Statement of Originality and Previously Published Work

Much of the work presented in this thesis has been previously published as listed below. Although some of these papers have co-authors, the work appearing in this thesis is entirely my own, with the exception of parts of chapter 3, which presents work jointly carried out by myself and Adrian Thompson. The respective contributions to this work will be explicitly stated at the beginning of the chapter.

List of Previous Publications

Kuntz, P., Layzell, P., & Snyers, D. (1997). A Colony of Ant-like Agents for Partitioning in VLSI Technology. In Husbands, P., & Harvey, I. (Eds.), *Proc. 4th European Conf. on Artificial Life (ECAL'97)*, pp. 417–424. MIT Press.

Layzell, P. (1998a). The 'Evolvable Motherboard': A Test Platform for Research of Intrinsic Hardware Evolution. Csrp 479, School of Cognitive and Computing Sciences, University of Sussex.

Layzell, P. (1998b). A New Research Tool for Intrinsic Hardware Evolution. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98)*, Vol. 1478 of *LNCS*, pp. 47–56. Springer-Verlag.

Layzell, P. (1999a). Inherent Qualities of Circuits Designed by Artificial Evolution: A preliminary study of populational fault tolerance. In Stoica, A., Keymeulen, D., & Lohn, J. (Eds.), *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*, pp. 85–86. IEEE Computer Society, Los Alamitos, California.

Layzell, P. (1999b). Reducing Hardware Evolution's Dependency on FPGAs. In *Proc. 7th Int. Conf. on microelectronics for neural, fuzzy and bio-inspired systems*. IEEE Comp. Soc. Press.

Layzell, P., & Thompson, A. (2000). Understanding Inherent Qualities of Evolved Circuits: Evolutionary History as a Predictor of Fault Tolerance. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd Intenational Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 133–145. Springer.

Thompson, A., & Layzell, P. (1999). Analysis of Unconventional Evolved Electronics. *Communications of the ACM*, *42*(4), 71–79.

Thompson, A., Layzell, P., & Zebulum, R. (1999). Explorations in Design space: Unconventional Electronics Design through Artificial Evolution. *IEEE transactions on Evolutionary Computation*, *3*(3), 167–196.

Hardware Evolution: On the Nature of Artificially Evolved Electronic Circuits

Paul Layzell

Submitted for the degree of D. Phil.

University of Sussex

May, 2001

Declaration

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree.

Signature:

Acknowledgements

Special thanks to the following people:

Phil Husbands, Adrian Thompson, Ricardo Zebulum, Inman Harvey, Julian Miller, Jon Bird, Chris Winter, Adrian Stoica, Nick Jakobi, Giles Mayley, Matthew Quinn, Suzy Gordon, Kathleen Layzell, Mark Shackleton and the Future Technologies team at British Telecom PLC.

I am grateful to British Telecom for funding this work, to Hewlett-Packard for the loan of test equipment, and especially to the Centre for Computational Neuroscience and Robotics (CCNR) and the School of Cognitive and Computing Sciences (COGS) for providing the stimulating and welcoming environment without which this work could not have progressed so far.

This thesis is dedicated to the memory of Derrick Layzell.

Hardware Evolution: On the Nature of Artificially Evolved Electronic Circuits

Paul Layzell

Summary

Artificial evolution is capable of deriving electronic circuits of very different nature to those designed using established top-down methodologies. Such circuits may have no clear functional decomposition, and can rely heavily on subtle 'parasitic' properties not normally exploited in conventional design. They can exhibit more efficient component usage, but less tolerance to fabrication and environmental variations. If an evolved circuit's operation is understood, steps can be taken to increase its tolerance to these factors, thereby enhancing its utility. However, analysis is problematic if the circuit possesses complex dynamics dependent on unknown parasitic properties. The topology and operation of evolved circuits are derived incrementally from the evolutionary process. Analysis is hence facilitated if established reverse-engineering techniques are also applied to ancestor circuits normally considered evolutionary by-products. But this is difficult and laborious for circuits evolved directly on commercial reconfigurable FPGA chips, which have little or no accessibility to internal nodes for probing.

As well as impeding analysis, FPGAs offer little choice of circuit primitives, restricted interconnection architecture, and most are susceptible to self-destruction. A new, less restrictive configurable research platform is presented, which possesses a comprehensive interconnection architecture and circuit primitives implemented as plug-in daughterboards, allowing a huge variety to be hosted with direct access to their pins for probing. The platform cannot be destroyed by illegal configurations if certain basic rules are followed.

A series of experiments demonstrates the platform's capacity to aid research of fundamental issues dominating hardware evolution, yielding useful insights on analysis, genotype encoding, choice of basic elements, portability, evolved topologies, exploitation of configuration circuitry, and extrinsic versus intrinsic evolution. The platform is then used to show that populations of evolved circuits contain individuals inherently tolerant to major faults. Through applying automatic analysis of ancestor circuits, the underlying mechanism responsible is revealed.

Submitted for the degree of D. Phil.

University of Sussex

May, 2001

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Topic

The study of hardware evolution — the application of evolutionary search algorithms to electronic circuit design and synthesis — has progressed in the last decade from a handful of devotees to an important research domain involving major institutions worldwide. The domain now covers virtually every aspect of electronic design including digital and analogue circuit synthesis, VLSI[1] placement and routing, adaptive systems and fault-tolerance (See Sipper, Mange, and Pérez-Uribe (1998), Stoica, Keymeulen, and Lohn (1999), Prieto (1999), Miller, Thompson, Thomson, and Fogarty (2000) for examples).

However, despite the increased attention and resources, it is clear that there is still a long way to go. Only a few engineering applications have been produced (T.Higuchi, Iwata, et al., 1999), and the promise of large-scale evolvable machines suggested by early experiments (for example Hemmi, Mizoguchi, and Shimohara (1996), de Garis (1995), McCaskill, Tangen, and Ackermann (1997)) remains largely unfulfilled. We would like to evolve circuits that are 'bigger' and 'better' than those produced so far, and to do so faster than we presently can, but achieving such a broad capability involves a large number of major issues. Many of these issues — such as scalability, robustness, portability, and difficulty of analysis — are well-known. In spite of endeavours to address them, they stand unresolved for all but a few specific cases.

This thesis is devoted to *understanding* the operation and derivation of evolved circuits, and the qualities possessed by them. I will approach this goal through exploring the principal issues that currently impede the progression of hardware evolution as an engineering design methodology, and will argue that the techniques and tools in our existing repertoire are only sufficient to tackle a few distinct cases where such issues are manifest. By examining cases where current analytical approaches are in some way unsatisfactory, I will develop new techniques and tools that may be applied more generally. In particular, I shall present and demonstrate a new hardware platform for the research of hardware evolution, that does not suffer from the difficulties inherent in devices originally intended for conventional engineering. I will also show how the by-products of the

[1] Very Large Scale Integration (VLSI) refers to circuits with very high component count implemented on a single piece of silicon.

evolutionary approach — a vast quantity of circuit descriptions produced on the way to deriving one that satisfies the desired criteria — may be harnessed to aid further our understanding of circuits produced by artificial evolution.

I do not claim that current analytical techniques should be discarded, indeed the proposed framework embraces them. Only a few novel additions are necessary to provide a general approach for understanding why evolved circuits look and behave like they do, and what prevents them from being 'better'. My arguments will be backed up by extensive experiments and case studies to illustrate their practical worth. We shall see light shed on the operation of previously impenetrable evolved circuits, and experiment with hitherto unused components as basic evolutionary elements, observing the consequences more directly than is possible with present evolvable media. The way in which our current engineering knowledge governs — and sometimes compromises — the design and application of evolutionary electronic systems will be explored, and domains for which evolution is a practical alternative to conventional design will be suggested.

Regarding the major issues, I do not attempt to resolve them, but will show how the majority can be investigated without recourse to large-scale evolvable architectures, complex evolved circuits and evolutionary algorithms, or vast computing power. They will be explored here by applying basic techniques to small, simple circuits derived using only the fundamental principles of artificial evolution. In the absence of an excessive quantity of factors hindering the comprehension of a given issue under examination, its principal components are more likely to emerge. This approach will reveal several tantalising glimpses of the qualities possessed by evolved circuits and the evolutionary process. Finally, as a showpiece for my methodology, it will be applied to one such quality hinted at in preliminary experiments, and lead to the discovery, understanding and prediction of an entirely new form of fault tolerance, inherent in genetically converged populations of evolved circuits.

1.2 Hardware Evolution

Human designers have used biological systems as a source of inspiration for their inventions throughout our history. In the past few decades, attention has also been paid to the processes which produced those systems. Biological systems are diverse, complex and robust, and it is widely accepted that Darwinian evolution is the principal mechanism responsible for their emergence. The major components of evolution — variation, heredity and selection — can be implemented artificially by using an 'Evolutionary Algorithm' (EA) to solve complex problems and to provide helpful insights for refining theories of natural evolution (for example Lund, Webb, and Hallam (1997)). EAs can also be applied to the design of engineering systems, including electronic circuitry. There are a variety of terms in current use to describe circuit synthesis through artificial evolution. I will use the term 'Hardware Evolution' (HE), and will elaborate on what is encompassed by it as the appropriate context is established in chapter 2. First, I will introduce the reader to the type of evolutionary circuit design framework used throughout this thesis by a concrete example of how hardware is evolved.

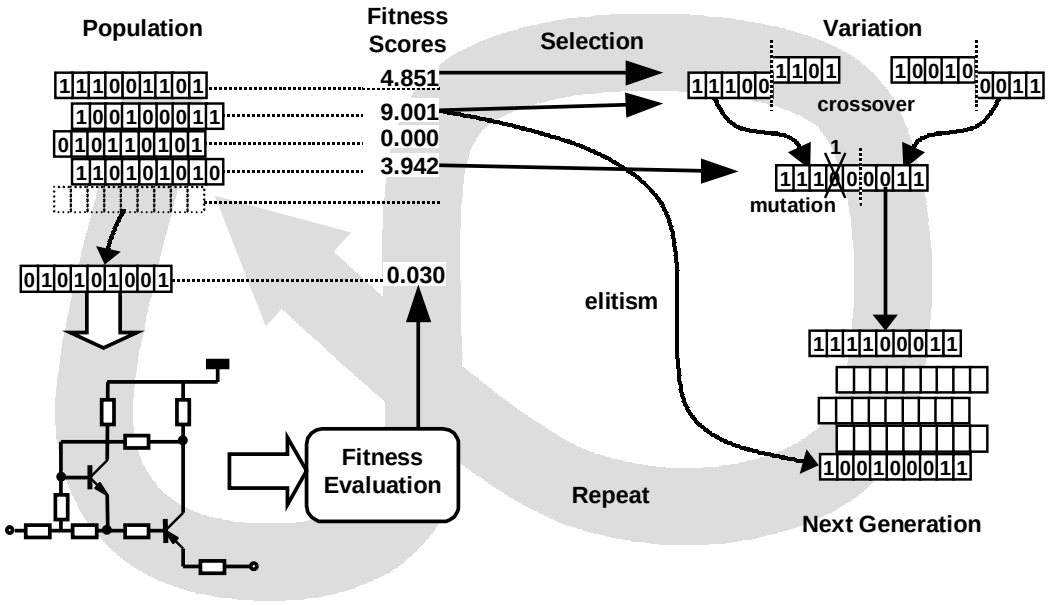


Figure 1.1: Applying an evolutionary algorithm to evolve hardware

1.2.1 Automatic Circuit Design through Artificial Evolution

Evolutionary Algorithms are search algorithms. Their task is to search the parameter set of some problem domain for a combination of distinct values that correspond to a solution within that domain. An EA requires the parameter set to be coded as a finite-length structure of characters from some finite alphabet; very often the structure is a binary string, or a string of finite-precision real numbers (Goldberg, 1989). For electronic circuit design, the problem domain is the derivation or construction of a circuit, and the solutions are circuits which behave according to some given specifications. The parameters define the quantities and types of available components and how they interconnect, hence each set of distinct parameter values corresponds to an individual electronic circuit. It is customary in hardware evolution to use the biological term *genotype* to refer to a set of distinct parameter values, and the term *phenotype* to refer to the electronic circuit derived from the set. To evolve hardware, an EA searches a space containing all possible circuits allowed by the parameter set, for one that behaves as specified. Figure 1.1 depicts the evolutionary search method used throughout this thesis.

A population of initially random bit-string genotypes, or 'individuals' is created, and each is taken in turn to construct a circuit in configurable hardware or simulation. The circuit is then evaluated and assigned a 'fitness score' reflecting how closely it approximates the specified behaviour. Once the entire population has been evaluated for fitness, a new population which will become the next generation of candidate circuit designs is created as follows. First, the best-scoring individual is copied once into the next generation (this is called *elitism* (de Jong, 1975, p.102)). The remaining individuals of the next generation are created by choosing individuals to 'breed' offspring which will inherit variations of their parents' genetic material. The parents are selected from the evaluated population in a stochastic manner such that individuals with high fitness scores are more likely to be selected than those with low scores. A number of selection strategies exist. All of the

evolutionary runs documented in this thesis selected the parents with a probability determined by a linear function of their rank within the population, as defined by the ordering of the fitness scores (Baker, 1985). The lowest scoring individual has zero probability of selection, while the highest scoring individual has twice the probability of the median. Variation is introduced by applying genetic operators, the most common of which is *mutation*: each genotype bit of the parent is inverted with some small probability (defined as the 'mutation rate') which is applied independently for each bit position. The offspring is the genotype attained having applied mutation. Another common operator, *crossover*, mimics sexual reproduction by forming the offspring from two or more parents. With a certain 'crossover probability', two parent genotypes are divided at the same bit position or 'crossover point' chosen randomly. The genotype bits preceding the crossover point in one parent are recombined with those following it in the other parent to become a new genotype (figure 1.1). Mutation is applied at this stage to form the offspring. A number of other genetic operators exist as well as variations of the two described here. When the next generation contains as many individuals as the initial population, it is used to configure candidate circuits for evaluation. The cycle of evaluation, selection and breeding is repeated until either a candidate circuit yields a fitness score equating to the desired specifications, or the run is abandoned.

The algorithm described above belongs to a class of EAs known as 'Genetic Algorithms' (GA). EAs within this class vary according to the selection strategy and genetic operators they employ, as well as how the parameter set is structured. For example, some use genotypes whose size may vary during the course of a single run. Several other classes of EA are in common use. As well as GAs (Holland, 1975), the principal ones are Evolution Strategies (Schwefel & Rudolph, 1995), Evolutionary Programming (Fogel, Owens, & Walsh, 1966), and Genetic Programming (Koza, 1992). The experiments described in the following chapters applied an elitist GA with the most basic types of evolutionary operator (bit-mutation and single-point crossover) to fixed-length genotypes, so that any results would reflect the fundamental mechanisms of artificial evolution applied to circuit design. Hence much of this thesis applies to other classes of evolutionary algorithm, and even where the conclusions are only applicable to the GA used, they should be of general relevance due to the widespread use of this type of EA.

1.2.2 A Reconfigurable Medium for Hardware Evolution

Using information contained in the genotype, candidate circuits must be constructed or synthesised in some way so that their behaviour can be evaluated. This procedure is called 'genotype⇒phenotype mapping', and can be carried out using either software circuit simulators or physical hardware. It is impractical to build separate circuits for the latter case, to be discarded after evaluation. Instead, a single piece of hardware is used whose architecture can be rapidly and repeatedly changed, or reconfigured, to produce a variety of different behaviours for different configurations.

Figure 1.2 depicts a generic reconfigurable architecture suitable for hardware evolution. A two-dimensional array of reconfigurable cells is bordered by a number of connections, or I/O blocks, which can be configured either as input or output connections to the external environment. The cells themselves have individual input and output connections to their horizontal and vertical neighbours, as shown. Each cell consists internally of a basic functional element, and programmable switches that determine the routing between the functional element and the neigh-

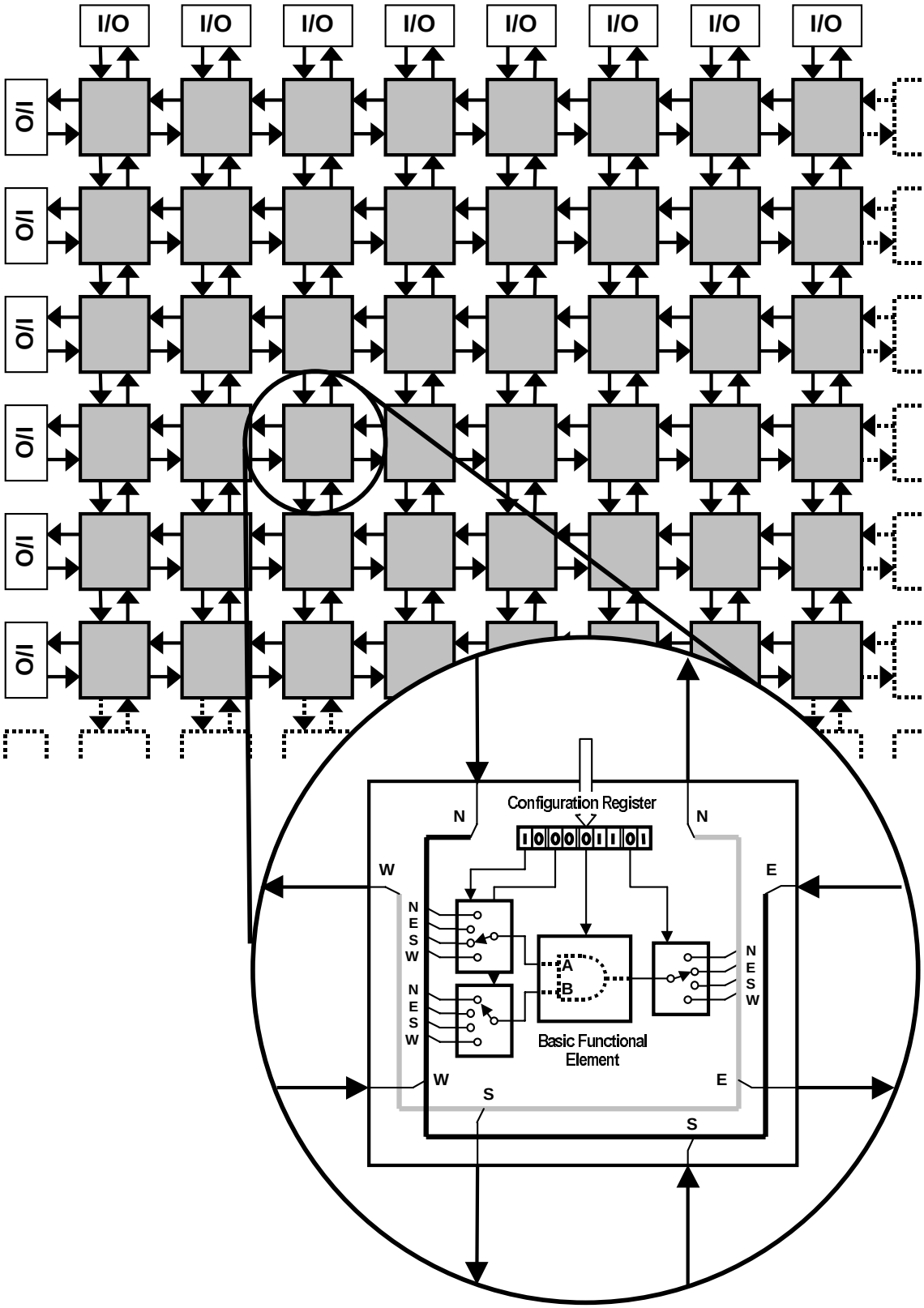


Figure 1.2: Schematic representation of a generic reconfigurable device

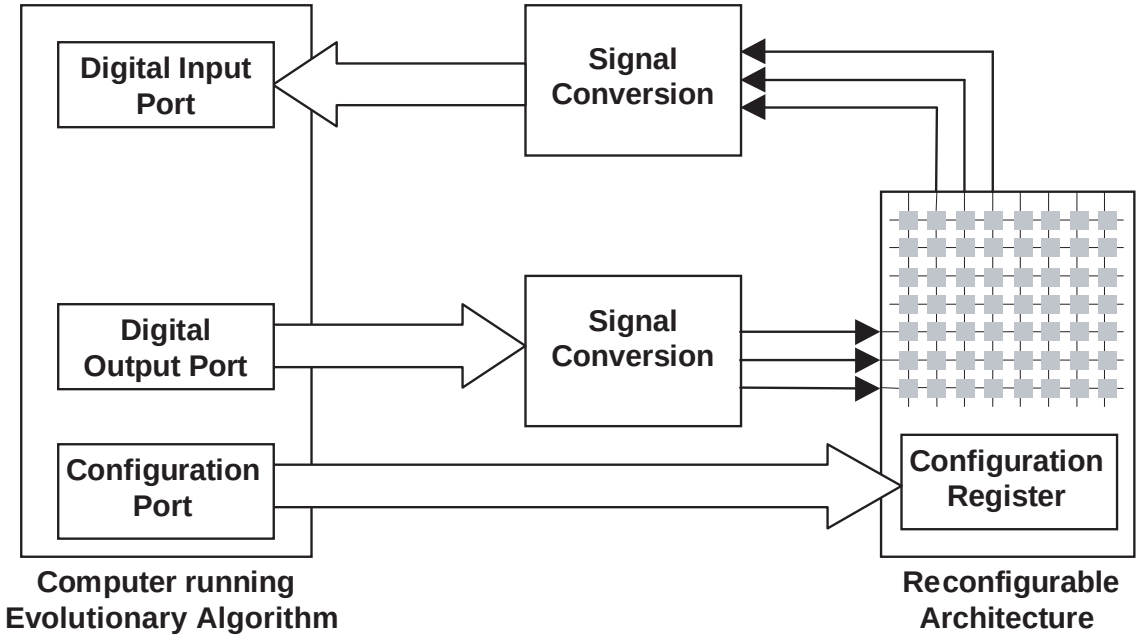


Figure 1.3: A framework for performing intrinsic hardware evolution

bouring cells. Both functional element and switches can be configured at any time according to the cell's 'configuration register', an on-board memory that can be manipulated by software running on a host computer. The state of bits in the configuration register determine what operation the functional element performs, and the neighbours to which its inputs and outputs are routed. In this example, the functional element performs a logical '**AND**', taking its inputs from its south and north neighbours, and outputting to its east neighbour. By changing the configuration bits from software, the settings of the switches and functional elements distributed throughout the architecture can be altered to produce a huge variety of different circuits. Given an array of say 10×10 cells, each with 9-bit configuration registers, and an additional bit to configure each of the 40 I/O blocks, then a total of $2^{10\times10\times9\times40}$, or 10^{10837} circuits can be configured. The majority of these circuits will exhibit trivial behaviour of no practical use for engineering applications, and even where a particular configuration performs a useful function, it is likely that many other configurations exist which will perform the same function. However, even if a tiny fraction of all possible configurations yields useful circuits whose behaviour differs significantly from each other — say 10^{-10000} percent — then this subset is still vast, containing over 10^{835} circuits. It is important to distinguish between configuring this type of architecture and programming a sequence of instructions to be executed by some fixed processor. While the structure of a circuit is determined by software, it is physically instantiated on the architecture and behaves in real-time according to the laws of physics (Thompson, 1998a). Hence the circuit's operation may be governed not only by the basic functional elements, their properties, configuration and interconnection, but also by any physical properties possessed by the configuration switches and routing tracks. Evolution makes no distinction between circuit elements we would categorise as primary and ancillary. However, I shall make such a distinction: The functional elements as described above are hereafter referred to as 'basic evolutionary elements' or 'basic elements' where their purpose is clear, and remaining parts of the medium as 'configuration circuitry'.

The architecture of figure 1.2 is hypothetical, however its structure is similar to certain commercially available Field Programmable Gate Array (FPGA) chips, which generally incorporate larger arrays of reconfigurable cells, more comprehensive routing, and additional resources. FPGAs are used extensively within the field of hardware evolution, and are predominantly digital devices. However, they are just one among several different types of architecture suitable for HE that will be described in later chapters. Figure 1.3 shows how such a device may be connected to a computer for configuration and evaluation. Having set up the configuration register according to the state of bits in the genotype, a set of test inputs is supplied. The outputs of the architecture are fed back to the computer, which compares them to the desired behaviour for assignment of fitness score. The computer's input and output ports may be connected directly to the architecture or if analogue behaviour is required, some form of signal conversion may be necessary; typically a/d or d/a converters[2].

The above use of reconfigurable devices as an integral part of the evolutionary framework is referred to as 'intrinsic' hardware evolution (de Garis, 1993). An alternative 'extrinsic' approach — in which the construction and evaluation phase is carried out using software circuit simulators — is also possible. In this case information in the genotype is used to write a circuit 'netlist' file which can be used by commercial simulators such as PSpice (MicroSim, 2000). A netlist is usually a text file, specifying each basic element used in the circuit, its physical characteristics where appropriate, and any nodes to which its pins are connected. In a typical extrinsic HE system the simulator combines each netlist with other files specifying input and environmental conditions, then calculates the operating voltages at all nodes in the circuit and dumps the results in an output file. Information in the output file is subsequently used by the EA to perform fitness evaluations.

1.3 Conventional Design versus Hardware Evolution

It will be clear to the reader schooled in electronic engineering that hardware evolution is an entirely different circuit design methodology. The notion of progressively tinkering with initially random circuit structures solely according to behavioural measurement, with no human intervention beyond design of the evolutionary framework and evaluation method, raises questions concerning the complexity and reliability of circuits that this method can be expected to produce. Conventional electronic engineering is a discipline based on the laws of physics, and with the benefit of a century of development can produce extraordinarily complex systems. It works because we understand the materials used, their physical characteristics and the interactions between them, and can apply our understanding at all stages in the design process. This enables us to rely on a given circuit's behaviour within some specified bounds. It can then be used as a 'building block' in larger circuits without the designer having to know every detail of its internal operation. Further complexity is made possible by the development of strategies by which an engineer can design electronic subsystems with little knowledge of the main system into which it will be incorporated, and have complete confidence that it will work. Many such strategies exist. Their objective is to provide a means by which complex tasks can be transformed or 'abstracted' into a form that can be understood and manipulated by human designers. Those whose effectiveness has been proven in common use are now considered a staple part of 'conventional design' — a term I shall use

[2] a/d = analogue to digital; d/a = digital to analogue.

throughout this thesis to refer to the systematic design of electronic circuitry using such strategies. Hardware evolution does not need the same kind of abstraction, because *understanding* a developing circuit is not a part of the basic evolutionary design process. As a consequence, evolved circuits may have entirely different properties and topologies compared with conventional ones, and analysing them to determine how they work can be difficult or impossible. As we shall see, they may be in some sense 'better' than conventional ones for certain tasks. Where this is so, the usefulness of hardware evolution as a design tool increases beyond merely automating the design process; it may be used to tackle applications for which conventional methods are inappropriate. Conversely, the rigorous strategies that ensure predictability and reliability in conventional circuits cannot necessarily be incorporated within an evolutionary framework without severely impeding the evolutionary search or restricting the attainable circuits to a tiny subset of all those possible. Hence, an evolved circuit may be 'better' in some sense than a conventional one, but it does not follow that it is practical.

1.4 The Thesis

With the above background, the principal achievements of my thesis can now be explicitly stated.

1. **A general evolvable platform for the research of hardware evolution is presented, with greater accessibility to the basic evolutionary elements than is possible with commercial FPGAs. This allows unusual circuits evolved on the platform to be analysed significantly faster and with less difficulty than those evolved on an FPGA. The platform's flexibility with regard to its interconnection architecture and the large range of basic elements it can host, allows the evolution and study of circuits beyond the scope of commercial FPGAs.**

2. **Combining evolution and analysis of *small* circuits is a viable means by which major issues predominating hardware evolution can be investigated. Its application here provides insights into fundamental factors governing:**

   - **the extent to which evolution may harness parasitic properties**
   - **evolved circuit topologies**
   - **portability of evolved circuits**
   - **intrinsic versus extrinsic evaluation**
   - **choice of interconnection architecture**
   - **the evolutionary search to locate the first stable prototype circuit.**
   - **inherent qualities of evolved circuits**

3. **Genetically converged populations of certain generic classes of evolved circuit can contain individuals inherently robust to major component failure, even if the failure is so severe as to render the best performing individual within the population useless. There is strong evidence that this phenomenon occurs because configurations of ancestor circuits which did not use the faulty component remain largely intact in later genotypes.**

The context for these claims is prepared in the following two chapters. Chapter 2 examines the state of the art at the time this work commenced, with emphasis on the trade-off between qualities peculiar to evolved circuits, and their practicality. The major issues influencing this trade-off are

then discussed and more recent strategies for tackling them compared. Chapter 3 discusses ways in which understanding an unusual evolved circuit's operation can improve its engineering usefulness. The aim of 'analysis' is defined clearly with regard to evolved circuits as traditional notions are not necessarily appropriate. A set of practical techniques for understanding and analysing an evolved circuit's operation are then presented, and applied to a particularly difficult example. Chapter 4 examines several disadvantages in using FPGAs for the research of hardware evolution, and the 'Evolvable Motherboard' (point 1, above) is introduced as a potential solution. It is then employed in chapter 5 for a series of open-ended experiments for evolving and analysing small circuits, investigating many of the important issues in HE research (point 2). Chapter 6 explores how certain nonbehavioural qualities possessed by a circuit are related to the design procedure, and introduces the concept of 'inherent qualities'. The analysis techniques of chapter 3 are then combined with the evolvable motherboard to discover and investigate a complex and potentially valuable quality inherent in evolved circuits: Populational Fault Tolerance (Point 3). All of the experiments detailed in the thesis could be extended to increase further our understanding of the issues they address. Ways in which this could be achieved are suggested in chapter 7. The conclusion brings together the principal arguments and evidence justifying points 1-3 above. While techniques for the evolution and analysis of large, complex circuits and systems are still in the early stages of development, the approach set out here is a practical means by which hardware evolution can progress to become a credible engineering tool.

Chapter 2

Context

This chapter describes the body of work that provided the inspiration for this thesis. Fundamental issues that remain unresolved are then considered in the light of more recent work. Finally, the evolutionary algorithm adopted for experimentation is justified, and the position of the thesis within the field of HE is clearly set out.

2.1 Background

2.1.1 The Field

Much of the early research specifically devoted to evolving electronic circuitry concentrated on extrinsic evolution of small functional building blocks, which could in principle be implemented on some programmable logic device (the implementation was rarely carried out in practice). In general, they were presented not as serious engineering applications, but as evidence to justify some evolutionary strategy or principle. Examples include: 6-input multiplexer and multiple XOR circuits (Kitano, 1996b); 4-bit comparator (Higuchi, Iwata, Kajitani, et al., 1996a); Sequential Adder (Hemmi et al., 1996); digital string generator (Zebulum, Pacheco, & Vellasco, 1996), and slow oscillator (Thompson, 1996c). Evolutionary techniques were also applied to analogue circuit design, focusing in the main on filters and amplifiers (Kruiskamp & Leenaerts, 1995; Horrocks & Khalifa, 1994; Koza, Andre, Bennett III, et al., 1996). Other work applying EAs to some aspect of electronic design include the evolutionary optimisation of various stages of VLSI synthesis, such as placement and routing or logic optimisation (Schnecke & Vornberger, 1995; Lienig & Brandt, 1994; Benten & Sait, 1994; Miller & Thomson, 1995); hardware implementation of cellular automata (for example M. Goeke et al. (1996)); and the application of EAs to high-level computer architecture design (Hirst, 1996; Teich, Blickle, & Thiele, 1996).

It is tempting to place some dividing line somewhere within this large domain space to define what does and does not constitute ‘hardware evolution’. Several authors have attempted such a classification. Yao and Higuchi (1996) exclude hardware implementations of EAs where the hardware architecture itself does not change, while Sanchez, Mange, Sipper, et al. (1996) compare biologically-inspired systems to three levels of organisation distinguished in living systems; the temporal evolution of the genome in individuals or species (phylogeny), the developmental process

of a single multicellular organism from successive divisions of the mother cell (ontogeny), and the learning processes during an individual's lifetime (epigenesis). Within this 'POE' framework, hardware evolution is placed firmly along the phylogeny axis, with self-reproducing automata and artificial neural nets considered to be more suitably placed along the other two respectively. My own concern is the *nature* of electronic circuits designed by artificial evolution; what they look like and what qualities they can be expected or encouraged to have, particularly where these qualities are in some way different to those possessed by conventionally designed circuits. This is the basis on which the literature reviewed in this chapter and throughout the thesis has been selected. 'Hardware evolution' will refer specifically to the application of artificial evolution to circuit synthesis, but work in peripheral fields that has implications regarding the nature of evolved circuits will also be introduced where appropriate.

2.1.2 Unconstrained Hardware Evolution and Continuous-time Dynamics

The research program adopted throughout this thesis is founded extensively on work carried out at the University of Sussex by Thompson, Harvey and Husbands, and indeed was funded specifically to extend it. This research attempts to map out the boundaries and benefits of an evolutionary approach to circuit synthesis that does not blindly accept the constraints necessary for conventional design (Thompson, 1998b).

The Sussex group highlights the widespread use of abstract models in electronic engineering, which simplifies design by allowing some aspects of reality to be ignored, but which effectively imposes constraints on what circuits can be produced in order to ensure that a given model is valid. Using digital design as an example in which transistors are considered not as analogue devices, but as ON/OFF switches, they specify the conditions required to allow circuits designed at this level of abstraction to work in reality (Thompson, Harvey, & Husbands, 1996). First, the transistors must always be kept either in the ON state or the OFF state except during short transient periods while switching between states. Second, the transients (a feature absent from the designer's model) must be prevented from affecting the system's overall behaviour. This is achieved by the use of a global clock in synchronous design, and more local co-ordination mechanisms in asynchronous methodologies. Clearly, circuits which adhere to the above conditions form but a small subset of all possible transistor circuits; the designer is constrained to work within this subset as a direct consequence of using abstraction. Were abstract models to be dropped from the design process, our notions of what an electronic circuit should be could alter:

> Intrinsic hardware evolution, by observing the consequences of variations made to the real hardware, avoids the need for design abstractions and the accompanying constraints. Our notion of the nature of electronic systems is heavily biased by our design methodologies and the constraints applied to facilitate their abstractions, so [hardware evolution] demands a radical rethink of what electronic circuits can be. (Thompson et al., 1996).

