SPLASH 2

JEFFREY M. ARNOLD
DUNCAN A. BUELL
ELAINE G. DAVIS

Supercomputing Research Center
17100 Science Drive
Bowie, Maryland 20715

INTRODUCTION

The Splash attached processor board (referred to as Splash 1) was designed and built at the SRC to provide very high performance on a range of bit-processing problems. It proved to be highly successful [GOHKa]; notwithstanding the known dangers of Second System Syndrome[1] [BROO], a follow-on system, Splash 2, is being designed and built. The purpose of this paper is to describe Splash 2, to compare it with Splash 1 and to discuss both its programming and two algorithmic applications.

Splash 1 was designed to be used as a systolic processing system [KUNGa, KUNGb]. Although it was very successful in that mode, there were many other applications which were not systolic but nonetheless successful on Splash 1 or which 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 which are Splash's processing elements (see, for example, [MOOR] or [SHAN]), Splash was unique in its goal to be programmable in a general sense.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

[1] "When one is designing the successor to a relatively small, elegant, and successful system, there is a tendency in one's success to become grandiose and design an elephantine feature-laden monstrosity."

SPAA '92 - 6/92/CA
©1992 ACM 0-89791-484-8/92/0006/0316 ..... $1.50

THE HARDWARE

Splash 1.

Splash 1 is a single multiwire board that plugs into the VMEbus of a Sun workstation. Each board contains 32 Xilinx 3090 Field-Programmable Gate Array (FPGA) chips X0 through X31 as processing elements. The FPGAs are connected in a linear array by a 32-bit-wide path. Chips X0 and X31 can be similarly connected to form a ring. Data on and off the board is handled by a pair of FIFOs controlled by X0 and X31, respectively. Between each pair of interior Xilinx chips is a $128K \times 8$ RAM with an 8-bit-wide path to the FPGAs.

The Xilinx 3090 chips have a maximum clock rate of 32 Mhz. To accommodate Splash 1 designs which could not be run at maximum speed, usually due to placement and routing problems or to the inability of the VMEbus to deliver data at a sufficiently high rate, the clock rate could be set in factors of 2 from 1 Mhz to 32 Mhz.

Programming of Splash 1 was originally done with a Xilinx-supplied software tool (the XACT editor). Later tools included the Viewlogic schematic capture package. To make the machine more accessible, researchers at SRC developed the Logic Description Generator (LDG), a higher-level language whose output could be mapped to the Xilinx chips [GOKHb]. In addition to the software for direct execution, a debugger called Trigger was extensively used.

Splash 2.

Comparisons with Splash 1.

The architecture of Splash 2, shown in Figures 1 and 2, reflects to a small extent the improvements in technology since the design of Splash 1 but shows much more the results of experience with Splash 1 and some of its correctable limitations. Before going into a detailed description of the hardware, we compare it with its predecessor.

Splash 1 was hosted by a Sun workstation over the VMEbus, and there were many applications on Splash 1 which were entirely I/O bound. Splash 2, in contrast, uses a SPARC II as a host over the accompanying SBus. While comparisons of peak ratings of buses and of buses as used by particular machines can be made, they do not usually reflect reality. In brief, it is expected that the sustainable I/O rate of Splash 2 should be somewhere between 8 and 10 times that of Splash 1.

The second technology-based change is to use the new Xilinx 4010 chips. Three major differences between the 3090 and the 4010 chips are that the 4010's have 400 Combinatorial Logic Blocks (CLBs) instead of the previous 320, that each CLB has 9 input lines instead of 5, and that the maximum speed is 40 Mhz instead of 32Mhz.

An additional feature of the new chip is a fast carry internal to the CLBs, which makes arithmetic computations faster and requires less programming and fewer CLBs. Further, the number and quality of the interconnection lines as well as the quality of the automatic place-and-route (APR) software used to configure the chips have improved; this should help more applications run at higher speeds. Finally, the new chips allow for the use of CLBs as a 32-bit RAM, configured either as 32×1 bit or as 16×2 bits.

Splash 2 has 17 Xilinx chips on a board instead of the previous 32. This is both a conscious decision and a necessity. The newer chips are physically larger, and it is not possible to put 32 of them on a single board along with the memories and the crossbar (to be described below). However, since the newer chips are each more powerful than the old chips, and since it was more often the case with previous applications that they were I/O-limited rather than processor-limited, it does not seem unrealistic to expect that a reasonable processor-to-I/O balance now exists.

Beyond mere technology upgrades are several architectural enhancements.

