An Evolvable Hardware FPGA for Adaptive Hardware

Pauline C. Haddow and Gunnar Tufte
The Norwegian University of Science and Technology
Department of Computer and Information Science
O. S. Bragstadsplass 2E, 7491 Trondheim, Norway
pauline@idi.ntnu.no, gunnart@idi.ntnu.no

Abstract. Can we realise the opportunities that lie in design by evolution by using traditional technologies or are there better technologies which will allow us to fully realise the potential inherent in evolvable hardware? In this paper we consider the characteristics of evolvable hardware — especially for adaptive design, and discuss the demands that these characteristics place on the underlying technology. We suggest a potential alternative to today's FPGA technology. The proposed architecture is particularly focused at reducing the genotype required for a given design by reducing the configuration data required for unused routing resources and allowing partial configuration down to a single CLB. In addition, to support adaptive hardware, self-reconfiguration is enabled.

1 Introduction

The design productivity gap in the electronic industry is a well known fact. How can the design community utilise the design capacity that technology is offering and at the same time ensure its correctness? To find solutions to the problem of developing large and complex designs, new design paradigms are required [Sem97, ÅN95].

One approach is to turn away from traditional design techniques following the various design and testing phases and instead allow hardware to evolve until a correct solution is found. The principles of artificial evolution are based on Darwin's theories of evolution by natural selection. These theories were first adapted to the field of artificial evolution in 1975 [Hol75]. The main features of reproduction by cloning, mutation and crossover through the exchange of and alteration of genetic material are still present. However, the selection process is artificial instead of natural and selection is based on how well the individuals solve a predefined task. Evaluation of individuals is based on a fitness function which generates a fitness value for the given individual. The fitness value is used by the selection mechanism to select individuals after a given algorithm.

Two main methods have been established for applying artificial evolution to the design of hardware systems. These are Extrinsic and Intrinsic evolution [dG93, ZPV96]. In Extrinsic (off-line) evolution a design is evolved using software simulation to evaluate intermediate solutions whereas in Intrinsic evolution the intermediate solutions are implemented in hardware. An interesting combination of these two methods termed Mixtrinsic evolution has been introduced at JPL [SZK00] where part of the population is evaluated extrinsically and part intrinsically. A further method exists where both the evolving design and the evolution process itself are implemented in hardware e.g. CHE [TH99] and PIG [Mac99b]. When individuals of the evolution process are tested out in the actual hardware, this provides opportunities for the evolution process to explore the underlying technology's physical properties [THH96].

If a design which meets specified goals is created and the evolution process is designed to be triggered by changes in the surrounding environment so as to restart evolution, then a flexible design may be found i.e. an adaptive design. Adaption to hardware failure may be found today but mainly in a more traditional sense where adaptivity is built into the design at design time. That is, failure cases are identified and adaption to these specific failures is built into the design. Research into adaptive machines takes a slightly different approach in that adaption to failure may be learned.

Evolution not only introduces testing to each phase or generation of the design process but provides the opportunity to search a larger design space than that available using traditional design techniques. This is achieved by removing some of the design constraints built into traditional design techniques which are often built into traditional design tools. In addition, evolution removes constraints imposed by the human designer. These constraints are limitations to the designer's thought process which limit the ability to think out all possible solutions for a given design.

The main focus of the research community has mainly been directed towards traditional technology platforms and applying evolutionary techniques to these platforms to evolve new and hopefully better designs and thus prove the worth of evolvable hardware as a design technique. However, can we really realise the opportunities that lie in design by evolution by using traditional technologies or are there better technologies which will allow us to fully realise the potential inherent in evolvable hardware?

In section 2 we discuss the underlying technologies in the field of evolvable hardware. Our focus is on digital technology and particularly FPGA technology. FPGA technology is further discussed in section 3. To obtain a picture over the requirements that evolution places on technology we first, in section 4, discuss the characteristics of evolvable hardware and then in section 5 discuss the technology requirements themselves. In section 6 we present our proposed structure for an evolution oriented FPGA and finally in section 7 we present a summary of the paper.

