FPGA COMPUTER ARCHITECTURES

Marek Perkowski

Department of Electrical Enginering, Portland State University.
P.O. Box 751, Portland, Oregon, 97207.

ABSTRACT

We present the recent research and development in the area of computer architectures designed from regular assemblies of FPGA devices (we will call them FPGA computers for short).

1. INTRODUCTION

FPGA technology proved itself already practical in two areas: to realize parts of digital systems (especially computers), and in the area of fast prototyping/hardware emulation of digital systems.

In the *first area* there are three types of products:

1. In case when a product of not high volume is expected, FPGA is preferred to ASIC to decrease the cost. A verified product that is in high demand is next migrated to an ASIC.
2. One piece of hardware plays several roles. This is done to decrease the cost or power consumption, to allow further modifications by software upgrades, decrease size, etc.
3. The circuit cooperates with a smart sensor or other analog/digital/mechanical subsystem which is changing its operation with time and needs therefore tunning and modification as the time progresses. Or, this subsystem is "user-upgradeable".

The *second application area* is to replace the system's software simulation during its design, with its emulation in hardware. This allows for quick design cycles and concurrent development of software and hardware. Also, it allows to observe effects that are not possible in standard simulation, for instance observing images, or hearing to sounds while developing multimedia systems. Companies like Quickturn, PiE Design Systems, and INCA [25] are selling massive general-purpose hardware emulation engines, and a patent for another concept of a general purpose hardware emulator has been awarded to employees of Mentor (now in Quickturn). Recent merger of Quickturn and PiE Design Systems to Quickturn Design Systems will perhaps consolidate market and provide even better service to customers. In addition, several other companies are developing all kinds of in-house specialized systems for fast prototyping and experimental FPGA-based emulators.

Fast development of FPGA and hardware emulation technology suggest that there may be some other opportunities open for them in the area of general computing. Are they? The goal of this paper is to present recent research and developments, investigate these chances, and try to predict the future.

2. THE IDEA OF THE FPGA COMPUTER.

Obviously, the next step of emulation technology will be to improve software platforms and migrate to higher level languages for design specification. This will have also strong influence on the emerging arena of *FPGA computers* outlined below. Hardware emulation will soon use VHDL, Verilog, or perhaps even higher-level specification languages. Similiar software base would be then required for the existing "hardware emulation" and emerging "FPGA computer" markets.

While the "glue logic" and "hardware emulation" areas can be treated as pretty well established, with quickly increasing and potentially large markets, there has been much controversy about another concept made possible by FPGAs - the *configurable computer architectures, or "FPGA computers"*. Such computers have been proposed in the last few years by several research groups.

Although several very impressive numerical results of FPGA computers have been demonstrated [3,6,14], there seems to be no major industrial company pursuing this idea commercially. The existing companies such as Algotronix and DEC seem to be rather oriented at the education and government markets. To our knowledge there is no company that did find a large market, other than research and education. This seems at first amazing, since the idea of creating a hardware-programmable computers, with supercomputer performance and at a fraction of cost, is very captivating. One can then wonder, what is the reason of such situation, is it the industry's conservatism, or just time is needed for the technology to mature and become widely accepted?

At this point we can recall some other computing ideas from the past. Babbage never completed his "Engine". Although everybody appreciated the idea of "cellular computer" of Ulam and Von Neumann, it took forty ears to build one [37]. IBM didn't believe in large market for computers, and next, for personal computers. One can give many examples such as those in the area of computing. Is there then a bright future for FPGA computers, and only time is needed for their common acceptance?

On the other hand one cannot forget another examples from the past of computing. There have been several seemingly good ideas that didn't meet commercial appeal, or are still striving for practical acceptance: Lisp computers, hypercube parallel computers, object-oriented hardware, systolic and cellular processors, or Japanese Fifth Generation logic computing plan. All these projects met one basic obstacle: standard computer technology (both hardware and software) was progressing so rapidly that the companies were not able to deliver competitive products based on their distinguishably new technologies. Will this be also a future of FPGA computers? What can we learn from history? In the enthusiasm for FPGA computers one cannot forget that the competing technologies of ASICs, standard computers, general purpose DSP processors and general purpose parallel processors are also not standing in place.