- Data path: The only data path on Splash 1 was a 32-bit wide linear connection of the 32 Xilinx 3090 chips, with the end chips connected to the host in FIFOs. Splash 2 retains this path but also has broadcast (to multiple Splash boards), a memory connection to the host, and interchip connections on the Splash boards themselves. Also, in Splash 1, the data from the host passed directly into the Splash board, so that handling of the FIFOs and any preconditioning of the data (merging of two input streams, for example) had to be done by the Xilinx chips on the Splash board. These functions have now been moved to an interface board, so the Splash 2 boards will not be exposed to these irregularities of data movement. Further, Splash 2 is designed throughout to be a 32-bit machine with extra tag bits allowed where possible. On Splash 1 it was often observed that 32-bit data widths were sufficient but that extra tag bits to be sent along with the data would have been very useful. The extension of the input width in the FIFOs from 32 to 36 bits should remedy this shortcoming. In addition, the extra 4 bits should accommodate many of the requirements of a SIMD computing mode—with a 36-bit-wide input word, 32 bits of immediate data can be passed in along with a 4-bit instruction opcode, or the extra bits can be used to allow the Splash 2 boards to distinguish instructions from data.
- Memory: The primary uses of the memory chips in Splash 1 were for lookup tables and for storing microprograms to be executed by state machines implemented in the Xilinx chips. The memory use was, however, encumbered by problems in sustaining peak rates, by the fact that memory loads had to be done down the systolic data path of the Splash board, and by the fact that the memories were connected to two Xilinx chips on the systolic data path. This last problem required programmers to exercise great care to separate in time the access to memory and the transmission of data from Xilinx to Xilinx. The memory chips on Splash 2, by contrast, are directly connected to a single Xilinx chip. The previous $128K \times 8$ bit RAMs have been replaced by $256K \times 16$ bit RAMs. These two changes make their memory much more accessible by and useful to the Xilinx chips. The memories can now be directly read/written from the Sun host over the SBus. They are not, however, dual-ported; the FPGAs must be inactive during the read/write operations. This change allows tables to be loaded in bulk and results to be read from the memories without requiring the circuitous path through the Xilinx chips. One result of this, as discussed below, is the natural use of Splash 2 for SIMD computing.
- Crossbar: Splash 1 had only a single data path—a systolic route through the 32 Xilinx chips. While the systolic paradigm is very powerful and its application to Splash has been very successful, there were many applications that either could not be done or whose efficiency suffered because the systolic path was the only data path. After a number of possible configurations were discussed, it was decided to use programmable crossbar chips to implement a full 16×16 crossbar among the Xilinx chips. Up to eight configurations can be preloaded into the crossbar control, and selection among the

eight configurations can be done dynamically. 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.

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 16 boards.) The Splash 2 boards plug into an enclosure containing two 9-slot backplanes. An interface board and its expansion board plug into the two center slots to drive the two backplanes in parallel, and an SBus adapter board plugs into a Sun SPARC 2 workstation to run the Splash 2 system via the interface board.

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 (which we take to be 23 since we deal only with data on 32-bit-word boundaries), which is inadequate to address the 16(boards) $\times$ 16(memories) $\times$ $512K$(bytes) of Splash 2 memory, so the interface board contains an address bank register which 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 "systolic 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 which can include as many as 16 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 the SIMD instruction or data into the crossbar and thus broadcast to the other Xilinx chips on that board.

Remarks:

(1) A multiplexor exists on the Splash 2 board for selecting whether the data path into X1 is from the SIMD bus or from the J1 connection from the "previous" Splash 2 board.

(2) The address/data path into the memories on a Splash 2 board, the SIMD bus, and the Rbus are bus connections in the backplane. The connections from the output of X16 on one board into X1 of the next board use connections on the backplane.

(3) A Splash 2 board must be present in the first slot of a system. A selection register exists on the interface board to enable "the last Splash 2 board" to deliver data to the Rbus. The selection register can be dynamically changed so that the interface board can enable the delivery of data to the Rbus from any Splash 2 board in a multi-board system.

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.

(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 di-

rectly to Splash 2. Further details of this are given below.

The Splash 2 Interface.

The interface board and its expansion board plug into the center slot pair of the enclosure and are responsible for generating all the signals necessary in the backplane for running up to 16 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 systolic 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 thence 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 which 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). Switching among chosen configurations, for example, would allow cube-connected data paths to be used in one dimension 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 systolic 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 (which lines must *not* be driven by X16 when broadcast is to take place).

To allow X0 to be sent "subroutine calls" in SIMD mode and to execute stored subroutines, and to allow for the lookup tables which 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 boards 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 possible by lines 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 explicitly programmed 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 systolic 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 also used 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.

Two Applications

Keyword searching/dictionary searching.

References for this include [LOPR, MCHEa, MCHEb].