2 Technologies for Evolvable Hardware

Most work to date has focused on using digital technology although it may be used in a way so as to exhibit analogue behaviour. Most research may be split into two categories: those using existing re-configurable chips and those that have created systems for evolvable hardware.

Designs have been evolved using either digital or analogue chips. Within digital design there are a number of people working on FPGAs [TLZ99, FMT98, TH99] where the chips chosen have varied between non-partially re-configurable FPGAs i.e. XILINX 4000 series [XDA97] and Lattice GAL PLD [HIM94] and partially re-configurable FPGAs i.e. XILINX 6200 series [XRE97]. It should be noted that the Xilinx 6200 series is no longer available and therefore Xilinx virtex series [XVE98] which is also partially re-configurable is now the focus of a number of researchers attention. Analogue work is based on FPAAs [FS98, ZPV99, Ozs98] using Zetex devices or Motorola devices.

Researchers that have put together EHW systems include Thomson [Tho95] using RAM, TTL logic sensors and motors; de Garis [dG96] — several FPGAs and RAM chips and Sipper [SGM$^+$97] — several FPGAs with additional logic.

However, if evolution aims to reduce the constraints of traditional design is it possible to reduce the constraints of the technology itself by introducing technologies better suited to evolution?

One approach aimed at trying to find a suitable technology for evolution is that of Layzell [Lay98] involving the evolvable motherboard. This work looks at how evolution exploits the underlying hardware, both the logic blocks and their interconnections. Different logic elements may be placed in the logic blocks for evaluation and a programmable crossbar switch allows a number of interconnection topologies to be realised. In FPGA technology it is hard to analyse the hardware as measurements are only available at the input and output pads. On the evolvable motherboard, measurements are available at the transistor level.

Other approaches involve the creation of chip prototypes for use in EHW. One such approach is the Processing Integrated Grid (PIG) re-configurable platform [Mac99a]. Each cell may be reconfigured by its neighbouring cells i.e. a self-re-configurable chip. A distributed ringed genetic algorithm is implemented which requires that all GA operations are in hardware, the results of these operations i.e. new individuals, are implemented on the same hardware.

Other solutions in the digital field include ETLs GRD chip [MYK$^+$99]. This chip combines different technologies — a RISC core processor combined with a re-configurable area on a single chip. In the analogue field several researchers are currently involved. At ETL an application specific chip has been designed called an Analogue EHW filter chip [MYA$^+$98] used as the name suggests to evolve filters. At JPL an analogue neural chip [SFHSL98] was evolved to provide visuo-motor coordination for a mobile robot. The Novel Palmo Analogue Signal Processing IC [HPTB98] is a new re-configurable analogue chip which although not specifically designed for EHW is intended to be tested out for EHW. The approach uses modulated digital pulses instead of an analogue current or voltage. At JPL a field programmable transistor array (FPTA) [SKTSL99] has been developed for evolvable hardware offering reconfiguration at the transistor level. This chip provides a platform for evolution of both analogue, digital and mixed signal designs.

3 FPGA Technology

In digital EHW, FPGA's may be seen to be the target technology as they provide a re-configurable platform and chips are commercially available. The main elements of FPGA chips are configurable logic blocks (CLBs) connected together in a grid format and configurable routing resources. In addition, configurable input/output blocks (IOBs) are connected to the grid at the perimeter of the chip.

Complexity of the CLBs ranges from simple logic blocks with thirteen multiplexor elements (Xilinx 6200) to small processor elements (Matrix, Garp) [MD96, HW97].

Configuration can be categorised into non-partial or partial configuration. The former may be applied to any FPGA and enables the complete chip to be configured. The latter is offered on some devices and allows a portion of the chip to be reconfigured whilst the remaining parts of the chip retain their current configurations.

Using traditional design techniques, illegal configurations are avoided by constraining configurations. Our aim with evolution is to avoid constraints but at the same time avoid illegal configurations that may damage the underlying chip. Therefore, it is desirable to build robustness against this fault into the technology itself. To date two different methods to solve this problem may be seen. Either the logic blocks themselves have an architecture where it is not possible to configure them illegally (as seen in XILINX 6200 series) or the users themselves do not have direct access to the routing resources during the evolution process (such as in Xilinx virtex series).