Several important related questions can be asked:

1. Will the FPGA systems always remain a domain of research and educational tools, hardware emulation and fast prototyping?
2. Do we know a single commercial application that requires boards heavily loaded with FPGAs for *final, sustaining* applications? What is it?
3. Are there any broader areas where FPGA computers will be in the *long run* competitive to general purpose processors, DSPs, ASICs, or general purpose parallel processors?
4. What are the application area niches, if any, that are particularly suitable for FPGA-based computers, and in which there is definitely no danger of competition from other technologies?
5. Is there a need to create new instruction sets for domain-oriented general purpose computers? If yes, why should such FPGA computers find better acceptance than (similar to certain extent) microprogrammed computers and VLIW computers?
6. Will the costs of developing completely new programming environments for FPGA computers be always too prohibitive?

3. UNIVERSAL VERSUS SPECIALIZED ARCHITECTURES.

There are two basic ways of designing computer hardware that are used today: to design it from fixed (hard-wired) components (off-the shelf and ASICs), and to design from FPGAs and/or other programmable devices. The second approach is used in emulators and several FPGA computers, where the entire architecture is mapped to an array of programmable components. While the first approach provides better speed and smaller cost, the second approach gives total flexibility of creating new architectures.

As observed by Hartenstein [17-23] and few other researchers, there is a third way between the two extremes of a totally electrically programmable logic (FPGAs), and a totally hard-wired one (microprocessors). Regular computer architecture have been proposed with fixed hierarchical floor-plans in which some blocks and some connections are electrically programmable. This restriction of programmability allows to create efficient and still flexible compromises and to find the best possible trade-off.

Moreover, because this new approach introduces certain fixed architectural blocks with fixed connections, it suggests to re-think the general Von Neumann and Harvard computer paradigms in light of programmability of blocks. One can think about this approach as of taking a well-known architectural paradigm, such as pipelined, hypercube, data-flow, systolic or other architecture, and next considering which fixed blocks of the architecture can be made programmable and what is gained by that generalization. For instance, when one realizes control logic of a processor as a microprogrammed unit, new instructions can be created by changing PROM programming of the unit. These instructions, however, will be always combinations of some elementary micro-instructions realized in ALU. It can be then observed that with FPGA technology also ALU can be easily generalized. In some ALUs, any of 16 Boolean functions of two binary variables can be realized in a bit slice, and next repeated bit-by-bit for the length of the word. If now we take a pair of bits as a basic chunk of information, a universal logic cell with four inputs and two outputs realizes any function on information encoded on pairs of bits (4-valued logic) [33]. Similarly as in ALU, this programmable pattern is repeated n times bit-by-bit for the length of the word of $2n$ bits. In this easy way we obtain an astronomical number of new mappings from input words to an output word. Thus, we were able to create many new micro-instructions in our "generalized ALU". Now, combining the programmability of control (microprogrammed control) with the programmability of data (generalized ALU) we obtain even more powerful pattern of system's programmability. This is however only the first step. While a simple carry signal is used in ALUs for few arithmetic operations, adding more "carry signals" going from the least to most significant bit, and adding few "confirm signals" going from the most to least significant bit allows to realize many more operations on input words. Such "generalized iterative unit", ILU, has also many other applications to realize non-standard arithmetic, logic, shift, string-, image-processing, and algebraic operations [33]. Since there is a feedback from the ILU's output through a register to one of its

inputs, this generalized ALU emulates very efficiently the "one-dimensional cellular automata", that have many applications in physics, image processing, mechanics, and biology [27,37] (the well-known Gardner's "game of life" is an example of a two-dimensional cellular automaton).

There are also many other ways to use the power of programmability to increase the power of the "generalized ALU". Such "generalized ALUs", "Programmable ALUs", "Cache logic", or Reconfigurable ALU", have been the main idea introduced in all FPGA computers proposed until now, and are responsible for dramatic speed-ups of DEC's Perle [6] or Splash II [3] computers.

Can we play longer this game of increasing the efficiency by making fixed blocks of traditional architectures, programmable? The questions that may be asked why designing these architectures are: (1) What are the main blocks? Which of them should be done programmable, and which not? (2) How should be the blocks connected? (3) What is the best technology (FPGA type) to implement such blocks. Definitely, by taking a well-known Von-Neumann or Harvard architecture prototype and making its blocks programmable we can design new architectures that will be more powerful for some applications. The same can be done for pipelined, Very Long Instruction Word (VLIW), systolic, hypercube, butterfly, or data-flow architectures. However, each of these generalizations provides us with a different trade-off between architectural gains (hardware's structure better fitting the problem to be solved) and technological losses (inevitable loss of speed of basic gates and increased cost in FPGA). The question may be thus asked, the generalization of which particularly architecture is the best? Or better, what *new* architectures will be even more suited to the advantages provided by the programmable technology. The conclusion of some authors is then: "why shall we be slaves of the existing architectural paradigms, let us design new general computer paradigms that the new programmable technology allows." In his well-documented critique of existing architectures, Hartenstein shows that his new paradigm, XPUTER, can be essentially better on several classes of problems than the Von Neumann and other known architectures [17-23].