Thompson has investigated the potential benefits of relaxing the constraints normally applied in digital circuit design, through several important case studies. As a first step, clock synchronisation was placed under evolutionary control in a robot controller (Thompson, 1995a). The task was to make an autonomous two-wheeled robot move continuously in the centre of a square, walled

arena. The d.c. motors powering the wheels were not allowed to run in reverse, and the robot's only sensors were a pair of sonars. Thompson proposed a conventional controller model for tackling the task, which consisted of timers to measure the sonar fire—reflection interval, a hardware implementation of a finite-state machine (FSM), and binary-coded pulse-width modulators to control the wheels. This FSM-based implementation would require all signals to be synchronised to a global clock to give clean, deterministic state-transition behaviour as predicted by the model. However, he argued that relaxing these temporal constraints would endow the system with a range of dynamical behaviour potentially rich enough to satisfy the wall-avoidance task, with no need of pre- or post-processing stages: the sonar outputs could be fed directly to the controller, which in turn could directly drive the motors. The relaxation was achieved by placing under genetic control both clock frequency, and the choice of whether each signal is synchronised by the clock or whether it is asynchronous. Thompson called this architecture a 'Dynamic-State Machine' (DSM), and used it to evolve a successful wall-avoiding controller consisting of only 32 bits of RAM and a few flip-flops.

The success of this experiment was attributed by Thompson not to the DSM architecture, but to evolution's ability to exploit it. A designer would only be able to work within a small subset of possible DSM configurations — the ones that are easier to analyse. The resulting continuous-time dynamical system cannot be simulated in software because the effects of the asynchronous variables and their interaction with the clocked ones depend upon the characteristics of the hardware, such as propagation delays and meta-stability constants, which cannot be specified with precision in a software model. The robot controller experiment revealed evolution's potential for producing functional systems even when components seemingly essential for the task at hand (in this case timers and pulse-width modulators), were denied it. This potential was further manifested in a landmark experiment for which all temporal constraints were relaxed.

The task was to evolve a circuit — a configuration of a 10×10 corner of a Xilinx XC6216 FPGA that would discriminate between audio signals of 1kHz and 10kHz frequency, outputting +5V when one signal was present at the input, and 0V for the other (Thompson, 1996c). The circuit had no access to a clock or other off-chip resources such as RC time constants, by which the input could be timed or filtered. A continuous-time recurrent arrangement had to be found to discriminate between intervals five orders of magnitude longer than the input⇒output propagation time of each cell. Although the robot-controller experiment suggested a benefit in providing a clock of evolvable frequency — as an optional resource rather than an imposed constraint — it was excluded primarily to explore the possibility of evolving very unusual circuits. Also, from an engineering perspective, the components needed for an external time reference would be bulky compared to the 1% of the FPGA's silicon area used by the final evolved circuit.

Near-perfect tone discrimination was achieved. The final circuit used only 32 of the FPGA cells — far fewer than would have been necessary had conventional design been used — and displayed some remarkable properties. A preliminary analysis revealed that some of the cells affected the circuit's behaviour when their functional units were clamped to output stable logic levels, even though there was no connected path by which they could influence the output. They were influencing the circuit by some means other than the normal cell-to-cell wires. Electromagnetic coupling or interaction through the power-supply wiring were suggested as possible mechanisms (Thomp-

son, 1997a). The circuit's behaviour was also influenced by variations in ambient temperature. If this lay outside the 10^{o}C range to which the circuit was exposed during evolution, its behaviour was degraded. Indeed the threshold frequency — above which the output is high (+5V) and below which the output is low (0V) — altered even within this range. The circuit's behaviour was also degraded when instantiated onto a completely different 10×10 array of cells. These and other observations led Thompson to conclude that:

> Evolution has been free to explore the full repertoire of behaviours available from the silicon resources provided, even being able to exploit the subtle interactions between adjacent components that are not directly connected. The input/output behaviour of the circuit is a digital one, because that is what maximising the fitness function required, but the complex analogue waveforms seen at the output during the intermediate stages of evolution betray the rich continuous-time continuous value dynamics that are likely to be internally present. (Thompson, 1996b).

So unconstrained hardware evolution is a powerful means of harnessing the full potential of a group of electronic components, and can result in effective, parsimonious solutions to complex engineering problems. But is it a practical method? The operational temperature range of the tone discriminator is small compared to that expected of commercial engineering applications. Furthermore, the evolved schematic cannot be used as a 'blueprint' to manufacture tone discriminators, since the circuit is too dependent on subtle physical properties of the medium on which it was evolved. Finally, commercial circuits and systems are not released into the market if their operation is not understood because their failure modes will also be unknown, yet the precise operation of circuits like the evolved robot controller and the tone discriminator remains largely unknown.

2.1.3 Fault Tolerance

Conventional circuits are very fragile to component failure unless special measures are taken. The failure of a single transistor or connection very often results in breakdown of the entire system. The most common approach to producing fault tolerant systems is the use of *redundancy*; incorporating spare parts into a system for use when a fault is identified (Chean & Fortes, 1990), particularly in safety-critical systems such as aircraft. Where constraints of size, cost or weight make redundancy impractical, some form of *graceful degradation* may be applied. In this case there are no 'spares' in the system, instead functional parts are arranged such that faulty ones do not result in catastrophic failure. Innovative use of multi-processor arrays for graceful degradation has achieved some success in the spacecraft industry (Castro, 1995).

Several biologically inspired approaches to fault tolerance have emerged as a result of the high level of robustness observed in living systems. A group at EPFL (Ecole Polytechnique Fédérale de Lausanne and CSEM (Centre Suisse d'Electronique et de Microtechnique SA, Neuchâtel), have investigated a framework by which the self-repair and self-reproduction properties of cells in a multicellular organism can be implemented using FPGAs (Mange, Goeke, Madon, et al., 1996). Within the framework, a cell's behaviour is determined both by information in the genotype and by the state of neighbouring cells. Each cell has a built-in self test capability, and a row/column index as part of its state, which describes its relative position in the circuit. If a fault is detected in a cell its index becomes null, and neighbouring cells on detecting this new state alter their own indices

to be one positional unit from the row/column *preceding* that of the faulty cell. The cumulative effect is that the column (or row) containing the faulty cell becomes excluded from the circuit; the circuit dynamically redistributes itself over the remaining functional cells.

Mange et al's above 'Embryonics' framework requires only local interactions between cells to produce self-repair. The absence of a global control or coordinator is also a characteristic of *swarm intelligence* (see for example, Nonaneau, Dorigo, and Theraulaz (1999)): Entomologists studying social insects have proposed that many complex behaviours observed in ant or termite colonies arise from individuals following simple rules, and carrying out local interactions with other individuals and/or their environment (Franks, 1989; Forrest, 1990). Swarm intelligence models have been applied to various difficult or NP-complete problems, including telecommunications routing (Schoonderwoerd, Holland, Bruten, & Rothkrantz, 1997), placement in VLSI (Kuntz, Layzell, & Snyers, 1997), and object clustering (Beckers, Holland, & Deneubourg, 1994). The latter model used simple autonomous robots to form a single group from randomly scattered pucks within a square arena. The system is inherently fault-tolerant because the removal or failure of individual robots has little or no influence on those remaining.

The above examples rely on traditional techniques of redundancy and graceful degradation, albeit within a quasi-biological framework. An alternative possibility is to allow a faulty component to be exploited as a resource. If a defect persists while a system is evolving, then the behaviour of the faulty part becomes another component to be used: The evolutionary algorithm does not 'know' that the part was supposed to do something else. In an extension to the wall-avoiding robot experiment, Thompson induced individual single-stuck-at (SSA) faults, in the 32 DSM RAM bits, and the evolved controller was allowed to evolve some more (Thompson, 1995b). At first, the fitness of the population was dramatically lowered, but recovered to previous levels within 10 generations. In this particular experiment, the faulty part was tolerated rather than used, but in general this need not be the case. While this mode of fault-tolerance is a consequence of the evolutionary procedure rather than an engineering technique, it could potentially be harnessed in a manufacturing process that included some small phase of further evolution, either to adapt an evolved circuit to slightly different hardware, or to allow the circuit to operate on hardware with minor defects. However, the phenomenon must be better understood for such a process to be practical.

Many evolved hardware strategies use a one-to-one mapping between genotype bits and the hardware configuration registers specifying circuit connection or logic function. The type of fault produced by bit mutation is therefore similar to SSA faults in conventional circuits. Mutation is a common operator in evolutionary algorithms, and a characteristic of evolved systems is that fitter individuals in later generations are relatively insensitive to bit mutation. This has already been observed in molecular evolution (Eigen, 1987; Huynen & Hogeweg, 1994) for a problem whose fitness landscape was such that the optimum peak suffered severe fitness degradation from single mutations, whereas mutations on another slightly sub-optimal peak would result in only small degradation. A similar experiment was performed using genetic algorithms (Thompson et al., 1996). In almost all cases the population moved away from the isolated global optimum in favour of a slightly inferior fitness peak. The fitness landscape used in this experiment was highly contrived, having only two optima. To see if the same result would be manifest in more typical situations, a large series of GA searches was carried out (Thompson, 1996b) using NK fitness

landscapes (Kauffman, 1993) with N=20 and various values of K and GA parameters. Taking the fittest individual from each run, the mean fitness decrease caused by a single mutation averaged over all N possible single mutations was measured. The single-mutation neighbours of the individual were then assigned random fitness values typical for an optimum with that individual's fitness and for the values of N and K used. The mean fitness decrease caused by a single mutation was then re-measured and compared to the original. The experiment showed that for NK landscapes of intermediate ruggedness, optima found by basic GAs are less likely to be degraded by single mutations than would be statistically expected.

Thompson concluded that although limited in magnitude and the range of faults to which it can apply, for certain types of EA:

> [graceful degradation] arises 'for free' out of the nature of the evolutionary process, without any special measures having to be taken (Thompson, 1996b).

The magnitude of the effect is not known outside of the NK model, but if it is significant in implementations of GAs for circuit synthesis, then evolved circuits may possess a degree of inherent fault tolerance which would be of benefit to engineering. The concept of inherent qualities, particularly fault tolerance will be explored in detail in chapter 6. It may be possible to provide evolutionary pressure to extend the mutation-insensitivity of EAs to other faults not so directly analogous, by incorporating fault tolerance into the fitness function, and deliberately introducing serious faults when the hardware is configured (Thompson, 1996a). For larger systems, determining serious faults requires testing for every possible fault, which is prohibitive. One suggestion is to co-evolve a population of test cases (Hillis, 1992), or faults that concentrate on the weak spots of the evolving circuits. Another potential method is the use of repeated re-evaluations of the more successful individuals to build up gradually an accurate picture of their performance in the presence of a set of faults (Paredis, 1994).

2.1.4 Adaptive systems

Closely related to fault tolerance, adaptive systems can change their configuration or behaviour during the *operational phase*, to cope with unexpected challenges from their environment or faults occurring within them. An adaptive robot for example, might be able to change its gait if a leg joint seized, right itself if placed upside-down, or find some way of walking around or over a previously unencountered obstacle. This is a different scenario to that in which an FPGA's configuration is 'adapted' to meet the behavioural requirements of an evolutionary algorithm: once the requirements are met, the configuration remains static when the FPGA is used in an application. A large group at ETL (Electrotechnical Laboratory, Japan) have concentrated primarily on the use of artificial evolution to produce adaptive electronic systems, for which they use the term 'Evolvable Hardware' (EHW)[1]:

> EHW can change its own hardware structure to adapt to the environment whenever environmental changes (including hardware malfunction) occur (Higuchi et al., 1996a).

[1] In recent years EHW has become a much broader term, encompassing virtually every aspect of evolutionary circuit synthesis including non-adaptive systems. It is because of this ambiguity that I use the general term 'Hardware Evolution' in preference to the more popular 'EHW', which I shall reserve for referring to adaptive systems.

The ETL group regard EHW as a potential replacement for artificial neural networks (ANN), highlighting advantages over the latter in execution speed and comprehension of the final result (Since they do not adopt the unconstrained approach, they are able to convert gate-level EHW circuits into boolean expressions for analysis). An initial case-study was performed which comprised a welder controlling circuit and an EHW system implemented on Xilinx XC4025 FPGAs, which received the same sensory inputs as the controller (Higuchi & Hirao, 1995; Higuchi et al., 1996a). As welding progressed, the EHW system was scored according to how closely its outputs approximated that of the existing controller. The basic idea was that by the time any malfunction occurred in the conventional control system, the EHW duplicate would have evolved the same function sufficiently closely to assume control.

A general EHW ASIC[2] was proposed (Higuchi, Iba, & Manderick, 1994) in which a whole population of individuals could be instantiated at once on a single chip, leading to the realisation of a gate-level EHW LSI chip consisting of GA hardware, reconfigurable hardware logic, a chromosome memory, a training data memory, and a 16-bit CPU core (NEC V30) (Kajitani, Hoshino, Nishikawa, et al., 1998). The chip was developed to serve as an off-the-shelf device for implementing adaptive control logic, and was successfully used to implement a control circuit in a myoelectric artificial hand. Currently, the user must adapt to the hand through a long period of training lasting several weeks. Using frequency spectra generated by a data glove repeatedly performing three pairs of actions, six control circuits evolve to provide the 'correct' myoelectric signals for the hand. Effectively, the hand adapted to the user, rather than *vice versa*. As similar research had been undertaken using neural networks (Uchida et al., 1993), the authors made a direct comparison with ANN controllers using back-propagation. Learning by the ANNs required around three hours, whereas the EHW system needed only a few minutes (Kajitani et al., 1998).

The same chip was used as part of a wheeled-robot controller in a programme that compared off-line model-free and on-line model-based hardware evolution (Keymeulen, Iwata, Kuniyoshi, & Higuchi, 1998). In the off-line approach, both the hardware and environment are simulated to find a controller capable of tracking a coloured object, and to avoid obstacles. The best controller found by evolution is used to control the real robot. The on-line approach allows the robot system to 'experiment' in an approximated world model of the environment (Booker, 1988; Grefenstette, 1996). In this case each controller is evaluated in the world model using an *experience replay* strategy (Lin, 1992). The strategy supports learning and execution taking place concurrently, but Keymeulen et al. prefer a sequential system by which an (initially random) controller is selected and during execution, the robot gathers data and builds up a world model as a series of state transitions. After a maximum discrete-time period of unsuccessful tracking, or if the robot hits an obstacle, an evolution phase commences to find a controller that does not hit an obstacle or get stuck in a loop and the execution/data gathering phase recommences.

The ETL group has also proposed several application-specific EHW architectures. In Tanaka, Sakanashi, Salami, et al. (1998), a hardware system was devised to compress individual image stripes in real time. This is a requirement of the translation process from postscript data to printed image in electrophotographic printers. The compression/decompression phase is a major bottleneck in the translation, due to the computational overhead required. In practice, a prediction

[2] Application-Specific Integrated Circuit

mechanism is used to predict the value of a given pixel based on the values of its neighbouring pixels. If the value can be predicted correctly, it is not necessary to store it separately, and the size of the image data is reduced. Consequently, the compression rate is related to the accuracy of the predictions. However, since the data is constantly changing the most suitable prediction function must be continually reselected. The EHW system achieves this with a GA implemented in hardware, conducting one run of 100 generations per stripe. Once the function is found, the hardware prediction mechanism is reconfigured accordingly. Tanaka et al. claim compression ratios twice as good as the industry standard, JBIG (ITTC, 1993).

The above examples exhibit varying levels of adaptability. In the welding controller, evolution takes place off-line, and it may be argued that the system as a whole — rather than the EHW component — is adaptable to faults. In this case however, there is no reason why an additional EHW controller should not be added to cope with continuous faults and environmental changes rather than a maximum of one. While on-line evolution takes place in the printer compression system, the structure of the data set is known in advance, and even if a prediction function cannot be found for certain data, it can be stored uncompressed without affecting the printer's operation. It is difficult to see how the system could adapt to faults. The execution phase of the myoelectric hand has two distinct components; learning and using, with on-line evolution and adaptation taking place in only the first of these. Only the robot controller is adaptable according to the definition made at the beginning of this section. These observations are made to highlight the 'grey' area between off-line evolution and adaptation in the context of completely unknown environments and equipment malfunction. They are not intended as any criticism of the studies described here, which in my view are among the best examples of HE's pertinence for real engineering applications. While on-line evolution is outside the scope of this thesis, the above research has implications for portability, fault-tolerance and robustness, the investigation of which will form a large part of the coming chapters.

2.1.5 Extrinsic Evolution of Analogue Circuits

There are various means by which information contained in the genotype can be used to construct netlists for commercial circuit simulators, and hence to evolve circuits in simulation. The advantages of doing so are that a large range of components — or models of them — is available, and that waveforms at all internal nodes of the evolved circuit can be displayed, facilitating analysis. Bennet, Koza, Andre, M., and Keane (1996) use a very simple 'embryonic' circuit as a starting point for a circuit-constructing program tree of the type ordinarily used in genetic programming (GP) (Koza, 1992). The embryonic circuit is then developed by progressively executing the functions in the program tree, which modify component values and interconnections, and insert new components into the circuit. Using this method, the group were able to evolve a two-band crossover filter and a 60dB amplifier sharing some of the features found in commercial operational amplifiers (Koza et al., 1996). The circuit-constructing tree approach allows phenotypes containing repeated structures to be evolved. These could be discerned on the schematics of the two circuits, however the schematics did not otherwise manifest the modularity found in conventional designs. For example, partly as a consequence of abstraction, a crossover filter would generally comprise two functionally independent components; one for each frequency band. The authors explained their

evolved topology as follows:

> In this run, genetic programming has no reason to evolve a circuit that employs this kind of neat decomposition of the problem into two disjoint parts ... The evolved circuit is holistic in the sense that there are numerous interconnections between the parts feeding [each output] (Koza, Bennet, Andre, & Keane, 1996).

This argument would appear to support those concerning design abstraction made by Thompson (section 2.1.2). But there are interesting differences between the respective test cases. First, the evolved crossover filter used *more* components than would be necessary had for example, standard butterworth filters been used (Horowitz & Hill, 1989, p1064). Second, notwithstanding abstraction, there are good engineering reasons to provide independent subcircuits for each frequency band, since the specifications of one filter is essentially the*reverse* of the other. The interactions present in the evolved circuit should result in undesirable attenuation of frequencies that should be left unmodified, though the frequency-domain graphs provided by the authors show that this is not the case. Third, since the SPICE simulator used was not able to model interactions though non-connected paths such as parasitic capacitance or electromagnetic coupling, evolution could not exploit these 'proximity effects', as it could for the intrinsically-evolved tone discriminator. Therefore, contrary to the tone discriminator, the crossover filter's bizarre topology is not in any way due to proximity effects. So why does the filter look like it does? Is this particular circuit typical of those that would be produced by many runs? Does it possess any non-behavioural qualities that we don't yet know about? There is not sufficient information to answer these questions; the above quotation is certainly correct in that evolution has no need of abstraction, but it may not tell the whole story. The topology is potentially due to the evolutionary development of the circuit from its initial 'embryo', in other words a direct and possibly inevitable consequence of the methodology employed. I believe these questions to be important, because the engineering utility of unusual evolved circuits could be increased by a better understanding of their operation and development; an argument I will develop further in chapter 3 and demonstrate in chapter 6. We have already seen in section 2.1.3 how certain qualities may arise for free from the way evolution searches through design space for an optimal circuit. There may be other qualities inherent in evolved circuits, desirable or not, so far unknown, but whose discovery and understanding would further increase HE's usefulness as an engineering methodology. Chapter 6 will show that such qualities do exist, and can be understood through a combination of analysis and empirical study on evolved circuits no more complex than the crossover filter.

2.2 Benefits and Difficulties Inherent in HE

The above body of work reveals a clear distinction between retaining digital design constraints to evolve practical engineering applications, and relaxing them so that evolution can harness the capabilities of semiconductor configurations in a way not possible with current techniques. Characteristics like parsimony and fault tolerance have obvious engineering value, but the value is lessened if they are achieved at the cost of temperature stability and component tolerance. This distinction forms part of the inspiration behind my own research program, which I now develop and will support with more recent research where this influences my arguments. While the unconstrained approach seems restricted by its likelihood of producing circuits dependent for their

operation on subtle physical properties, it has thus far only been applied in extreme cases — 'forcing' basic elements intended for digital design to behave in an analogue manner. To develop it for more practical use requires investigating a combination of potentially more suitable basic elements, and constraints tailored to the evolutionary approach rather than those inherited from conventional design. The remainder of this chapter will be devoted to the factors involved in achieving this and how they may best be investigated. Ultimately, the potential benefits of HE implied by the previous section are sought. My own interpretation of them is summarised below, but it must be stressed that some are still far from being fully realised.

2.2.1 Benefits of the Evolutionary Approach

No requirement for 'Divide and Conquer'-type abstraction: Through relaxing constraints which human designers deem necessary to cope with circuits of high behavioural complexity, evolution can produce functional circuits with richer dynamical and spatial structure than is within the scope of conventional design.

Exploitation of the medium: When implemented intrinsically, evolution is capable of exploiting subtle physical properties of the evolvable medium that cannot be predicted using the conventional approach, thereby increasing further the spectrum of behaviours that the medium can produce.

Parsimony: Complex behaviours can be achieved with smaller component count than normally expected of the conventional approach.

Effective use of available components: Evolution is capable of producing functionality in the absence of components that would be necessary under conventional design maxims. This has significant implications for VLSI where components such as capacitors, resistors and inductors are difficult to implement in small size.

Graceful degradation: The relative insensitivity of the phenotypes produced by some EAs to mutation in the genotype can be exploited to produce systems which are inherently robust to certain types of fault.

Adaptation: Artificial evolution is a viable approach to constructing systems that can change their configuration in real time to cope with unexpected circumstances in their environment.

As a source of knowledge: The circuits produced using evolution may reveal electronic properties or component configurations not yet exploited in conventional design. Analysis of such circuits may lead to the properties being used to create new and innovative designs in the conventional manner.

Design: For certain types of system, an HE framework can be envisaged for which the only human requirement is to produce a behavioural specification, which can be achieved with little or no knowledge of electronics, thereby opening up the field of hardware design to non-specialists.

2.2.2 Challenges for the Evolutionary Approach

Notwithstanding the above benefits, there a number of difficulties and uncertain issues whose resolution will be an important factor in the progression of HE as an engineering tool. Principal issues are portability, robustness, scalability, and circuit verification/testing.

2.2.3 The Influence of Physical and Methodological Constraints on Portability

A circuit is defined here as being portable if it can be transferred from the device on which it was evolved to a different but nominally identical device, and be relied upon to exhibit the same behaviour within some well-defined bounds. By this definition, neither the tone-discriminator nor the wall-avoiding robot are portable. This test has already been conducted with the tone discriminator (see section 2.1.2 above), but not with the robot. However, since we do not fully understand every factor governing the robot's operation, we cannot rely upon it to work in a different hardware implementation, hence it cannot be deemed portable. By contrast, it is possible to deduce clearly whether or not the evolved printer compression system or the welder controller (section 2.1.4) are portable, because their function can be expressed comprehensively in terms of boolean equations. In the case of the welder controller, it is possible to make analytical assessments of the extent to which for example variations in motor speed of the welder arm will affect the overall behaviour, hence it can be determined in advance whether or not implementing the system in different hardware will result in behaviour within the defined bounds.

If we wish evolved circuits to be portable, are we necessarily forbidden from relaxing the constraints that enable systems to be expressed by abstract models, or are there certain classes of constraint which may be relaxed during evolution without affecting an evolved circuit's portability? For the purposes of this discussion, I shall place the various types of constraint into two broad categories — *methodological* and *physical*.

Methodological constraints are imposed to reduce the number of possible interactions between components of a complex system, to a quantity small enough to be understood and manipulated by human designers. An obvious example is the well-known functional decomposition approach (also known as 'Divide and Conquer'), through which a system is constructed by connecting together largely independent subsystems having relatively few input/output connections, and whose internal dynamics are unaffected by those of other components of the system.

Physical constraints reduce the influence of certain physical characteristics whose value cannot be specified with precision in the manufacturing process, or whose value varies significantly according to certain operational conditions such as voltage or temperature. Such constraints may be as simple as ensuring that the circuit's power supply is well regulated, but more complex constraints are frequently used, for example through the analytic modelling of devices such as transistors: Mathematical manipulations can be performed on the models to provide configurations whose behaviour is not affected by excessive variations in the 'rogue' characteristic. In this case the designer is constrained because he or she can use only a subset of all possible configurations when employing the device in a circuit.

One highly successful design technique — synchronous digital design — imposes constraints which are both methodological and physical. Communications between interconnected circuit elements (normally logic gates) are only allowed to take place on the 'ticking' of a global clock. This technique reduces the state-space of the system, since it eliminates effects such as free-running 'glitches' (pulses of very short duration caused by differing gate propagation delays) from percolating through the circuit. Furthermore, physical characteristics which affect the time taken for a glitch to percolate, or a gate to reach a stable logic state, need not be precisely understood; the designer only needs to know the maximum time required to reach the state, and adjust the clock

frequency accordingly.

Methodological constraints can be relaxed without affecting a system's portability. The resulting circuits may have unusual topologies but analysing and understanding them, though difficult, is possible with existing tools and techniques[3]. However it seems that we relax physical constraints at our peril (as far as portability is concerned). Although it is theoretically possible to construct analytical models[4] which take into account all the physical properties of a circuit, this is currently infeasible (Thompson, 1996b).

A recent extension of the tone-discriminator experiment casts new light on the constraints issue. Thompson (1998b) wondered whether evolution could be encouraged to find its own mechanisms for increasing what he calls the 'operational envelope' of evolved circuits; the range of environmental conditions within which a system can be expected successfully to operate. Examples of environmental conditions are temperature, power-supply voltages, load conditions and fabrication variations. If evolution could be so encouraged, then the need to impose *a priori* physical constraints would be reduced. To this end, an evolvable machine (jocularly named the 'Evolvatron') was constructed to allow five different FPGAs to be configured and evaluated simultaneously with the same genotype. Though nominally identical, FPGAs manufactured by two different foundries, and with different packaging were used. They were also run at different temperatures (12 to 60°C), with one FPGA having a thermal gradient of 15°C/cm maintained across the central region of the package. A variety of power-supply voltages and load impedances were incorporated into the system, and individual genotypes were instantiated on geographically different 10×10 CLB regions within the FPGAs. Initial evolvatron experiments were seeded with the original tone-discriminator configuration, and met with only partial success (Thompson, 1998b). Later, a 6MHz clock was provided as an optional resource that could be coupled synchronously or asynchronously, in a similar manner to the wall-avoiding robot controller. Near-perfect behaviour on all five chips was successfully evolved (Thompson & Layzell, 2000). The evolved genotype was used to configure six other FPGAs that had not been used during evolution, with all displaying the same near-perfect behaviour across a temperature range of -50°C to room temperature. The circuit also worked in simulation, which was used to provide a preliminary analysis. Although much of the circuit was synchronised to the clock, the simulation waveforms showed many spikes and glitches of various duration resulting from the analogue time delays through the circuit. Presumably the design evolved to be insensitive to these, perhaps completely ignoring them. The circuit worked first time in simulation, without any need to fine-tune the component timings. This means it does not rely on any physical properties of the chips not captured in simulation.

These findings show that evolution can be encouraged to find its own physical constraints, *without them having to be specified a priori*:

> The circuit has evolved to achieve robustness by finding a well-behaved, clocked, digital mode of operation. However, it is not at all clear what digital design rules could

[3] In fact the use of EAs involves methodological constraints such as those on component choice or interconnection architecture. Rather than relaxing methodological constraints altogether then, the HE designer chooses among different ones.

[4] In mathematics, the term 'analytical model' denotes a set of equations from which a result can be formally derived. However, it is used here in a broader context to denote any model constructed to facilitate some understanding of a system's operation. This could be a set of formal equations, but could also be a software simulation.

> have been formulated in advance, that would have guaranteed robust digital evolution and yet permitted this evolved circuit (Thompson & Layzell, 2000).

There is no guarantee that the circuit would work on yet another different chip. However, as for the welder controller and printer compression circuits, the abstract simulation model could be used to determine bounds on the physical qualities involved, so that the circuit's portability could be deduced. On the other hand, if circuits produced using the evolvatron method do not work in simulation, it is difficult to envisage how their portability can be predicted. Stoica, Zebulum, and Keymeulen (2000) suggest simultaneous configuration and evaluation in both hardware and simulation, and have produced some preliminary results. In this case, CMOS transistors were used as basic elements for the 'mixtrinsic' evolution of an AND gate. The group found that the transistors' physical properties had to be modelled with high accuracy by the simulator before the AND behaviour could be successfully evolved (R. Zebulum (2000), personal communication). Therefore, while the mixtrinsic approach forbids component interaction through unconnected paths, the simulation is tailored to a particular hardware implementation. Portability cannot be assumed.

So far I have discussed the conditions under which an evolved circuit configuration can reliably be used as a blueprint for direct instantiation on multiple devices. An alternative, indirect method is to incorporate an evolutionary phase into the manufacturing process. The basic idea is to use unconstrained intrinsic evolution to provide a functional prototype whose genotype would seed further evolution on a different reconfigurable device. For this approach to be viable, the 'post-prototype' phase would have to craft the circuit's configuration to suit the new device's physical characteristics using far fewer generations than were required to evolve the prototype from scratch. The only example to my knowledge of this being carried out prior to this thesis is the tone-discriminator's instantiation on a different region of the same FPGA (Thompson, 1997a). In this case only 100 further generations were required to recover perfect performance, compared with 5000 to produce the initial prototype. Similar tests will be conducted in chapter 5, but much more work is needed to assess the feasibility of the method.

2.2.4 Transferral from Simulation to Reality

Transferring extrinsically evolved circuits from simulation to reality presents additional portability problems to those already discussed. First, only particular 'preferred' component values for discrete components are readily available. Second, real components have parasitic properties, for example an inductor has resistance as well as inductance. These factors would inhibit the construction and performance of say, Koza et al.'s crossover filter. Third, circuit simulators consider devices as mathematical entities, ignoring some of the physical *limitations* they may posses. No warning is given by most commercial simulators when overcurrent or overvoltage conditions occur, hence a device may be destroyed when the circuit is implemented in hardware. Several authors have provided evidence that these problems can be overcome. Horrocks et al. have shown that evolution can be successfully constrained to use a particular series of preferred component values (Horrocks & Khalifa, 1994), and that by incorporating a component-value sensitivity analysis into the fitness function, component values which include certain parasitic properties can be used in simulation to evolve circuits that would work well in the real world (Horrocks & Khalifa, 1996). Zebulum, Vellasco, and Pacheco (1998) provide a set of precautions applicable to extrinsically

evolved transistor models. The precautions include checking that the base-emitter voltages do not surpass the nominal value (usually 0.7V); checking that the collector current does not surpass the maximum possible value, and adjusting the device models so that transistor parameters known to vary widely for different specimens of a given type match the values of the transistor to be used in the real implementation. By following these precautions, the group has been able to evolve amplifier circuits in simulation, whose behaviour is closely matched by their physical implementations. An alternative 'minimal simulation' approach (Jakobi, 1998) has already achieved some success in evolutionary robotics, and could potentially be adapted for HE. The basic idea is to model only those robot-environment interactions that are necessary to underpin a desired behaviour. Everything else is made unreliable by careful use of randomness, hence an evolved controller is forced to use the minimal set of interactions picked out by the simulation designer. In the context of HE, the fundamental interactions desired in a basic element could be included in the simulation (for example, output voltage as a function of input voltage in an operational amplifier), but known parasitic properties replaced with noise to avoid the need to model them and to ensure that they do not affect the physical circuit's operation.

2.2.5 Robustness

Ensuring that a circuit evolved on one medium behaves identically on another is of little use if its original behaviour was flawed. Reliable behaviour is largely dependent on the exhaustiveness of the fitness evaluation. If the evaluation method excludes some input or environmental conditions, then predicting an evolved circuit's behaviour under those conditions must be carried out by applying analytical models and inevitably, physical constraints. Where this is undesirable or impractical, hardware evolution necessitates a compromise between fully exhaustive evaluations and the computational cost of conducting them (Thompson et al., 1996; Yao & Higuchi, 1996). There are many factors governing how exhaustive a given evaluation method can be, particularly for intrinsic HE. For example, evaluating over a variety of temperatures requires not only a time factor, but a knowledge of whether the configurable device will withstand the stress imposed by continuous thermal modulation. Even if ambient conditions are ignored, evaluating a candidate circuit under all possible input states does not guarantee similar behaviour in later use. We shall see in section 5.4.1 how an evolved circuit's operation may depend on the *sequence* in which a device's configuration switches are set when it is instantiated in hardware. There are no established methods to guarantee robust operation of evolved analogue circuits. A highly constrained, 'brute-force' approach might be to use coarse-grain components as basic elements whose input/output function is well known, such as operational amplifiers, and restrict their interconnection to configurations known to cause predictable behaviour. In the case of operational amplifiers, positive feedback would be forbidden, and any phase-shift in negative-feedback configurations carefully controlled to prevent oscillation. Amplifier outputs could only be connected together indirectly, via some impedance whose value is restricted to some minimum level satisfying the manufacturer's output load specifications. To ensure robust operation under varying temperature, restrictions might also be placed on the maximum gain a given element could be configured for, and highly stable reference voltages and signals enforced. If under these severe restrictions, a system displaying dynamical behaviour could still be evolved, robust behaviour could only be ensured by an evaluation

method that covers all possible basins of attraction attainable by the system. Such an approach may well produce circuits not already achieved by conventional design, but it would be difficult to put into practice, and may result in inappropriate search spaces for evolutionary algorithms to explore (see *evolvability*, below).

Although I have made a clear distinction between robustness and portability, an approach designed to improve the former may also improve the latter, and *vice versa*. The tone-discriminator produced by Thompson's Evolvatron was portable and also robust to variations in ambient temperature. There are also parallels with the HE systems incorporating on-line evolution discussed in section 2.1.4. Adaptive systems should by definition, be robust. However, as evolved systems increase in complexity, robustness becomes more difficult to guarantee. It may be that the degree of robustness currently expected of engineering systems will have to be relaxed for evolved circuits of genuine utility. There are already precedents for this in software systems, where it is no longer uncommon to release operating systems and applications on to the marketplace, even though they have known bugs and reliability problems. In this case, the usefulness of a system outweighs its inherent unreliability. Therefore, while efforts to improve the robustness of evolved circuits must continue (as they do in the software industry), the notion that only perfectly behaved circuits are of use in engineering applications should not deter us from exploring what evolution is capable of, albeit within a restricted operational envelope.

2.2.6 Scalability

To my knowledge, no circuit with 100+ functional basic elements has yet been evolved; the greatest number so far attained seems to be around 30-40 (Koza et al., 1996; A.Thompson, Layzell, & Zebulum, 1999; Bennett, Koza, Wu, & Mydlowec, 2000). There are examples with greater element count, for example Thompson et al.'s slow oscillator (Thompson et al., 1996) and Hemmi et al.'s evolved artificial ant (Hemmi et al., 1996), but tests have not yet been conducted on these examples to determine how many of the elements have a functional role, and how many are 'junk' (Note that I use my own definition of 'basic element' here (section 1.2.2), but acknowledge that it could be susceptible to different interpretations).