FPGA technology has seen a rapid growth in both performance and popularity in the last 10 years. Originally it was seen as a prototype technology but is now being used as a flexible technology for many applications.

FPGA development may be said to follow a similar path to that of processor development in the 80s. That is the development of the CISC architecture where more and more complex operations were enabled by more and more complex instructions. To offer the possibility for complex designs, FPGA is both expanding in size i.e. increasing the number of combinational logic blocks per chip, increasing the complexity of these blocks and introducing vast and varied routing resources. If we consider the difference between Xilinx XC4000 series introduced in 1990 and Xilinx virtex series introduced in 1999 we see that whereas XC4000 has up to ca. 3000 CLBs Virtex has up to ca. 16000 CLBs. Each CLB of XC4000 can handle functions with up to 9 inputs and has two flip-flops whereas each Virtex CLB can handle functions with up to 19 inputs and has 4 flip flops and has considerably more routing resources than that of the XC4000.

4 Characteristics of Evolvable Hardware for FPGAs

It is important to consider the characteristics of evolution and what requirements they place on technology when considering a new technology. We have attempted to highlight some of the features of evolution which effect evolvable hardware, focusing on FPGA technology and considering possibilities for adaptive hardware.

4.1 Genotype

A genotype may be expressed in terms of RAM content or as configuration data. That is, the design that an individual represents is implemented in memory (RAM content) or logic (configuration data) during evaluation of the individual. This also means that the final design is based on memory or logic respectively.

The genotype required to represent a RAM-based design will, in most cases, be much smaller than that required to represent logic or, in other works, configuration data. If we look at, for example Thomson's work where a RAM-based genotype was used for a robot controller, a 32 bit genotype was required [Tho95]. However in his tone-discrimination circuit where logic is represented, an 1800 bits genotype is required [Tho96]. A smaller genotype means that each individual in the population is smaller and, as such, less storage is required to store the population.

4.2 Population and Genetic Operations

An on-chip implementation of the genetic algorithm involves storage of the population on-chip.

If, as in the PIG proposal [Mac99b] and in [PRH00] all individuals of the population are implemented in parallel on the chip then size is a limiting factor for the population. If the population is limited, this reduces the amount of genetic information available to help the evolution process towards an acceptable fitness level i.e. an optimal solution. If the individuals of the population are stored in their genotype form on-chip and just implemented when the given individual(s) are under evaluation then the chip area required for the population is reduced. Therefore, the population size in an on-chip solution places a requirement on the technology that there is sufficient resources for a sufficiently large population such that the application may reach an optimal solution. It should be noted that saving the population off-line, such as in Intrinsic evolution removes the population area requirement but restricts the possibilities within adaptive design and increases the configuration time for a given individual.

Genetic operations may be conducted in hardware either on the genotype form or the implementation of the individuals. Either way this is a design decision which is not so effected by the underlying technology.

4.3 Fitness

In the field of evolvable hardware we are still trying to find out which applications are best suited to evolutionary methods. Therefore, the main aim may be said to be to evolve a functionally correct design. However, fitness may be extended to handle features such as area and speed. These factors are both dependent on the evolved design and the underlying technology. Including these features in the fitness evaluation enables a greater exploration of the design space for the given application.

Fitness evaluation is often the bottleneck of the evolution process. This is particularly important when dealing with real-time solutions. Real time designs require that fitness evaluation be faster than the incoming data rate. An example is real-time solutions for signal processing of audio applications. The incoming signal must be sampled, evaluated and refined in accordance with the filter type and reference signal in real time. This places a requirement that fitness evaluation is faster than the data rate of the audio signals. A number of researchers have looked at the problem of filtering using evolutionary methods [FS98, ZPV99, Kea97].

4.4 Non-constrained Evolution

If no limits are set as to how evolution may use the underlying technology then new solutions may be found that are not achievable with traditional (constrained) design methods. Evolution may use characteristics and parameters of the underlying technology which are not available for the traditional design process.

If an evolved solution does not involve a clock (synchronous) or handshaking (asynchronous) to control a design or parts of a design then the flow of signals is dependent on the propagation time of the different signals and transistor switching times. If a transistor has not fully switched before the signal is read then a value in between valid digital values will be allowed to progress in the circuit. During evolution of Thompson's tone-discrimination [Tho96] circuit, the experiment pushed the circuit behaviour beyond that of a typical digital circuit. That is to say almost as far as analogue behaviour.