Several authors describe various kinds of what we call *Field-Programmable Cellular Tissue (FPCT).* This is a two-dimensional tissue of cells that realize systolic or cellular calculations. It is a "generic data processor", personalized to its specific functions by electrical programmability of some of its component blocks and/or connection networks. The programmable blocks are of the following types: RAM - programmable algorithm, FPGA - programmable logic of the hardware, FPID - field programable interconnection devices - to dynamically reconfigure connections between other blocks.

While several authors invented independently various new programmable components and subsystems of *specialized FPGA computers* for certain tasks, the effort of Hartenstein's group is unique in its long-ranging attempt at *re-inventing the general-purpose computer.* Rather than designing an efficient architecture for a class of problems, as done by Paris DEC Labs or Supercomputing Research Center, they try to find a single ideal structure of a new-generation computer based on programmable logic. His main idea is to have a data driven rather than control driven calculations and to demonstrate that regular data-flow methods known from specialized address generator-based DSP processors (Honeywell, Sharp) can be also efficiently realized for many other applications.

4. THE GENERALIZED COMPUTER

Building on our previous concepts of Universal Logic Machine [33], Hartenstein's ideas of non-Von Neumann general processor, and the newest developments in programmable logic (fine grain FPGAs [12,16,7,1,28,32], and FPID devices, like I-Cube IQ160) our group developed a concept of a *generalized computer.* The concept of the generalized computer is based on the similarities of formal systems used in various areas and is difficult to explain without introducing those formalisms. Not to loose the main idea in the technicalities, we will try below to focus informally on the main concepts, and use these formalisms only as illustrations.

The generalized computer can be then presented as a generalization to many existing formalisms but should be linked to none of them in particular. This is like with a standard computer, that may be explained to a novice as a machine to perform arithmetical operations, but the arithmetical computer can do much more than arithmetic. The same is true with the generalized computer, which was presented as the "Logic Design Machine", the "Cube Calculus Machine" or the "Universal Logic Machine" to point out its certain aspects. This is a machine based on logic operations, but it can be used not only for logic. It is a *general purpose computer* and a *superset of a standard computer.*

Moreover, the presentation of ideas related to FPGA computers is difficult because they have several distinct but mutually related aspects. First, one has a physical hardware composed of: FPGAs and other chips, boards and racks. Next, there is a set of application architectures that are mapped to this hardware and run on it. In case of our generalized computer, or in a similar concept of XPUTERS, there is one more architectural layer between the application architecture and the real hardware, the general realization/programming model, which helps to translate from the high level specification to the physical level, and plays therefore a role similar to assemblers in classical computers. The things are also complicated by the fact that the separation between software and hardware is practically non-existent.

One can look at our generalized computer in one of the following ways:

1. A general CAD accelerator with its own software.
2. A general purpose accelerator of existing software packages which use certain general-purpose operations on sets and Boolean functions (for instance, the logic synthesis tools from U.C. Berkeley use the set covering, binate covering and tautology checking algorithms, which can be essentially speed-up using our machine).
3. A universal non-numerical/numerical computer that internally uses a formal system called Generalized Multiple-Valued Cube Calculus (GMVCC) [33] rather then only the binary arithmetics used in standard computers.
4. A system for fast prototyping of algorithms to solve combinatorial problems that occur in logic synthesis, graph theory and computer vision.
5. A computer vision algorithm emulator that contrary to the existing systems speeds-up not only low-level image processing tasks (such as filtering) but also the medium-level ones (such as Hough transform), and the high level ones (such as the recognition of solids based on consistent labeling, maximum cliques, or other combinatorial models).
6. An "FPGA Compiler" - the method of translating high-level descriptions to FPGA hardware.
7. A methodology to design ASICs directly from high-level specifications, and which links high-level, logic and physical design stages through the notion of "cellularity".

The main idea of the generalized computer is to reduce many combinatorial problems to few, and next solve those few efficiently. This is a well known mathematical idea and the book by Garey and Johnson is an excellent example of reducing all well known NP-hard problems to just one - the 3-Satisfiability Problem. Another close approach is to reduce a problem to unate or binate covering, satisfiability, or tautology.

