AN ANALYSIS OF FPGA-BASED CUSTOM COMPUTERS FOR DSP APPLICATIONS

Neil W. Bergmann and J. Craig Mudge

CSIRO/Flinders Joint Research Centre in Information Technology,
Flinders University, GPO Box 2100, Adelaide, AUSTRALIA

ABSTRACT

Field Programmable Gate Arrays (FPGAs) can be rapidly reconfigured to provide different digital logic functions. When such FPGA logic circuits are incorporated within a stored-program computer, the result is a machine where the programmer can design both the software and the hardware that will execute that software. This paper first surveys this area of custom computing. It then describes a new custom computing architecture which uses a processing node with three sections: a standard arithmetic chip, static RAM and reconfigurable logic for operand handling. Finally an analysis of the suitability of this new approach for implementation of DSP applications shows it to be worthy of further investigation.

1. INTRODUCTION TO CUSTOM COMPUTING

Field Programmable Gate Arrays are now a popular implementation style for digital logic systems and subsystems [1]. Where the programming configuration is held in static RAM, the logic function implemented by those FPGAs can be dynamically reconfigured, in fractions of a second, by rewriting the contents of the SRAM configuration memory.

When such FPGA logic circuits are incorporated within a stored-program computer, the result is a machine where the programmer can design both the software and the hardware that will execute that software. Such a machine, where the hardware can be reconfigured and customised on a program-by-program basis, is called a *custom computer*. With the advent of SRAM-programmable FPGAs, there is now an increasing worldwide interest in custom computing, as evidenced by a recent workshop in this area [2].

Several researchers report algorithm speed-up rates of hundreds or thousands of times compared to conventional desktop workstations, and often significantly greater than the best reported results using "conventional" supercomputers. This paper surveys some of the most important custom computers, presents the authors' work on a new custom computing architecture specifically designed to support DSP applications, and analyses its performance with respect to implementation of DSP algorithms.

Note: This work is supported by the Australian Research Council under Grant A49132307.

2. PREVIOUS METHODS OF CUSTOMISING COMPUTERS FOR DSP

There is a constant tension in computer design between being general purpose, i.e., doing a wide range of computational tasks moderately well, and being application specific, i.e., doing a smaller range of computational tasks much better (at the cost of either increased system resources or of poorer "general purpose" performance). Previous approaches to improving the performance of a general purpose computer for specific applications include the addition of application specific hardware such as image compression boards, the addition of a parallel processing sub-system to a host processor [3], and the use of a writable control store within a microprogrammed computer to implement application-specific machine instructions [4].

3. EXISTING CUSTOM COMPUTERS

3.1. SPLASH

SPLASH and SPLASH II [5] are custom computers which have been developed at the Supercomputer Research Corporation, Maryland. The SPLASH II processor consists of an extendable number of processor boards, with each board containing 16 Xilinx 4010 FPGAs [6], connected as a linear array, plus an extra Xilinx 4010 for control, with all of the FPGAs having some additional interconnections via a central crossbar switch. The SPLASH boards can communicate with a SUN Sparcstation host via input and output FIFOs, which are on an additional interface board connecting the SUN and SPLASH array.

Once configured, data is streamed through the SPLASH processor using the systolic array programming model, with results streaming back to the host. The SPLASH processor is most suited to simple streaming operations, and has shown significant

0-7803-1775-0/94 $3.00

speedups over conventional supercomputers for tasks such as text searching and genetic database searching.

SPLASH is usually programmed by specifying the function of the FPGAs using VHDL, which is then automatically translated into an FPGA configuration file. Current research [5] is examining the use of other programming languages, such as data parallel C.

3.2. Programmable Active Memory

The PAM (Programmable Active Memory) has been under development at DEC's Paris Research Labs for several years [7], [8]. The latest version consists of a 5x5 array of Xilinx 3090 FPGAs, connected to local 32-bit wide RAM banks, and also, via a 100MB/s TURBOchannel interface, to a DEC desktop workstation. The workstation writes data to the RAM banks, which is processed by the Xilinx array and returned to the RAM banks, and thence back to the host.

Programming the PAM consists of designing software components for the host, and hardware components for the PAM array. The latter can be done by writing a program in a conventional programming language (Lisp, C++, and Esterel are used) using a specialised library. The program describes logic modules by their bit-level logic equations, or by using standard library modules such as adders, and registers.