Taking advantage of the physical parameters may, however, give chip dependence problems. During fabrication, a chip is deemed satisfactory if it meets the fabricant's requirements. Over and above these requirements, it cannot be guaranteed that each chip has exactly the same physical parameters due to variations in the semiconductor material. The question being are we interested in non-constrained evolution with its chip dependence. For a one-off solution this may be said to be no problem. However, can the circuit be shown to be robust? How long will it continue working in the given manner? For a solution that is to be repeated, then this chip dependence provides an unreliable solution. What is important for evolvable hardware is that technology does not restrict this feature of evolution but allows the evolution designer to place constraints into the evolution process if chip independence is sought.

5 Technology Requirements

5.1 Flexibility and Speed

Both Intrinsic and Complete Hardware Evolution (CHE) use a hardware implementation of each individual in the evaluation phase of the evolution process. This places a requirement of flexibility and, therefore, re-configurability on the implementation technology. Another important property of the target hardware is the number of possible re-configurations. A near unlimited number of re-configurations is desirable since the number of re-configurations required for a given design depends on the number of individuals in the population and the number of generations required to evolve a solution. This also places a requirement on configuration speed to prevent the evaluation phase becoming a bottleneck in the evolution process.

5.2 Partial Reconfiguration

If the genetic algorithm is off-line i.e. in a host processor, then configuration is driven off-line and as such incurs additional off-board delays To reduce this cost, partial reconfiguration enables the current individual to be compared with the previous individual and those configuration bits that differ can be used to configure the required changes, thus achieving faster configuration. Another solution is off course to move the host onto the same board and thus incur only off-chip delays rather than off-board delays.

If the genetic algorithm is implemented on-chip and the technology is non-partially re-configurable then the evolving design must be implemented as memory. That is that an area of the chip is set-off as memory which may be reconfigured i.e. written to, during the evolution process. However, if partially re-configurable technology is used then the evolving design may be implemented in either memory or logic since it is possible to reconfigure logic during the evaluation process.

5.3 Self Reconfiguration

Since the evaluation of how well the proposed design solves the specified task is done by fitness evaluation, exploration of a new design proposal could be started up when the current design no longer solves its task satisfactory. If the evolution process is on the same chip or on a local microprocessor, adaption does not need to be decided at design time but the required change in the circuit is free to evolve dynamically with changing external factors. For an on-chip solution, the chip needs to be self-reconfigurable — the ability of a part of the chip to reconfigure another part of the same chip. The logic detecting the external change needs to be able to restart the evolution process, generate and evaluate new individuals until a new solution which meets the changed external factors is found.

6 Proposed FPGA for Evolution

If we consider again the processor industry and consider the move from CISC to RISC then we see a move towards small instructions where a large number of instructions may be used to complete a given operation. That is simplicity of the unit i.e. the instruction, but an increase in the number of units.

What would happen if we reduced the complexity of the units of the FPGA chip? This would involve perhaps reducing the complexity of each CLB, thus allowing a greater number of CLBs on a single chip, requiring several CLBs to complete an operation that may be today handled by one CLB. To further reduce complexity the enormous routing resources should also be reduced. These routing resources, although providing fast routing across the complete chip, require an enormous amount of configuration data to configure all the routing alternatives. As stated earlier, the size of the genotype is proportional to the amount of configuration data required and a reduction in the size of the genotype will speed up the evolution process. This leads us to our proposed structure for an evolution suitable FPGA chip.