It seems that there is no escape from using hardware when an exact or a good quality solution is searched for large problems that can be categorized as "pure search models" - satisfiability or tautology verification are here the best examples. So is the chess, with its specialized hardware. Pure search model can be best solved by a computer hardware that is logical in nature and can reconfigure itself easily and quickly to the problem. FPGAs and particularly fine-grain FPGAs are currently the best technology, if not the only one, to build such systems.

In general, the following calculations are particularly efficiently realized in our computer: NP-hard combinatorial search problems using sets and logic; systolic or broader "regular" architectures such as orthogonal transforms, convolutions, and matrix operations; complex operators based on logical functions that do not appear in standard ALUs; applications that require CAMs and bit-serial arithmetic; applications that require non-standard arithmetics; applications that require non-standard algebra and solving equations other than algebraic or differential/difference.

An argument often used against specialized and parallel architectures is that because of the Amdahl Law such architectures will be never competitive to general-purpose sequential processors. This is, however, not true with respect to our machine for two main reasons:

1. In several important problems of our interest a simple algorithm is run on massive data and takes more than 95% of the total time.
2. The class of applications that are efficiently reduced to such problems is quite large.

Concluding, in theory the generalized computer can be applied to every problem, but it is especially efficient on search problems that are expressed in certain formal (search) model. Hopefully, the class of these problems is wide enough to guarantee the practicality of the computer.

5. THE REQUIREMENTS FOR A SUCCESSFUL PRODUCT.

After a study of application areas and market's survey it is the opinion of this author that the following aspects are crucial to the success of FPGA computers.

1. The *single product* must be found (such as a personal supercomputer, general-purpose accelerator, multimedia engine, biomedical imaging, engineering workstation) which will prove *an absolute advantage* of FPGA computer over competing technologies for uses other than research and education. The success of personal computers will be difficult to repeat since much more sophistication will be now required from the programmers without an early payoff. While the personal computers benefited from the available software technology of mainframes, it is unlikely that easy to use tools will be soon available for FPGA computers. Therefore the product we are looking for must give the users not only the incremental improvement with respect to similar existing products, but the higher order of usefulness and the momentous advantage, such as was the spreadsheet program on early personal computers or the first digital TV game.
2. This product must have the existing market base large enough to develop a new system, but not large enough to develop an ASIC-based solution. (Successful introduction of this FPGA-based computer may in turn create a new, currently non-existing market, which will subsequently lead to ASIC-based system. This would prove again that the only role of FPGAs is to create transitory products).
3. The key point to create an FPGA computer niche of the computer market is to find the product that will allow *various* applications that are user programmable/definable, and not just one application. We will call this property *flexibility of applications on the same architecture*. By the application base we understand the number of *various* applications to be run on the architecture. The application base should be nei-

ther too wide nor too narrow. If the application base of the product were too large, a standard next generation computer or a parallel system would always win the market competition because of the price, installed software base, and ease of programming. If the application base were too small, an ASIC chip or ASIC-based special purpose system would be a better solution because of speed and cost. They would win with the FPGA computer either from scratch, or after a transitory period of the FPGA-based system. Therefore, the market segment of FPGA computers seems to be relatively narrow and in any case the road to the FPGA computers is through some way of expanding the FPGA-based fast_prototyping/emulation technology and other current applications of FPGAs.

4. It can be predicted that two separate market segments can be created in future for FPGA computers:

4.1. *scientific "very high performance" applications* that require sophisticated users who will write programs on the "assembly code level" (by which we mean specifying hardware architecture in detail). This group of users is located in educational and government institutions, and at the moment would not create a big enough customers' base to justify the product. Most of the examples of FPGA systems defined until now belong to this category. Although not yet sizable, this group of users will enthusiastically accept new software tools and development environments and can become an important drive of the new technology.

