions. The full English language would thus hash into a Splash 2 memory that is about 1/8 full, and 16 independent hash functions would produce false hits with a probability of 1 in $2^{48}$, which is certainly low enough for nearly any application. A "real" dictionary or keyword list could be expected to be much smaller (the Unix spell-checking program uses a dictionary of about 30,000 words); the corresponding substantial reduction in the probability of false hits will be more than make up for the fact that text is in fact not "random".

The program as written for Splash 2 runs at 25 MHz, processing two bytes of text per clock, for an effective text throughput rate of 50 Mbytes per second.

### 4.2 Macromolecule Sequence Comparison

Splash 2 has been programmed by the third author to perform DNA and protein sequence comparisons [4]. The well-known dynamic programming algorithm is used to compute the edit-distance between two strings of characters. The edit distances for all prefixes of a source sequence $S = s_1 s_2 ... s_n$ and a target sequence $T = t_1 t_2 ... t_m$ form a matrix. With $s_i$ labelling row $i$ and $t_j$ labelling column $j$ of the matrix, the entry in the $(i, j)$ cell of the matrix is the edit-distance from $S_i = s_1...s_i$ to $T_j = t_1...t_j$. This distance is the smaller of three quantities:

1) The distance from $S_{i-1}$ to $T_j$ plus the "cost" of appending $s_i$ to $S_{i-1}$.

2) The distance from $S_i$ to $T_{j-1}$ plus the cost of appending $t_j$ to $T_{j-1}$.

3) The distance from $S_{i-1}$ to $T_{j-1}$ plus the cost of substituting $t_j$ for $s_i$.

For DNA sequences, the costs in 1) and 2) are usually taken to be 1 and the cost in 3) to be 0 if $t_j = s_i$ and 2 if not.

The locality of the information needed for edit distance computation permits a systolic method to be used which is also highly parallel because all the cells on a given antidiagonal can be updated simultaneously. In Splash 2 implementation of this algorithm for DNA comparisons, 14 copies of a cell-update processing element fit on a single Xilinx chip for a total of 224 processing elements per Splash 2 board. This implementation can process 12 million characters per second. In a protein sequence version, 8 million characters per second can be processed.

### 4.3 Edge Detection

The high I/O bandwidth and the pipelined nature of Splash make it well suited to many image processing applications. One such application which has been implemented is a 3x3 gradient operator edge detector. The input image is streamed into the array at video rates (up to 16 MPixels/sec). For the duration of one scan line, the input stream is written to one PE memory while the previous two scan lines are read from two other memories. The input scan line together with the two previous scan lines are used to compute the X and Y partial gradients. Another PE memory is used as a table lookup to compute the magnitude and the angle of the gradient. At the end of each scan line, the three scan line buffers exchange roles so that the second previous buffer is over written with the new scan line, and the most recently written buffer becomes the previous. This is accomplished by rotating among three crossbar configurations at the end of each scan line. An "image" consisting of the angle and magnitude of the gradient at each pixel is streamed out at the input data rate with a latency of three scan line times.

## Acknowledgements

We acknowledge all those who have contributed to the Splash 2 project, including D. Burns, N. Coletti, S. Cuccaro, B. Fross, M. Gokhale, W. Holmes, E. Keith, W. Kleinfelder, D. Kopetzky, A. Kopser, J. Kuehn, S. Lucas, M. Mascagni, M. McGarry, R. Minnich, M. Moran, F. More, L. Podrazik, C. Reese, J. Schlesinger, P. Schneck, B. Schott, D. Smitley, D. Sweely, C. Tscharner, and K. Wallgren.

## References

[1] J. M. Arnold, "The Splash 2 software environment," *Proc., IEEE Workshop on FPGAs for Custom Computing Machines*, 1993, to appear.

[2] J. M. Arnold, D. A. Buell, and E. G. Davis, "Splash 2", *Proc., 4th Annual ACM Symp. on Parallel Algorithms and Architectures*, pp. 316–322, 1992.

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

[4] D. Hoang, "Searching genetic databases on Splash 2", *Proc., IEEE Workshop on FPGAs for Custom Computing Machines*, 1993, to appear.

[5] *IEEE Standard VHDL Language Reference Manual*, IEEE Std 1076–1987, New York, NY, 1988.

[6] D. Pryor, M. Thistle, and N. Shirazi, "Text searching on Splash 2", *Proc., IEEE Workshop on FPGAs for Custom Computing Machines*, 1993, to appear.

[7] *The XC4000 Data Book*, Xilinx Inc., San Jose, CA, 1992.