6.1 Architecture

We propose to retain the grid structure of the FPGA with blocks — *sblocks*, laid out as a symmetric grid but without the external connections between neighbouring sblocks — as illustrated in figure 1. That is, each sblock neighbours onto sblocks on its 4 sides. Each sblock consists of both a simple logic/memory component and routing resources. The sblock may either be configured as a logic or memory element with direct connections to its 4 neighbours or it may be configured as a routing element to connect one or more neighbours

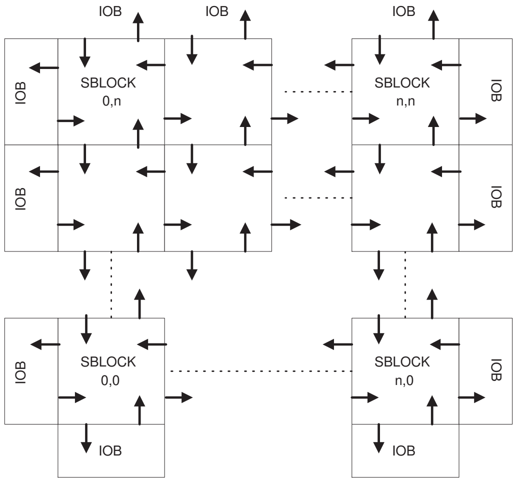

Figure 1: Grid Structure of Sblocks and IOBs

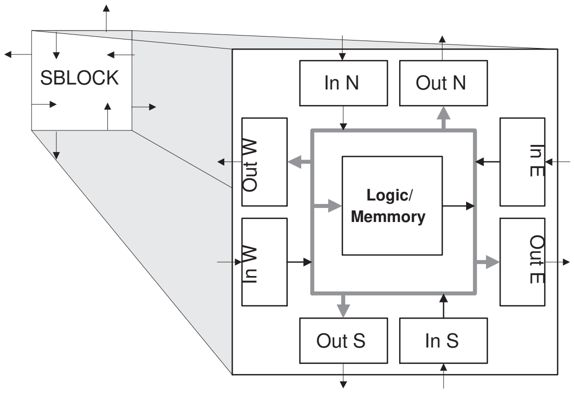

Figure 2: Sblock — Routing and Logic/Memory Block

to non-local nodes. By connecting together several nodes as routing elements, longer connections may be realised.

The internal connections include 4 pairs of unidirectional wires attached through routing logic to a routing channel. Each pair includes one input and one output connection. The routing logic chooses the appropriate wire of the routing channel and forwards this to the neighbouring sblock. Incoming data is forwarded to the appropriate channel wire for either forwarding to logic/memory or to the given output channel. The sblock is illustrated in figure 2.

Each sblock is responsible for processing input signals and routing outgoing signals to its neighbouring sblocks. Since each sblock may communicate on each of its edges to neighbouring sblocks then this provides a symmetrical and scalable architecture.

In our case, the internal routing of each sblock will only allow inputs to be routed to outputs and the interface between sblocks will only allow outputs to be routed to inputs. In this way, it will not be possible for evolution to generate illegal configurations for the sblocks where we define illegal configurations as configurations that may cause damage to the chip.

IOBs (input/output blocks) are placed at the perimeter of the chip. It is desirable that each column or row has its own IOB at the perimeter of the chip. This places a requirement on the minimum size of the sblocks and the IOBs. Reducing the size of the sblocks will increase the number of rows and column on a given sized chip. This will thus increase the number of IOBs required. Each IOB will thus be limited to the column and row width of the grid. However, IOBs cannot just be reduced to a size to meet the sblock requirements. IOBs cannot be smaller than that they can hold the logic required to drive the required signals to its neighbouring sblock. An IOB has also an associated pin. Placing the pins too close to each other may give practical problems when soldering the pins to a circuit board. Therefore, to retain the design structure that we propose, the minimum size of the sblock is restricted by the minimum size of the IOBs.

In earlier versions of FPGA devices, the on-board oscillator has often been found unreliable and an external clock has been used in designs. When we are aiming to achieve on-chip adaptivity, an on-chip oscillator is desirable. However, the new virtex series, if not having solved all the problems, has come close to a good oscillator solution and therefore we propose an on-board oscillator for our architecture.

Synchronous control of a design may be seen as a design constraint. Asynchronous design gives more freedom but less robustness but evolution may use this non-constraint control to achieve designs not achievable using traditional design techniques. It is therefore important that technology provides the opportunity to allow the design or parts of the design to be controlled asynchronous or synchronous by allowing evolution access to a clock if required. In addition, various clock speeds within the same design are often an advantage. To provide these features we propose global clock lines and an oscillator to drive the clock signal. Logic on the oscillator will allow evolution to select clock speeds for a set of three global clock lines. Although we have reduced routing to that inbuilt in the sblocks and IOBs, it was chosen to retain global clocking to avoid clock skews.