4.2. *mass applications,* such as multimedia computer games with real-time transformations of acquired video/image/sound, now available only in very expensive graphic workstations, or not available at all. This is a large market of unsophisticated users. The application will be the ready "computer code" (hardware programming information); it will be delivered to the final users by the software house and just loaded by the final user to the FPGA computer. On the other hand this market will create the "system integrator" market and the "software house" market. Such software house will need sophisticated tools internally and will not sell them. This will create tool developers market which will ultimately merge with the developers of software tools for scientific applications. This mass product market is now non-existent since such a single powerful application has not yet been found. The closest to achieving this goal seem to be some health-related and biomedical imaging companies which use FPGA-based image processing boards [10]. Pap-smear analysis machines, NMR imaging or SCEPT imaging are just few good possibilities. High performance animation, image/speech recognition and processing, gene-splicing research, drug-design, three-dimensional transformations, digital signal processing and various CAD and scientific accelerators (physics, biology) are other possible candidates. Finally, flexible multimedia-related product linked to QuickTime or similar product, that would allow to realize different algorithms for compression, tiling, morphism in the same hardware, could make a breakthrough in personal computer imaging/multimedia market. These are potentially large and interesting markets, but the long-term rationale of using FPGAs versus ASICs in them is still questionable.

5. The main obstacle to accept the FPGA computer technology will be software. As a comparison, there are no widely accepted methods for writing pipelined or parallel programs, or converting sequential to parallel programs. According to the existing approaches, programming of FPGA-based computers will require at least these techniques. Similarly, there are no widely accepted and commercially available tools for converting high-level descriptions (such as VHDL) to systolic processors. Even much simpler related approaches such as microprogramming, which are known in industry for years and for which a number of tools have been developed, are difficult, time consuming and less used in practice than their already existing technologies make possible. New methods will be also required, such as implementation of recursion and complex data structures in hardware. All these techniques and many other would be necessary to realize a truly universal FPGA computer, unless some powerful niche would be found with narrower software base, or unless some breakthrough in programming technology will occur. Again, finding an "application niche" seems then to be the key to the step-by-step development of this required software technology to support the FPGA computers.

6. A notably higher mathematical and computer science sophistication of engineers and program developers will be needed, and the concepts such as program verification, theorem proving, software/hardware codesign, transformational design, and very high level languages will have to be accepted. This can be done (see how the concepts of logic synthesis and VHDL have been acquired in the eighties) but will take much time and will require new educational efforts.

7. For easier acceptance, the FPGA computers should adhere, as much as possible, to all existing software/hardware, CAD-tool, interface, language, and even mechanical, standards.

6. CONCLUSION.

The concepts of specialized, general-purpose, and generalized FPGA architectures have been introduced. At Northcon presentation, all those concepts will be illustrated with diagrams of the introduced above machines and numerical examples of their work. The reconfigurable architectures built from general-purpose FPGA chips [6,20] were on several problems superior to supercomputers, and proved the usefulness of the concept of electrical reconfigurability applied to special purpose massively parallel processors. It is highly probable that despite existing obstacles in few years a combination of outlined above architectural ideas with new cellular FPGAs [1,7,31,34,35] and their forthcoming respective MCMs will allow to build relatively inexpensive PC-based "supercomputers on a board" for multimedia acceleration.

LITERATURE.