Yao and Higuchi (1996) consider the fundamental limiting factors in producing large-scale, complex systems to be computational complexity and genotype length. They point out that "Neither the worst nor average time complexity has been established for any EA used to solve any particular [HE] problem". Using a hypothetical example which assumes $O(2^n)$ time complexity in an EA applied to hardware evolution, they compare 1 microsecond for evolving a system with 10 basic elements to 10^{13} years for one with just 100. The example is useful in that it illustrates the danger of relying too much on hardware speed, but $O(2^n)$ implies that no new solutions appear as the search space increases, therefore it is not unrealistic to expect better orders of complexity. Thompson (1996b, chapter 2) uses a more sophisticated example when disputing de Garis' vision of 'evolution at electronic speeds' (de Garis, 1995). Using educated guesses for future best-cases of efficiency of future EAs, use of parallel hardware for concurrent fitness evaluations, and evaluation speeds, he estimates a factor of around 10^4 increase in speed of a future evolutionary process over existing techniques.

If the genotype⇒phenotype mapping employed for the tone discriminator were used to con-

figure an entire X6216 FPGA, genotype lengths of the order 10^5 would be required. Several authors believe evolution using genotypes of such sizes to be inefficient (Kitano, 1996a; Yao & Higuchi, 1996; Iwata, Kajitani, Yamada, et al., 1996). Solutions to this perceived problem include variable-length GAs (VGA) for more compact encoding (Iwata et al., 1996; Higuchi et al., 1996a) and grammar-based morphogenetic systems which allow the use of repeated substructures in the evolving system (Kitano, 1998; Hikage, Hemmi, & Shimohara, 1998). Other approaches endeavour to provide smaller search spaces rich in potential solutions by incorporating domain-specific knowledge into the evolutionary process. In their Field Progammable Transistor Array (FPTA) chip, (Stoica, 1999; Zebulum, Stoica, & Keymeulen, 2000) restrict the interconnection architecture to "a sufficient number [of configuration switches] to allow a majority of meaningful topologies for the given transistor arrangement", in this case 24 programmable switches. The restrictions are based on how transistors are conventionally wired, and the group give a number of implementations of standard building blocks as examples. However, the FPTA's interconnection architecture is also versatile enough to allow very unusual circuits to be configured. Higuchi, Iwata, Kajitani, et al. (1996b) suggest that fine-grain, or gate-level evolution is insufficient for practical applications and propose the use of higher level hardware functions (such as adders, oscillators, etc.) as basic elements, presumably due to their widespread use as 'building blocks' in conventional design. The danger here is that even small genetic manipulations in such systems may cause radical changes in behaviour, and the evolutionary process degenerates to random walk (Kauffman, 1993). The application of domain-specific knowledge about known ways the problem can be solved, so that evolution will not have to waste time rediscovering them is inadvisable according to Thompson (1996b), who warns:

> This *is* sensible if this information inevitably would have to be rediscovered, but if the information represents just *some* ways of setting about solving the problem — perhaps ways suitable for human designers, or for evolution in biology, but not suitable for the electronic medium — then forcing evolution to use this information unnecessarily restricts the space of possible solutions. Even worse, it could steer evolution in ways incompatible with the nature of the evolutionary process or of the reconfigurable medium.

Clearly, there is more to scalability than vague notions of complexity and genotype length alone. The third major factor is *evolvablility*, which concerns the nature of the search space. It is not so important to have a search space rich in potential solutions as it is one that allows a gradual path to any solutions to be found by the chosen genetic manipulations of the EA. Unfortunately, very few direct comparisons between different electronic search spaces have been conducted for a given task. Indeed closely related work by Miller and Thomson (1998) suggests that direct comparison should be used with care. Their work assesses the evolvability of different genetic representation and evaluation methods. The authors compare the evolution of digital functions using two different genetic encodings, which allow different interconnection capabilities between functional cells. Each encoding equates to a particular cell geometry in hardware: the first, to an array of functional cells with limited routing capabilities, and the second to an array in which half the cells are devoted purely to routing (referred to as 'undifferentiated' and differentiated models, respectively) . When a GA was applied, the undifferentiated model yielded significantly more optimal solutions than the differentiated one. However, in their endeavour to make a direct

comparison, the authors found that the number of *functional* cells in the differentiated model was close to the theoretical minimum for the chosen task, and point out that one should be cautious in assuming one model to be generally better than the other. The paper also included an interesting comparison between exhaustive and partial fitness evaluations, to determine whether evolution could 'learn by example', and produce combinatorial logic circuits given only samples of the full truth table. The results do not bode well for the evolution of complex digital circuits; optimal fitness could only be achieved when evaluating against a complete truth table.

Evolutionary roboticists have studied *incremental evolution* — in which the target behaviour is broken down into a sequence of tasks of increasing difficulty — as a means of increasing behavioural complexity (Kodjabachian & Meyer, 1998; Harvey, Husbands, & Cliff, 1994; Jakobi, 1996). The sequence can be constructed by using domain knowledge to specify increasingly difficult fitness evaluations. Alternatively, a population of problems or constraints can be made to co-evolve with the circuits such that the difficulty increases as the circuits gain greater efficiency. Lohn, Haith, Colombano, and Stassinopoulos (1999) compared both approaches with the conventional use of a static fitness function, conducted over 25 runs each of an analogue amplifier task. The domain knowledge approach was applied using a sequence of 100 predetermined difficulty steps. Runs were conducted for both regular intervals between step changes (fixed schedule), and changing step only when the fitness of at least one individual solves the current difficulty (adaptive schedule). The different approaches were compared using the mean fitness, computed over all runs of each schedule. Both the fixed and adaptive schedules showed poor performance (in terms of the generations required to evolve to a given fitness) relative to the co-evolutionary one, but none showed performance superior to that of the static fitness function. Lohn et al. do not attempt conclusive explanations for this result, but they suggest that "the fixed and adaptive schedules are potentially 'handicapped' by the somewhat arbitrary choice of manually-crafted schedules", implying that the step changes altered the search space too radically for the evolving population to maintain individuals of adequate fitness. More research is needed to clarify this issue.

2.3 Evolvable Architectures

When this research commenced, commercial reconfigurable devices were becoming increasingly numerous and diverse (Sanchez, 1996), and it appeared that there would soon be a large choice available for the HE researcher. However, those with most potential as media for intrinsic HE proved less suitable for industrial engineering and have since been discontinued, with few new products to take their place. Notable examples are the Xilinx XC6200 series FPGA (Xilinx, Inc., 1996) upon which a number of evolvable machines have been based (for example Korkin, Nawa, and de Garis (1998), McCaskill et al. (1997)), and the Motorola/Pilkington Field Programmable Analogue Array (FPAA) (Bratt & Macbeth, 1996), which offered switched-capacitor amplifiers as basic elements (for example, Zebulum et al. (1998). The XC6200 series is particularly missed as it was the only FPGA that met all of the requirements for intrinsic HE: *Reconfigurable an unlimited number of times; Fast Reconfiguration; Indestructability; Flexible Input/Output; Observability (through an open architecture); Fine-grain reconfigurability* (Thompson, 1996b).

Nearly all current commercial FPGAs are susceptible to configurations that can damage or destroy them, through the interconnection of tri-state outputs. It is the user's responsibility to ensure

that all but one of the connected outputs is in a state of high impedance at any time; if two outputs simultaneously attempt to drive the same signal to opposite logic levels, the resulting large current levels can damage the device. This situation is known as 'bus contention', and can in principle be avoided by restricting the set of possible genotypes to those encoding non-destructive configurations. However, most FPGAs have a 'closed' architecture – the way in which the configuration bits determine what circuit is present on the chip is kept a proprietary secret of the manufacturer — in order to deter any reverse-engineering of user designs. Therefore, validity checking must be carried out by the manufacturer's software, which produces the configuration bits from a user's design (for example the JBits package (Guccione & Levi, 1998)). Running this software for every candidate genotype introduces a severe computational overhead to an evolutionary system that uses the FPGA. Additionally, analysis of any circuit evolved on a closed-architecture device must take place at the digital model level, because more precise implementation details are unavailable to the user. Slorach and Sharman (2000) propose a 'system on a system' solution to combat the potential of current FPGAs to self-destruct. They give the design and implementation of an indestructible XC6200-type architecture and show how it can be emulated using 'off-the-shelf' FPGAs such as the Xilinx XC4010E or Altera EPF6010A. Due to the limited CLB count (< 1000) on these devices, the emulated architecture has a maximum of only 28 cells, but this could be increased if devices such as the Xilinx Virtex series (Xilinx, Incorporated, 1999) were used.

Several groups have proposed and in some cases manufactured custom evolvable architectures implemented in VLSI. As well as the EHW and FPTA chips already described, examples include Hamilton et al.'s Palmo device, a mixed-signal VLSI which can process analogue signals specified as pulse streams (Hamilton, Papathanasiou, Tamplin, & Brandtner, 1998; Hamilton, Thomson, & Tamplin, 2000), and Moreno et al.'s FIPSOC device, which comprises programmable analogue and digital sections plus a microcontroller on a single chip (Moreno, Madrenas, Faura, et al., 1998). Additionally, a large number of reconfigurable computing devices and neural networks implemented in VLSI continue to be introduced (see for example Prieto (1999)). Though interesting, these latter devices lie outside the scope of this thesis; I want to stay clearly focused on evolutionary circuit synthesis at the component level. Evolvable architectures and aspects involving their design and use will be further discussed in chapter 4, where I will present my own architecture intended specifically as a tool for further research into the issues surrounding HE.

2.4 Navigating Design Space: Neutrality and SAGA

I now describe the factors governing my choice of EA for the experiments documented in the coming chapters. In essence, the choice is a compromise between an EA effective enough to evolve non-trivial test cases, but basic enough for any conclusions to be widely applicable. My experiments will cover aspects of evolvable architecture, fitness evaluation, analysis and search space, but their results are not to depend on the nuances of some problem-specific EA. Consequently, I will employ Harvey's Species Adaptation Genetic Algorithm (SAGA) (Harvey, 1992b) throughout this thesis, because it already has an established record for circuit synthesis, and can be applied using a basic GA with very few alterations (Harvey & Thompson, 1997; Thompson, 1996b). SAGA was originally developed as a conceptual framework, to extend the application of a basic GA to constantly changing search spaces, for example where variable-length genotypes are

employed. Unlike the standard GA, for which genetic convergence (achieved through many cycles of variation and selection) equates to termination of the evolutionary search, SAGA provides the framework for a relatively genetically converged population to search continuously within the space. The difference can be illustrated by imagining a large three-dimensional space containing points which represent individual genotypes. Whereas for a standard GA the points are sparsely scattered within the entire space, and gradually converge to a specific locality, SAGA may be visualised as a localised 'cloud' of points, navigating the space as a single entity (Harvey, 1992a). The only modification to the standard GA required by SAGA is to maintain a constant selection pressure (for example, by using rank-based selection), and to set the mutation probability to, on average, one mutation that affects fitness per genotype.

2.4.1 Elitism

The EA chosen for the forthcoming experiments will be a standard GA with rank-based selection, single point crossover and mutation, with mutation rates chosen according to the SAGA framework. As mentioned in section 1.2, I will also incorporate elitism, since it has already proved instrumental in successful HE both in the literature (Miller & Thomson, 1998; Thompson, 1996b; Kajitani et al., 1998), and in my own initial experiments.

The simplicity of the elitist operator belies its profound affect on the evolutionary search process. EAs without elitism have a stochastic likelihood of leaving promising areas of search space to explore other areas which may be initially poorer in terms of fitness, but which may ultimately yield better scores. By contrast, an elitist EA is committed irrevocably to finding solutions as good as or better than the previous best[5]. Elitism would therefore seem deleterious to SAGA. Continuing the above visual metaphor, the selection pressure on the genetically converged cloud of points confines its movement to areas of increasing fitness. Elitism allows no 'back-tracking', hence the cloud appears destined to become lodged in some local sub-optimal region, unless it is known in advance that the search space does not contain them. This is only possible if all the solutions are known in advance, in which case there is no need for an evolutionary search. Why then, do empirical results suggest that elitism is advantageous?

2.4.2 Neutrality

The previous argument is an intuitive one based on a 3-D visualisation of search spaces. However, the visualisation is misleading for the 'real' search spaces extant in the majority of HE experiments, which possess very high dimensionality. Genotype mutations within these spaces can be advantageous or deleterious regarding the corresponding fitness score. They can also be *neutral*, if they leave the fitness unchanged. Paths accessible by neutral mutations can percolate through vast volumes of high-dimensional genotype space. Consequently, a population can undergo a neutral drift through such pathways, preventing the evolutionary search from being terminally impeded by the presence of local optima (Kimura, 1983; Barnett, 1997).

Assessing the degree of neutrality inherent in a given search space is extremely difficult, because there are many factors determining whether or not a mutation can be expected to be neutral.

[5]There are exceptions to this argument, for example if the fitness evaluations are noisy. This will be explored further in chapter 4, but the present discussion assumes noiseless evaluation

For example, neutral mutations can occur in 'junk DNA' — parts of the genotype which do not form part of the evolving circuit. However the same part may be incorporated into the circuit at a later stage, resulting in mutations of that part no longer being neutral. Mutations occurring in non-junk DNA can also be neutral if either the behaviour of the phenotype is unchanged, or if the evaluation method apportions equal fitness to different behaviours. Thompson et al. (1996) give a more detailed discussion on neutrality, together with examples of the types of neutral mutation described above.

At the time of writing, the most extensive study of the nature and role of neutrality in the context of HE is Vassilev and Miller (2000). The authors use three-bit multiplier search spaces or 'landscapes' as test cases to derive various metrics for the type and frequency of neutral mutations during evolutionary runs and random walks. The experiments revealed that the evolutionary process is accompanied by neutral mutations, the number of which decreases with a lower rate compared with the decrease in length of the neutral pathways (as measured by random walks) as fitness increases. This observation is used to argue that neutral mutations occur as a natural consequence of the high proportion of junk DNA in low-fitness genotypes produced at the early stages of evolution — but as fitness increases and the proportion of junk DNA reduces, the neutral mutations no longer just 'happen', but are promoted by the selection mechanism and become an important component in exploring the search space.

The success of multiplier evolution where neutral mutations are permitted is then compared to that where they are not. The permitting/forbidding of neutral mutations is achieved by a small alteration of the selection mechanism. The authors concluded that:

> The landscape neutrality appeared to be vitally important in that it firstly prevents the evolved sub-circuit from deleterious mutations, and secondly, it allows the evolutionary search to avoid entrapment at local optima[6]

I believe that the arguments presented in this section are sufficient to justify my choice of EA, although I acknowledge that there are many additional factors that influence population dynamics. There will be little further discussion of evolutionary theory in this thesis. The work is tackled from an electronic engineering perspective; I do not pretend any authority on theoretical approaches to evolutionary computation. For the same reason, I will avoid comparisons with evolutionary biology. Whether or not this thesis has implications for those fields, the focus is on the nature of electronic circuits produced by artificial evolution, for which I will use the simplest and most general EA practicable. It is left to the reader to infer whether other sorts of EA may have achieved faster evolution or different results, but where my arguments are dependent on the choice of EA it will be clearly stated.

2.5 The Position of This Thesis Within the field

In preparing this research program, I have taken a specific viewpoint: That small circuits evolved with as few constraints as possible may be of increased utility to engineering applications if they are properly understood. In addition to the factors governing a circuit's operation, *understanding*

[6]The authors used an evolutionary strategy (ES) in preference to a GA. This method retains the best individual of former generations, and so is relevant to my argument concerning elitism.

encompasses the influence an evolutionary design process has on its structure and properties, and whether altering the process — for example by changing the types of basic element or reconfigurable architecture — could improve the circuit in some way. It will be clear to the reader from the preceding pages that there are many ways of tackling this, and that some of the foundations have already been laid by the authors cited above. The engineering approach I have adopted and the experiments to be presented achieve some milestones; the first evolvable architecture suitable for general HE research, the first examples of intrinsic HE at the transistor level, and the discovery and partial understanding of an entirely new form of fault-tolerance inherent in evolved circuits. While I acknowledge the importance of large-scale systems for the evolution of complex circuits, they will not be explored here. My work is intended to provide insights into the types of circuit at which evolution excels, and further foundations for the methodologies by which they can be produced. I believe that the most appropriate starting point for achieving this is through the evolution and analysis of relatively small circuits and systems.

Chapter 3

Analysis of Evolved Circuits

This chapter discusses how the domains to which an evolved circuit can be practically applied depend on how well its principles of operation are understood. The limitations of common analytical techniques are explored, and some alternatives suggested. These are then applied to a well-known evolved circuit, and result in a better understanding of its internal dynamics and expected long-term behaviour. However, the analysis fails to reveal the core mechanism underlying the circuit's operation, and it is difficult to envisage how this mechanism will ever be discovered.

Clarification of Originality

Part of the material in this chapter has previously appeared in (Thompson & Layzell, 1999) and (A.Thompson et al., 1999). The experimental program was conceived and conducted jointly by Adrian Thompson and myself. Both parties contributed equally with such a degree of interaction that it is not possible to attribute any single part of the program to either one of us. However, the text of the above publications was written chiefly by Dr. Thompson with my co-operation, as a description of our joint work. While I have substantially modified and extended the original text for this chapter, which includes previously unpublished research of my own, I have not deemed it necessary to rewrite passages which already suffice perfectly, 'in my own words'.

3.1 Promises and Pitfalls of Current Techniques

We have already established in section 2.2.3 that it is possible (and in some cases preferable) to constrain evolution to produce circuits that adhere to conventional design rules, and can be guaranteed to work both in simulation and physical hardware. These classes of evolved circuit are considered suitable for analysis using well-known techniques, and are therefore not covered by this chapter, which is concerned primarily with unconstrained evolved circuits.

3.1.1 Applying Analysis to Increase the Utility of Evolved Circuits

One of the potential application domains for HE is the design of small circuits that can be incorporated into the electronic engineer's library of building blocks (see section 2.2.1). If the circuits themselves are unsuitable for this purpose, novel strategies adopted by evolved circuits may be

able to contribute to the 'cook-book' of circuit design tricks currently used by designers. However, if an evolved circuit is of such an unusual form that its operation cannot be understood, then its usefulness is reduced. Clearly, if we are to use an evolved circuit to inspire new design tricks, then we need to be able to discern at least some of its principles of operation. Even if the goal is merely to evolve a circuit that works (and we don't need to know how), some degree of analysis may still increase its utility as an engineering product. In particular, if bounds on possible long-term changes in the circuit's behaviour can be derived, then the circuit can be applied with confidence more widely. The need for *extended* consistent performance is difficult to accommodate within an evolutionary framework, because usually the tests for fitness measurement of candidate solutions (the bottleneck in the evolutionary process) are designed to be as brief as possible. The evolutionary approach can be made more viable if, through analysis, it can be predicted that a circuit will perform adequately in the long term, even though it was never tested for long during its evolution.

There are two components to long-term stability of behaviour. First, the circuit must be insensitive to certain variations in its implementation or environment, or able to adapt to them. Examples include thermal drift, noise, and ageing effects in semiconductor devices and integrated circuits. Second, it is possible for even simple dynamical systems to display *intermittent* behaviour over long time-scales (Pomeau & Manneville, 1980). This is not due to any external fluctuations, but is a property of the system's own dynamics. Jeffries (1998) shows how simple electronic circuits can be constructed which will inevitably — though after a long period of normal operation — suddenly and unpredictably change in their qualitative mode of behaviour; possibly forever, or perhaps to return to normal operation for another long interval. An evolutionary algorithm, unless using debilitatingly long fitness evaluation tests, would be blind to this pathological behaviour, and could present such a circuit as a solution to the engineering specification. Inherently erratic dynamics of this kind can also interact with the time-dependent exogenous influences mentioned above. If analysis can provide reassurance against long-term sporadic misbehaviour, the circuit is rendered more useful.

In critical applications, complex circuits can be embedded within an error-recovering framework (Anderson, 1985). The error recovery mechanisms themselves can be simpler, and perhaps formally verified. For example, if a failure condition is detected, the circuit responsible could be automatically reset to a safe initial state. The more a circuit's potential failure modes are understood, the more feasible it becomes to construct a resilient system containing it.

Failure conditions are one of a number of nonbehavioural requirements that a circuit suitable for commercial applications must fulfil. They define the resources available to a circuit and the environmental conditions under which it must continue to operate (A.Thompson et al., 1999). Common environmental conditions are temperature, electromagnetic interference (EMI) and fabrication variations, but there are many more. Evolved circuits that rely for their operation on properties severely affected by environmental conditions are less likely to satisfy nonbehavioural requirements. We have already seen examples of intrinsically evolved circuits that work only on the device on which they were evolved, and extrinsically evolved ones that are impractical to build. If the properties essential to a circuit's function can be discerned, especially those least tolerant of changes in environmental conditions, then steps can be taken to redesign the appropriate subcircuit by hand. Its ability to satisfy the nonbehavioural requirements is thereby increased, as is its

usefulness.

When encouraging evolution to explore beyond the scope of conventional design, there can be engineering benefits to some sort of analysis even without a complete understanding of how the circuit works. The next section discusses what 'analysis' is, and how it can be performed. These techniques are then applied to illuminate an unconventional evolved circuit, showing that one can indeed address some analysis questions even if the circuit is so strange that parts of its inner workings remain unknown.

3.2 The Analyst's Toolbox

Analysis of exotic evolved circuits is different to that undertaken as part of orthodox design. It is especially important to recognise that an evolved system may not have a clear *functional decomposition.* A functional analysis decomposes the system into semi-independent subsystems with separate roles; the subsystems interact largely through their functions and independently of the details of the mechanisms that accomplish those functions (Simon, 1996). Systems designed by humans can usually be understood in this way, because of the 'divide and conquer' approach universally adopted to tackle complex designs.

Although an evolved system *may* have particular functions localised in identifiable subsystems, this is not always so. Dynamic systems theory (Burton, 1994) provides a mathematical framework in which systems can be characterised without a functional decomposition. Process algebras such as CCS (Milner, 1989) or its stochastic variants SCCS and WSCCS[1] (Tofts, 1994; Christodoulou, 1999) provide similar characterisation of discrete-time dynamical systems. Hence, what to many people is the essence of *understanding* — being able to point at parts of the whole and say what function they perform — is not always appropriate for evolved systems. In such cases, 'analysis' must address more precisely formulated questions regarding the organisation of behaviour. These questions, such as those regarding long-term dynamics, are centred around the suitability of an evolved circuit for engineering applications.

The successful action of a circuit can be considered as a property of the *interface* between its inner mechanisms and the external environment (Simon, 1996): the inner has been adapted so that the behaviour at the interface satisfies the specification. Observations at the interface (for example, at input and output connections) during normal circuit operation may reveal little about the inner mechanisms, but instead will largely reflect the demands of the specification. Analysis therefore requires internal probing, and/or observation under abnormal conditions, either internal or external.

There are surprisingly many tactics that can be used to piece together an analysis:

> **Probing and abnormal conditions**, discussed above, are the core of analysis based on experimentation. Examples of abnormal conditions include manipulation of the input signals; varying temperature or power-supply voltage; replacing parts of the circuit with alternative or non-functional pieces; injecting externally generated signals at internal points. Monitoring an internal voltage always has some side-effect, often placing a reactive load at the probing point. This may have negligible consequences, but potentially perturbs the measurement, or even stops the circuit from working altogether. Probing internal signals of

[1] Weighted Stochastic Calculus of Communicating Systems

a circuit implemented on a VLSI chip can require routing extra connections to reach the external pins of the device, with a danger of further disrupting the circuit under study.

Mathematical techniques, and the theory of analogue and digital circuits developed for conventional design, are preferable for their rigour and generality. If a whole unconventional evolved circuit is mathematically intractable, there may still be parts of it which yield.

Simulation of a circuit allows rapid and extensive interactive exploration. As we shall see, circuits evolved not in simulation, but using real reconfigurable hardware, may rely on detailed hardware properties not easily modelled in a simulation. Attempts at simulation can at least help to clarify the extent of this dependence.

Synthesis: a circuit can be implemented using alternative electronic devices. For example, a circuit evolved on a single VLSI reconfigurable chip, might then be constructed out of a number of hard-wired small-scale chips. This provides easy access for probing and manipulation of 'internal' signals, and again can clarify what aspects of the hardware are important to the circuit's operation.

Power consumption for the most common VLSI technology ('CMOS'), is related to the rate of change of the internal voltages. After removing any power-supply smoothing capacitors, power consumption can be monitored with high temporal precision, while the circuit is exposed to test conditions.

Electromagnetic emissions, resulting from rapidly changing electrical signals, can sometimes be detected using a tuned radio receiver. Circuit activity within a chip, which might be difficult to monitor directly, can thus be roughly characterised.

Evolutionary history: The mechanism underlying a task-achieving behaviour may be more apparent soon after its evolutionary origin, rather than after evolution has refined it closely to match the specification. It may be possible to identify the innovation (perhaps caused by one or more mutations) giving rise to the behaviour's origin in an ancestor, and to relate this to means used by the final circuit to achieve the more polished performance.

Population diversity: Sometimes there can be several slightly different (but related) forms of high-fitness circuit in an evolutionary population, which can help to reveal the basic mechanisms used.

Although unconventional evolved circuits can seem dauntingly unfamiliar, with such a toolbox, the analyst is far from powerless.

3.3 Case Study

Some of the analytical techniques described above are now applied to Thompson's tone discriminator (Thompson, 1998a) — probably the most bizarre, mysterious, and unconventional unconstrained evolved circuit yet reported. The aim is two-fold. First, to explore the extent to which analysis can reveal the mechanisms underlying the operation of an exotic circuit evolved in VLSI (with few points available for direct probing by external measuring equipment). Second, how analysis may be able to abate some of the worries associated with employing very unusual evolved circuits in an engineering application.

3.3.1 Thompson's Tone Discriminator

The tone discriminator experiment was described briefly in section 2.1.2, and the reader is referred to (Thompson, 1998a) for full experimental details. The circuit was evolved intrinsically on a small corner of a Xilinx XC6216 FPGA. Its task was to distinguish between two audio tones of 1kHz and 10kHz frequency, producing a steady high output when one tone was present at the input, and a steady low output for the other.

Each genotype contained configuration settings for a 10×10 corner of the FPGA's array of cells. Having configured and enabled the FPGA, candidate configurations were evaluated by injecting a series of half-second tone bursts — five each of 1kHz and 10kHz, in random order — at the chosen input pin. Fitness was scored according to how well a configuration approximated the desired discrimination behaviour. The logic cells and multiplexers within the FPGA operate in a time-scale of nanoseconds, and one of the original motivations of the experiment was to see if the dynamics of the FPGA could be organised to give an orderly behaviour on a very different time-scale: the periods of the two tones to be distinguished are 1ms and 0.1ms. No clock or other off-chip resources were provided, by which the period of the input could be timed or filtered.

Evolution was allowed to exploit the capabilities of the FPGA as freely as possible. No constraints were placed on the circuit's structure or dynamical behaviour. Since no clock was provided, evolution was searching the space of continuous-time dynamical systems for which the FPGA can be configured. The experiment was successful, and the behaviour of the resulting circuit (figure 3.1) was near-optimal.

At this stage, Thompson had already completed a preliminary analysis. Only the cells which have a role in the circuit's operation are shown in figure 3.1. All of the cells not shown can be simultaneously 'clamped' without affecting the output. To clamp a cell, the configuration sent to the FPGA was changed so that the cell's function produced a constant output, and this sourced all four of the cell's output connections. Where a connection is shown passing through a clamped cell, all of the cell's attributes except for this connection could be clamped.

The circuit contains a number of continuous-time recurrent loops. Their presence, together with oscilloscope readings taken as evolution progressed, led Thompson to deduce that the circuit was exploiting continuous-time *analogue* properties of the VLSI medium. The FPGA chip is intended for digital designs. However, being made of real transistors, a logic cell is really an analogue component behaving in continuous time. These analogue cells have very high *gain* (amplification); for most inputs their output rapidly saturates either fully high or fully low. Consequently, if certain design rules are followed, then the system's behaviour can be abstracted to a *binary* description; a digital logic system. The basic digital design rule is that *recurrent* connections — paths through the circuit by which a component's output can (indirectly) later affect its own input — must only be allowed to operate at particular discrete instants. This gives the components time to saturate to their extreme (digital) states before their inputs change again. In *synchronous* design, this is done on the ticking of a global 'clock', whereas in asynchronous digital design more local co-ordination is used. With no constraints imposing this design rule, the output voltage of a cell can lie within the saturation extremes, significantly affecting the operation of other cells to which it is connected. When this occurs, the shape of the curve describing the cell's output voltage for a given input voltage — the 'transfer function' ((Horowitz & Hill, 1989),

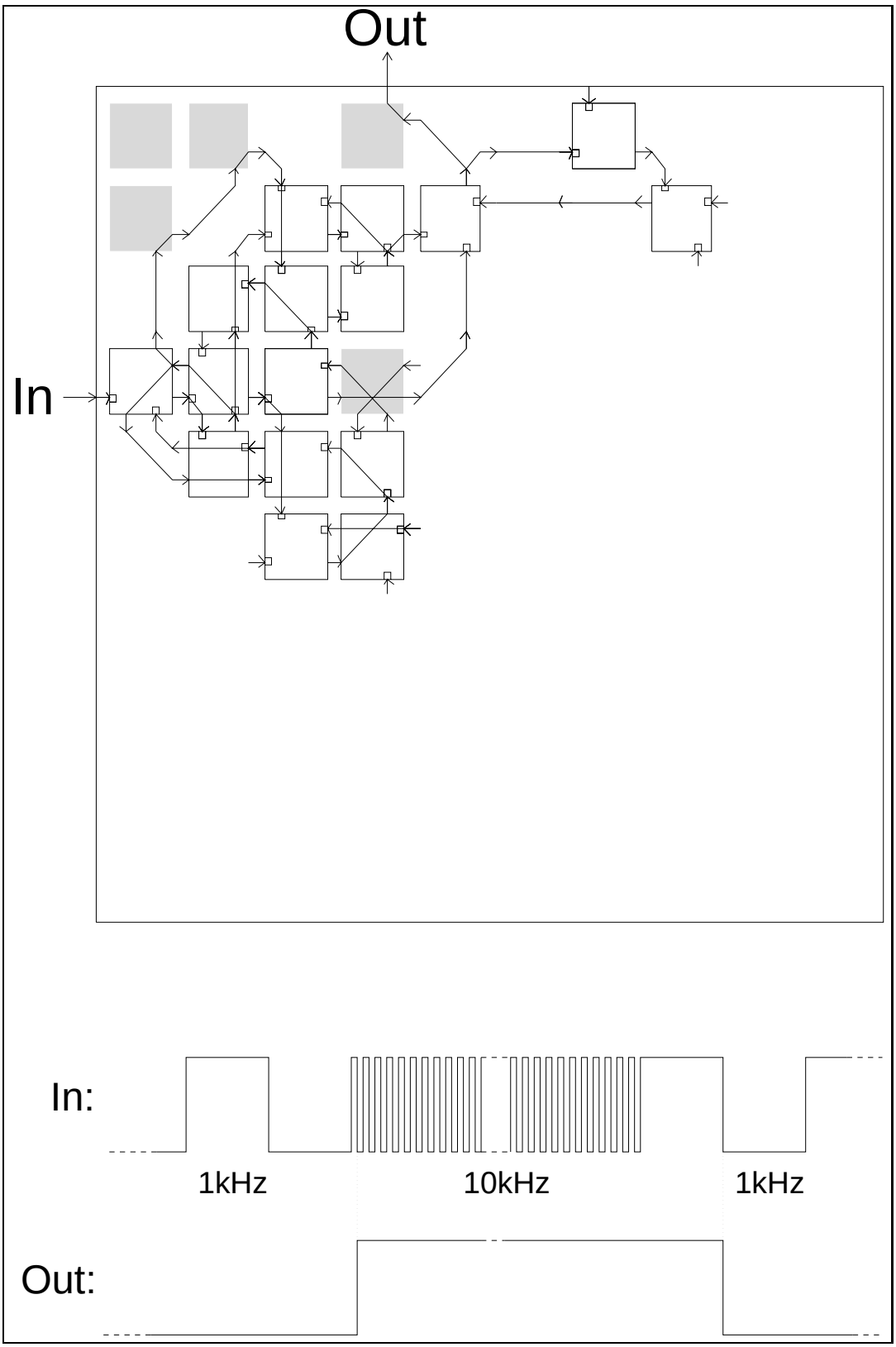


Figure 3.1: The circuit for analysis, after pruning. For simplicity, only the connections between the cells are shown, not their functions, which are also under evolutionary control. Of the signals arriving from its four neighbours, those used as inputs to a cell's function are marked with small boxes. Each of a cell's four outputs (to the four neighbours) can be driven either directly from one of the cell's inputs, or by the output of the cell's function.

p.7) — is dependent on physical properties of the semiconductor material, such as capacitance and inductance. Hence important mechanisms underlying the circuit's operation may rely on very subtle physical semiconductor characteristics. Indeed this is likely to be the case: Thompson's deduction was reinforced by observations of the circuit's operational temperature range, which is an order of magnitude smaller than would normally be expected of digital VLSI. Even within this range, the discrimination frequency varied with temperature. If the circuit was configured onto a different, but nominally identical FPGA chip (not used during evolution), then its performance was degraded. Finally, a number of cells (shaded grey in figure 3.1) cannot be clamped without degrading performance, even though their functional unit is not used as an input to any other cells. They seem to be influencing the rest of the circuit by some means other than the normal inter-cell routing. One of the aims of further analysis was to learn the influence of these 'grey cells'.

This was the extent to which the circuit had been analysed until the study documented here. One could only speculate as to the circuit's means of operation, so unusual are its structure and dynamics. Although the circuit was evolved to address specific academic questions and was never intended as a practical engineering application, its publication raised doubts as to whether practical circuits could be realised by the unconstrained approach to hardware evolution.

3.3.2 A Logical Analysis

If the tone discriminator's input frequency was gradually changed from 1kHz to 10kHz, the output (low at 1kHz) would begin rapidly to alternate between low and high, spending more time high as the frequency was increased, eventually staying steady high for frequencies near 10kHz. This binary behaviour of the output voltage suggested that at least *part* of the system could be understood in digital terms. By temporarily making the assumption that all of the FPGA cells were acting as Boolean logic gates in the normal way, the FPGA configuration could be drawn as the logic circuit diagram of figure 3.2. In the diagram, even numbers of inverters are cancelled out (the path from one functional cell to another is subject to as many as 8 inversions via the inter-cell routing). The inverters are shown explicitly in Part C, but in Parts A & B they are replaced by hexagons containing estimates of the time delays (in ns) incurred in the path. Hexagons are also shown within each gate, indicating its estimated input⇒output propagation delay. The estimates are based on the maximum values specified by the manufacturer (Xilinx, Inc., 1997).