6.2 Configuration

Two global configuration buses, an address and a data bus traverse the grid. Each bus is proposed to be 16 bits wide. The address bus can thus address approximately 64000 CLBs. Each CLB thus having its own private address, an 8-bit x address and an 8-bit y address. The buses traverse each row in the grid thus allowing both serial or column parallel configuration of the complete grid.

Addressing each CLB individually opens up the possibility for partial reconfiguration. In Xilinx 6200 partial configuration is achieved through accessing either a particular cell or a range of cells. In the new virtex series, a portion of each column (a single bit per cell) may be addressed at a time — a frame. The number of frames needed to address a complete column is 48 frames as 48 bits are needed to configure a single CLB. This restricts partial configuration to one or more columns.

For effective designs, one can assume that a given function, or design component will be placed in a set of sblocks with the maximum number of local neighbours i.e. a square area of the chip rather than columns of the chip. As such, it is more desirable to configure an area of the chip rather than columns.

One possible approach to configuration by area is to split the chip into a number of areas of a fixed size. However, if a given design is slightly bigger than this fixed size, the design may overlap several areas thus requiring each of the overlapped areas to be configured in full.

Instead of this fixed size it is desirable to be able to configure any given area of the chip. That is that the range and position of the area be given in the configuration data i.e. the individuals of the evolution process.

For adaptive hardware it is also desirable that any sblock can drive a reconfiguration of another part of the chip. Configuration logic should, therefore, be on-chip where it is accessible to any node of the grid. As such the FPGA will be a self-re-configurable device.

6.3 Configuration Data

In the current FPGAs, a vast amount of the configuration data is attributed to routing. All routing must be configured and with the vast amount of routing available this entails a lot of routing configuration data even if a lot of the routing resources are unused. In the structure given, routing is not an independent resource from that of logic/memory. Each sblock's configuration data includes its necessary routing and logic data. Removing the large routing overhead in traditional chips will drastically reduce the amount of configuration data required to configure unused resources. If the genotype for evolution is based on configuration data then this will reduce the size of individuals required and the space needed to store the population. This will also speed up evolution, since the process will handle smaller individuals and configuration time will be less due to less configuration data.

Being able to partially reconfigure the chip, especially for a given size means that individuals may be off different sizes either within the same population or between generations. The latter is especially relevant if we wish to implement the given design in the smallest possible area. As fitness can include an area parameter for evaluating the quality of a given individual then individuals meeting the tighter area constraints may be prioritised i.e. given higher fitness. The range requirement for the given individual needs to be built into the individual and as the range reduces then the size of the individual i.e. configuration data part of the individual should also be reduced. This means that as we near a perfect solution meeting both functionality and size requirements the size of the individuals in the population will be smaller and smaller and processing time will reduce for each generation.

6.4 Feedback of Information

One important area within evolvable hardware where there has been little research is on analysis of the evolution process. To aid development of analysis tools for evolution it is important that as much information about the intermediate configurations (individuals) and their evaluations be available to analysis tools. In this proposed structure any CLB or IOB or range of CLBs/IOBs may be accessed through the configuration buses to read back the data lying in these blocks.

7 Summary

In this paper we have considered the characteristics of evolvable hardware and the requirements these characteristics place on the underlying technology. Further we have proposed an FPGA chip structure which we believe to be more suitable to evolution than that offered in today's FPGA devices.

Thomson [Tho98] suggested three technology characteristics that are important for Evolvable Hardware. These include the number of reconfigurations, reconfiguration time and smallest configurable unit. Our proposed chip has unlimited number of reconfigurations — as is the case for standard FPGAs, but unlike standard FPGAs we propose a structure enabling reduced configuration data for complete configuration thus reducing configuration time. In addition, the structure allows partial configuration where our smallest configuration unit is our reduced size CLB. One important application area of evolvable hardware is adaptive hardware. To meet the needs of adaptive hardware our structure also offers self-reconfiguration.