One kind of dictionary searching is a *spell* program such as exists in Unix. One has a dictionary of several thousand words and wishes to determine if any words in a document or documents are *not* in the dictionary. Another kind of searching is exactly the opposite of a spelling checker; one has a list of several thousand key words and wishes to determine if any words in a document or documents *are* in the list. A computational difference between these two applications is that in checking spelling one is likely to have a single document which may be quite long. In keyword searching, one is probably going to have many documents, most of which will be rather short.

This application is common in what the information business calls SDI–Selective Dissemination of Information–in which a user or users list keywords as a profile of interest and a computer run is made periodically against bodies of text to pull out articles of interest (i.e., that match the profile). The bodies of text frequently are news stories, product announcements, compendia of trade magazines, the text of legal decisions, etc. In this scenario, we assume that the text is not indexed but appears simply as free full text of the articles. Another application might have indexed articles in a bibliographic system. Given an index, searches are almost always faster using the index, and Splash 2 is unlikely to be able to be of much assistance. However, it is becoming more and more common even in bibliographic databases to have text present (such as abstracts or summaries of journal articles or technical reports) which is not indexed. In such circumstances, one still has a full text search to perform.

The basic technique one is likely to explore in Splash 2 was outlined by Dan Lopresti in his short note [LOPR] and resembles the Unix spell checker. Given a dictionary D of N words, one constructs a hash function $f(w)$ and a memory array. Stored in the memory array is a 1 at locations $f(w)$ for words $w \in D$ and a 0 at locations to which no word in the dictionary hashes. Given an input string of characters s, $f(s) = 0$ implies $s \notin D$. If $f(s) = 1$, then with some probability p (depending on f and on the lengths of the dictionary and the memory array) we know that $s \in D$. If we apply n hash functions $f_n(w)$, for each of which we have probability p that $f(s) = 1$ implies that s is in the dictionary, and we assume the functions are independent, and if $f_i(w) = 1$ for all i, then with probability $(1-p)^n$ the string s is not in the dictionary. Searching for words in the dictionary or words not in the dictionary is merely a matter of inverting the 1's and 0's in the memory array.

This basic approach has been programmed on Splash 1 using a number of hash functions. Characters are streamed into the processors, each of which implements a single hash, and those words which "survive" through the pipeline of hash functions are scored as hits. The same approach would work for Splash 2, but some enhancements are obvious. Certainly the larger memories will allow for larger memory arrays and thus hash functions with a higher probability of success. Next, the crossbar or the more global communication lines could be used to transmit success or failure information. The limiting factor in this application will still be bandwidth—Splash 2 can process words (compute hash functions) as fast as they can be input to the system over the SBus.

DNA Pattern Matching.

The DNA pattern matching problem is by now fairly well known. Given two strings of DNA, represented as strings in a four-character alphabet (A, C, G, T), the problem is to compute the edit distance between the two strings (based on character insertions, deletions, and substitutions in the usual case of imperfect matching of the strings). A common algorithm is the construction of a matrix of distances between strings $a_1a_2...a_m$ and $b_1b_2...b_n$ using the recursive formulas

$$d_{i,j} = \min \left\{ \begin{array}{l} d_{i-1,j} + del(a_i) \\ d_{i,j-1} + ins(b_j) \\ d_{i-1,j-1} + sub(a_i, b_j) \end{array} \right\}$$

where $del(\cdot)$, $ins(\cdot)$, $sub(\cdot,\cdot)$, the deletion, insertion, and substitution costs, are often taken to be 1, 1, and 2, respectively.

This application maps well to Splash [GOHKa] but on Splash 1 was only able to run with a 1 Mhz clock due to I/O limitations. Even with these limitations, it was competitive with or much faster than supercomputers; with the improved speed due to increased I/O on the SPARC SBus, yet another order of magnitude speed increase can be expected.

Acknowledgements.

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

References

[BROO] Fred Brooks, *The Mythical Man-Month*, Addison-Wesley, Reading, Massachusetts, 1975, The notion of "second-system effect" seems to come from Brooks, although this precise definition comes from *The Hacker's Dictionary*, by Guy L. Steele, Jr..

[GOHKa] 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.

[GOKHb] Maya Gokhale, Andrew Kopser, Sara Lucas, and Ronald Minnich, *The Logic Description Generator*, SRC Technical Report SRC-TR-90-011 (1990).

[KUNGa] H. T. Kung, *Why systolic architectures?*, IEEE Computer **15** (1991), 37-46.

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

[LOPR] Daniel P. Lopresti, *Fast dictionary searching on Splash*, SRC report (1991).

[MCHEa] John T. McHenry, *Dictionary search application on Splash*, SRC report (1991).

[MCHEb] John T. McHenry and Andrew Kopser, *Keyword searching on Splash*, SRC report (1991).

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

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