The logic circuit diagram shows several continuous-time recurrent loops (breaking the digital design rule described above), so the system's behaviour is unlikely to be fully captured by this Boolean level of abstraction. However, it is easy to show that whenever the input is a logic **1**, all of the gates in parts A & B will output predictable, constant logic values. They will reach this static state within about 20ns of the input becoming **1**, and remain there until it returns to **0**. This occurs because if one input of an AND gate is **0**, then the output will be **0** irrespective of the second input; similarly the output of an OR gate will be **1** as long as any one of its inputs is **1**. Such functions — where one input can assume a value which makes the other inputs irrelevant — are called 'canalysing' functions (Kauffman, 1993). It so happens that whenever the circuit's input is **1**, all of the recurrent loops in Parts A & B are broken by a cascade of canalysing functions. Once this happens, A and B satisfy the digital design rules, and all of their gates deterministically switch to fixed, static logic values, remaining constant until the input next goes to **0**.

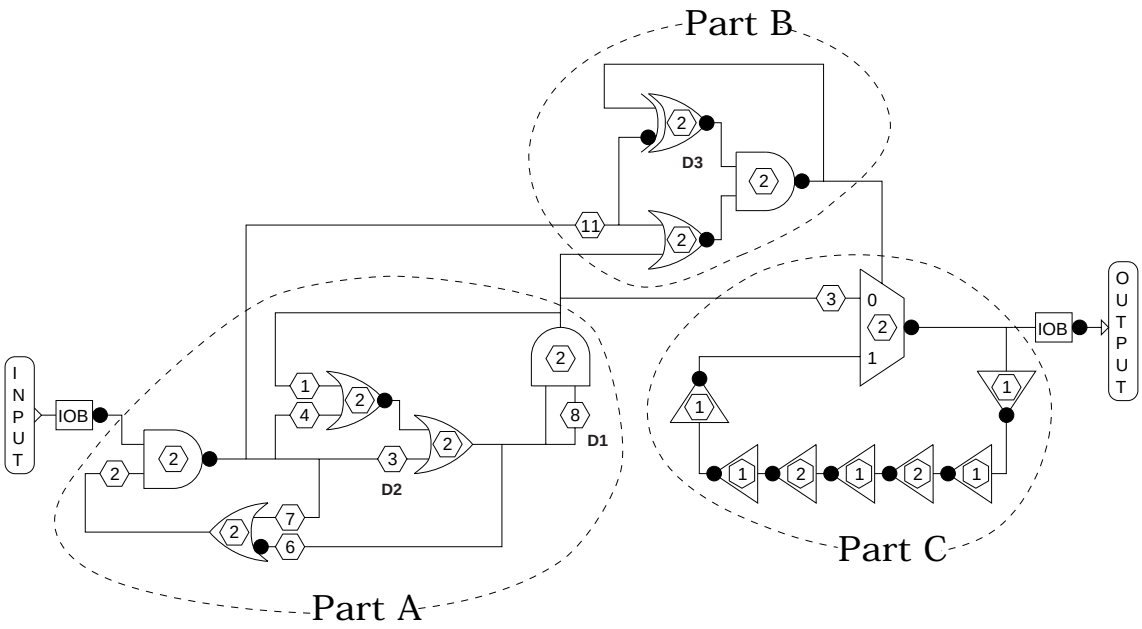


Figure 3.2: The logic circuit representation. The numbers in hexagons are rough estimates of time delays (in ns), based on the maximum values specified by the manufacturer. 'IOB' refers to the inverting Input/Output Blocks that interface the edges of the reconfigurable array to the physical pins of the chip.

Part C of the circuit is based around a 2:1 multiplexer. When Part B is in the static state, the multiplexer selects the input marked '1' to be its output. This input comes from the multiplexer's own output via an even number of inversions, resulting in no net logic inversion but a time delay of around 9ns. Under certain conditions, it is possible for such a loop to oscillate (at least transiently), but the most stable condition is for it to settle down to a constant logic state. The output of the whole circuit is taken from this loop. As this Part C loop provides the only possibility for circuit activity during a high input, the next step in the analysis was to inspect the output very carefully while applying test inputs.

We had already observed that the output only ever changes state (high⇒low or low⇒high) on the *falling* edge of the input waveform (figure 3.1). Although never required to do so during evolution, we discovered that the output also responds correctly to the width of *single* high-going pulses. Figure 3.3 shows a low⇒high output transition occurring after a short pulse; further short pulses leave the output high, but a single long pulse will switch it back to the low state. The output assumes the appropriate level within 200ns after the falling edge of the input. The circuit does not respond to the width of low-going pulses, and recognises a high-going pulse delimited by as little as 200ns of low input at each end of the pulse. The output is perfectly steady at logic **1** or **0**, except for a brief oscillation during the 200ns 'decision time' which either dies away or results in a state change.

This is very surprising. During the single high-going pulse, we know that parts A and B of the circuit are almost immediately 'reset' to a static state, as shown in figure 3.3. Observations at the output showed that Part C is also in a static state during the pulse. Yet somehow, within 200ns of the end of the pulse, the circuit 'knows' how long it was, despite being completely inactive during it!

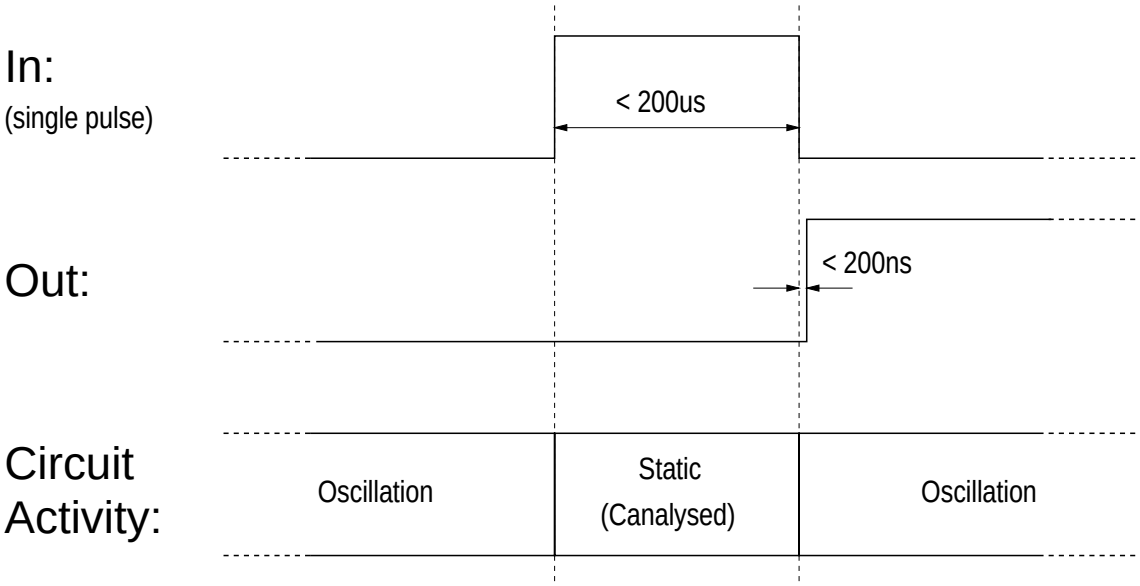


Figure 3.3: The response to a single high-going pulse.

3.3.3 Assessing Alternative Possibilities to a Static Timing Mechanism

The implausibility of a canalysed logic circuit containing some mechanism capable of timing such relatively long pulses prompted alternative explanations. Perhaps the behaviour of the circuit was dependent on some other quality of the input than its pulse width, for example the falling edge of a 500μs pulse could have slightly different properties to that of a 50μs pulse. Perhaps some characteristic not encompassed by the logic diagram led to some undetected circuit activity during a high pulse.

The first of these possibilities was unlikely, since the circuit analysis was performed with a completely different signal generator than had been used to evolve it, and in any case the FPGA's IOB (figure 3.2) contains circuitry to ensure analogue input voltages quickly saturate to digital levels. Furthermore, an opto-isolator was placed between the signal generator and the FPGA's input pin during analysis, thereby removing parasitic influences such as cable impedance. To alleviate concerns that the opto-isolator was behaving differently for different pulse-widths, it was bypassed, and the circuit evaluated with both sine and square-wave tones. In every case it operated flawlessly.

The second possibility of undetected circuit activity was investigated through many separate types of observation, all of which agree that the circuit is inactive during the pulse. Removing all smoothing capacitors, powering the circuit with a battery, and monitoring the power lines confirmed that power consumption returns to steady, quiescent levels during the pulse. Many of the internal signals were (one at a time) routed to an external pin and monitored. Sometimes this extra routing altered or destroyed the circuit's behaviour, but we have observed at least one signal from each recurrent loop while the circuit was successfully discriminating pulse-widths, and there was never activity during the pulse. We were concerned that perhaps, because of the way the gates are implemented on the FPGA, it was possible that *glitches* (very short-duration pulses) were able to circulate in the circuit while the logic-analysis predicts it should be static; possibly these glitches could be so short as to be unobservable when routed to an external pin. Hence, we hand-designed

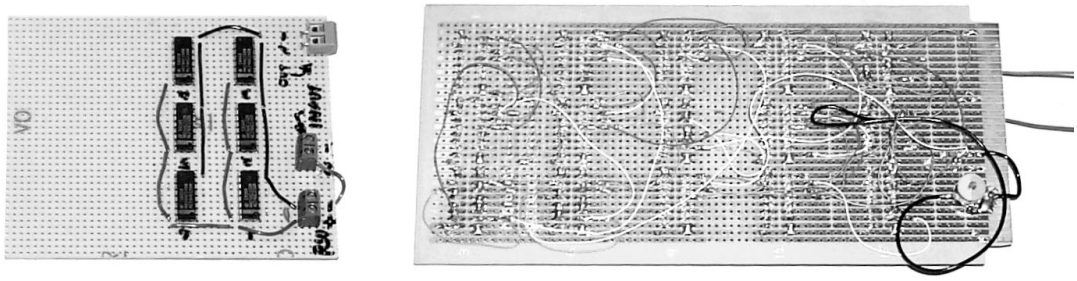


Figure 3.4: Hand-built versions of Parts A (right) and B & C (left) of the tone-discriminator, constructed so that internal signals could be directly monitored.

a high-speed 'glitch-catching' circuit (basically a flip-flop) as a configuration of two FPGA cells. Glitches sufficiently strong to circulate for tens of microseconds could be expected to trigger the glitch-catcher, but it detected no activity in any of the recurrent loops during an input pulse.

We performed a digital simulation of the circuit using PSPICE A/D (OrCad, 1997), extensively exploring variations in time-delays and parasitic capacitances. The simulated circuit never reproduced the FPGA configuration's successful behaviour, but did corroborate that the transient as the circuit enters its static state at the beginning of an input pulse is just a few tens of nanoseconds, in agreement with experimental measurements of internal FPGA signals, and according with the logic analysis. We then built the circuit out of separate CMOS multiplexer chips (figure 3.4), mimicking the way that the gates are actually implemented by multiplexers on the FPGA, and also modelling the relative time-delays. Again, this circuit did not work successfully, and — despite our best efforts — never produced any internal activity during an input pulse.

We then went back to find the first circuit during the evolutionary run which responded at all to input frequency. Its behaviour is shown in Figure 3.5. During a pulse, the output is steady low. After the pulse, the output oscillates at one of two different frequencies, depending on how long the preceding pulse was. These oscillations are stable and long-lasting. The differences are minor between this circuit and its immediate evolutionary predecessor (which displays no discrimination, always oscillating at the lower of the two frequencies). In fact, there are no differences at all in the logic circuit diagrams; the changes do things like alter where a cell's function takes an *unused* input from. This lends further support that circulating glitches are not the key: there was no change to the implementation of the recurrent loops.

3.3.4 Simulating the Circuit's Operational Dynamics

The circuit does indeed appear to contain some timing mechanism that operates during the static state. Observations of internal nodes of Part A and of Part C (multiplexer input 0) showed that on being released from the canalysed stable state, the circuit enters one of two different oscillation modes reminiscent of figure 3.5. The mode entered depends on the state of the timing mechanism. We re-routed the inter-cell connections such that Parts B & C were effectively disconnected and observed similar behaviour; bistable oscillation whose mode depended on the input frequency. The timing mechanism must therefore be located in Part A. Neither the simulation nor the hand-built version could reveal anything about this mechanism, but it was possible to simulate the circuit

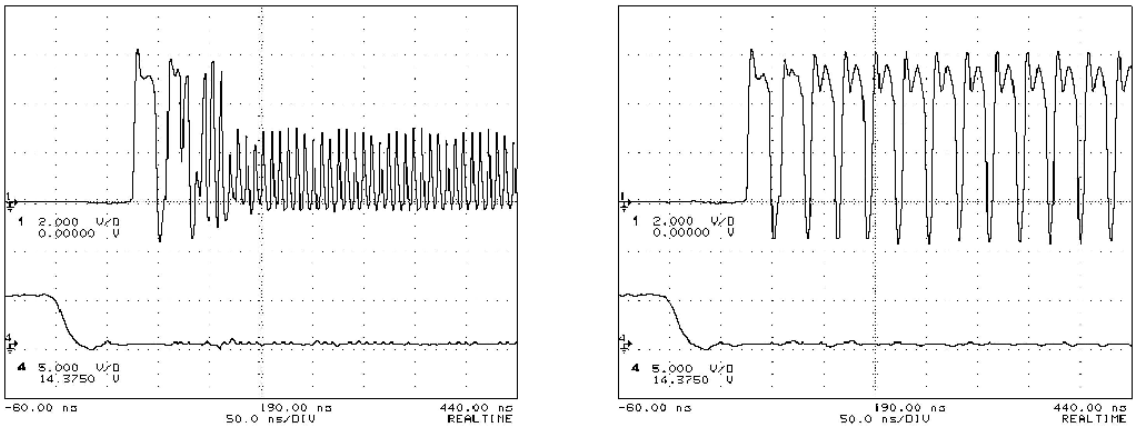


Figure 3.5: Behaviour of the first frequency-discriminating ancestor. The upper waveform is the output, and the lower the input; we see the behaviour immediately after the falling edge of a single input pulse. *left* long pulse, *right* short pulse. The input to the FPGA is actually delayed $\sim$ 40ns by an intervening buffer, relative to the wave seen here.

dynamics that followed its operation. Parts B & C were modelled in simulation and injected with square-wave oscillations of identical frequencies to those observed in the real circuit at the two 'outputs' of Part A (see figure 3.2). The simulation showed how Parts B and C conditioned the oscillations to derive a steady output according to the pulse width. The entire circuit was then simulated to discover how stable these two modes were once entered, and the conditions for which a given mode was assumed.

Adjusting all but one of the hexagonal delays in simulation made only gradual changes to the circuit dynamics. However, the 8ns delay separating the two inputs of the AND gate of Part A was critical. Adjusting this delay in simulation from 4ns to 12ns revealed three distinct modes of oscillation over the delay range. Each mode is separated from the other by delays of only a few tens of picoseconds, and two of the three have virtually identical frequencies to those observed in the actual circuit. It seemed likely that the mysterious static timing mechanism determined the oscillation mode by influencing this delay. If this were true, it should be possible to reproduce the rest of the circuit's dynamics entirely in a digital simulation. This proved to be the case. Almost identical behaviour to the physical circuit was achieved by adjusting only two other delays, marked D2 & D3 on the schematic[2]. Figures 3.7 and 3.6 shows real and simulated waveforms for comparison. The waveforms are those observed at the multiplexer of Part C; oscilloscope traces of the physical circuit for input tones of 1kHz and 10 kHz, and simulated waveforms for D1 delays of 8.1 and 8.2 ns. While not conclusive (the digital simulation does not take into account the FPGA's analogue properties, and the real circuit may require hysteresis to maintain the mode of oscillation if the value of D1 is not constant), there is strong evidence that the simulation reflects the true operation of the circuit; Delays D1-3 are set within 10% to those estimated in the original circuit, and the other delays can be varied by the same degree, with no major change in circuit dynamics. All the simulated waveforms at other internal nodes concur with those we had observed in the physical circuit. Appendix B contains plots of these waveforms, the three oscillation modes

[2]Specifically, D3 influences the settling time before the circuit output reaches a stable state. Altering D2 introduces additional oscillation modes appearing between the three mentioned, and which can be discerned over very small ranges (around 0.1ns) of D1.

achieved by varying D1, and the schematic used for the simulation.

3.3.5 Summary of the Tone-Discriminator's Operation

We see bistable oscillations at internal nodes of part A of the final circuit. On being released from the canalysed stable state, the difference in the oscillatory behaviour in part A is used by parts B & C to produce a steady near-optimal output corresponding to the input pulse width. There is some initial state of the part A dynamics which is determined by the input pulse length. This initial state does not arise from any circuit activity in the normal sense: the circuit over the entire array of cells was stable and static during the pulse. We still do not understand fully how it works: the core of the timing mechanism is a subtle property of the VLSI medium. We have ruled out most possibilities: circuit activity (including glitch-transients and beat-frequencies), metastability (Marino, 1981), and thermal time-constants from self-heating. One guess is that the change in initial state results from some slow charge/discharge of an unknown parasitic capacitance during the pulse, and that this affects the delay D1, but we cannot yet be sure. Whatever this small effect, we understand that it is amplified by alterations in bistable and transient dynamics of oscillatory loops, and in detail how this is used by parts B & C. The time delays D1, and on the connections from A to B are crucial to deriving an orderly output. This explains the influence of the 'grey cells', which are all immediately adjacent to (or part of) the path of these signals. Varying the time delays in the simulation produces a similar result to interfering with the grey cells. The loop of part C serves to maintain a constant and steady output even while the rest of the circuit oscillates, but immediately after an input pulse it has subtle transient dynamics resulting from those of A & B.

3.4 Discussion

While the analysis of this bizarre circuit has not yielded a comprehensive understanding of its operation, we have learned enough to increase its utility as an engineering application. On the key question of long-term consistency of behaviour, we know that the entire FPGA circuitry is strongly reset to a deterministic stable logic state for every high half-cycle of the input waveform. Long-term pathologies arising from the circuit's own dynamics are therefore highly unlikely. We also know from varying the propagation delays in simulation, and by observations of the physical circuit and its ancestors, that the two oscillation modes are stable.

Varying the temperature alters the circuit's discrimination frequency, rather than causing catastrophic failure. Hence the principal temperature-dependent elements are those in Part A which determine the threshold at which each of the two oscillation modes occur. We know that each mode can be produced by a minuscule variation in the delay D1. Therefore, it should be possible to increase the circuit's operational temperature range for an engineering application by adding some circuitry (for example a small RC network) to make larger variations in this delay according to the input pulse width. However, it must be acknowledged that the addition of external components would diminish this particular circuit's size advantage over a conventionally designed one.

It is unclear whether any new design tricks are to be gleaned from the circuit. It certainly employs a number of novel techniques — the use of bistability to amplify a subtle physical effect, and the 'sample and hold' subcircuit of Part C. However there already exist a large number of

amplification and sampling techniques. There would need to be a significant benefit in selecting the methods used here in preference to those already known before we can claim that any new design tricks have been discovered.

The results of this analysis raise questions about the conditions required to make a circuit 'evolvable'. For example, the representation and unconstrained approach adopted by Thompson allowed a configuration to be found for which several distinct asynchronous modes of oscillation were available through minor adjustments of the circuit layout, and these were crucial in obtaining the final solution. Would evolution be more or less likely to discover such a configuration with a different genotype⇒phenotype mapping, or if a synchronous clock were provided?

Analysis of highly unusual evolved circuits enhances their utility, but requires novel approaches. There are numerous tactics that can be used to piece together answers to analysis questions, even for seemingly impenetrable circuits. Many of these techniques were applied here to one of the most advanced unconventional circuits yet produced, with good results. However, the analysis took a great deal of time and specialist equipment, and many questions still remain regarding the circuit's operation and the evolutionary dynamics that produced it. The next chapter proposes and evaluates a way to tackle these and further issues.

Summary of Chapter 3

On removing the constraints on circuit structure and dynamics normally needed to allow design abstraction and to ensure that nonbehavioural requirements are satisfied, artificial evolution is free to explore the entire repertoire of behaviours that the physical hardware can manifest. However, circuits evolved using the unconstrained approach cannot necessarily be analysed using established techniques, since they may have no clear functional decomposition, and may rely heavily on subtle physical properties inherent in the medium. If a circuit's principles of operation are revealed through analysis, its usefulness as an engineering product is increased. Where its failure modes are understood, steps can be taken to enclose it within an error-recovery framework, or to redesign portions heavily susceptible to temperature or fabrication variations. Even if the circuit's operation is not comprehensively understood, novel strategies discerned through analysis can be incorporated into the electronic engineer's library of building blocks. The following analytical tactics are suggested: Probing and imposing abnormal conditions, mathematical techniques, simulation, synthesis using different hardware, monitoring of power consumption and electromagnetic emissions, use of evolutionary history and population diversity.

The tactics are applied to a well known, but particularly mysterious evolved circuit; Thompson's tone discriminator. The basic elements constituent to this circuit's operation had already been revealed through clamping their function to a constant value. Through applying a variety of input signals, it was discovered that the circuit satisfied its behavioural requirements by measuring the length of single, high-going input pulses. A boolean description with time delays corresponding to estimated physical propagation delays was then constructed that suggested that the circuit was inactive (canalysed) whenever the input was high, even though this was the crucial period at which the discrimination was performed. This was confirmed by a series of experiments, which included simulation, synthesis using discrete hardware, temporal monitoring of power consumption and of the signals present at internal nodes where possible, and constructing an *in-situ* glitch

detector. During a low-going pulse, bistable oscillations were observed at internal nodes, whose mode depended on the input frequency. Through applying the techniques above, it was possible to discern those parts of the circuit that conditioned the bistable oscillations to produce a constant high or low output. The subcircuit suspected of containing the discrimination mechanism was unchanged since the first frequency discriminating ancestor.

The data produced by this analysis made it possible to construct an accurate digital simulation model whose internal behaviour resembled closely that of the physical circuit. In the simulation, a given oscillation mode could be induced by altering one of the delays by just 100ps. It is conjectured that some subtle physical property is affected sufficiently during a high going-input to alter this delay when the circuit enters its oscillation phase. The circuit's usefulness is improved as a result of this analysis. We know that long-term pathologies arising from the circuit's own dynamics are highly unlikely. We also know what steps to take to increase the circuit's operational temperature range. However, the core of the timing mechanism remains a mystery.

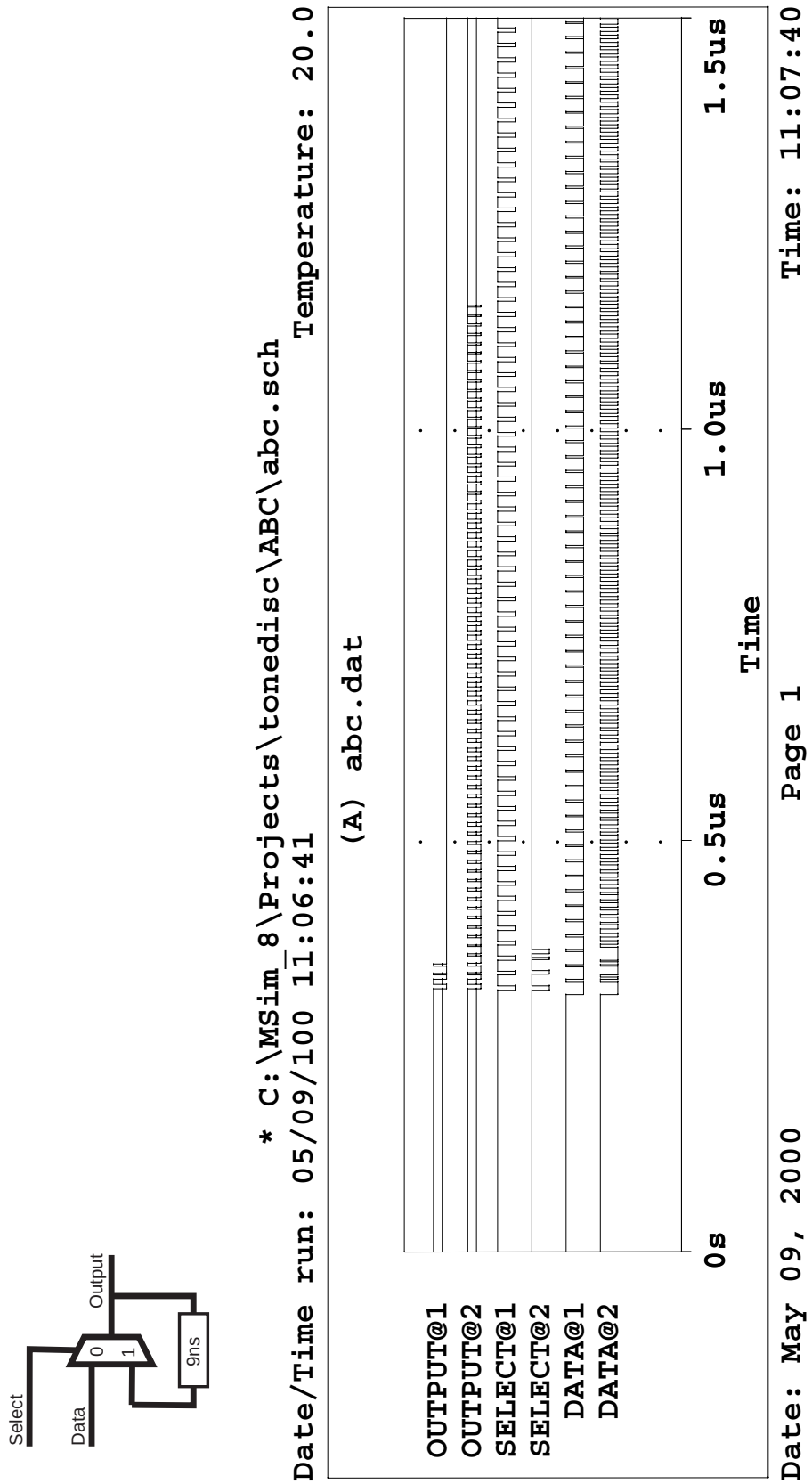


Figure 3.6: Simulated waveforms present at the multiplexer, Part C, for delays D1 of 8.1ns (1) and 8.2ns (2). These delays correspond to the two oscillation modes produced for 1kHz and 10kHz tones, respectively.

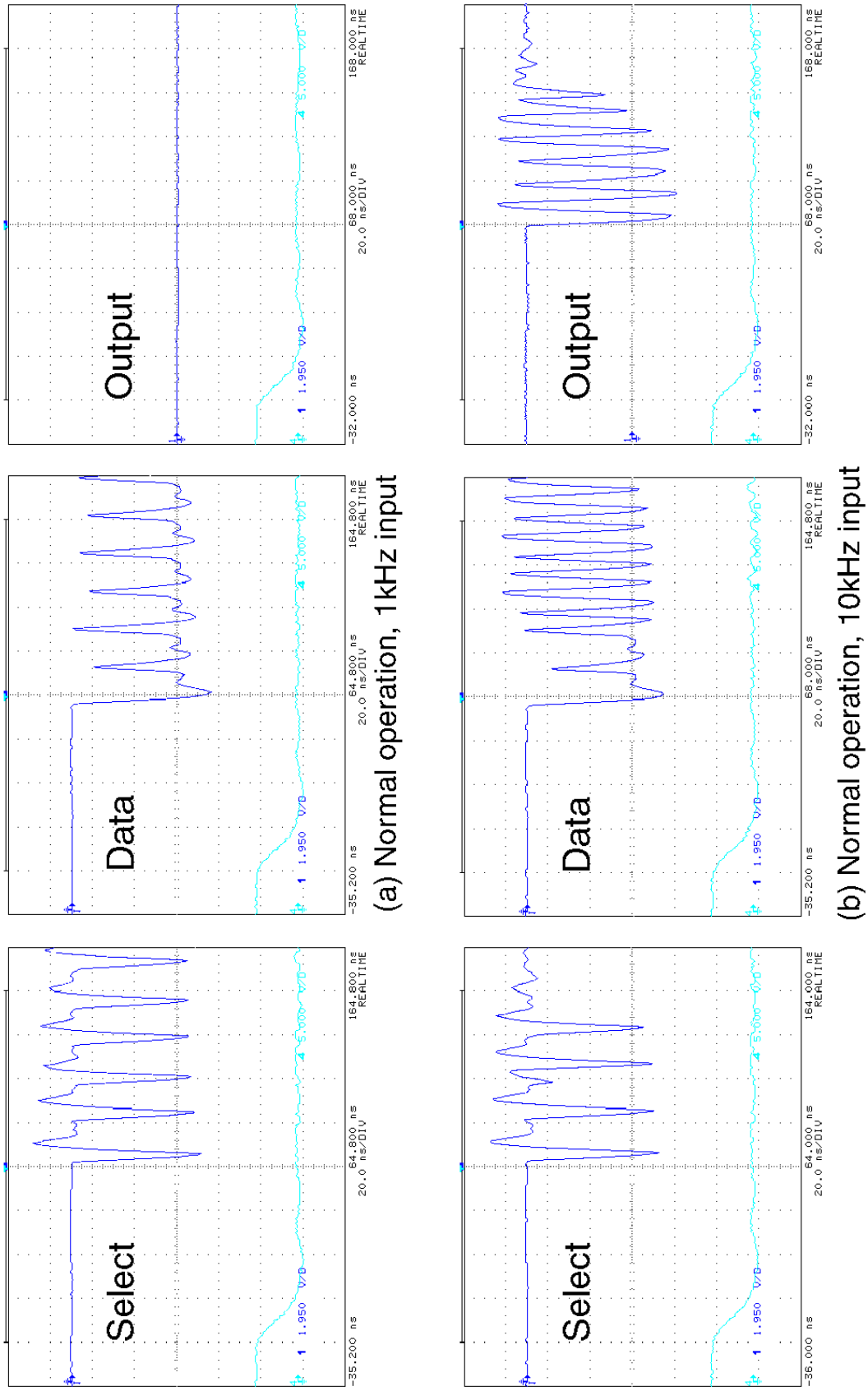


Figure 3.7: Waveforms present at the multiplexer, Part C, during normal operation. **above:** 1kHz tone, **below:** 10kHz tone. The lower traces in the figures are the input to the FPGA, which is delayed $\sim$ 40ns by an intervening buffer, relative to the wave seen here.

Chapter 4

An Original Research Tool for the Study of Hardware Evolution

---

Some of the drawbacks of FPGAs as an evolutionary medium for intrinsic HE research are now discussed, and a discrete test platform designed specifically to tackle them is presented. The platform — the 'Evolvable Motherboard' — was used for all of the evolutionary experiments documented henceforth. A description of its architecture and factors governing its design is followed by a simple experiment exemplifying its use. When first reported (Layzell, 1998a), this was the first case of intrinsic hardware evolution at the transistor level.

4.1 The Need for a General Evolvable Architecture

We have seen in section 2.3 that reconfigurable devices suitable for intrinsic hardware evolution abound, and continue to be developed. The majority of these devices are implemented in VLSI, and require considerable resources to produce. Hence, commercial FPGA chips remain the first choice for many groups. The previous chapter highlighted one difficulty associated with analysing evolved circuits in VLSI; the only way to access internal nodes with measuring equipment is to change the circuit's configuration, routing the node under inspection to an external pin. This is a time-consuming process and prone to errors. The routing circuitry places additional reactive loads in the signal path, potentially altering its unloaded behaviour, and thereby affecting the measurement. In extreme cases the extra loading sufficiently alters the node's physical characteristics to cause the circuit to fail. Other parts of the circuit lying adjacent to the new route may also be affected. If analysis is expected to form a principal phase of evolutionary circuit design, this difficulty may be addressed in advance. For example, a representation could be conceived such that nodes are connected at all stages during evolution to cells whose sole purpose is external routing. However, this is only a partial measure; successive stages of buffering and inversion contained in the route would still condition the signal. By the time it reached an external pin, its waveform and phase would be altered. Analysis is further restricted when evolving on current FPGAs which have closed architectures (see section 2.3). For example, it was only possible to deduce the influence of the tone discriminator's grey cells without probing, because an open-architecture FPGA was used, and their geographical location was known.

The difficulty of monitoring internal nodes is not the only issue associated with using commercial FPGAs as a medium for HE research. First, there is very little choice over the type of basic configurable element employed in commercial devices. Nearly all use boolean functions implemented as look-up tables or multiplexers. This is particularly restrictive if analogue applications are sought. With the demise of the Motorola FPAA (Bratt & Macbeth, 1996), there is at present only one reconfigurable analogue device commercially available, the Zetex TRAC020 chip (Zetex plc, 1996), for which evolved circuits have been reported (Flockton & Sheehan, 1998; Ozsvald, 1998). However, we do not yet know exactly what the most appropriate basic element for hardware evolution might be. Depending on the task it could be extremely basic, such as a transistor; some higher level functional unit; or combinations of different components including passive resistors or capacitors. Second, the interconnection highways between circuit elements in FPGAs are designed with the conventional, modular circuit design methodology in mind. Routing resources are tailored to CAD Automatic Placement and Routing (APR) tools for efficient cell usage, minimum longest-path delays, and other physical considerations such as fanout[1] (Xilinx, Incorporated, 1999). Usually, they consist of nearest-neighbour connections together with a hierarchy of semi-local and global buses. The architecture can be effectively altered by selecting a genotype⇒phenotype mapping which excludes some of the routing switches. This *restricts* the set of available interconnections; the set cannot be expanded if the switches do not exist in hardware. We don't currently know whether FPGA interconnection systems are the most appropriate for HE. Evolution may be better able to exploit a more comprehensive interconnection system (or one that can be restricted in a different way). Questions regarding the suitability of a given interconnection architecture for HE generally centre on its impact on *evolvability* (see section 2.2.6), but there are other factors, for example, evolving for fault-tolerance (Ortega & Tyrrell, 1999). Without a fully comprehensive interconnection system available for empirical tests, there is little consensus on what comprises a system 'ideal' for HE. Finally, we have seen in section 2.3 that preventing configurations that can damage or destroy commercial FPGAs can add a severe computational overhead to the evolutionary design process, and/or restrict the set of interconnections between basic elements.