One key feature of the proposed structure is a reduction in routing resources and a more flexible division of logic and routing on the chip. This along with a smallest configuration unit of a CLB reduces the size of the genotype required for a given circuit with respect to that required for today's technology.

This work is at an early stage and experiments will be conducted to test out this structure in practice in the near future. One of the key features missing from the structural description is the structure of the logic/memory of the CLBs. We see that where the XL 6200 series used multiplexers other series use LUTs. Multiplexers may be seen to be a reasonable base unit for evolution [Tho96]. These may also be used by the evolution process to simulate delays required in the circuit. LUTs, on the other hand, provide a very flexible and fast logic block. As such, both alternatives will be tested out before a CLB structure is proposed.

References

[ÅN95] R. Åserud and I.R. Nielsen. Trends in Current Analogue Design. *Analogue Integrated Circuits and Signal Processing*, 7(1), 1995.

[dG93] H. de Garis. Artifical Life: Growing an Artificial Brain with a Million Neural Net Modules Inside a Trillion Cell Cellular Automata Machine. In *4th International Symposium on Micro Machine & Human Science*, 1993.

[dG96] H. de Garis. CAM -BRAIN The Evolutionary Engineering of a Billion Neuron Artificial Brain by 2001 Which Grows/Evolves at Electronic Speeds Inside a Cellular Automata machine (CAM). In *Towards Evolvable Hardware : the Evolutionary Engineering Approach*, volume 1062 of *Lecture Notes in Computer Science*, pp 76–98. Springer, 1996.

[FMT98] T. C. Fogarty, J. F. Miller, and P. Thomson. Evolving Digital Logic Circuits in Xilinx 6000 Family FPGAs. In *Soft Computing in Engineering Design and Manufacturing*, pp 299–305. Springer, 1998.

[FS98] S. J. Flockton and K. Sheehan. Intrinsic Circuit Evolution Using Programmable Analogue Arrays. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 144–153. Springer, 1998.

[HIM94] T. Higuchi, H. Iba, and B. Manderick. *Massively Parallel Artificial Intelligence*, chapter 12. Evolvable Hardware, pp 398–421. MIT Press, 1994.

[Hol75] J.H. Holland. *Adaption in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[HPTB98] A. Hamilton, K. Papathanasiou, M. R. Tamplin, and T. Brandtner. Palmo: Field Programmable Analogue and Mixed-Signal VLSI for Evolvable Hardware. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 335–344. Springer, 1998.

[HW97] J. R. Hauser and J. Wawrzynek. Garp: A MIPS Processor with a Reconfigurable Coprocessor. In *IEEE Symposium on FPGAs for Custom Computing Machines (FCCM97)*, pp 24–33, 1997.

[Kea97] J.R. Koza et al. Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. *IEEE Transactions on Evolutionary Computation*, 1(2) pp 109–128, July 1997.

[Lay98] P. Layzell. A New Research Tool for Intrinsic Hardware Evolution. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 47–56. Springer, 1998.

[Mac99a] N. Macias. The PIG Paradigm: The Design and Use of a Massively Parallel Fine Grained Self-Reconfigurable Infinitely Scalable Architecture. In *The 1st NASA/DoD Workshop on Evolvable Hardware*, pp 175–180, 1999.

[Mac99b] N. Macias. Ring Around the PIG: A Parallel GA with Only Local Interactions Coupled with a Self-Reconfigurable Hardware Platform to Implement an O(1) Evolutionary Cycle for Evolvable Hardware. In *Proc. of the 1999 Congress on Evolutionary Computation (CEC99)*, pp 1067–1078. IEEE, 1999.

[MD96] E. Mirsky and A. DeHon. MATRIX: A Reconfigurable Computing Architecture with Configurable Instruction Distribution and Deployable Resources. In *IEEE Symposium on FPGAs for Custom Computing Machines (FCCM96)*, pp 157–166, 1996.

[MYA+98] M. Murakawa, S. Yoshizawa, T. Adachi, K. Suzuki, K. Takasuka, M. Iwata, and T. Higuchi. Analogue EHW Chip for Intermediate Frequency Filters. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 134–143. Springer, 1998.