Ten applications are described in [8], including long multiplication, RSA cryptography, data compression, string matching, heat and Laplace equations, a Boltzmann machine neural network, 3-D graphics acceleration, and the discrete cosine transform. Results are very encouraging; e.g., the PAM implementation of 512-bit RSA cryptography was faster than any other reported implementation in any technology as of February 1990, and 10 times faster than the next best reported implementation on a custom VLSI circuit.

3.3. The Virtual Computer

The Virtual Computer [9], from the Virtual Computer Corporation, provides approximately 500,000 programmable logic gates, using an array of 52 Xilinx 4010 FPGAs and 24 ICUBE Field Programmable Interconnect Devices, 8 megabytes of SRAM, and 16k x 16-bit 25ns dual-port RAMs. The system also has 3 x 64-bit I/O ports - one for hardware configuration / readback, and two for general purpose I/O such as connection to a host workstation. The central processing array of 40 Xilinx FPGAs, called the Virtual Array, consists of four Virtual Pipelines, each with 10 FPGAs connected to dual-port RAMs at each edge of the pipeline, with the other port of these dual-port RAMs connected to a control FPGA with a RAM buffer.

The array is intended to be a flexible computing resource, with a typical application on a host processor loading data into SRAM, which is streamed through the pipeline under local control via the dual-port RAMs. Results would take the reverse route back to the host. All of the FPGAs are designed to be reprogrammed in parallel, so that the function of all 500,000 logic gates can be changed in 25 ms.

It is intended that programming could be at several levels. At the highest level, a run-time function library would be provided. Next would be translation of VHDL or C++ code into hardware implementations, and at the lowest level would be full-custom hand placement and routing. The Virtual Computer is interesting in that it is designed as a commercial product (initially targetted at the research community), rather than as a research vehicle.

4. A NEW CUSTOM COMPUTER

4.1. Weaknesses of Existing Architectures for DSP

An analysis of those applications where custom computing has shown speedups of several orders of magnitude reveals that those areas where current custom computing architectures show the greatest promise is where an application can be decomposed into many components, each of which can be computed in parallel, and where each of these parallel computations can be implemented by a low-gate-count processing element, with many such elements being able to be implemented on the available FPGA resources of the custom computer. Typical applications in this category include text searching and genetic database matching.

DSP applications often exhibit a high degree of potential parallelism, but are less suitable for implementation on a custom computer because of the high gate count of the arithmetic operations (addition and multiplication) typically required for each parallel processing element. For this reason, the authors are now exploring a new custom computing architecture explicitly targetted at the efficient implementation of digital signal and image processing applications.

4.2. Our New Architecture

Our decision to concentrate our efforts on a custom computing architecture which is specifically designed for the support of DSP algorithms leads us naturally to the provision of specific hardware support for the arithmetic operations which dominate many DSP algorithms, but which are costly to implement using FPGA programmable gates.

Our proposed architecture has developed from previous work into the rapid prototyping of DSP algorithms [10]. A current experimental system consists of an Algotronix CHS2x4 (Configurable Hardware System) [11] PC plug-in board containing 8 CAL1024 chips and 512 kbytes of RAM, with the addition of a second plug-in board with 4 custom VLSI chips, each containing four 16-bit, bit-serial multiply-accumulate units. The major drawback of our initial architecture is its reliance on a custom designed arithmetic resource chip. Our limited design and fabrication budget means that such custom-designed chips have only moderate arithmetic performance, and even a small array of such

chips will at best be able to match the arithmetic performance of modern CPU chips.

We have therefore commenced work on a more ambitious architecture [12], which more closely couples an arithmetic chip, static RAM, and reconfigurable logic within a processing node. This node is then replicated a number of times to produce a complete custom-computing co-processor for a workstation. It seems likely that this architecture will use commercially available arithmetic chips, providing of the order of 20 MFLOPs each. Eight such processing nodes would give some 160 MFLOPs of processing power. The architectural and algorithmic challenge then becomes to ensure that the reconfigurable logic chips and static RAM chips can store, communicate and organise operands for these arithmetic chips to allow such a peak processing performance to be sustained.