These issues do not by any means exclude the use of commercial FPGAs from valuable HE research, as the literature clearly testifies. Nonetheless, if we are to investigate all the possible benefits of hardware evolution as an engineering tool, we must tackle questions such as those concerning the most appropriate basic elements and interconnection architectures for a given task. The more tools available to facilitate analysis within a practical timescale, the more we can learn about the types of circuits a given strategy can produce, and the qualities both strategy and circuits can be expected to possess. To this end, I propose a general-purpose architecture for which virtually any type of component can be used as the basic element, with each element pin externally available for monitoring by test equipment, and with a fully comprehensive interconnection architecture. This architecture is intended not for use as an evolutionary medium for practical applications or large systems, but as a low-cost research tool to address the issues described above, and others discussed in section 2.2.2.

[1] Fanout refers to the maximum quantity of inputs within the same logic family that a given digital device output can drive, and still perform within specifications.

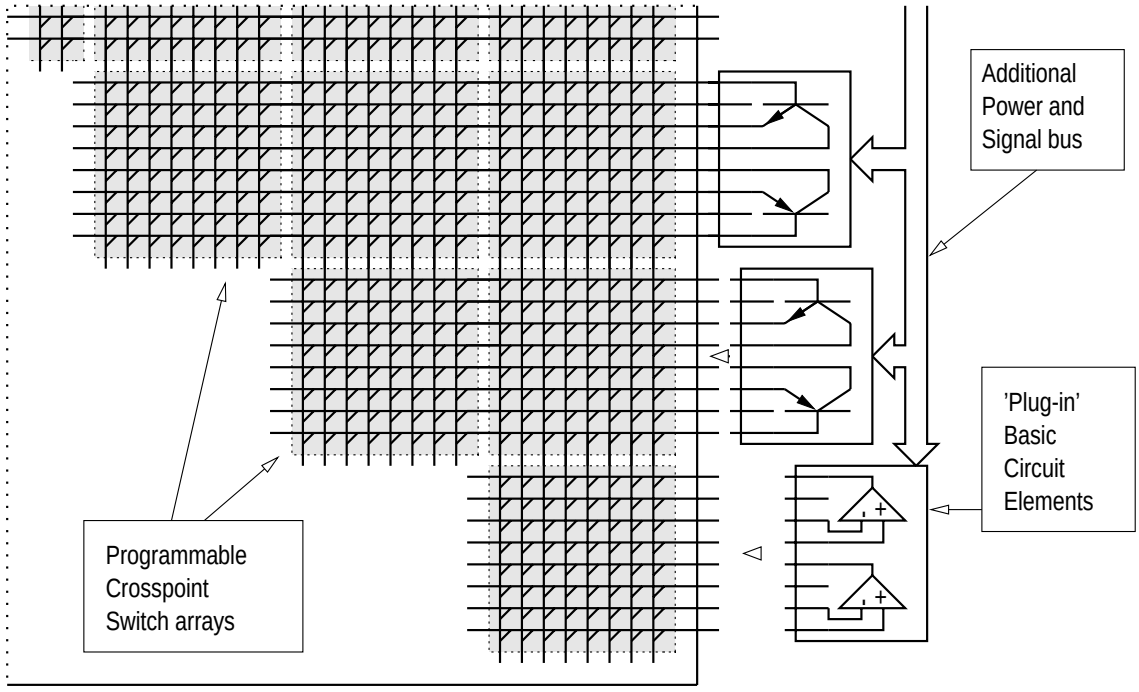


Figure 4.1: Simplified schematic of part of the Evolvable Motherboard.

4.2 The Evolvable Motherboard - Overview

Conceptually, the general-purpose architecture outlined above is a discrete motherboard containing a comprehensive interconnection architecture capable of routing both digital and analogue signals. The motherboard does not itself contain any basic elements. Instead, ports are provided into which daughterboards containing them can be plugged. Separating the basic elements from the interconnection architecture in this way allows a variety of different types of basic element to be evaluated, and direct access to their pins for measurement. I have named this system the 'Evolvable Motherboard' (EM).

4.2.1 General Architecture

Figure 4.1 is a simplified diagram of one corner of the EM and the plug-in daughter-boards. The short diagonal lines represent digitally-controlled analogue switches which allow row/column interconnection. The minimum number s of switches required to ensure all possible combinations of interconnections between basic elements is equal to the number of different pairs of the total of their pins:

$$s = \frac{n(n-1)}{2} \tag{4.1}$$

where n is equal to the total number of basic element pins.

Equation 4.1 can be realised using a triangular matrix of switches situated on a wiring grid of n rows $\times$ n columns. This switch matrix is approximated on the EM using analogue crosspoint switch devices. The configuration pins and chip-select lines of these chips are routed to a configuration port on the EM so that any switch may be individually configured by a host computer with a simple interface card. While the majority of possible switch configurations will effectively short out the wiring — resulting in useless circuits — the capability for individual switch configuration

combined with their arrangement allows almost any interconnection architecture (for example the local and global bus system used commonly by FPGAs) to be investigated by the appropriate choice of genotype⇒phenotype mapping. Also, via software, the board can be subdivided (for example into 2 matrices) to allow mapping individuals to two sets of components, thereby allowing the impact of manufacturing tolerance to be assessed.

Each set of eight lines on the wiring grid terminates in a port, into which the daughterboards containing the desired basic elements for evolution can be plugged. Certain types of basic element require additional connections. For example, digital logic requires a single rail power supply; operational amplifiers require a dual-rail supply; digital potentiometers require a configuration line to program their value. For this reason, the ports must also provide access to general-purpose I/O lines and independent power lines, so that a combination of digital and analogue components may be used as basic elements.

A principal function of the motherboard is to allow any point of the circuit to be accessible for measurement. If evolution exploits the components allowed it in an unexpected way, it should be simpler to deduce the contributing properties than is the case for an FPGA. Note however that the analogue switches are themselves semiconductor devices, contained within integrated circuits. The analogue switches have resistance and capacitance, and are therefore expected to form an integral part of any circuit configured. This is unavoidable using present technology. The only configurable devices for which these characteristics may be considered negligible are electromechanical relays, whose size renders them impractical, and whose magnetic field would be liable to influence the operation of an evolved circuit in any case. Analysis is hindered if the switches themselves play an active role in an evolved circuit's operation. Should this circumstance arise, a circuit can be configured with various different switches to determine whether or not subtle properties of particular switches are indeed essential.

It will be clear to the reader from figure 4.1 and equation 4.1 that an architecture allowing all possible combinations of component wiring imposes a severe limit on the number of basic elements that can be incorporated in any practical implementation. Whereas an EM with 10 daughterboards would require 3160 programmable switches, a 100 daughterboard EM would need 319600, clearly infeasible to implement using discrete devices.

A number of alternative, O(n) interconnection strategies were investigated during conception, but their adoption would mean sacrificing the EM's generality, giving an increased risk that any conclusions drawn from evolutionary runs would only be valid for other EM users (This risk already exists due to the electrical characteristics of the programmable switches as described above). Furthermore, the alternative designs were heavily based on the structure of existing circuits and conventional design maxims. I have repeatedly reiterated the danger of prejudicing the design of evolutionary electronic systems with knowledge from the conventional domain (see section 2.2.6), and since no reliable method could be found of conceiving an architecture through the exclusive application of information gained from previous HE experiments, the alternatives were not adopted.

With such a small quantity of basic elements available for evolutionary runs, it is difficult to conceive of a means by which the EM could be used to investigate scalability. But the other issues need not require evolved circuits of high component count for their exploration, and I have already

warned against using large, complex circuits to explore fundamental issues of HE (section 1.1). However, there is a fear that the restrictions imposed on the basic element count by the EM's architecture are *too* severe to produce any meaningful data concerning these issues. Fortunately, the following chapters will show that this fear is ungrounded.

4.2.2 Preventing Destructive Configurations

The interconnection architecture permits configurations which may damage the EM. An obvious example of this is the power lines becoming shorted together. Individual crosspoint switches implemented in VLSI can only handle a relatively small current (typically around 30mA) and if this current is exceeded, the switch could be destroyed. One way of preventing this would be to check for such configurations in software, but there are so many that the risk of omitting some is extremely high. Care would also have to be taken when designing the software verification to guarantee safety under unusual circumstances, for example if the software or operating system crashes during a long run. There is little value in putting a current limiter on the power supply — many circuits are likely greatly to exceed 30mA current drain without any single switch having to cope with more than this value — a great many 'legal' circuits would trip the current limiter, and not be evaluated.

Self-destruction can be prevented by limiting the power supply voltage. The power supply can be arranged such that any path leading to direct shorts requires at least two programmable switches. When 'closed', the switches have a small resistance whose approximate value is known. Using Ohm's law, it can be shown that if the power supply does not exceed a certain maximum voltage, then the testbed itself cannot be blown up. However, the problem does not end there: Certain devices that could be used as basic elements may be destroyed with much smaller currents. In this case, software verification is practical, since unforeseen configurations would not be fatal; blown plug-in components are easily replaced. Alternatively, additional resistors can be incorporated into the daughterboards.

4.3 Practical Implementations of the Evolvable Motherboard

4.3.1 A Physical EM

Physical EMs, used for all the experiments detailed in the coming chapters, were constructed with a 48×48 wiring grid, admitting up to six daughterboards. The grid requires approximately 1500 switches, giving a search space of 10^{420} possible configurations. The complete circuit schematics are shown in figures 4.2 and 4.3. Twelve Harris CD22M3494SQ (Intersil Corp., 1997)[2] 8×16 analogue crosspoint switch array chips (U0 to U11) are arranged to approximate the triangular switch matrix. DB1 to DB6 are 16-way connectors into which the daughterboards are plugged. Eight lines of each connector supply the daughterboards with power and I/O: Five power-supply/earth lines are provided, labelled $\{$VDD,VSS$\}$ and $\{\pm$VCC, VEE$\}$ in figure 4.2. This accommodates a single-ended supply and an independent dual-rail supply, to allow for combinations of digital and analogue basic elements. The three general purpose control lines may be used to provide configuration for programmable basic elements, or biasing for analogue ones. The remaining eight lines

[2] Harris have since become part of Intersil Corporation. Therefore, the Intersil CD22M3494 datasheet is cited in this chapter. Intersil's version of the chip is identical.

of each daughterboard connector are wired directly to the switch matrix, represented in figure 4.2 by octal buses V BUS 0 to V BUS 5 (Vertical), and H BUS 0 to H BUS 5 (Horizontal). U0 to U11 are staggered vertically over the horizontal buses to approximate most closely the triangular matrix of figure 4.1. For example, U1 and U7 both connect to buses H BUS 2 and H BUS 3, while U4 and U9 both connect to H BUS 3 and H BUS 4. The staggering leaves some redundant 8×8 switch blocks in U3, U8, and U11, which are routed to H BUS 6. Connectors V1 to V5 and H1 to H4 are arranged on the switch matrix to provide power and I/O to an evolving circuit, and to allow several EMs to be daisy-chained together, should the need arise[3].

Each crosspoint chip requires a number of address and control lines to configure it. Address lines AX0 to AX3 and AY0 to AY2 address the switch to be configured, while CS0 to CS11 select the desired chip. The state of the DATA line during a pulse on STROBE determines whether the switch is open or closed. An global RESET line is also provided to open all switches simultaneously on a selected chip. Of these lines, only the chip-select lines CS0 to CS11 are routed to individual chips. The rest are connected to every chip and are therefore buffered by U12 and U13 to prevent the host computer's interface card from being overloaded. The buffers are open-collector type so that the EM may be powered with a different supply voltage to the interface card.

The address and control lines terminate in ports PC1 and PC2. These are the ports to which the host computer's interface card is connected. Figure 4.4 is a schematic of the card used for all EM experiments in this thesis. The card is plugged into a personal computer's ISA bus, but could easily be adapted for other bus types such as PCI. U1 to U3 provide the address decoding, mapping the card's address space to the region reserved for prototype boards in the PC's I/O memory map (0300 Hex). U4 and U5 are octal latches connected between the PC's data bus and the EM's switch address and control lines, and U6 is a 4 to 16 line decoder used to select the desired crosspoint switch for configuration. The EM can therefore be configured by direct writes to the PC's internal I/O ports, hence genotypes can be instantiated in hardware in a very short time (less than 1ms). Sample code for configuring the EM using the ISA interface is supplied in appendix C. Also included in the appendix are printed circuit board (PCB) foil patterns and component layout. The PCB used in this thesis was through-plated, but the foil is designed so that pins can be soldered to connect each side if through-plating facilities are unavailable. Figure 4.6 is a photograph of an assembled EM, with daughterboards containing two transistors each attached.

4.3.2 Power-Supply Considerations

No on-board provision is made for powering the motherboard or its plug-in basic elements, apart from smoothing capacitors (not shown on the schematics). It is left to the user to provide the most appropriate laboratory power supply unit for the evolutionary experiment. Power can be fed to an evolving circuit at any point on the switch matrix using either connectors V1 to V5, H1 to H4, or by connecting the daughterboard supply lines through to the matrix (The latter method was adopted for all experiments described in the following chapters). Providing circuit power on *both* vertical and horizontal lines (for example, positive supply on H1 and ground on V1) is not

[3]Of course, two daisy-chained motherboards would no longer have a comprehensive interconnection architecture. A further two wired back-to-back to provide a full (i.e. not triangular) 48×48 switch matrix would be required to maintain it.

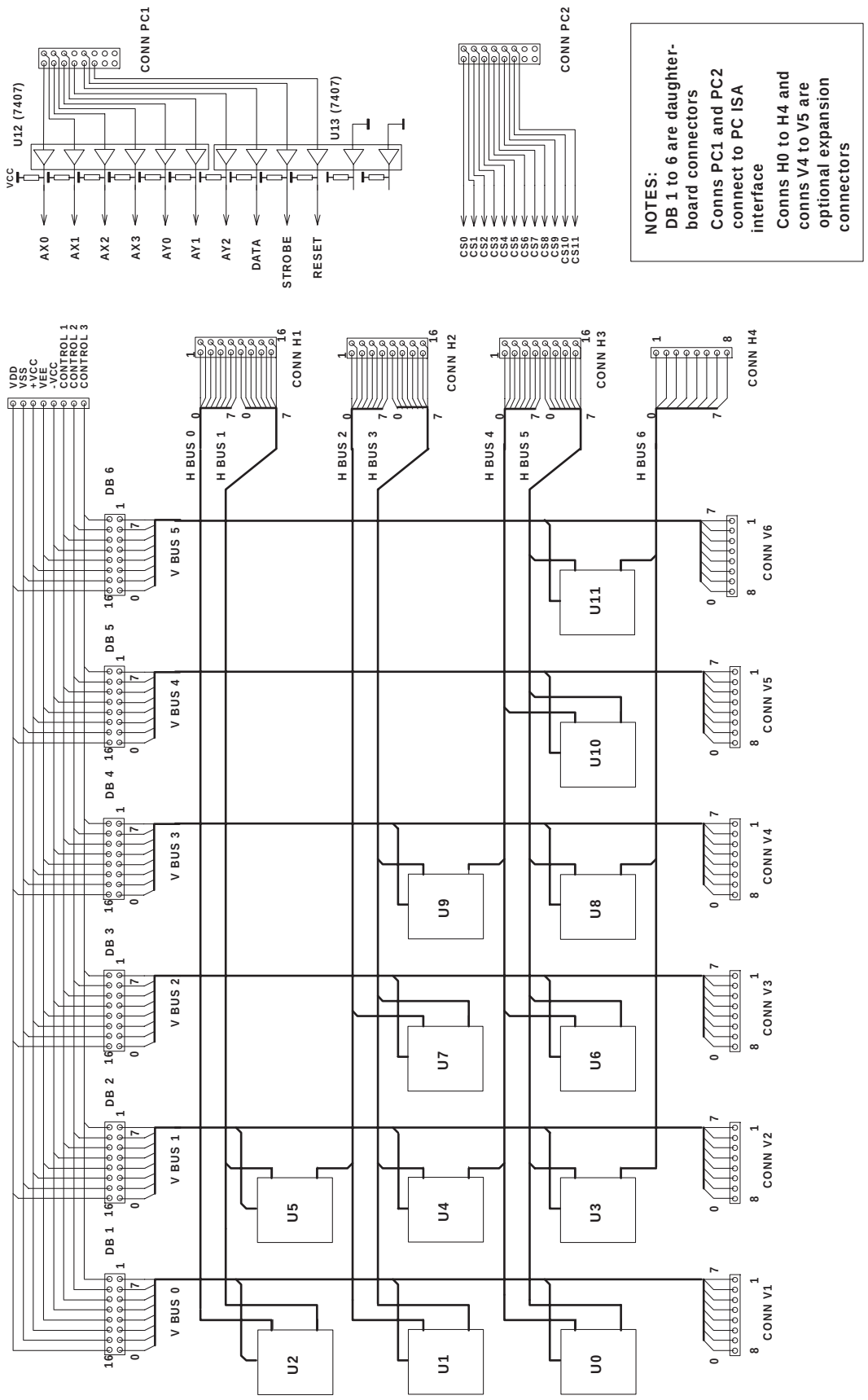


Figure 4.2: Circuit diagram for a 48×48 wire Evolvable Motherboard, showing daughterboard and expansion ports, switch select wiring, buffering, and additional power and I/O for the daughterboard ports

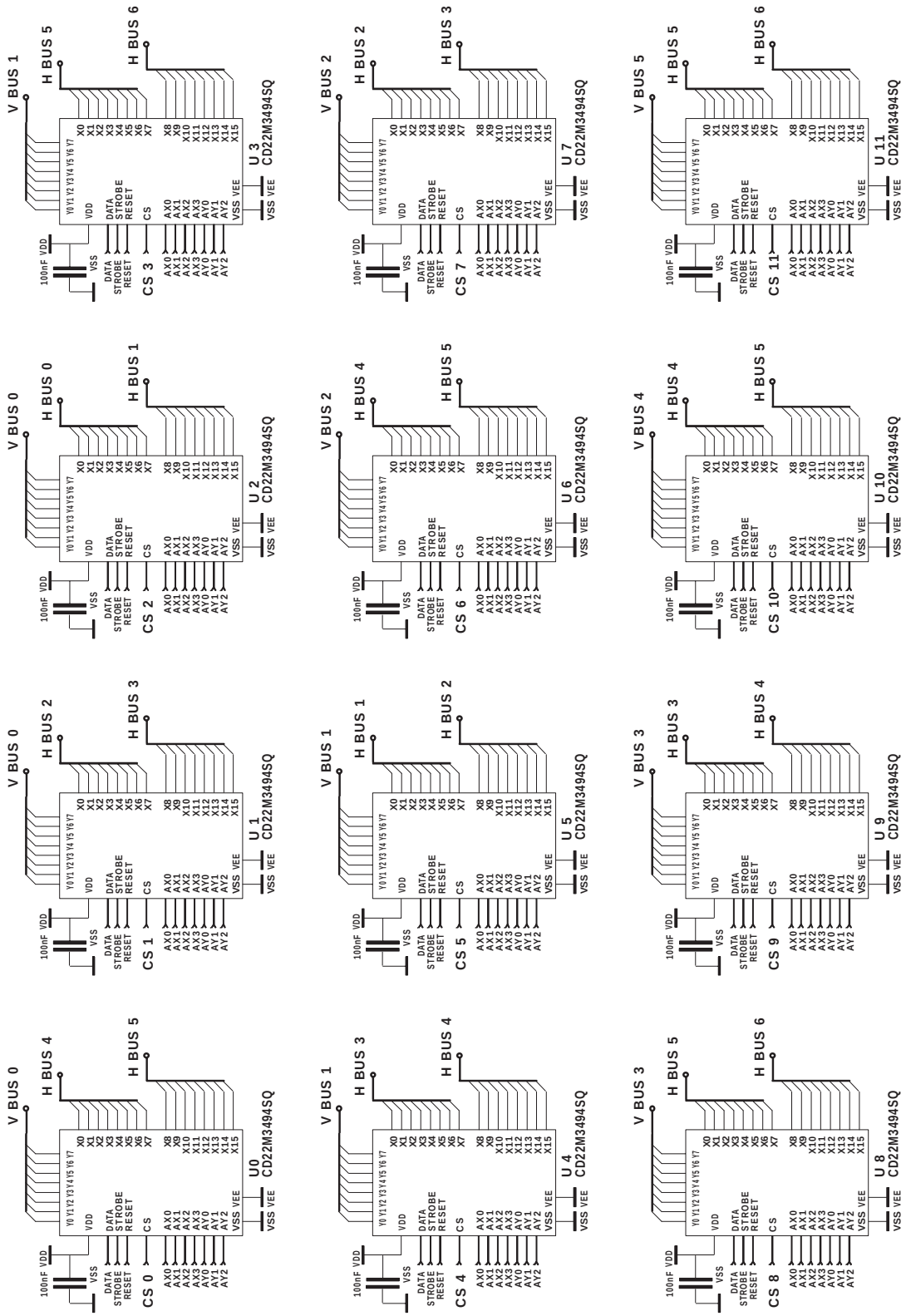


Figure 4.3: Individual pin-out connections for the crosspoint switch devices

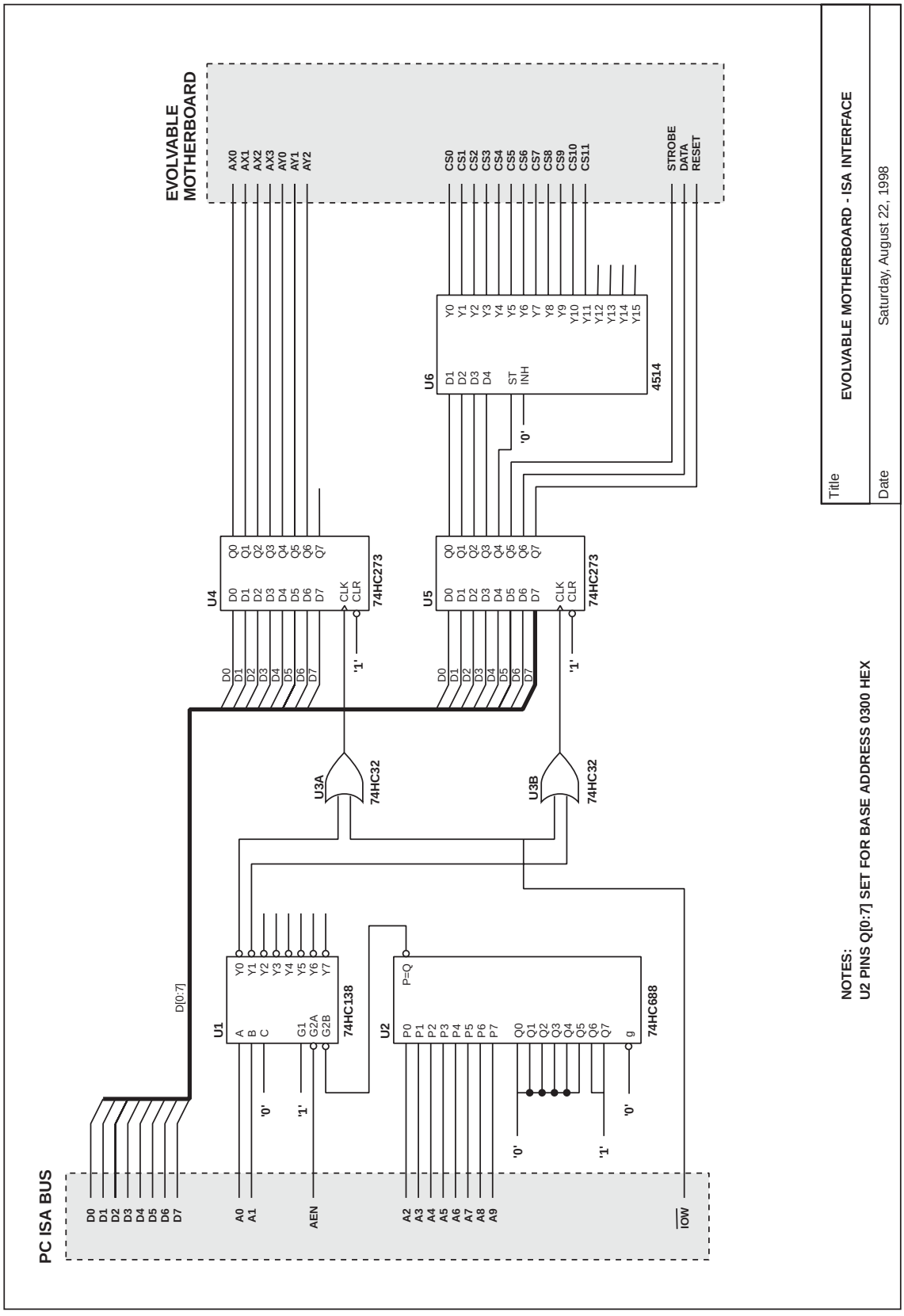


Figure 4.4: Circuit diagram for an ISA interface card suitable for EM configuration with a personal computer

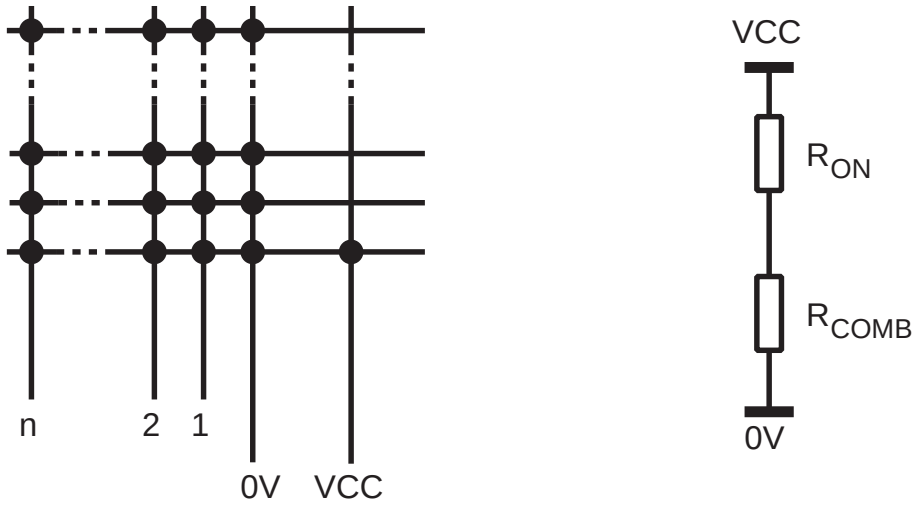


Figure 4.5: (Left) Worst-case configuration for the load on a given switch. In this case, all switches are closed, except those connecting to VCC, of which only one is closed. (Right) The equivalent circuit resistance (See text)

recommended — recall from section 4.2.2 that the path from positive to ground should require at least two switches to reduce the load on any single switch. If this precaution is followed, then for most configurations the maximum supply voltage permitted V_{MAX} is given by equation 4.2

$$V_{MAX} = I_{MAX}(2R_{ON} + R_{PSU}) \tag{4.2}$$

where R_{ON} is the switch resistance when closed, R_{PSU} the power-supply internal resistance, and I_{MAX} is the switch's maximum specified current rating. Using the manufacturer's data of $R_{ON} = 50\Omega$ and $I_{MAX} = 25$mA (Intersil Corp., 1997), and assuming negligible power supply resistance (true of most laboratory PSUs), then $V_{MAX} = 2.5$V.

It is possible to envisage configurations for which equation 4.2 does not apply. Such configurations have many switches closed on the same bus, and their combined resistance becomes less than R_{ON}. Those switches are unharmed, since the load is distributed across them, but if they are then connected via a single switch to the power supply, the single switch must bear the total load.

Figure 4.5 illustrates a worst-case scenario. All motherboard switches are closed, except those connecting the positive power line (VCC on the diagram), of which just a single switch is connected. For simplicity a full (not triangular) matrix of switches is assumed. In this case the total resistance from VCC to ground is that of the single switch plus the combined resistance of the others (R_{ON} and R_{COMB} on the diagram). Assuming an equal number of horizontal and vertical grid wires n in addition to those used for power, then Ampère's current law can be used to show that

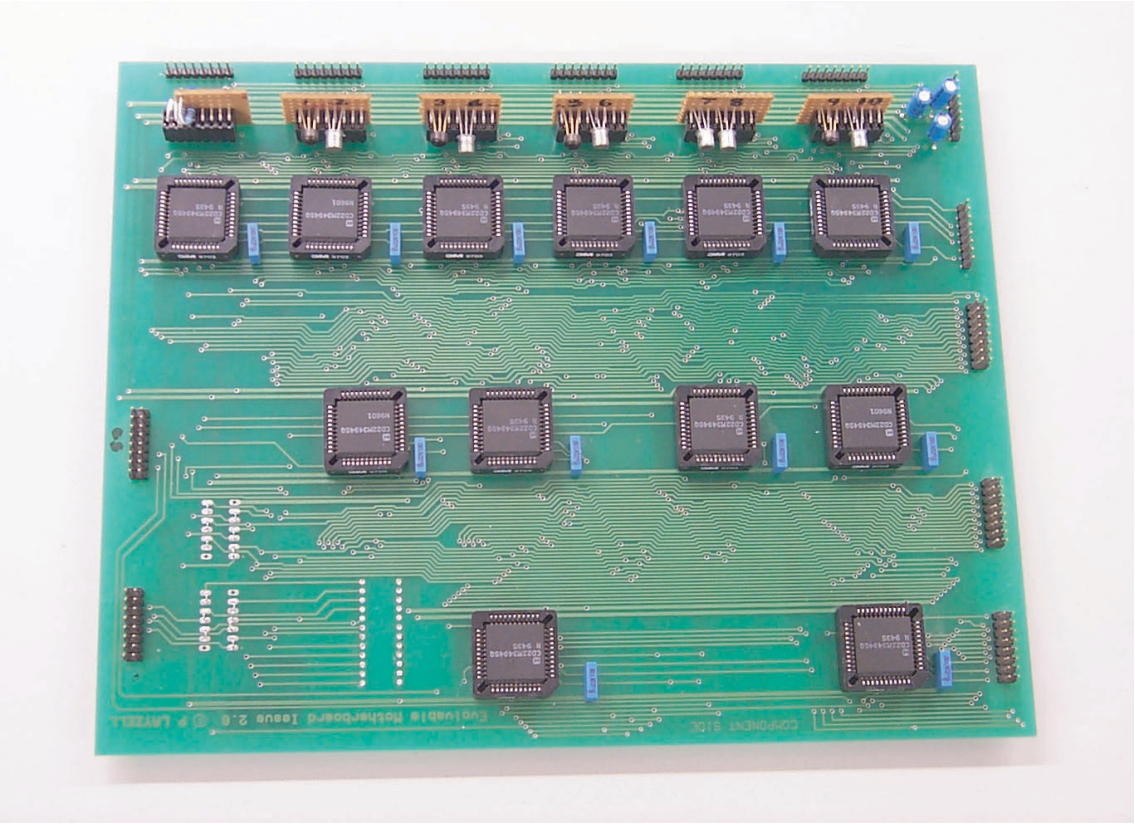


Figure 4.6: An evolvable motherboard, with daughterboards containing two transistors each attached.

$$R_{COMB} = \frac{R_{ON}(2n+1)}{n^2+2n+1} \tag{4.3}$$

For large values of n, R_{COMB} is negligible compared with R_{ON}. While an extreme case is shown here, similar configurations — albeit with fewer parallel switches — are no means exceptional, as will be shown later and in the following chapters. It is up to the user to determine their likelihood for a given genotype $\Rightarrow$ phenotype mapping. If the mapping permits such configurations, then equation 4.4 should be used to calculate the maximum permitted supply voltage.

$$V_{MAX} = I_{MAX}(R_{ON} + R_{PSU}) \tag{4.4}$$

For the values given above, V_{MAX}= 1.25V. Such a low permissible supply voltage is clearly restrictive. However, software verification is relatively simple since only lines connected directly to the power inputs need be taken into account. Alternatively, V_{MAX} can be raised by adding a low-value resistor in series with the power lines. Evolved circuits that consume low power will not cause a major voltage drop across the resistor, but it will protect the EM from those that are more greedy.

4.4 A 'Virtual' EM

Considerable care was taken to produce a software interface capable of storing all details of evolutionary runs, and automatically translating genotype information into circuit schematics to prevent errors. The state of the analogue switches can be displayed graphically with reference to their position on the wiring matrix. This information can be used to construct a *netlist* (a text file describing the basic elements, the analogue switches, and their interconnections), allowing the circuit to be displayed in standard schematic form to facilitate analysis. The netlist can also be used by a circuit simulator. If details of the power supply and input conditions are included, the simulator can model the physical circuit. Therefore, evolution can be carried out *extrinsically* without the need to connect a physical EM. Figure 4.7 shows all of these elements in use; the software interface employs the commercial simulator PSpice (MicroSim, 2000) to evolve an inverter circuit, displaying the current state of the EM switches, and mapping them to a basic schematic editor. There are several benefits to implementing the EM in simulation. In addition to the non-requirement of physical hardware, the switch matrix can be modelled using a linked-list structure. Hence much larger wiring grids, with a capacity for hundreds of basic elements are practical. This possibility will not be explored here, but the 'virtual' EM will be used in the following chapter to compare directly the relative merits of intrinsic and extrinsic hardware evolution.

4.5 Applying the EM to a Simple Evolutionary Experiment

To illustrate the use of the EM in a complete evolutionary framework, a simple experiment is presented in this section. The task was to evolve a digital inverter (a NOT gate) using commonly available bipolar transistors, a BC109C (NPN) and a BC179B (PNP)[4]. These were chosen as the basic elements because their physical characteristics are well understood (see Yang (1988) for details), and because they are adequately modelled in commercial simulators. While in itself a fairly trivial function, the inverter is an interesting initial experiment because the fitness measure must cater for the gradual improvement of what is essentially single-bit digital behaviour. The experiment is also significant in that the resulting circuit was the first ever to be evolved intrinsically at the transistor level.

4.5.1 The experiment — A Hand-Seeded Digital Inverter

Figure 4.8 shows the evolutionary framework. The digital input to the EM is provided by a personal computer via a digital I/O board, and the output is connected to an A/D converter card in differential mode to minimise external noise. The output is also connected to an oscilloscope so that the behaviour of the evolving circuit can be visually inspected. R_{IN} prevents the I/O card being damaged should its output be shorted to the power supply lines, and is of a fairly high value to encourage the evolved circuit to have a high input impedance. A high value of R_{IN} also prevents configurations that use the input to power the circuit. R_{BAL} is a necessary requirement when using the A/D converter in differential mode. The EM is powered by two separate supplies (not shown): one to provide +5V supply for the programmable switches, and the other to provide +2.8V supply

[4]NPN transistors are designed to conduct when the base voltage is positive with respect to the emitter voltage, while PNP transistors, to conduct when the base voltage is negative with respect to the emitter voltage.