[MYK+99] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N Kajihara, M. Iwata, and T. Higuchi. The GRD Chip: Genetic Reconfiguration of DSPs for Neural Network Processing. *IEEE Transactions on Computers*, 48(6) pp 628–639, June 1999.

[Ozs98] I Ozsvald. *Short-Circuting the Design Process: Evolutionary Algorithms for Circuit Design using Reconfigurable Analogue Hardware, Masters Thesis*. University of Sussex, UK, 1998.

[PRH00] S. Perkins, P. Reid, and N. Harvey. Everything on the Chip: A Hardware-Based Self-Contained Spatially-Structured Genetic Algorithm for Signal Processing. In *3rd International Conference on Evolvable Systems (ICES00)*, Lecture Notes in Computer Science, pp 165–174. Springer, 2000.

[Sem97] Semiconductor Industry Association. *The National Technology Roadmap for Semiconductors*, 1997.

[SFHSL98] A. Stoica, A. Fukunaga, K. Hayworth, and C. Salazar-Lazaro. Evolvable Hardware for Space Applications. In *2nd International Conference on Evolvable Systems (ICES98)*, Lecture Notes in Computer Science, pp 166–173. Springer, 1998.

[SGM+97] M. Sipper, M. Goeke, D. Mange, A. Stauffer, E. Sanchez, and M. Tomassini. The Firefly Machine: Online Evolware. In *Proc. of 1997 International Conference on Evolutionary Computation (ICEC97)*, pp 181–186. IEEE, 1997.

[SKTSL99] A. Stoica, D. Keymeulen, R. Tawel, and C.and Wei-Te Li Salazar-Lazaro. Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits. In *The 1st NASA/DoD Workshop on Evolvable Hardware*, pp 76–84, 1999.

[SZK00] A. Stoica, R. Zebulum, and D. Keymeulen. Mixtrinsic Evolution. In *3rd International Conference on Evolvable Systems (ICES00)*, Lecture Notes in Computer Science, pp 208–217. Springer, 2000.

[TH99] G. Tufte and P. C. Haddow. Prototyping a GA Pipeline for Complete Hardware Evolution. In *The 1st NASA/DoD Workshop on Evolvable Hardware*, pp 18–25, 1999.

[THH96] A. Thompson, I. Harvey, and P Husbands. Unconstrained Evolution and Hard Consequences. In *Towards Evolvable Hardware : the Evolutionary Engineering Approach*, volume 1062 of *Lecture Notes in Computer Science*, pp 136–165. Springer, 1996.

[Tho95] A. Thompson. Evolving Electronic Robot Controllers that Exploit Hardware Resources. In *The 3rd European Conference on Artificial life (ECAL95)*, 1995.

[Tho96] A. Thompson. An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In *1st International Conference on Evolvable Systems (ICES96)*, Lecture Notes in Computer Science, pp 390–405. Springer, 1996.

[Tho98] A. Thompson. *Hardware Evolution Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution*. Springer, 1998.

[TLZ99] A. Thompson, P. Layzell, and R. S. Zebulum. Explorations in Design Space: Unconventional electronics design through artificial evolution. *IEEE Transactions on Evolutionary Computation*, 3(3) pp 167–196, September 1999.

[XDA97] *Xilinx Databook: "XC4000E and XC 4000X Series Field Programmable Gate Arrays Product Specification*, 1.4 edition, 1997.

[XRE97] *Xilinx XC6200 Field Programmable Gate Arrays Product Specification*, 1.10 edition, 1997.

[XVE98] *Xilinx Virtex 2.5V Field Programmable Gate Arrays Product Specification*, 1.1-advance edition, 1998.

[ZPV96] R.S. Zebulum, M.A. Pacheco, and M. Vellasco. Evolvable System in Hardware Design: Taxonomy, Survey and Applications. In *1st International Conference on Evolvable Systems (ICES96)*, Lecture Notes in Computer Science, pp 344–358. Springer, 1996.

[ZPV99] R. S. Zebulum, M. A. Pacheco, and M. Vellasco. Artificial Evolution of Active Filters: A Case Study. In *The 1st NASA/DoD Workshop on Evolvable Hardware*, pp 66–75, 1999.