1. Algotronix, The CHS 2*4, the World's First Custom Computer, 1992.
2. Aoki, T., et al, *Proc. IEEE ISMVL,* pp. 364-367, May 1989.
3. Arnold, et al, "SPLASH 2", *Report,*
4. Ast, A., Hartenstein, et al, "Novel High Performance Machine Paradigms and Fast-Turnaround ASIC Design Methods: A Consequence of, and, a Challenge to, Field-programmable Logic", in *FPL '92.*
5. Barrie, P., Cockshott, P. at al, "SPACE: A Scalable Cellular Array Architecture", in *FPL '92.*
6. Bertin, P., Roncin, D., and J. Vuillemin, "Introduction to Programmable Active Memories", *Research Report No. 3, DEC Paris Research Laboratory,* 1989.
7. Concurrent Logic, Application Notes and other company materials, 1992.
8. Cox, C.E., and W.E. Blanz, "GANGLION - A fast field programmable gate array gate array implementation of a connectionist classifier," *Report No. RJ 8290 (75651), IBM Research Division,* Almaden Research Centre, 1990.
9. Cucchhiara, R., et al, and T.S. Cinotti, "FPGAs boost flexibility and performance of a fine grain mesh connected SIMD array for computer vision applications", in *FPL '92.*
10. ELTEC GmbH, "An Important Step in Image Processing," 1991.
11. Fortes, J.A.B., Fu, K.S., and B.J. Wah, "Systematic Approaches for Algorithmically Specified Systolic Arrays in Computer Architecture: Concepts and Systems, *Milutinovic ed.,* North Holland, 1988.
12. Furtek, F., Stone, G., and Jones, I.W., "Labyrinth: A Homogeneous computational medium", *Proc. IEEE Custom Integrated Circuits Conference,* May 1990, paper 31.1.
13. Furtek, F., "An FPGA Architecture for Massively Parallel Computing", in *FPL'92.*
14. Gokhale, M., et al, "Building and Using a Highly Parallel Programmable Logic Array", in *FPL '92.*
15. Gray, J.P., and Kean, T.A., "Configurable Hardware: A New Paradigm for Computation", *Proc. Decennial Caltech Conference on VLSI,* Pasadena, CA, March 1989.
16. Halverson, R., "Hawaii Parallel Computer Project: the Boolean Processor", in *FPL'92.*
17. Hartenstein, R.W., Schmidt, K., et al, "A Novel Compilation Technique for a Machine Paradigm Based on Field-Programmable Logic," in *FPGAs,* pp. 255-270.
18. Hartenstein, R.W., Lemmert, K., "A CHDL-based CAD-system for the synthesis of Systolic Architectures," *Proc. Intl. Symposium on Hardware Description Languages and their Applications,* Washington D.C., North Holland Publ. Company, Amsterdam 1989.
19. Hartenstein, R.W., Hirschbiel, A.G., and M. Weber, "A Novel Paradigm of Parallel Computation and its Use to Implement Simple High Performance Hardware", *Proc. of Joint Conference on Vector and Parallel Processing, CONPAR 90, VAPP IV, Zurich,* Sept, 1990.
20. Hartenstein, R.W., Hirschbiel, M. Weber, "XPUTERS: An Open Family of Non-von Neumann Architectures", *Nr. 195/89, Universitaet Kaiserlautern, Fachbereich Informatik,* Postfach 3049, D-6750 Kaiserlautern, 1989.
21. Hartenstein, R.W., et al, "A Novel Paradigm of Parallel Computation and its use to implement simple High Performance Hardware", *Intl. Conf. on Information Technology,* Tokyo, Japan. Oct. 1990.
22. Hartenstein, R.W., et al, "The machine Paradigm of Xputers and its Application to Digital Signal Processing", *Proc. of 1990 Intern. Conf. on Parallel Processing,* St. Charles, Oct. 1990.
23. Hartenstein, R.W., et al, "A Novel ASIC Design Approach Based on New Machine Paradigm", *IEEE Journal of Solid-State Circuits,* July 1991, Vol. 26, Nr. 7, pp..
24. Heeb, B., and C. Pfister, "Chameleon: A Workstation of a Different Colour", in *FPL '92.*
25. Virtual ASIC, "Concept Silicon partitions your design onto multiple FPGAs", *INCA company materials,* 1992.
26. Kappler, Ch. J., "The MusiCAL Lambda Reduction Machine", in *FPL '92.*
27. Katona, E., "Cellular Processing", Chapter 6 in "Fuzzy, Holographic, and Parallel Intelligence, By B. Soucek and the IRIS Group, pp. 215-229, John Wiley & Sons, 1992.
28. Kean, T., "Using CAL to Accelerate Maze Routing of CAL Designs", in *FPL '92,*
29. Linde, A., Nordstroem, T., and M. Taveniku, "Using FPGAs to Implement a Reconfigurable Highly Parallel Computer", in *FPL '92.*
30. Luk, W., and I. Page, "Parametrising Designs for FPGAs", in FPGAs, pp. 285-295.
31. Moore, W., and W. Luk, FPGAs, Edited from the Oxford 1991 International Workshop on Field Programmable Logic and Applications, *Abingdon EE &CS Books,* Abingdon England,1991.
32. Page, I., and W. Luk, "Compiling OCCAM into FPGAs", in *FPGAs,* pp. 271-283.
33. Perkowski, M.A., "A Universal Logic Machine", *Proc. of the IEEE ISMVL'92, the 21st International Symposium on Multiple-Valued Logic,* Sendai, Japan, May 27-29, 1992, pp. 262-271, invited address.
34. *Proceedings of ACM FPGA'92 Workshop,* Hotel Durant, Berkeley, 16-18 Febr. 1992.
35. *Proceedings of Workshop of Field-Programmable Logic and Applications,* FPL'92, Vienna, Austria, 31 August - 2 September 1992.
36. Proceedings of the Workshop on FPGA custom architectures, *Napa Valey, California, April 1993.*
37. Toffoli, T., and M. Margolus, "Cellular Automata Machines", *MIT Press, Cambridge, Mass., 1987.*