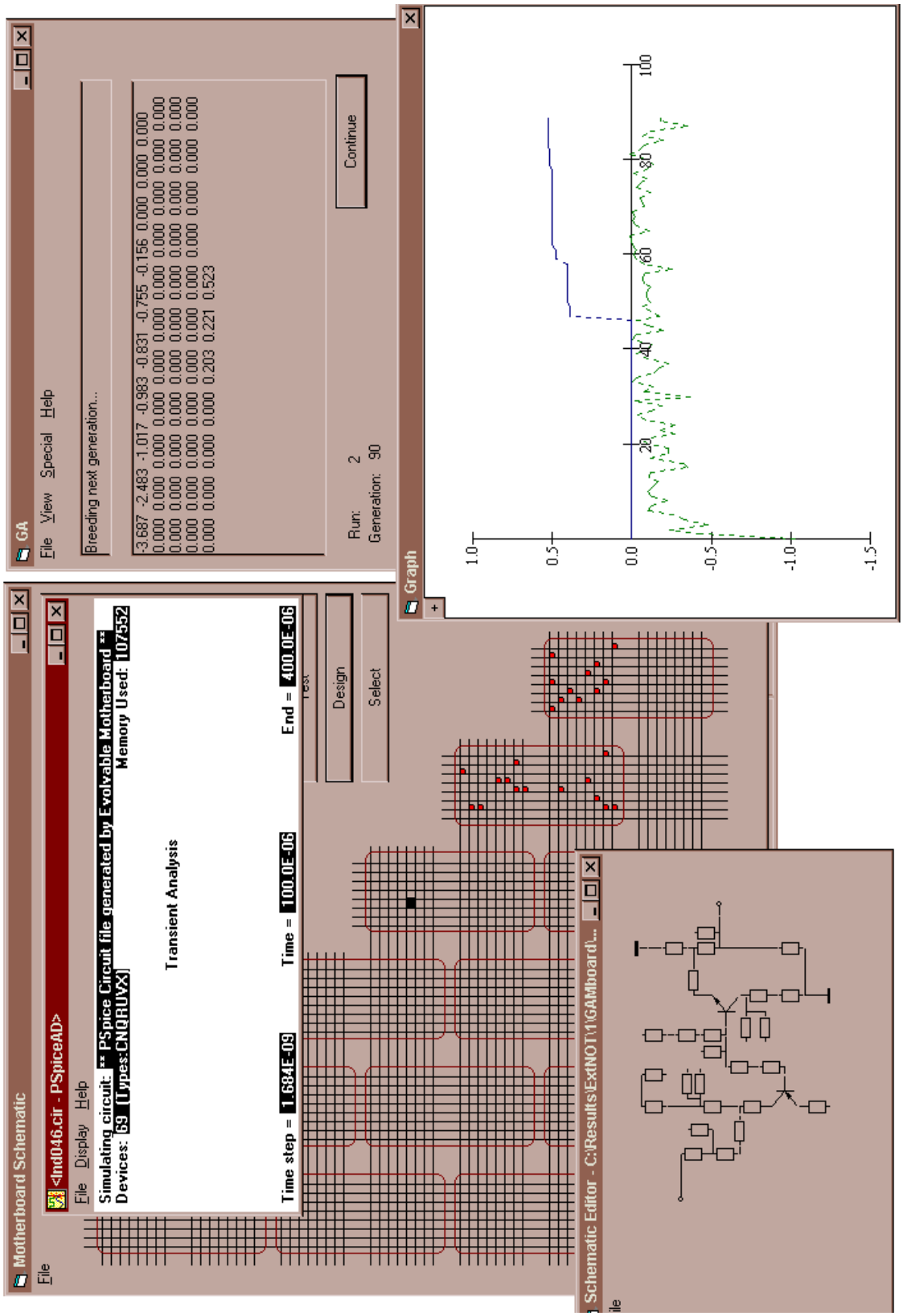


Figure 4.7: A ‘virtual’ version of the EM being used to evolve a simple circuit. A graphical representation of the wiring matrix and the analogue switches is shown (*centre left*). The state of the switches is used to construct a circuit netlist, which provides interconnection details to a simple schematic editor (*bottom left*) and the PSpice circuit simulator (*top left*). The switch configuration is generated by a GA, which displays the population and best fitnesses (*right*)

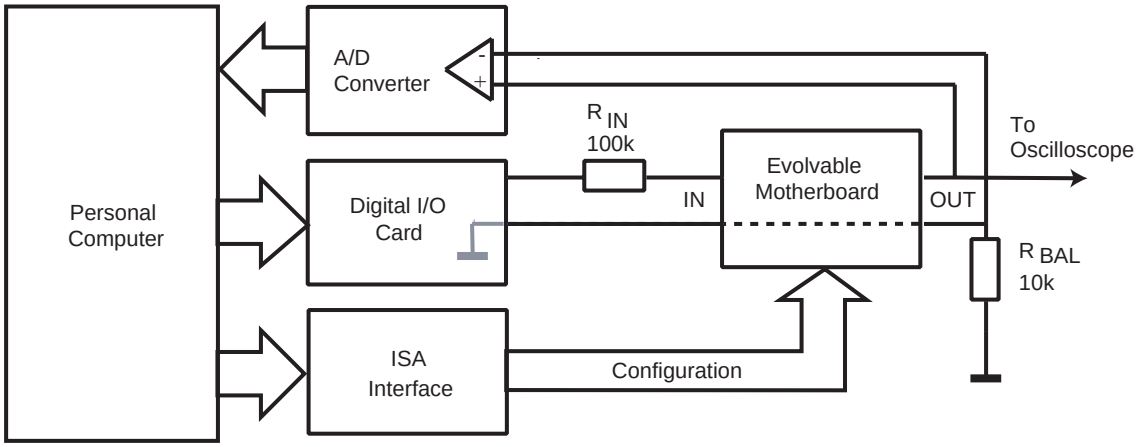


Figure 4.8: The framework used for the evolution of digital circuits.

to the evolving circuit. By configuring the appropriate switches, evolution can make whatever use of this supply it requires, including none, but it does not have access to the +5V switch supply.

The experiment involved seeding the initial population with a hand-designed circuit to determine how artificial evolution could improve on an existing prototype, and to what extent it made use of the resistance (or other characteristics) of the programmable switches. A generational GA was used with population size 50, single-point crossover, linear rank-based selection (as detailed in section 1.2.1), and elitism. Only a small portion of the EM was made available for this simple circuit, allowing the use of just one NPN transistor and one PNP. This was achieved by encoding for just 128 of the 1500 switches through direct genotype⇒phenotype mapping — the on/off state of each switch was determined by a unique bit in the 128 bit genotype. Per-bit mutation rates were set to 0.008 (1⇒0) and 0.0015 (0⇒1), and crossover probability set to 1.0. The mutation rate was biased in this way to avoid the active components being shorted out through too many switches closed at any one time. For the same reason, the initial random population was created with genotypes containing only 3% '**1**'s on average. The values used were found through experimentation at this task, but were not critical.

A single individual of the initial population was replaced by the 'poor' inverter circuit shown in figure 4.11 (*left*). During a logical **0** input, the transistor is switched off, providing no path for the output to ground, hence the output remains at the same voltage as the power supply. Input voltages greater than approx. 0.7V — logical **1** — cause the transistor to saturate resulting in very small resistance between its collector and emitter, and a path to ground for the output. However, the combined resistance of the configurable switches (represented as boxes on the diagram) acts as a potential divider, and the point at which the output is connected can never drop below three-quarters of the supply voltage. This circuit conforms to the NOT function in that its output corresponding to a **0** input is of slightly higher voltage than that corresponding to a **1** input, however this difference is too small to be of any practical use. In electronic parlance, its voltage swing is not great enough to cross the digital logic threshold.

The evolving circuits were evaluated as follows: A series of 100 test inputs containing 50 **1**s and 50 **0**s (logical Highs (+5V) and Lows (0V), respectively) was applied sequentially in random order. For each test input, the output voltage was measured five times with random delays between

measurements, and summed. The delays were uniformly distributed in the range 0 to 100ms. Fitness was scored according to equation 4.5, where t signifies the test input number, S_L and S_H the set of Low and High test inputs respectively, and v_t ($t = 1, 2, .., 100$) the summed output voltage in millivolts of the circuit corresponding to test number t. With a power supply of 2.8V, maximum fitness values in the range -250,000 to 140,000 are expected. The minimum value would be given by a wire connecting the input directly to the output, and the maximum by an inverter with a voltage swing of 2.8V. Circuits with constant output would have zero fitness.

$$\text{fitness} = \frac{1}{5}\left\{\left(\sum_{t \in S_L} v_t\right) - \left(\sum_{t \in S_H} v_t\right)\right\} \tag{4.5}$$

Equation 4.5 permits a gradual improvement in the evolving circuit; fitness increases with the voltage swing. Multiple measurements for each test input permit a gradual improvement in the speed with which the circuit responds to changes in input voltage.

4.5.2 Results

The results shown here are typical of those obtained from several runs. Over 200 generations the fitness score of the best individual increased threefold (figure 4.9) — from 32,000 to 96,000 — a dramatic improvement in performance when observed on the oscilloscope. While the circuit is still not ideal, it could now function as a practical NOT gate. Figure 4.11 (*centre*) shows the best individual when evolution was stopped at generation 470. In essence the circuit is the same as the seeded one; a single NPN transistor with power and i/o lines connected at the same points. The increased performance is due to the way evolution has utilised the resistance of the programmable switches, to ensure a low resistance path to ground compared with that to +2.8V, when the transistor is saturated. This has been a achieved by moving the output connection slightly, and placing several sets of switches in parallel to reduce the resistance between the transistor's emitter and ground (the actual arrangement of parallel switches, figure 4.10 (right), is more complex than that depicted in the diagram). Evolution has 'discovered' and exploited a fundamental law concerning resistance; that placing resistors in series *increases* the combined resistance, but placing them in parallel *reduces* it.

This experiment was originally conceived as a simple 'test' of the EM's capabilities as an evolvable medium. However, the results are not without significance. Referring to figure 4.10, the portion of the motherboard used for this experiment allows up to eight separate paths between external connections and the transistors. The evolved circuit used three paths to connect input, output, and +2.8V, and devoted all remaining ones to the 0V connection in order to achieve parallel resistance. This has implications concerning interconnection architecture. Evolution may not have been able to make such good use of parallel resistance with a more restrictive interconnection system.

It is not surprising that the programmable switches' physical characteristics were exploited; evolution does not *know* that they are intended only for signal routing. However the degree to which these characteristics were embodied in the evolved circuit may be significant. There are signs even from this simple experiment that the configuration circuitry cannot be considered separable from the basic elements in an evolvable medium intended for analogue circuits. Evolution is

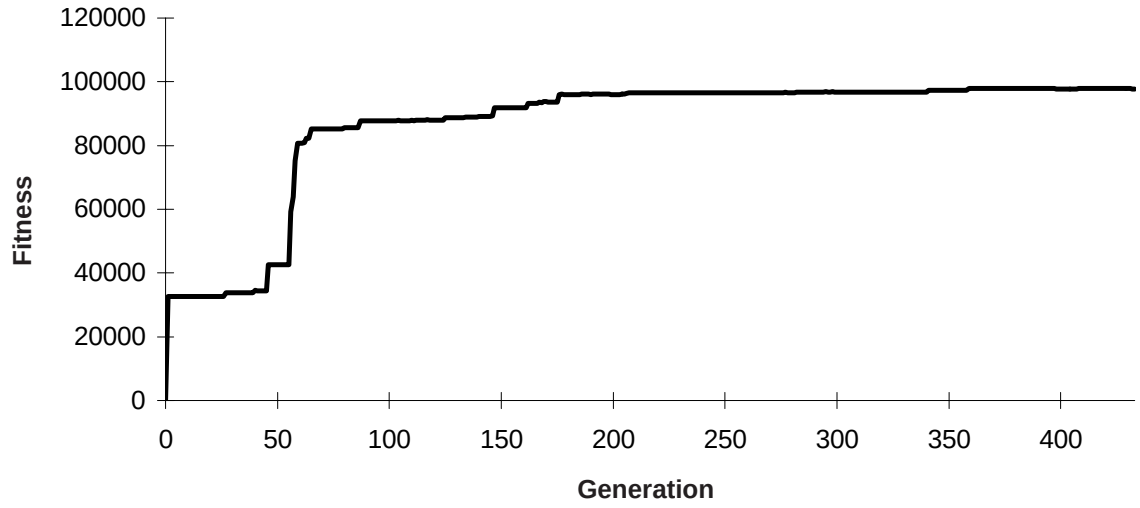


Figure 4.9: Best fitness of the population for each generation.

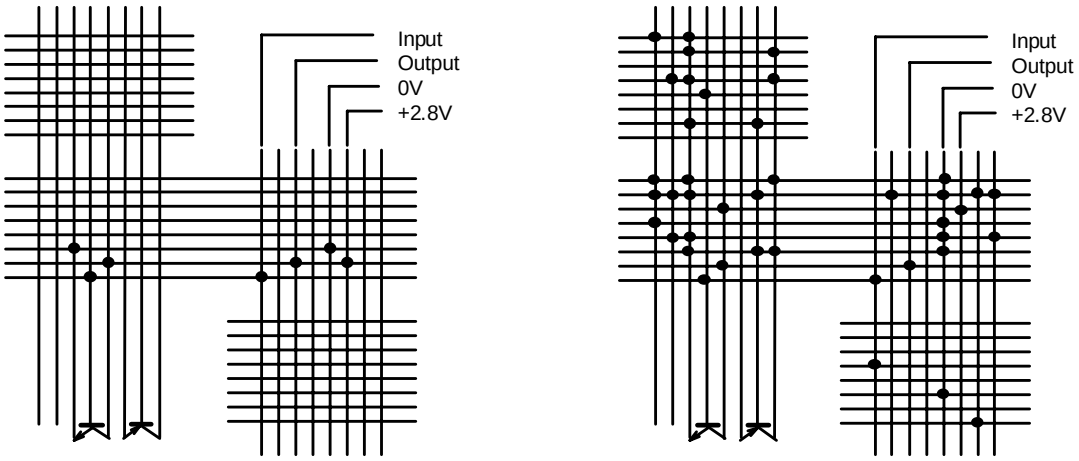


Figure 4.10: Circuits instantiated on the evolvable motherboard. The black dots represent closed switches. (*Left*) The original hand-seeded prototype. (*Right*) The fittest individual after 470 generations.

equally likely to exploit the physical characteristics of either. We shall see more examples of this in the following chapter.

Summary of Chapter 4

Commercial reconfigurable devices implemented in VLSI are commonly used in HE research. However, they are not ideal for this purpose. Monitoring waveforms at internal nodes is problematic and laborious, making analysis difficult; there is little choice of basic evolutionary element; their interconnection architecture is targeted at conventional design methodologies, but may not be the most suitable for evolution; they can be damaged or destroyed by a wide range of configurations. Reconfigurable VLSI devices aimed specifically at hardware evolution exist, but they require considerable resources to produce, and share some of the disadvantages above with their commercial counterparts. The Evolvable Motherboard (EM), a new reconfigurable platform implemented using discrete devices, is presented as an alternative tool for general HE research.

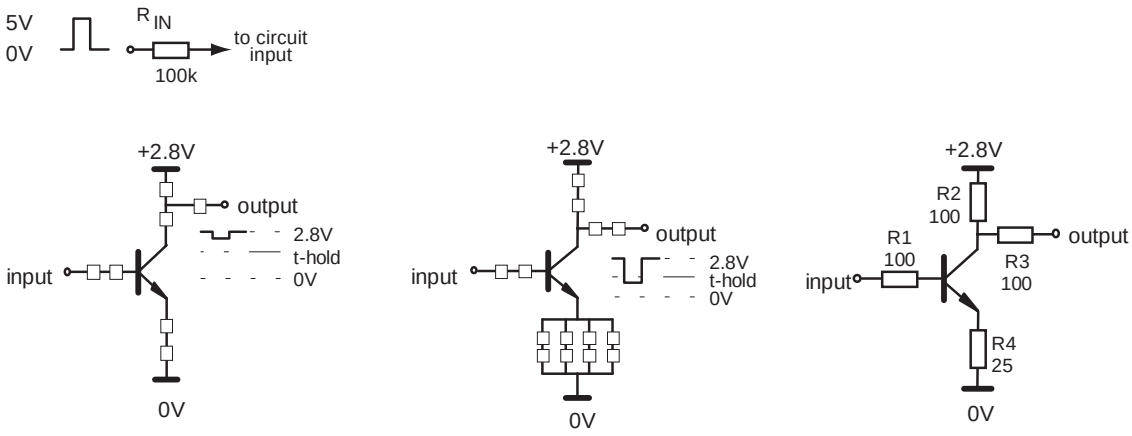


Figure 4.11: Circuit diagrams. (*Left*) Hand-designed prototype inverter. The boxes represent motherboard switches in the 'closed' state. (*Centre*) Fittest individual after 470 generations. (*Right*) Circuit representation replacing switches with resistors of equivalent value.

This platform possesses a triangular matrix of programmable analogue switches, situated on a two-dimensional wiring grid. Basic evolutionary elements are connected to the grid using plug-in daughterboards: This allows a huge variety to be hosted with direct access to their pins for probing. The EM's interconnection architecture is comprehensive, since the topology of the switch matrix allows independent interconnection of all combinations of basic element pins. However, more restrictive architectures can be investigated through an appropriate choice of genotype⇒phenotype mapping. The EM is economical to build, and cannot be destroyed by illegal configurations if the supply voltage is kept below a well-defined maximum value. Physical implementations of the EM limit the quantity of basic evolutionary elements that can be practically employed, nonetheless a vast quantity of different circuits can be configured on them.

EMs can also be implemented entirely in software using commercial circuit simulators. This permits the direct comparison of intrinsic versus extrinsic modes of HE to be carried out, and permits an increased quantity of basic elements to be configured in an evolving circuit. To illustrate how the EM can be incorporated into an evolutionary framework, a simple experiment was conducted to optimise a hand-designed inverter circuit. Within 200 generations, the circuit's performance increased threefold. Evolution achieved this primarily through exploiting the resistance of the configuration circuitry.

Chapter 5

Common Issues in HE Investigated Using the Evolvable Motherboard

---

The evolvable motherboard is now applied to several simple evolutionary experiments, each designed to address one of the issues arising in HE research, as discussed in chapter 2. The experiments are preliminary explorations and are presented to illustrate the flexibility of the EM in this context. Although they furnish useful insights, they are not intended to be comprehensive studies of any single phenomenon. As few constraints as possible are imposed in the experiments on how evolution should use the EM's resources, and the chapter closes with a brief discussion on the implications of the unconstrained approach when applied to intrinsic analogue HE.

5.1 Evolving an Inverter From Scratch

The first experiment extends the hand-seeded inverter experiment of the previous chapter. Having already observed circumstances whereby evolution can improve on an existing prototype, the inverter is now evolved from an entirely random initial population. The experiment addresses two questions:

Qn1 Is a directly-mapped, comprehensive interconnection architecture suitable for evolution to find its own prototype circuit?

Qn2 What are the relative merits in evolving from scratch in preference to optimising an existing design (presuming one exists)?

Both of these questions involve the nature of the search space. In the previous chapter, it was remarked that many of the possible EM switch configurations are likely to interconnect the wiring grid such that the basic elements (the transistors) are bypassed. Such configurations result in circuits either giving constant output, or outputting some attenuated copy of the input. We can infer that the search space is dominated by circuits displaying these trivial behaviours, but we do not know whether other potentially fitter circuits are isolated peaks distributed throughout the space, or clustered together in regions such that a circuit slightly better than trivial fitness can quickly improve by gradual modifications. Even though it is reasonable to suppose that the proportion of

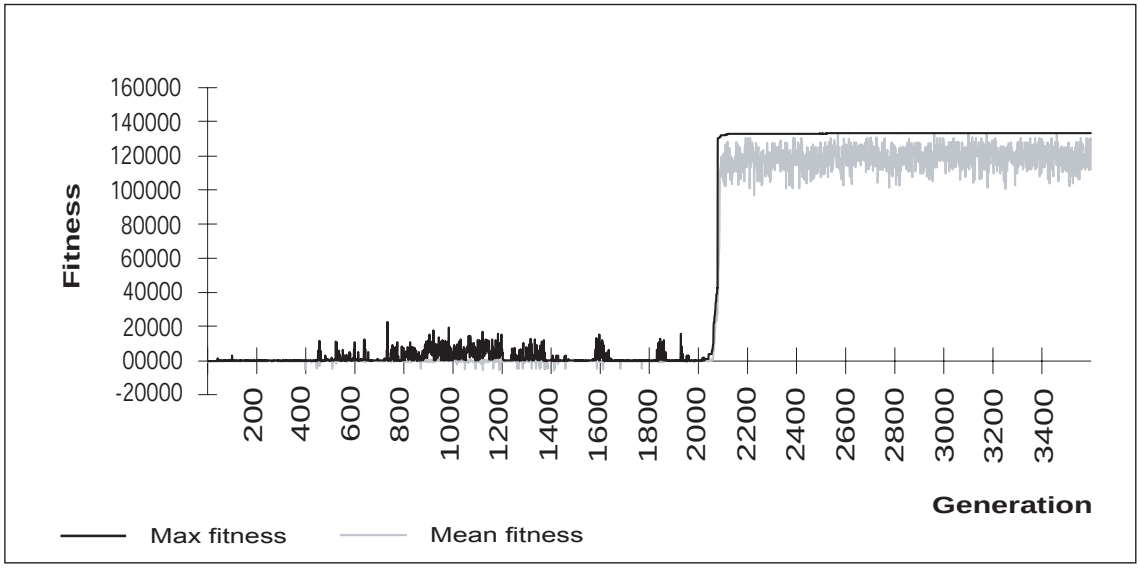


Figure 5.1: Typical mean and maximum fitness plots for an inverter circuit evolved from scratch. Once a stable prototype has been found, less than 100 generations are required to optimise it.

useless circuits within the space is extremely high, this has little bearing on our main concern; the space's evolvability.

The experiment employs the same GA parameters, evaluation method, and portion of the motherboard as for the hand-seeded inverter, but with an initial population of random genotypes, each containing an approximately equal proportion of '1's and '0's. Several runs were conducted, and each required many generations (between 2000 and 10000) before a circuit arose whose behaviour scored greater fitness than constant output. However, once this prototype had been found, only 50 to 100 more generations were required in almost every case to produce fitness scores of around 133,000 - significantly better than had been achieved with the hand-seeded experiment. Figure 5.2 shows one of the circuits.

This circuit is certainly unconventional; although near-perfect behaviour was observed on the oscilloscope, its operation is not at all clear at first glance. Evolution has made no use of the 0V line, implying that the circuit should not work at all! In fact a path to 0V is achieved via the 10MΩ input impedance of the oscilloscope (Ro on the diagram). Q2 is operating in the seldom-used reverse mode (see Yang (1988) for details), and for best operation, R7 should be small compared to Ro, which is evidently why evolution has exploited the oscilloscope in this manner. If the oscilloscope is unplugged, the circuit ceases to work. While Q1 appears to be 'meaningfully' connected, it has no obvious function, and indeed unplugging it has no apparent effect on fitness scores. The circuit is not exploiting any subtle properties particular to the transistors with which it evolved. It gave almost identical fitness scores when the transistor board was unplugged and substituted with other boards. A circuit simulation was also carried out using the SPICE simulator. It too worked perfectly, with a resistance substituting the oscilloscope (for which a simulation model was unavailable).

The best circuits of the generations just prior to that containing the first prototype were investigated. The transistors in these circuits were shorted out by many programmable switches in the closed state, and in most cases the input was disconnected, giving constant output with zero

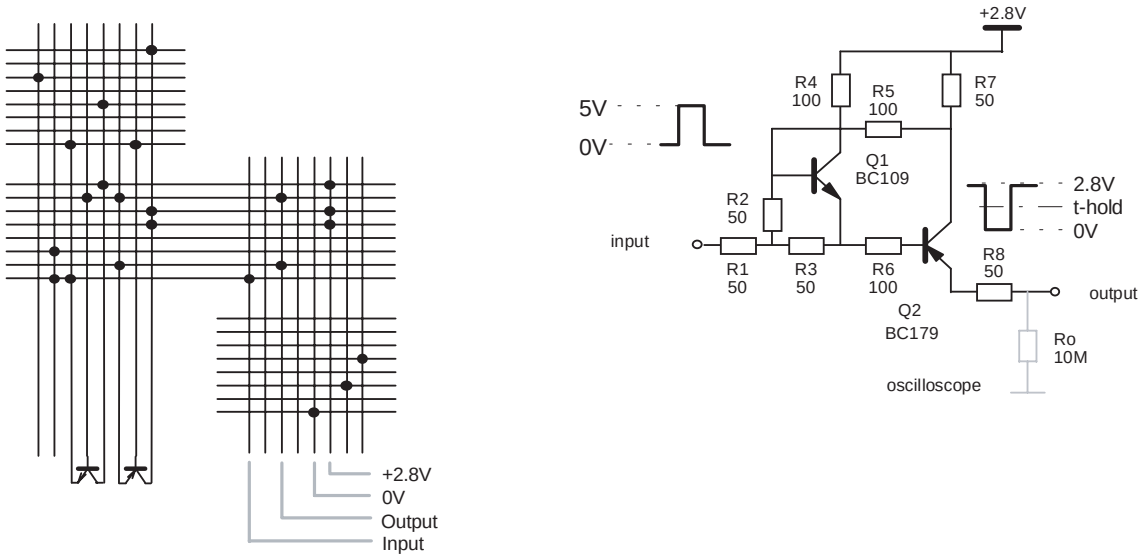


Figure 5.2: An inverter circuit evolved from scratch after 2500 generations. The circuit's only path to 0V is via the oscilloscope.

fitness. Since circuit configurations that provided any signal path from input to output yielded negative fitness scores, evolution quickly located solutions with the input or output disconnected. In this case, the circuits' performance was unaffected by bit values in the majority of the genotype. Despite biasing mutation in favour of open switches, there was therefore no selection pressure to incorporate the transistors in the circuits. This particular task contained sub-optimal circuits of constant output, but the suboptima for another task could be circuits directly copying the input signal to the output. This would also tend to prevent the transistors from playing any active role in an evolving circuit.

These observations do not appear to favour the use of a directly-mapped comprehensive interconnection architecture for locating a prototype by evolutionary search (Qn1). However, the combination of mapping/architecture/fitness function provides a search space in which inverter circuits are not isolated; once a prototype is found, a near-optimal circuit is consistently evolved within a few tens of generations. Indeed, the circuit consistently evolved faster from the prototype stage than was the case for the hand-seeded design. Averaged over five runs, the number of generations required was an order of magnitude smaller for this phase. Imposing a human-designed starting point seems to constrain evolution to a less 'evolvable' region of search space (Qn2).

5.2 An Alternative Interconnection Architecture

An alternative interconnection architecture is now investigated. It was conceived to prevent the shorting of basic elements observed in the previous experiment, without excessively restricting the EM's repertoire of circuit topologies. The new architecture is a consequence of altering the genotype⇒phenotype mapping so that only a limited number of switches can be closed. Figure 5.3 illustrates how the new mapping is encoded. The basic idea is to limit the quantity of closed switches available per row, so that the pins of basic elements are not highly connected. This is consistent with many conventional circuits where each pin is usually only connected to two or

three other pins. In the encoding, each column is assigned a corresponding row. This can be observed in figure 5.3: Taking the first row/column to be the top right, Vcc (col 2) is assigned to row 2; Output (col 5) to row 5, and so on. The genotype encodes the switches a row at a time. For each row, one bit specifies connection to the corresponding column, followed by the column address and on/off connection bits for n additional switches.

The electrical equivalent of this mapping is n single-output, c-input programmable analogue multiplexers (or single-pole, c-throw rotary switches) placed on each row, with the output connected to the row, and each input connected to a unique column. c is the total number of EM columns, and each multiplexer has an **enable** input corresponding to the on/off bit in the mapping. This new mapping/architecture shall be referred to as *multiplex* mapping. Restricting the row connectivity in this way does not place severe limitations on the number of closed switches per column, hence the use of multiple paths to exploit parallel resistance is still possible. An example of this is shown in figure 5.3 columns 3 and 8. The new encoding has the additional advantage of allowing more standard EA operators to be used; biased mutation rates are no longer necessary.

The task remains the inverter function, but now evolution takes place using the whole complement of the evolvable motherboard, with ten transistors on a 48×48 wire matrix. Thus, as well as exploring the new encoding, the experiment will give an indication of whether or not evolution will exploit all of the resources open to it, as ten transistors would normally be considered an excessive amount for an inverter. The genotype is encoded with $n = 3$ giving a length of 1056 bits. Although fairly large, such genotype lengths have not presented problems on the FPGA experiments detailed in chapter 2, and it is important to determine that the same is true for the EM. Per-bit mutation rate was set to 0.008, but otherwise the GA is unchanged from the previous experiment. The same fitness function was also used with the exception of the normalising factor, which was changed from 1/5 to 1/250 so that fitness scores reflect the output voltage swing in millivolts. In order that the circuit's output could be monitored by oscilloscope without the scope being exploited, a standard unity-gain FET-input buffer amplifier (Horowitz & Hill, 1989) was placed between the circuit output and the PC's A/D card input (to which the oscilloscope was attached).

The circuit was allowed to evolve for 4000 generations, but reached near-optimal fitness long before this period. Figure 5.4 displays the fitness graphs and population convergence — the expected hamming distance between a pair of randomly chosen genotypes. The population became genetically converged to a stable level before fitness began to increase. This does not occur with standard GAs, but is the essence of SAGA (section 2.4), in which a relatively converged population is allowed to explore design space as a single entity. It is important to note that the circuit configurations in the population are more similar than is implied by the convergence graph. The new mapping permits genotypes to contain large amounts of 'junk'. This is because genotype bits specifying switch addresses are only relevant if the bit that determines the open/closed state of the switch is *set*, i.e. the switch is closed. Altering the address bits of an open switch will not affect the circuit's configuration or its behaviour.

Figure 5.5 shows the fittest circuit after 600 generations, the prototype having evolved after only 200. The circuit uses eight out of the ten transistors. If any are unplugged, the circuit fails to operate. Many feedback loops are visible, with half of the transistors only partially connected and acting as diodes. It is interesting that so many transistors are necessary. The circuit performs

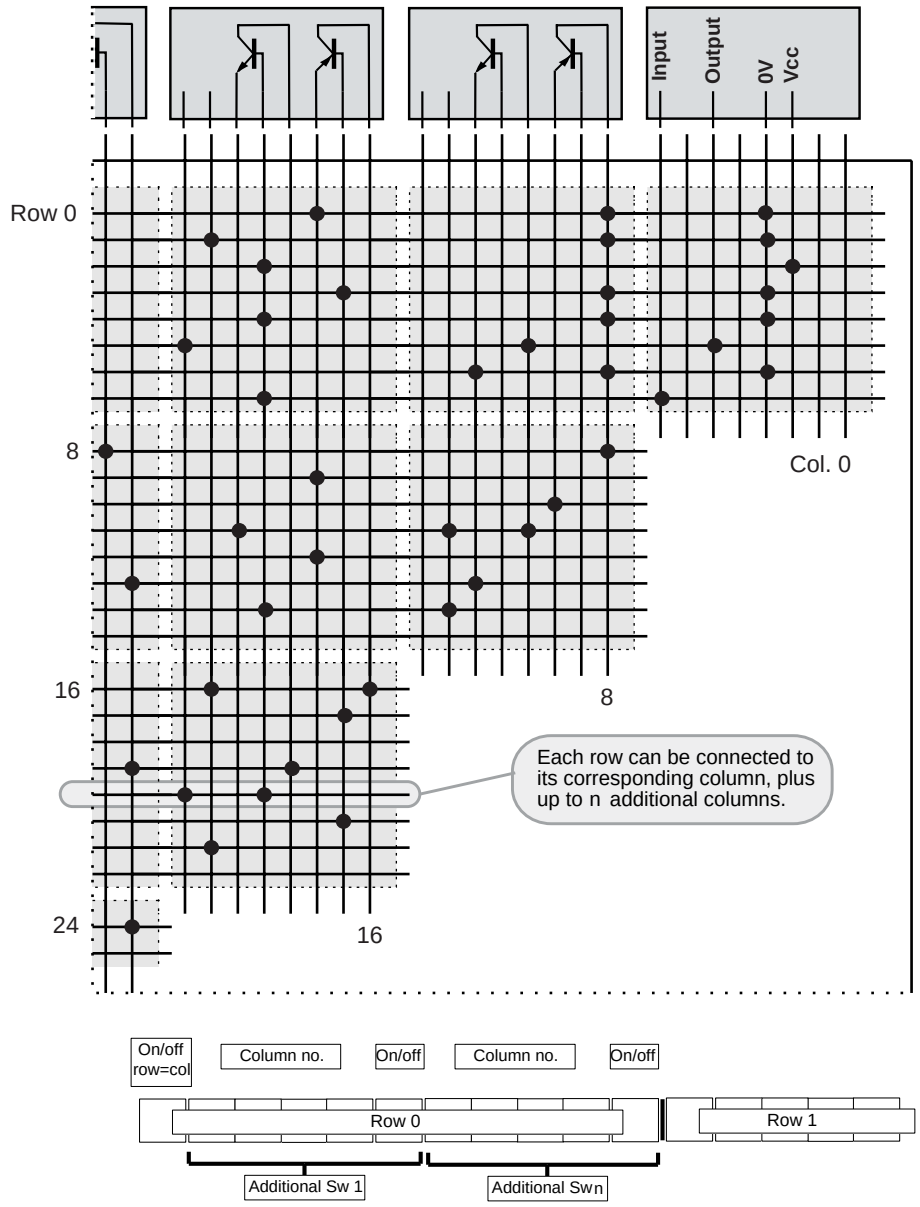


Figure 5.3: An alternative to direct mapping. In this scheme, each row is assigned a corresponding column. One bit specifies whether that row and column are connected. Up to *n* additional columns can be connected to the same row, each encoded by a number of address bits signifying which column, and an additional bit signifying whether the connection is made. In this example, $n = 2$.