5. ANALYSIS OF CUSTOM COMPUTING ARCHITECTURES FOR DSP

As mentioned previously, we claim that many existing custom computing architectures may be less suitable for DSP algorithms than our more specialised architecture presented above. Hence we present a relatively simple comparison between different custom computing architectures, and also between custom computing architectures and other programmable implementation methods for DSP algorithms. The following assumptions are made:

Assumption 1. The limiting factor in DSP performance is the computation of multiply-accumulate operations, which dominate many DSP algorithms, such as filters, transforms, and neural networks.

Assumption 2. For the previously-mentioned conventional custom computing architectures, which all use Xilinx FPGA's, the cost of a fixed-point multiplication is based on a recent design by Casselman [13]. This 24-bit fractional multiplier requires 48 CLBs (configurable logic blocks) within a 4000 series Xilinx FPGA, and produces a result in 16 clock cycles at 16 MHz. We assume that a fixed-point addition can be done in the same time using an extra 2 CLB's for a total of 50 CLB's for a fixed-point multiply-accumulate operation, at 1 million operations per second.

Assumption 3. For the same conventional custom computing architectures, the cost of a floating-point multiplication is based on the same design by Casselman [13]. A single-precision floating point multiplier requires 60 CLBs within a 4000 series Xilinx FPGA, and produces a result in 16 clock cycles at 16 MHz. We estimate, given some experience with previous floating-point operator designs [14], that a floating-point adder requires a similar-sized circuit to a floating-point multiplier, giving a total of 120 CLB's for a floating-point multiply-accumulate operation, at 1 million operations per second.

Assumption 4. For the same conventional custom computing architectures, it is assumed that 20% of chip area is required for control, and other overhead hardware. Most of the systems described use Xilinx 4010 FPGA's, with 400 CLB's. Accounting for the 20% overhead leaves 320 CLB's for the multiply-accumulate operations.

Assumption 5. For our new custom computing architecture, we assume 8 processing nodes, where each node is provided with a custom arithmetic support chip. While the exact custom arithmetic circuit to be used has not been decided, typical floating-point coprocessor chips can do parallel multiply and accumulate operations in 100ns. Assuming operands can be input at a sufficient rate to keep the FPU continuously processing, 10 million multiply-accumulate operations per second per node (80 million multiply-accumulate operations per second for the system) can be achieved. We assume that the reconfigurable logic chips in the system are used to control data flow to ensure that the FPU's are kept busy.

Assumption 6. For comparison, a fast uniprocessor, with hardware floating-point support is used - the DEC Alpha 21064 chip [15]. The machine has a pipelined ALU which can run at 200MHz. Linpack 1000x1000 results [15] suggest a best-case sustained computation rate of 75% of the peak rate. Reducing this to 50% to account for general algorithm and system overheads, this gives 50 million multiply-accumulate operations per second.

Assumption 7. For comparison, a parallel processing system using 4 TMS320C40 DSP processor chips, is used [3], with a peak processing rate of 4 * 80MFLOPs (each chip can do parallel floating-point multiply and add operations at 40 MHz). Since the processors have to deal with all overhead instructions, as well as the floating-point operations, we suggest the same average floating-point computation rate as in assumption 6, i.e., 50% of the peak.

Table 1 shows the results of a simplified performance comparison, based on the above assumptions, between a single SPLASH II processor board [5], the DEC PAM [8], the Virtual Computer [9], our new custom computer for DSP, referred to as CC-DSP, the TMS Parallel Processing Development System [3], and a workstation with a DEC Alpha CPU [15].

The "# of chips" column gives the number of chips required for processing and interprocessor communication, and includes FPGA's and field-programmable interconnection chips, but excludes host interface chips and memory chips. This is meant to give some measure of the "cost" of the system. The "(# of Xilinx)" gives the number of FPGA's used for implementing arithmetic operations in the first 3 systems. The last four columns give an estimate of the sustained and maximum multiply-accumulate computation rates for the systems, using assumptions 6 and 7 above for the last two systems, in millions of multiply-accumulate operations per second (megaMAC/s), and the same measure per