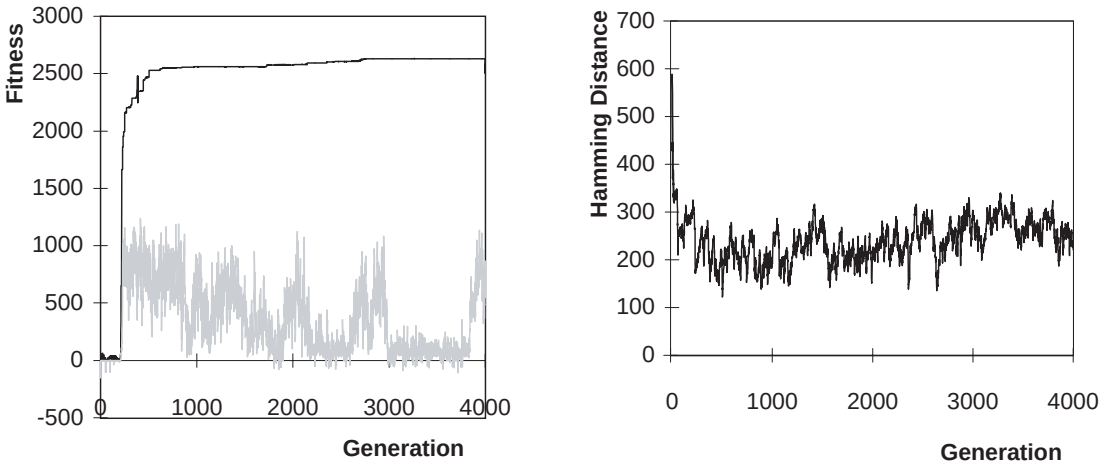


Figure 5.4: Population statistics for the multiplex-mapped inverter. *(Left)* Maximum and mean fitnesses of the population at each generation. *(Right)* Genetic convergence, measured as the mean hamming distance between the genotypes of pairs of individuals, averaged over all possible pairs.

no better than the previous one, neither does it exhibit any apparent tolerance to mutations or faults. On investigating the evolved topology, a follow-up experiment revealed a curious effect. The experiment consisted simply of removing one of the transistors from the motherboard, and re-evaluating all individuals of the final population. No matter which transistor was removed, at least one individual from the population scored higher than 75% of maximum possible fitness, while the elite circuit was effectively destroyed, scoring zero fitness. The population is fault/mutation tolerant while the elite individual is not. This is surprising, because the evolving population was already converged, and should not have contained diverse solutions to the inverter problem.

If this is true of evolved circuits in general, rather than being due to the encoding or the search space for this one task, then it is a highly desirable quality. Were a reconfigurable device containing an evolved circuit to be used in an engineering application, faults in the device could be tackled simply by reconfiguring it with another member of the population whence the circuit came. These other members are available at no extra ‘cost’; they are a by-product of the evolutionary process. This will be investigated in detail in chapter 6. There are also implications for the portability issue discussed in section 2.2.3. Evolved circuits may be made to be more portable for a manufacturing process that implements a population, rather than a single individual.

The interconnection architecture resulting from the new encoding was a more effective evolutionary medium than the comprehensive, direct mapping of the previous experiment. Repeated attempts to evolve inverters on the full motherboard failed when using direct mapping. The new encoding is retained for all of the evolutionary runs described in the following sections, unless otherwise stated.

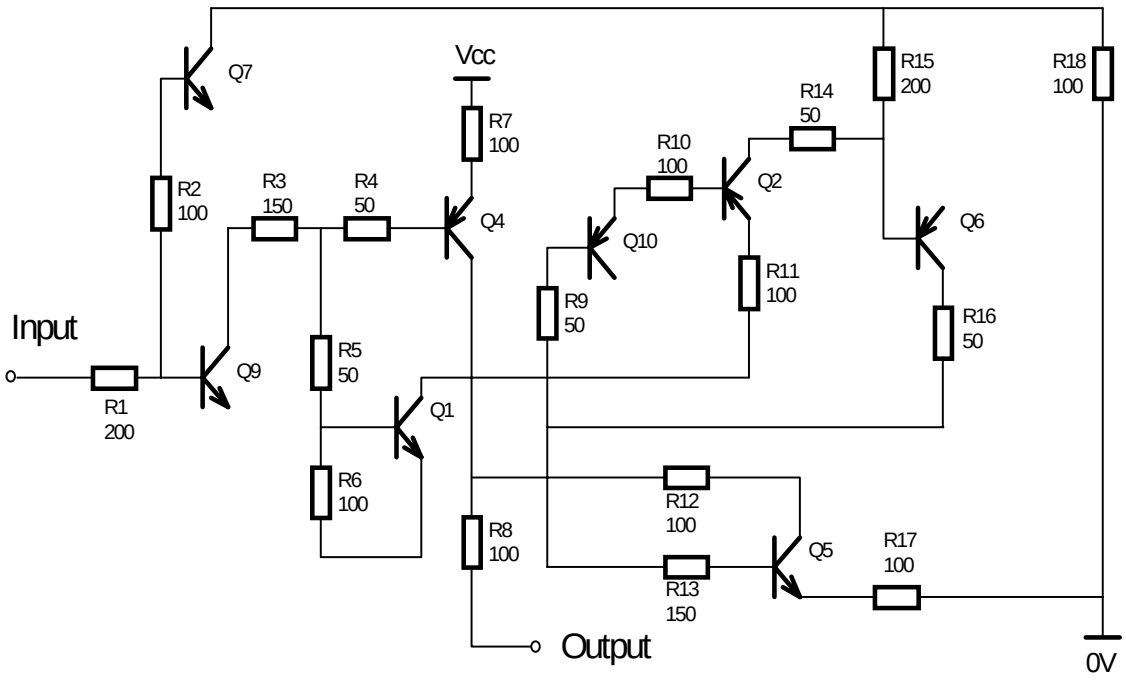


Figure 5.5: An inverter circuit after 600 generations, evolved using the new mapping, and the full EM.

5.3 Extrinsic Hardware Evolution

So far, all of the experimental studies documented here have been evolved intrinsically. Having discussed the relative merits of this mode of evolution with its extrinsic counterpart in chapter 2, a software implementation of the EM is now applied to extrinsic HE, in order to examine the two most common notions :

1 Extrinsic HE can be expected to be slower than intrinsic HE because there is more computational expense in modelling and evaluating the evolved circuits.

2 By definition, circuits evolved intrinsically are guaranteed to work in reality, whereas those evolved extrinsically may not.

(Zebulum et al., 1996; Thompson, 1998a; Huelsbergen, Rietman, & Slous, 1999; Stoica, Fukunaga, Hayworth, & Lazaro, 1998).

The latter does not apply where an abstract simulation model can be made predictably to reflect the behaviour of its physical implementation, as is the case for synchronous digital systems or feed-forward combinatorial logic. It concerns circuits whose behaviour may be dependent on physical parameters that are not accurately known or modelled.

The above notions seem reasonable, but they are based largely on common sense and studies of only one of the two modes. A direct comparison is a more sensible first step if they are to be taken seriously as heuristics by which the most appropriate mode can be chosen for a given task. To achieve this, evolutionary runs using both a physical architecture and a software simulation of that architecture can be conducted for the same task. Until the present study (first reported in Layzell (1999b)), the only direct comparison reported was for a slow oscillator (Thompson, 1996b). The

oscillator experiment was aimed at studying the unconstrained manipulation of timescales in noise-free digital environments versus noisy and essentially analogue hardware, rather than the issues of interest here. However the extrinsic phase took considerably more time than the intrinsic, due to the computational overhead required. This need not always be the case. Where behavioural changes over a long time-scale are desired, intrinsic HE may necessitate long evaluations, whereas lengthy real-time periods of constant behaviour can be simulated quickly. For example, consider evolving a 0.1Hz square wave oscillator from discrete components. Evaluating candidate circuits for this task would require taking measurements over at least one cycle of oscillation, a period of 10 seconds. In simulation this might only take 1 second of real time. The computation time required by a simulator is not the only factor governing the relative speed of HE modes; another is noise.

There are three principal means by which noise can be introduced into the HE framework. The first is *measurement noise*: The interface between the circuit and the host computer can be susceptible to variations in the external environment, such as mains hum propagated in the air, or to noisy internal components. The second, *stochastic circuit behaviour* results from similar susceptibilities in the circuit itself. Finally, *stochastic evaluations*: The fitness evaluation procedure can include a stochastic component. For example, in the preliminary inverter experiment of chapter 4, the output was measured five times with random delays between measurements for each test input, and the test inputs themselves were applied in random order. Each of these components results in identical circuits yielding different fitness scores each time they are evaluated, and their combined effect is cumulative. All three may be present or deliberately made so in both modes of HE, but only the latter can be entirely eliminated when evolving analogue circuits intrinsically. The following argument assumes that noise and stochastic circuit behaviour are *not* present for extrinsic HE, and that the fitness evaluation procedure does not include any stochastic component.

When evolving from scratch, a significant number of random candidate solutions must be evaluated before the first stable prototype solution for a task appears — one that gives better fitness scores than constant output, say. For extrinsic HE, this prototype need only give a fractionally better score, but for intrinsic HE its score must also exceed a threshold of stochastic circuit behaviour and measurement noise if it is to survive further rounds of modification and evaluation. Stochastic circuit behaviour is not necessarily constant or predictable; it is likely to depend on the circuit topology. For example, a circuit whose output is connected directly to 0V will be far less noisy than one whose output is floating. Of course, intrinsic evaluations may be made more thorough to reduce the influence of circuit and measurement noise, but these may last longer than simulated evaluations.

The notion that intrinsic evolution is guaranteed to work in reality is questionable. The only ‘real’ example of the circuit is the configurable device on which it has evolved. If the intended application incorporates the same device, then the statement is true (albeit subject to any of the long-term failure and environmental conditions discussed in section 3.1.1). But the statement is of little relevance if a different device, even if nominally the same, is the intended target. The constraints required to ensure that an intrinsically evolved circuit will work if instantiated in different hardware are similar to those for the extrinsic case. Where these constraints cannot be met, and where working in reality implies a different device, how can we assess the relative merits of

Run	Generations req. to reach fitness 2500	
	Intrinsic	Extrinsic
1	1186	31
2	1802	151
3	936	135
4	1066	131
5	1552	114
6	2333	116
7	120	174
8	1203	60
9	1104	218
10	446	141

Table 5.1: Comparison of intrinsic versus extrinsic inverter evolution.

intrinsically versus extrinsically evolved circuits where portability is concerned?

The following experiments focus on the above issues of noise impeding the search for a stable prototype and portability of unconstrained evolved circuits, by making direct comparisons between intrinsic and extrinsic modes.

5.3.1 The Influence of Noise in Finding a Stable Prototype

Intrinsic and extrinsic evolution was carried out for multiple runs of inverters. The inverter circuits were evaluated in a similar manner to previous sections, but the series of test inputs was reduced from 100 to 10, containing 5 '1s' and 5 '0s' applied sequentially to the input in alternating order. For each test input, the output voltage was measured twice (immediately on applying the input, then after a delay of 1ms) and summed. Fitness was scored according to equation 5.1:

$$\text{fitness} = \frac{1}{10}\left\{\left(\sum_{t\in S_L} v_t\right) - \left(\sum_{t\in S_H} v_t\right)\right\} \tag{5.1}$$

As before, only 10 bipolar transistors were made available as basic elements. The transistors used (BC109C and BC179B) are included in the libraries of most circuit simulation packages, while the analogue switches were substituted in simulation with resistors of 50Ω (closed) and 1GΩ (open). These figures were derived from the CD22M3494SQ datasheet (Intersil Corp., 1997), but the switch capacitance characteristics were not modelled. All runs were carried out using a rank based, generational GA with elitism, and an initially random population of 50 individuals. Genetic operators were mutation (per-bit probability set at 0.01) and single-point crossover (probability 1.0).

For 10 runs of inverter evolution, fitness values reached 90% of the maximum possible value in around 1000 generations for intrinsic evolution and 150 generations for extrinsic evolution in most cases. Individual cases are shown in table 5.1.

The time required to evaluate the candidate circuits was limited mainly by the speed of the

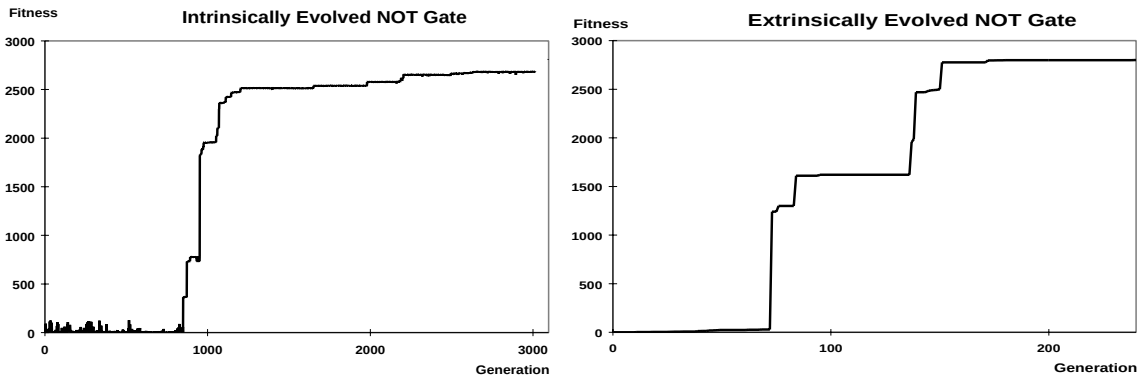


Figure 5.6: Typical fitness plots of the best individual for intrinsically (*Left*) and extrinsically (*Right*) evolved inverters.

digital I/O card used to supply the test input, and by the processor speed for the intrinsic and extrinsic methods respectively. Extrinsic evolution was carried out using a 266MHz Pentium P.C., resulting in inverter evolution taking an average of around 3 hours extrinsically, and 6 hours intrinsically. The real time advantage is not necessarily significant here, since the two approaches had different factors limiting the speed, either of which could make a profound difference if adjusted. However, that extrinsic evolution took an order of magnitude fewer generations to achieve good fitness scores is certainly significant. Figure 5.6 shows typical fitness plots of the best individual for both methods. In both cases, initial fitness is zero, given by constant output circuits. However, such circuits in reality have a stochastic component and have produced fitness scores of above 50 on many occasions. During this random walk stage, circuits which genuinely performed better than constant output did not remain in future generations until one was found with performance significantly above the noise level. This was not a problem for the extrinsic approach, which found a circuit giving fitness of 0.2 at generation 0. Since there was no noise, evolution was able to develop this circuit despite it being only marginally better than constant output. At generation 73, the fitness of the best individual jumps from 29 to 1240. Figure 5.7 shows the circuit diagrams before and after this jump. It is clear from the diagrams that the new high-fitness circuit was not found by chance, but by a minor variation of the best circuit from the previous generation.

5.3.2 Assessing Intrinsic and Extrinsic Modes for Portability

Simple transistor amplifiers were evolved as test cases for assessing portability. This task was chosen since the resulting circuits are likely to rely on the transistors' *current gain* (h_{FE}) parameter. h_{FE} varies widely for different transistors of a given type, hence conventional amplifier circuits rely only on it being above a certain minimum value (Horowitz & Hill, 1989, page 62). To evaluate the amplifiers, a 1kHz sine wave of 200mV peak to peak amplitude was applied to the input with a DC[1] offset of half the supply voltage. Both input and output were monitored (using an A/D converter in the intrinsic case), and 500 samples of each taken at 10μs intervals. Fitness was the average error between the output and amplified input:

[1] DC = Direct Current.

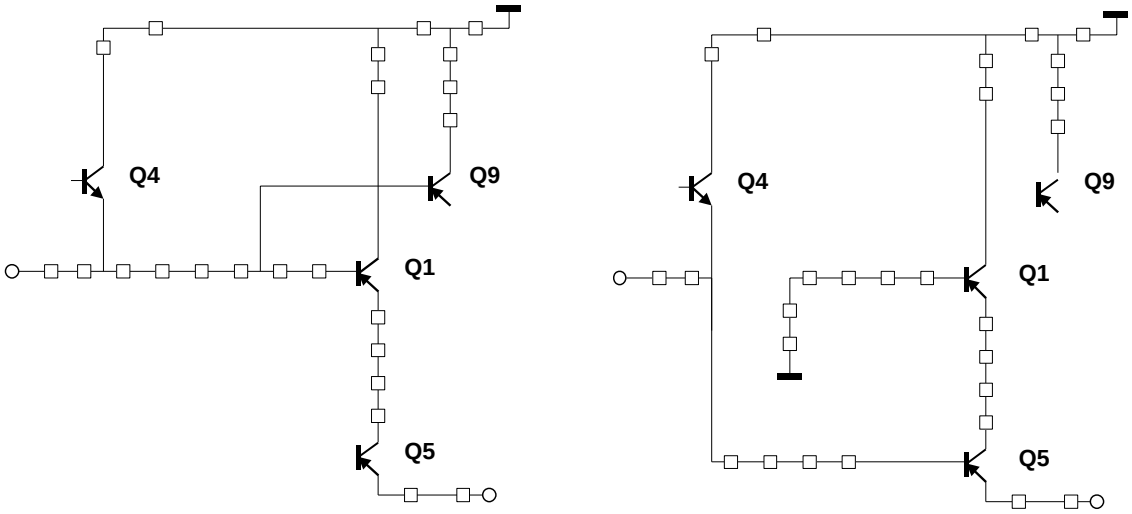


Figure 5.7: Inverter circuit diagrams during preliminary stages of extrinsic evolution. (*Left*) Generation 72, fitness 29; (*Right*) Generation 73, fitness 1240. The high fitness circuit is clearly a minor variation of the best circuit from the preceding generation.

$$fitness = -\frac{1}{500}\sum_{i=1}^{500} |a\,(V_{in_i} - O_{in}) - (V_{out_i} - O_{out})| \tag{5.2}$$

where a is the desired amplification factor, V_{in_i} and V_{out_i} are the i^{th} input and output voltage measurements respectively, and O_{in} and O_{out} are the DC offsets of input and output respectively. Amplification a was set to -10. The fitness measure equates to a simple inverting amplifier, however it is not intended to be a practical amplifier since no provision is made for phase shift, slew rate, or frequency response.

Five successful runs each of extrinsic and intrinsic amplifier evolution were conducted using GA parameters identical to those for the inverters. For each run, evolution was stopped after 6000 generations (extrinsic) and 8000 generations (intrinsic). On completing an intrinsic run, the transistors were unplugged and replaced with nominally identical ones. The final population was then re-evaluated, and in all cases fitness of the best individual was reduced — from 1% up to 12%. Evolution was then allowed to continue, with previous fitness being restored within 200 further generations. On completing an extrinsic run, the final population was used as a starting point for intrinsic evolution. Evaluating this population using a physical motherboard resulted in far greater fitness reduction — from 52% to (in one case) 100%. However, in 4 out of 5 cases, less than 200 generations were required to attain 90% of previous fitness scores, with fitness fully restored within 400 generations. Figure 5.8 shows a typical plot. The remaining case gave no better fitness than constant output, and re-evolving from this starting point gave similar results to evolving from scratch.

It is acknowledged that the above test for portability requires improvement; the intrinsically evolved final circuits only had to cope with different transistors even though they were also reliant on switch resistance. In the extrinsic case, *every* component was different and circuit capacitance, unmodelled in simulation, was introduced. Since performing this experiment, another EM has

been constructed. In a future version of the experiment, the intrinsic circuits will be transferred from one EM to another, and a more accurate simulation model used.

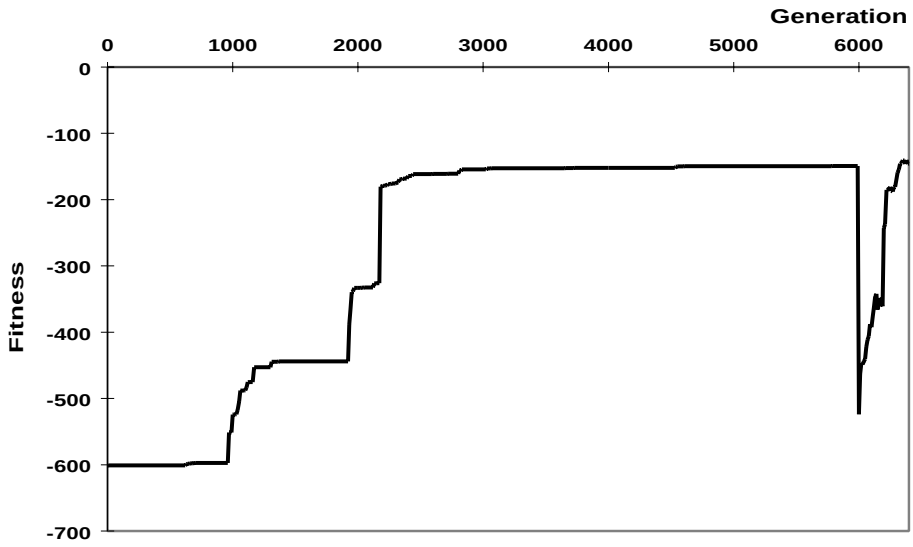


Figure 5.8: Using the 6000th generation of extrinsically evolved amplifiers as a seed for further intrinsic evolution.

5.3.3 Implications of the Direct Comparison

The direct intrinsic/extrinsic comparison of the inverter and amplifier tasks cannot be considered a thorough investigation of the relative merits of both evolutionary modes. However, we have seen at least one task for which extrinsic evolution has proved the faster. This is apparently due to external noise (a combination of measurement noise and stochastic circuit behaviour) impeding the intrinsic search for a stable prototype. When comparing the number of generations required for amplifier evolution the difference was far less marked. Extrinsic evolution required on average about half the number of generations compared with the intrinsic approach, to produce circuits with near-optimal fitness. For this task however, considerable care had been taken to minimise external noise. It was substantially lower than for the inverters, and further supports the suggestion that noise was the main factor. Given the simplicity with which the programmable switches were modelled, extrinsic evolution also performed surprisingly well in the 'fairer' test for operation in reality. While further experimentation is needed before we can say conclusively which is the better mode for a given EA, task and architecture, it is clear that the decision to use intrinsic in preference to extrinsic HE should not be taken solely on the basis of the computational overhead required by simulation software. It is dependent on the conditions under which candidate circuits are evaluated, the degree to which computer models reflect real components, the time required to evaluate the circuit, and the nature of any constraints imposed on the way evolution can use the components.

5.4 A More Difficult Evolutionary Task

The inverter and amplifier circuits have proven useful as test cases for experimentation, but they are not difficult challenges for circuit design. Conventional techniques can deliver equivalent cir-

cuits with lower power consumption, fewer components, and better performance than those we have seen in this chapter. In section 2.1.2 we observed that HE shows much potential for the design of continuous-time dynamical systems, at which it may eventually prove more successful than conventional design for all but the simplest systems. Dynamical systems — both artificial and biological — are often governed by systolic processes (Huelsbergen et al., 1999). Synchronous digital systems use the ticking of a global clock to coordinate state changes; petrol engines rely on phase information from rotating components to determine ignition and valve timing; oscillation is observed at microscopic and macroscopic levels in animals (for example, neuronal firing patterns (Kandel, 1976), distribution of blood by the heart). To date the intrinsic evolution of electronic oscillators has not been entirely successful; either the evolved waveform has been irregular (Thompson et al., 1996), or the attained frequency has differed vastly from the desired frequency (Zebulum et al., 1998; Layzell, 1999b; Huelsbergen et al., 1999). One reason that more effort has not been devoted to oscillator evolution is that we already know how to design them. However, there may still be advantages to evolving them. Conventional circuits employ charge storage devices (capacitors) which release their charge in a controlled manner through a resistance (Horowitz & Hill, 1989, pp. 284-300) to provide the necessary timing. This combination of components is known as an RC time-constant, the value of their product increasing as frequency decreases. Since large-value capacitors are difficult to implement in VLSI with any accuracy, low-frequency oscillators ($\leq$100kHz) generally require them to be provided externally, and are thus expensive. In the tone discriminator experiment, we have seen how evolution was able to alter an FPGA's configuration to 'tune' time-delays (chapter 3). RC time-constants were still required, but they were formed by parasitic capacitance present in the silicon. The following experiment attempts to harness this ability, to produce an evolved oscillator of a precise frequency without the use of capacitors.

5.4.1 The Oscillator Experiment

Oscillator evolution using bipolar transistors is a difficult task. Whereas oscillation is a likely outcome of recurrent loops of digital gates or operational amplifiers, precise operating points must be established before it can be produced by a network of transistors. The conditions necessary for oscillation are extremely unlikely to arise by chance (this was confirmed by initial experiments performed by the author that rewarded only oscillation frequency and amplitude). Therefore the search must be biased in some way to 'encourage' them. One set of conditions known to induce oscillation is an amplifier with positive feedback, and we already know from the previous experiment that transistor amplifiers are relatively easy to evolve. Hence the fitness function chosen for this task rewards high-amplitude AC[2] signals present at the output, even if the signal is just noise. In the hope that this produces an amplifier, configurations that allow positive feedback are further encouraged by rewarding cyclic signals.

In the publications cited in the previous section, the desired frequencies were not attained because the output of candidate circuits was sampled directly by an a/d converter. When using such a framework, the well-known Nyquist criterion dictates that the sampling frequency must be at least twice the oscillation frequency if it is to be measured accurately. There is no guarantee that an evolving circuit will meet the criterion and where it does not, erroneous results known as

[2] AC = Alternating Current

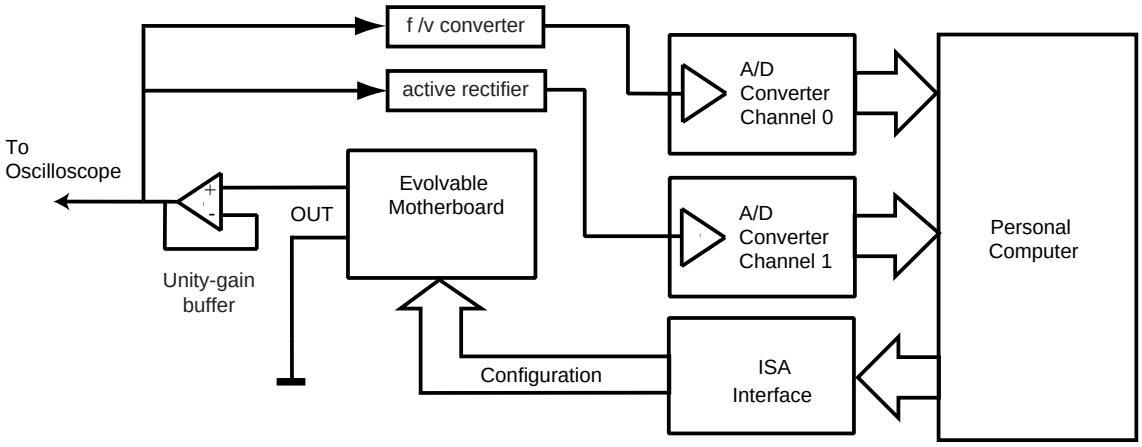


Figure 5.9: The framework used for oscillator evolution. The buffered output is fed to a frequency to voltage converter and to an active rectifier circuit. Both are implemented in hardware to prevent aliasing errors, and each is allocated an independent channel of the a/d converter.

'aliasing errors' will be produced: Integer multiples of the desired frequency that are above the sampling frequency will give equally high fitness scores.

The framework of figure 5.9 prevents aliasing errors by placing a frequency to voltage (f/v) converter implemented in hardware, between the EM output and the a/d converter. If a signal of amplitude greater than 10mV r.m.s.[3] is present at its input, the f/v converter outputs a d.c. voltage in the range [0..6V], equal to the input frequency multiplied by a timing constant, k. A separate measurement of the candidate circuits' amplitude is also provided within this framework by a hardware rectifier circuit between the same output and another a/d channel, giving a DC voltage which is the r.m.s. amplitude of its input signal. Fitness of candidate oscillators is given by

$$\text{Fitness} = \begin{cases} \bar{a} + k\frac{f_{min}}{f_{max}}\left(f_{target} - \left|f_{target} - \bar{f}\right|\right) & \text{if } \bar{f} > 60\text{Hz} \\ \bar{a} & \text{if } \bar{f} \leq 60\text{Hz} \end{cases} \tag{5.3}$$

where $\bar{a}$ and $\bar{f}$ represent respectively output amplitude and frequency, averaged over 20 samples, each taken at 100μs intervals. f_{min} and f_{max} are the minimum and maximum of the 20 frequencies sampled, and f_{target} the target frequency. The ratio of minimum and maximum frequencies serves to reward individuals with constant output frequency. This fitness function requires configuring the f/v converter's time constant such that the target frequency corresponds to 3V output (half its maximum output voltage), otherwise the function will not be smooth for frequencies above 60Hz[4]. The target frequency was 25kHz, giving $k = \frac{3}{25000}$.

GA parameters and encoding were identical to those used in the previous section. On configuring candidate genotypes in hardware, evaluation was delayed by 5ms to allow the f/v converter and rectifier circuits to settle to a stable value. From 20 runs, 10 resulted in successful oscillation, attaining the target frequency to within 1%, with a minimum amplitude of 100mV. Figure 5.10 is

[3] r.m.s. = root mean square. This is a standard measure for the amplitude of sinewaves, and is equal to the peak-to-peak amplitude times the reciprocal of the square root of 2.

[4] A lower bound of 60Hz (equation 5.3) was chosen to ensure that the f/v converter was detecting oscillation and to avoid circuits heavily influenced by mains hum (50Hz in the UK).

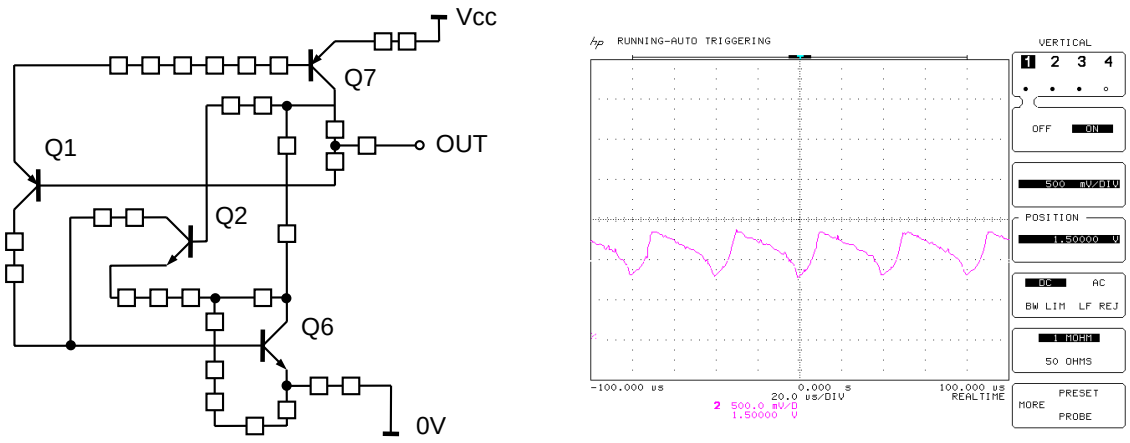


Figure 5.10: Circuit diagram and output waveform of a typical evolved oscillator, output frequency 25kHz, amplitude 500mV peak to peak

the circuit diagram of a typical evolved oscillator. Many feedback loops are present, all of which were probed using an oscilloscope. It was thought that the circuit would employ beat frequencies to achieve the relatively low target frequency, but probing revealed that this is not the case; all internal nodes display oscillation at the target frequency. If the transistors are substituted with nominally identical ones, the output frequency changes by as much as 30%. A simulation model was constructed that incorporated parasitic switch capacitance as well as various forms of capacitances possessed by bipolar transistors (Horowitz & Hill, 1989, p.102), but despite extensive parametric manipulation it could not be persuaded to oscillate. It may be possible for an experienced analogue designer to determine exactly how the circuit works from the schematics and oscilloscope waveforms. Since the transistors are easily unplugged, their physical characteristics can be measured individually to provide additional data. However, the programmable switches' characteristics are almost certainly important for the circuit's operation. While their input/output connections are accessible, the chips in which they are embedded contain additional circuitry that may also influence circuit function. It looks as though we have to fall back on the analysis techniques of chapter 3 before we can explain the circuit's behaviour precisely. Hence, while the EM's flexibility and observability has indeed facilitated analysis, this experiment has also revealed its limitations.

5.4.2 Environmental Influences on the Oscillator Experiment

The experiment has demonstrated that the difficult task of evolving a transistor oscillator at a precise frequency *can* be successfully achieved using the EM, but in fact the unsuccessful runs are potentially more interesting. All were able to achieve good fitness, yet did not manifest stable oscillation. Recall that configurations resulting in amplifiers were encouraged by the fitness function. Monitoring the (unsuccessful) circuits' output with an oscilloscope revealed signals of the order of 10 to 50mV amplitude with rapidly changing frequency. The signals resembled those produced by a nearby a.m. radio, which was unable to pick up any but the strongest stations due to the proximity of computers in the lab. Connecting the circuits to an audio amplifier produced an identical sound to the detuned radio. It seems likely that evolution had amplified signals present in

the air which were stable enough over the 2ms total sampling period to gain good fitness scores. Amplifiers capable of this must have extremely high input impedance, and this was certainly the case with many of the non-oscillating circuits; their DC output voltage rose if a person's hand was placed closer than half a metre from them. If the hand remained there, the output voltage remained high, but rapidly fell if the person was earthed. In effect, evolution had 'cheated' by producing radios and highly sensitive electrometers, using as an input the printed circuit tracks connected to an 'open' programmable switch whose impedance in that state is at least 100MΩ. Notwithstanding the undesired behaviour of these circuits, evolution had not only imbued the programmable switches with a functional role in the circuit, but also the printed circuit to which they were soldered.

Some of the evolved oscillators worked reliably until a laboratory soldering iron several meters away was disconnected from the mains, at which point oscillation ceased. This occurred in spite of the high-quality laboratory power-supplies and extensive mains filtering employed. Further tests were conducted with one of these circuits. Clearly the circuit was extremely sensitive to tiny transients in the supply voltage, nonetheless it worked perfectly every time it was re-instantiated (irrespective of whether the soldering iron was on or off). However, it failed to oscillate if during instantiation, the EM configuration switches were set in *reverse order* to that used originally. It seems that oscillation was dependent on some initial condition such as node voltage or capacitance, and the initial condition could only occur when the configuration switches were set in a particular sequence, but this is only a guess. However, repeated tests made it certain that the circuit's operation relied in *some* way on the configuration sequence, revealing yet another, more dynamic way in which the configuration circuitry can influence evolved circuits.