System	# of chips (# of Xilinx)	megaMAC/s (fixed-point) sustained/max	megaMAC/s/ chip (fixed-point) sustained	megaMAC/s (floating-point) sustained/max	megaMAC/s/chip (floating-point) sustained
SPLASH II board	32 (16)	96 / 96	3	48 / 48	1.5
PAM	40 (25)	150 / 150	3.75	75 / 75	1.9
Virtual Computer	76 (40)	240 / 240	3.1	120 / 120	1.6
Our CC-DSP	16	80 / 80	5	80 / 80	5
TMS-PPDS	20	80 / 160	4	80 / 160	4
Alpha Workstation	10 [nominal cost]	50 / 100	5	50 / 100	5

Table 1: Relative performance of DSP implementation solutions

logic & interconnection chip.

Two conclusions can be drawn from the above, simple analysis. The first is that our hybrid custom computer architecture for DSP gives a moderate performance improvement over other custom computing approaches for DSP applications, on a computation-per-chip basis, and so is worthy of further investigation. The second conclusion is that, for DSP applications, single-board custom computing approaches do not seem to give any significant improvement over the conventional parallel processing and uniprocessor approaches.

Finally, a general comment on the sustainability of special purpose, or custom, architectures can be made. The TMS320 parallel processing system and the Alpha workstation described here are both general purpose systems. Hence their technology progress functions will approximate that of logic technology. The new custom computing architecture presented here, because it is special purpose, will be manufactured in lower volumes and hence will tend to have a less steep progress function. Thus, it will be challenging for the custom architecture to sustain a competitive position. It is necessary that the arithmetic, SRAM and operand handling sections have the same slope of progress function as the general purpose systems. A further improvement might be to design an operand handling architecture which is scalable and so fully uses the technology improvement of each new generation of FPGA.

We plan further work to (a) get better performance measures than those used in the simple analysis presented in Table 1, and (b) discern a more scalable design.

6. REFERENCES

[1] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays," *Proceedings of the IEEE*, 81(7), July 1993, pp 1013-1029.

[2] *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*, Napa, April, 1993.

[3] "TMS320C40 Parallel Processing Development System", *TMS320C4x Technical Brief*, Texas Instruments, 1991, pp 6.5-6.8

[4] L. R. Morris, and J.C. Mudge, "Speed Enhancement of Digital Signal Processing Software via Microprogramming a General Purpose Minicomputer," *Proceedings of ICASSP '77*, 1977.

[5] M. Gokhale, R. Minnich, "FPGA Computing in Data Parallel C," *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*, Napa Valley, April, 1993.

[6] The XC4000 Data Book, Xilinx, Inc., San Jose, 1992.

[7] P. Bertin, D. Roncin, J. Vuilemin, "Introduction to Programmable Active Memories", *DEC Paris Research Labs, Research Report #3*, June, 1989.

[8] P. Bertin, D. Roncin, J. Vuilemin, "Programmable Active Memories: a Performance Assessment", *DEC Paris Research Labs, Research Report in preparation*, 1993.

[9] S. Casselman, "Virtual computing", *Proceedings of IEEE Workshop on FPGA's for Custom Computing Machines*, Napa Valley, April, 1993.

[10] N.W. Bergmann, H. Wong, P. Sutton, "A Rapid Prototyping and Implementation Method for Systolic Array Digital Signal Processors", *Proceedings of 1991 Microelectronics Conference*, Melbourne, pp 91-93, 1991.

[11] *CHS2x4 User Manual*, Algotronix Ltd, Edinburgh, Scotland, February, 1992.

[12] N.W. Bergmann, J.C. Mudge and L.R. Cirroco, "FPGA-Based Custom Computers", *1993 Australian Conference on Microelectronics*, Gold Coast, October 1993.

[13] S. Casselman, "FPGA multiplier", *info-fpga electronic mail posting from sc@vcc.com*, October, 1993.

[14] D. Giarola & N.W. Bergmann, "Design and Testing of a Bit-Serial Floating Point Adder", & "Design and Testing of a Bit-Serial Floating Point Multiplier", *University of Queensland, Electrical Engineering Dept., Internal Reports No. EE89/5 & EE89/9*, May, 1989.

[15] E. McLellan, "The Alpha AXP Architecture and 21064 Processor", *IEEE Micro, 13(3)*, June 1993, pp36-47.