5.5 Discussion

The experiments documented in this chapter show that the evolvable motherboard can be used to investigate many important issues arising in current HE research, including analysis, genotype encoding, basic elements, portability, evolved topologies, exploitation of configuration circuitry, and extrinsic versus intrinsic HE. The genotype⇒phenotype mappings explored here are two of many that are possible, due to the array of switches allowing any combination of pin connections. Mapping is analogous to the interconnection architecture of a configurable device, and analysis of various mappings is likely to lead to a picture of the sort of architecture that an FPGA ideally suited to evolution should have. While I have made suggestions throughout this chapter to account for the phenomena observed and their possible implications, I reiterate that they are not, nor are they intended to be, conclusive. The experiments are presented primarily to illustrate the capabilities of the EM as a research tool. Taken as a whole however, they provide strong evidence that unconstrained evolution is likely to exploit any physical characteristic that can influence circuit behaviour. Exploitable characteristics are not confined to the basic elements and interconnection architecture. They are present in the entire evolutionary environment. The implications of this are clear; it is infeasible to produce a reconfigurable architecture for unconstrained analogue hardware evolution, that has a 'transparent' interconnection architecture. If we are to design and build evolvable analogue machines for engineering applications, we must either provide precise physical models of them, or constrain their use. Some constraints are inevitable for ensuring reliable op-

eration within specified bounds, but they should not prevent evolution from exploiting electronic components and architectures in novel ways. It seems more beneficial to accept that evolution *will* exploit unusual environmental resources and to attempt to understand the processes that cause this, than to impose constraints of such severity that its capabilities may be limited to producing circuits we already know well how to design. The next chapter explores in detail some of the underlying processes that give evolved circuits qualities that are absent in their conventionally designed counterparts.

Summary of Chapter 5

A experimental program was conducted to illustrate the EM's capabilities as a research tool for addressing a variety of fundamental issues affecting HE. First, the relative evolvability of two different interconnection architectures is explored for inverter evolution. The results suggest that a directly-mapped comprehensive architecture seems less evolvable than one that restricts the average quantity of connections to basic element pins; the former yields a search space containing an excessive amount of circuits that short out the basic evolutionary elements.

An inverter evolved using the more restrictive mapping failed altogether to function if any constituent transistor was removed, but the (genetically converged) final population contained individuals relatively unaffected by removal of the same transistors. This phenomenon, if generally true, could be exploited to render evolved circuits more portable or fault tolerant.

A direct comparison of intrinsic versus extrinsic HE was performed to determine the factors which affect the relative speeds of the two modes, and the resulting circuits' portability. Two situations where extrinsic HE might be the faster are conjectured; the first where behaviour over long time-scales is desired, the second where evaluation noise prevents the intrinsic search from retaining a stable prototype. The second conjecture was tested empirically using physical and virtual EMs. Extrinsic HE consistently required an order of magnitude fewer generations to achieve 90% of maximum fitness than intrinsic HE. Analysis of ancestor circuits from an extrinsic run showed how a high-fitness ancestor was a minor modification of one with fitness much lower than the intrinsic noise threshold. The low fitness prototype would have been ousted by a worse but noisy circuit in intrinsic mode. Portability in an engineering context was compared by implementing extrinsically evolved amplifiers in physical hardware, and intrinsically evolved ones with different though nominally identical basic elements. While the subsequent drop in fitness was less marked for the intrinsic case, approximately the same number of generations was required for both modes to re-evolve to previous fitness in the new medium.

The final experiment investigated HE in the absence of components normally considered essential for a given task. Using additional evaluation hardware to prevent aliasing errors, a series of oscillators were evolved which achieved the target frequency to within 1%, even though evolution was denied the use of capacitors and resistors.

Though not designed to be conclusive, the experimental program achieved some milestones; the first intrinsically evolved transistor circuits; the first assessment of the relative merits of intrinsic versus extrinsic HE through direct comparison; the first evolved oscillators to achieve their target frequency. This program demonstrated the EM's effectiveness for addressing HE issues, yielding useful insights into genotype encoding, portability, evolved topologies, exploitation of

configuration circuitry, and the influence of evaluation noise.

Chapter 6

Inherent Qualities of Evolved Circuits: Evolutionary History as a Predictor of Fault-Tolerance

---

The principal ideas of the thesis are now gathered together in a detailed investigation of a complex new phenomenon, hinted at in the previous chapter. Novel techniques of analysis, the evolvable motherboard, and the evolution of small circuits are combined to produce a startling conclusion: Genetically converged populations of certain generic classes of evolved circuit are inherently tolerant to major faults where individuals within the populations are not. Moreover, through investigating the mechanism underlying this phenomenon, the types of fault for which it can be expected to occur can be predicted with a high level of confidence.

6.1 Inherent Qualities of Electronic Circuits Defined

In principle, the well known techniques of abstraction and analysis in conventional circuit design are capable of producing circuits which conform perfectly to a given set of behavioural specifications the very first time they are implemented in hardware or simulation. By contrast, circuit design by artificial evolution requires the implementation of *every* candidate design, to determine which ones to select for further cycles of evaluation and modification. When compared with conventional design, the evolutionary approach may be deemed an *incremental* process. Even where in practice, conventional design incorporates an apparently similar phase of trial-and-error, any modifications are made with an *expectation* — based on knowledge or analysis — of their consequence on the circuit's behaviour. Hence, applicable modifications are limited to a subset containing those that can be analysed or understood. However, modifications by evolutionary operators are made with no *a priori* expectations of their consequences. Evolution does not need to analyse its modifications — it only needs to *observe* the subsequent circuit behaviour. Hence, its modifications come from a very different subset, containing all those permitted by the architecture, genotype⇒phenotype mapping, and genetic operators, regardless of whether they can readily be analysed or understood. The fundamentally different nature of these two design processes is reflected in the kinds of circuits they produce. Both exhibit *inherent* qualities, which are not explicitly specified as behavioural or nonbehavioural requirements, but arise from the procedural implementation of the design method. For example, conventional design generally derives circuits

with a clear functional decomposition (see section 3.2).

Some inherent qualities may be undesirable; we have seen examples throughout this thesis of circuits that are not portable or whose behaviour changes with temperature. In the next section, I will argue that isolating and understanding inherent qualities can reveal promising new directions for HE research. To justify the argument, a potentially highly desirable quality will be shown empirically to be both inherent and predictable in circuits evolved under certain conditions.

6.2 Rationale

As the field of hardware evolution progresses as an engineering methodology, it becomes increasingly important to ascertain and understand areas in which it excels. As we have seen in section 2, a large proportion of the field is devoted to areas that are problematic for conventional design (for example Higuchi and Kajihara (1999), Sipper et al. (1998)), but comparatively few analytical tools are available to help determine whether such areas are any less problematic for HE. Isolating and understanding inherent qualities is a beneficial approach to making such determinations, since many inherent qualities are likely to be common to general classes of evolutionary algorithm, rather than just resulting from specific nuances of some particular method. Furthermore, inherent qualities by definition arise 'for free' from the design process. Where an inherent quality is desirable, an understanding of the mechanisms that produce it may permit tailoring the design strategy to encourage it further. However, there are no universal methods for providing such understanding. Useful insights are gained by analysis of evolutionary runs performed on fitness landscapes such as Royal Road (Nimwegen, Crutchfield, & Mitchell, 1999), NK (Kauffman, 1993), and NKp (Barnett, 1998), for example (Thompson, 1995b; Vassilev, Miller, & Fogarty, 1999). All of these share certain characteristics with real design space, but otherwise bear little resemblance to it. I suggest an alternative first step of applying the analysis tactics of chapter 3 to observe the phenotypic changes in behaviour and internal operation at *intermediate* stages of evolutionary history. This approach tracks the movement of a population though *real* design space, and therefore has the disadvantage of being difficult to formalise, but it is likely to provide answers as to why, for example, transistors are connected in unusual ways, or parasitic properties so often exploited. Vassilev and Miller (2000), in remarking on the importance of landscape neutrality for the evolutionary design of digital circuits, have posed similar questions: How exactly do circuits change during neutral walks? How does evolution preserve any attained circuit modules? Phenotypic analysis at interim historical points can tackle these questions for specific cases, and if applied methodically may lead to more general notions of how we can expect evolved circuits to be. This tactic will prove particularly powerful in the following study.

6.3 Populational Fault Tolerance

As a case study, this chapter endeavours to isolate and understand a potentially highly desirable quality inherent in circuits designed by Genetic Algorithms. Hereafter referred to as 'Populational Fault Tolerance' (PFT), it is the potential for a population of evolved circuits to contain an individual that adequately performs a task in the presence of a fault that causes the fittest individual to fail. The study was originally motivated by the experiment described in section 5.2, in which some

individuals in the final population of inverters maintained good fitness scores upon the removal of a transistor, even though the fittest member of that population suffered a catastrophic fitness drop. The phenomenon was manifested again when the final populations of extrinsically evolved amplifiers were used to seed intrinsic evolution (section 5.3.2). In each case, the previously best individual gave fitness scores equating to constant-output circuits, while other individuals exhibited a degree of amplification in the new environment. The study was also motivated by reports of similar effects from other authors, for example the tone discriminator circuit. When using the final population at generation 5000 to configure a completely different 10×10 region of the same FPGA, Thompson (1997a) observed that:

> When used to configure this new region, the individual in the population that was fittest at the old position deteriorated by $\approx 7\%$. However, there was another individual in the population which, at the new position, was within 0.1% of perfect fitness. Evolution was allowed to continue at the new position, and after only 100 generations had recovered perfect performance.

In a series of experiments using genetic programming to evolve movement controllers for random-morphology robots, Dittrich, Bürgel, and Banzhaf (1998) reports a case where an important structural joint broke during an evolutionary run. The breakage caused a significant, but not catastrophic drop in fitness, and fitness scores recovered to pre-break levels within 250 generations. In this case it is not reported which individual in the population performed best when the break occurred.

If a fault that manifests PFT is persistent, the new 'best' individual may be used to seed further evolution, possibly attaining performance equal to that before the fault occurred in a fraction of the time it took to evolve the circuit from scratch (Thompson, 1997b).

The phenomenon of PFT and the notion that it may be an inherent quality of evolved circuits was first postulated in Layzell (1998b). A preliminary study was conducted (Layzell, 1999a), and provided enough evidence to justify the more detailed study documented here. The next section analyses over 80 test cases on which faults are effected, to determine whether PFT is truly an inherent quality of evolved circuits, and examines various hypotheses for the underlying mechanisms that produce it. The results produced by the final experiment, which examines how constituent components were added to the design during evolutionary history, allows us to predict whether a given component will manifest PFT when faulty, with a high degree of confidence. The conclusion is that PFT is indeed an inherent and predictable quality of certain generic classes of evolved circuit. While it is hoped that PFT may eventually supplement existing fault tolerance techniques (for example Avizienis (1997), Ortega and Tyrrell (1999), Bradley and Tyrell (2000)), the primary aim of the case study is to *understand* it, not to assess its usefulness.

6.4 The Experimental Framework

The study in the following sections comprises two elements. The first is an empirical study consisting of many evolutionary runs of several different tasks. On completing the runs, hardware faults are effected, and data is taken concerning the subsequent performance of individuals in the final population and some of their ancestors. The second element is an analysis of the evolved

circuits to determine the relative importance of their constituent components, thus providing an indication of the severity of a given fault. The runs were carried out intrinsically using the evolvable motherboard. The EM is ideal for such a study as faults can be effected by physically removing the plug-in components and/or replacing them with faulty ones. Alternatively, the connections can be re-routed by software, which allows the effective removal or substitution to be carried out automatically. Both methods were used in the following experiments. As in the previous chapters, only bipolar transistors (and the analogue switches in a static state) were provided as basic elements, up to five each of PNP and NPN. The tasks used in the study — digital inverters, inverting amplifiers, and oscillators — were chosen as test cases for the following reasons:

- They are simple enough to evolve many times from scratch within a tractable time-scale
- They pose increasingly difficult challenges for fault tolerance. Faults are expected to be particularly troublesome for oscillators if they rely for their operation on chains of components within a loop
- They are expected to exploit the transistors in different ways: the inverters to use saturation mode, the amplifiers to use linear mode, and the oscillators to exploit parasitic capacitance.

The tasks were evolved using the same EA as in previous chapters; a rank-based, generational GA with elitism, and an initially random population of 50 individuals. Genetic operators were mutation (probability 0.01 per bit) and single-point crossover (probability 1.0). The genotype⇒phenotype encoding from the previous chapter (section 5.2) is also retained, using either 48 EM rows/columns with 3 additional switches per row, or 40 rows/columns with 4 additional switches, giving genotype lengths of 1056 and 1160 bits respectively. The inverters, amplifiers and oscillators were evaluated as described in sections 5.3.1, 5.3.2 and 5.4.1 respectively. For all tasks, the supply voltage was set to +2.8V, and the EM output was buffered by an FET-input operational amplifier.

6.4.1 Verifying the Existence of PFT

The first experiment was devised to give an idea of the extent (if any) to which PFT occurs for the test cases described above. The fittest circuit from the final population of each test case is subjected to a series of emulated transistor faults. This circuit is hereafter referred to as the BBF (Best Before Fault) circuit or individual. The final population is then re-evaluated in the presence of each fault to establish whether it caused the BBF circuit to fail and if so, whether other members continue adequately to perform the task. In this case, PFT is manifested. Only the *constituent* transistors of the BBF circuit will be subjected to faults. A transistor is considered constituent if its removal gives a measurable reduction in the BBF's fitness.

Multiple runs were conducted until each task had been successfully evolved 20 times. By the end of each run, the population had settled at a stable level of genetic convergence.

The BBF circuit's constituent transistors were then classified by analysing their role in the evolved circuits. Transistors wired such that their collector-emitter current was clearly modulated by base voltage were termed *active*, and the rest *passive*. Faults in both types may be equally severe in terms of their effect on the performance of a single individual. However, if passive transistors are configured as 'wires', a population may contain individuals of similar fitness that

bypass them with configuration switches. It is important to establish whether PFT is limited to such simple cases. Constituent transistors were rendered faulty simply by removing them. This is equivalent to a fault common with bipolar transistors — both PN junctions becoming open-circuit.

The extent to which reduced performance of a circuit in the presence of a fault can be considered *adequate* is clearly dependent on the task. Here, the minimum adequate performance is defined as corresponding to a fitness score significantly above that obtained by a trivial circuit giving constant output. Such a circuit would be an appropriate seed for further evolution, even if it were no longer suitable for the intended task in its existing state.

Figure 6.1 displays the BBF circuit's constituent transistors for each run, each transistor represented by a circle. The circles are shaded according to the effect the transistor's removal (the fault) had on the final population's performance as follows:

1. **Grey** The fault caused every individual in the population to fail
2. **Black** The BBF circuit failed, but one or more other members of the population performed adequately
3. **Concentric circles** The BBF performed adequately, but one or more other members of the population gave superior performance
4. **White** The BBF performed adequately, and was superior to any other member of the population

For example, the BBF amplifier circuit of run 7 contained two constituent transistors, Q4 and Q9. When Q9 was faulty, every individual circuit in the population failed (case 1), and when Q4 was faulty, the BBF performed adequately, but another member performed better (case 3). PFT occurs for circuits in which we see case 2 above, although its definition could arguably be extended to case 3. I have avoided doing so because we know from section 2.1.3 that there are also mechanisms whereby *individual* evolved circuits may be inherently tolerant to faults (see Thompson (1997b) for more details). There would be a degree of uncertainty whether fault tolerance presumed to be PFT is in fact attributable to some other process. However, I have distinguished between cases 3 and 4 in the diagram to allow the reader to judge.

In accord with the original definition of PFT then, we are only concerned with the black and grey areas — faults that rendered the BBF individual useless. The larger the ratio of black versus grey for a given set of runs, the better the PFT. This ratio is more easily discerned in figure 6.2, where the faults are averaged over all runs for each task and displayed as a pie chart.

The diagrams show enough evidence of PFT for the test cases to justify further analysis. However, there is one significant concern. In nearly all cases where the entire population was rendered useless, the transistor removed had been active rather than passive. It is possible that faults in active transistors are too severe for PFT to occur, in which case its potential as a strategy for producing robust populations of circuits would be limited. Fortunately, the next sections show this concern to be unfounded.

6.4.2 Examining Various Hypotheses to Explain How PFT Occurs

Having established evidence for the existence of PFT, we now seek to understand the underlying mechanisms that produce it. The experiments described so far have been carried out using a particular evolvable architecture. We cannot predict the extent to which PFT might arise using

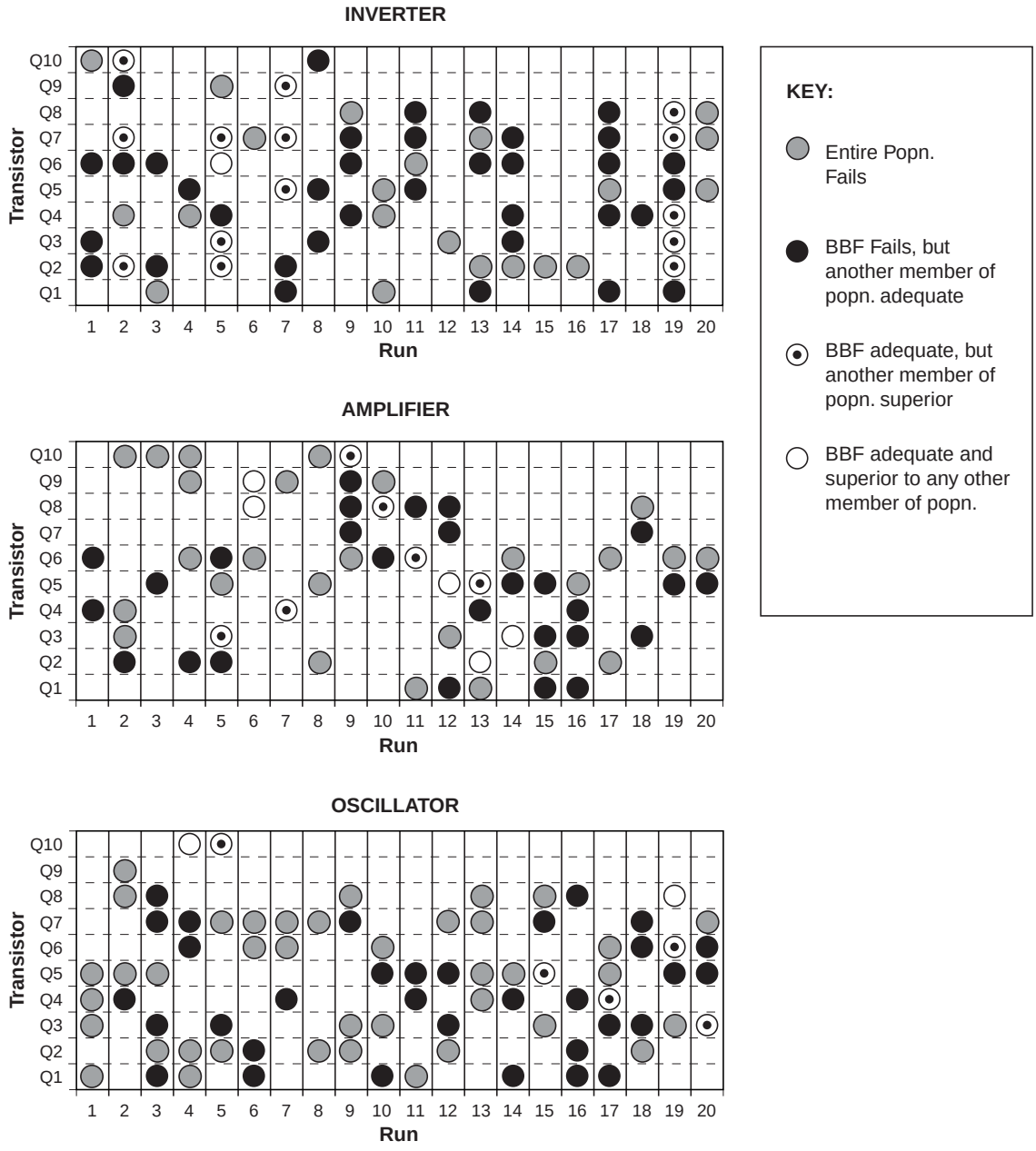


Figure 6.1: Constituent transistors for each of 60 test runs. A circle represents a single transistor and is shaded according to the effect the transistor's removal had on the population's performance.

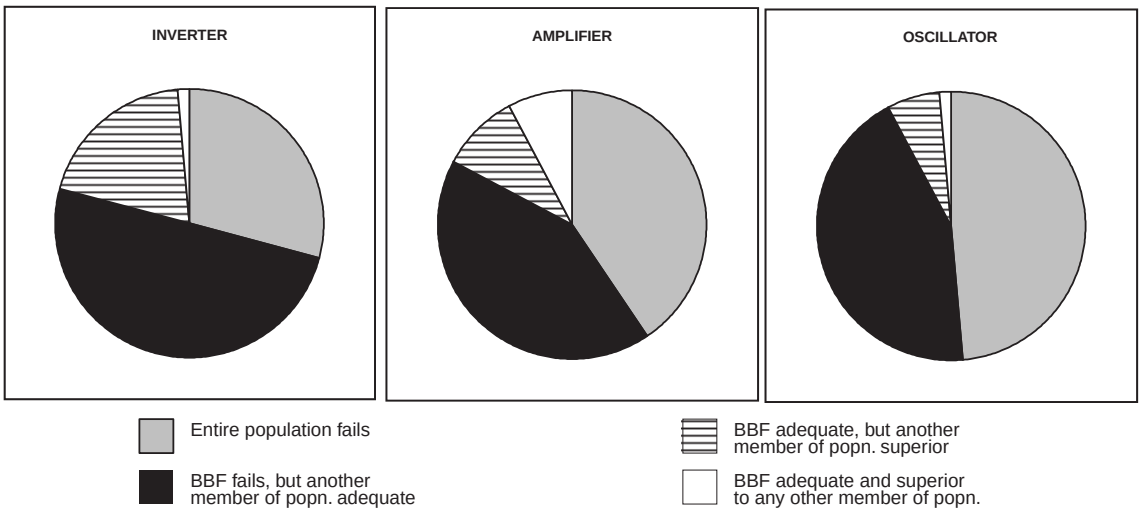


Figure 6.2: Pie charts showing the relative proportions of fault outcomes for each task, calculated by counting all occurrences of each outcome over the 20 runs

other architectures that have more or less redundancy, or different constraints on component use. Therefore, the search to explain PFT is simplified by confining further analysis to apply only to the architecture on which the test cases were evolved. Given a constant architecture, we can propose several hypotheses to explain how PFT occurs. Those that appear most likely are listed below:

H1 Although the final GA population was relatively genetically converged, it still contained diverse solutions.

H2 PFT is the result of the incremental nature of the evolutionary design process, as described in section 6.1 above. Configurations of ancestor circuits that did not make use of the transistors whose removal resulted in PFT still exist relatively undisturbed in the final genotypes.

H3 PFT is not an inherent quality of evolved circuits. It is a likely outcome of any design implemented on this architecture. Hence, a population of mutants of conventional circuits could be expected to give similar results.

H4 PFT is the result of some peculiarity of the evolutionary algorithm employed, for example the crossover operator.

H5 The transistors whose removal manifested PFT were not used in a functional capacity, but merely acted as resistances. They could be bypassed by mutation with little difference in circuit performance.

A thorough comparison of the BBF individuals' schematics with those that performed better in the presence of a fault showed that H5 was rarely the case. Indeed, the oscillator circuits consisted mainly of active transistors whose removal very often manifested PFT. The following experiment addresses H1 and H4. Ten each of inverter and amplifier circuits were evolved using a $(1+1)$ Evolution Strategy (ES) (Schwefel & Rudolph, 1995) with the same mapping. In the ES, each new variant was produced by applying bit mutations to the (initially random) parent with the same probability as for the GA[1]. It was not possible to evolve 10 oscillators using this method.

[1]The ES can be also be regarded either as a hill climber with random ascent, or as the same GA as used above, but with a population size of two and no crossover; this GA was generational with a rank selection function such that the

On completing the runs, 49 copies of the final individual were mutated (using the GA mutation rates) to form a population of 50 including the final individual, and the previous experiment was repeated using these populations. The mean hamming distance between two pairs of randomly selected individuals from this population is approximately 20, while for the GA it was of the order 200. The genotypes of the GA population are the more diverse, but the corresponding circuit configurations are expected to be converged (see section 5.2). Using the ES is intended to confirm this, provided that crossover is not a major factor: If the GA population contains diverse solutions (H1), or if PFT is due to the crossover operator (part of H4), then circuits evolved using the ES should not manifest PFT.

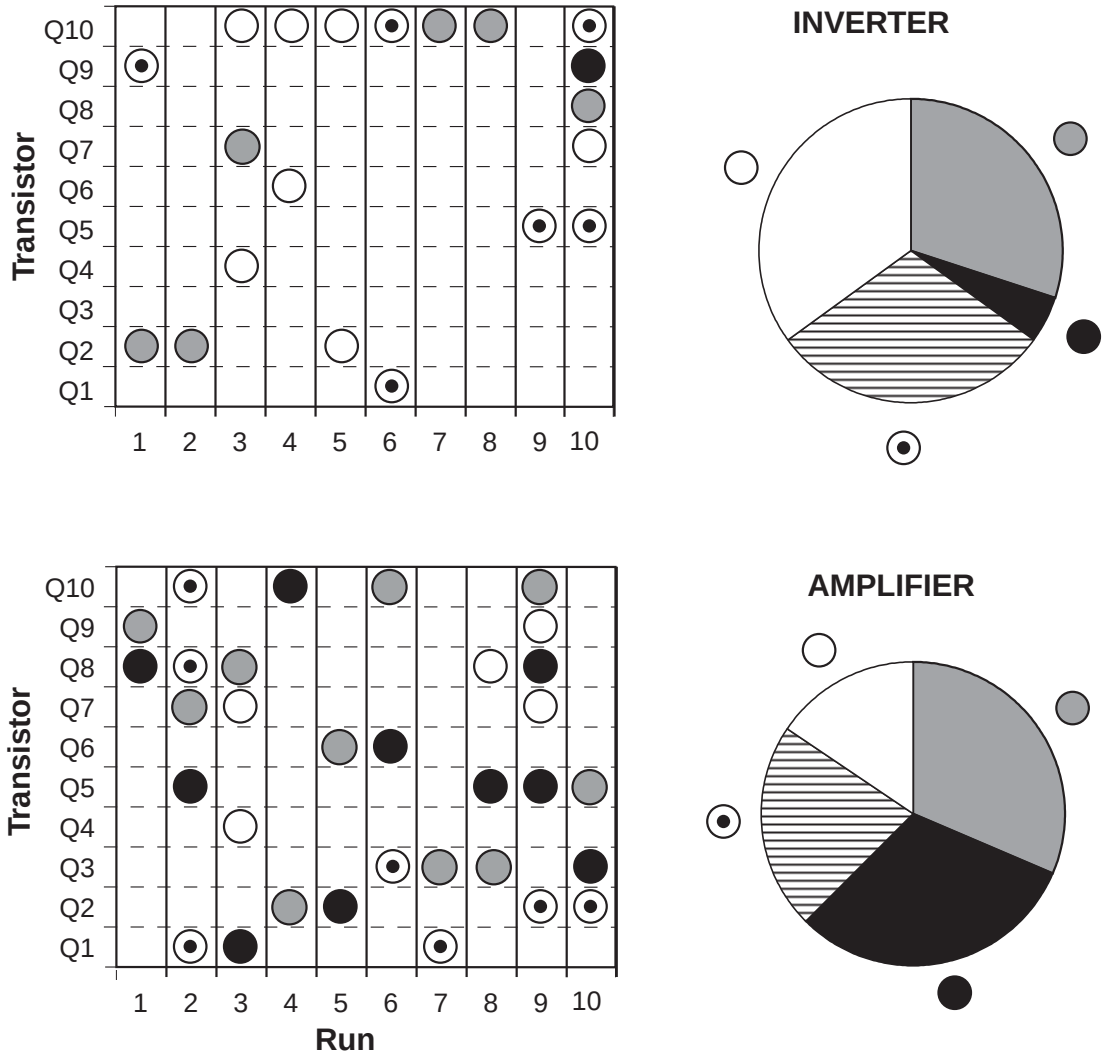


Figure 6.3: Constituent transistors of 20 ES runs, shaded as for figures 6.1 and 6.2.

Figure 6.3 shows the results for the 20 ES runs. The inverter does not help greatly to distinguish PFT faults, as very few severely affected the BBF individual (the ES parent). The reasons for this are known. The ES tended to produce inverter circuits with two transistors in parallel, the fractionally higher voltage swing attained giving slightly higher fitness. Hence removing ei-

lowest scoring individual had zero probability of selection. However, generational, rank selection GAs can exist for which the lowest ranking individual has a non-zero selection probability. There are also many variants of hill climbing algorithm, hence the algorithm is best described as an ES.

ther transistor made very little difference to performance. However, it is not known *why* the ES produced this type of circuit while the GA did not. While it is interesting that the ES produced *individual* circuits tolerant to faults, this is a different form of robustness to that which we seek here.

The amplifier shows similar results to the GA. Indeed for both tasks, a maximum of only one constituent transistor rendered the entire population useless when removed. These results imply that PFT is not due to any more diversity in the population than is present in one set of mutants of a single individual. Also, since crossover was not used by the ES, it too is ruled out as a factor.

6.4.3 Using Evolutionary History to Predict PFT

While hypotheses H1, H4 and H5 are not conclusively ruled out (particularly H4), the evidence against them makes the remaining two the most likely candidates. H3 proposes that PFT may not be due to the evolutionary design process at all, but is more a function of the circuit implementation. One approach to determining whether this is the case could be to describe a number of conventionally designed circuits by genotypes with identical mapping to that used for the first experiment. A population of mutants could then be made and instantiated in hardware, and the previous experiment repeated. However, there are a number of insuperable difficulties associated with this approach. Firstly, hand-designing a sufficient quantity of different circuits using only transistors and motherboard switches to provide statistically valid data would be extremely difficult. Additionally, care would have to be taken to extend the redundancy which evolution was afforded to the conventionally designed equivalents (the evolved circuits employed on average only about half of the available transistors). For example, a conventional circuit could be arranged on the EM such that its constituent transistors were geographically adjacent. If another arrangement of the same circuit were used — say alternate transistors — then different behaviour under mutation would be unsurprising.

Instead, consider H2: If the transistors whose removal manifests PFT are indeed later additions to some early ancestral prototype whose configuration remains largely intact, then there should be a correlation between such transistors and the point in evolutionary history at which they were incorporated. Transistors adopted by the earliest solutions would unlikely to manifest PFT when faulty because there would be no ancestors for which such transistors were redundant, that came close to achieving the task. The reverse is true for transistors adopted at the latter stages of evolution since many suboptimal solutions with no need of these transistors would have appeared in preceding generations. If one of them is present largely unchanged in the final genotype, then PFT is very likely. The following experiment was devised to test for such a correlation. Having completed an evolutionary run, the best circuits in previous generations are examined to determine which of the available transistors was constituent to a given circuit. Performing the examination for every generation builds up a profile of any particular transistor's importance to the best individual at all stages during the evolutionary history of that run. The completed 'importance profiles' will then be used to assess H2.

It would be too laborious and prone to errors to carry out the assessment by analysing the schematic of every individual. But the process can be automated by applying similar techniques as were used above when testing for PFT. One transistor at a time is disconnected from the best

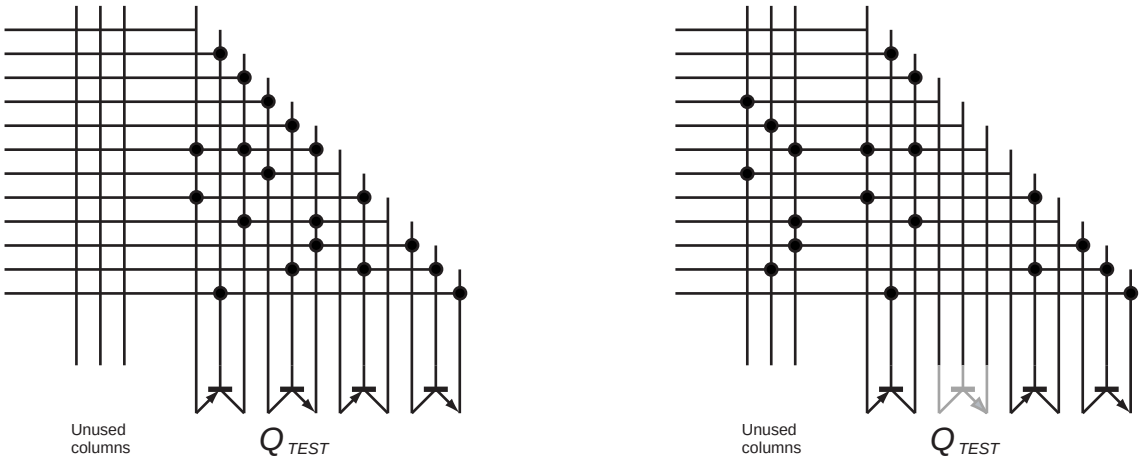


Figure 6.4: By re-routing the column connections for a given transistor Q_{TEST}, to unused columns, the transistor is effectively disconnected from the circuit.

individual of a given generation, then the individual is re-evaluated. If the individual's fitness is consequently reduced, then the disconnected transistor is assumed constituent. The importance of a given transistor is given by plotting the *reduction* in fitness (the original fitness minus the re-evaluated fitness with the transistor disconnected) for each generation. Hence a non-constituent transistor will not affect the best individual's fitness when removed, giving a reduction of zero. Likewise, removing a constituent one will reduce the best individual's fitness, giving a positive value.

To avoid manually removing and replacing transistors, the process was automated by re-routing the EM connections from the transistor in question to unused columns as shown in figure 6.4. This was only possible with ten of the twenty runs per task, since the other ten used all 48 of the EM columns. In practice, sufficiently accurate profiles could be constructed by carrying out the process every 5 generations.

6.4.4 Results

Three plot sets typical of those produced are given here, one for each of the three tasks. More appear in appendix D. Figure 6.5 shows the process applied to run 14 of the inverters. The upper trace is a plot of the best individual's fitness with all transistors intact, while the lower traces are of the reduction in that individual's fitness upon removal of a given transistor. Plots of fitness reduction are given here for each of the eight transistors, whether or not it was constituent in the final circuit. The graphs show clearly how the transistors were incorporated into the design as evolution progressed. The first improvement on constant-output — around generation 550 — required only Q2. At generation 700, Q1 and Q3 were briefly included and a few generations later Q1, Q7 and Q8 were added to Q2 as constituent elements. Just after generation 850, a major design change occurred. Q3, Q4 and Q6 were incorporated, and at the same point Q1 and Q8 were dropped.

Similar plots are observed for the amplifier in figure 6.6. Constant-output circuits are produced until generation 550, at which the first stable prototype is found using Q3 and Q5. Various transistors are incorporated and dropped in later generations, with major contributors being Q1,

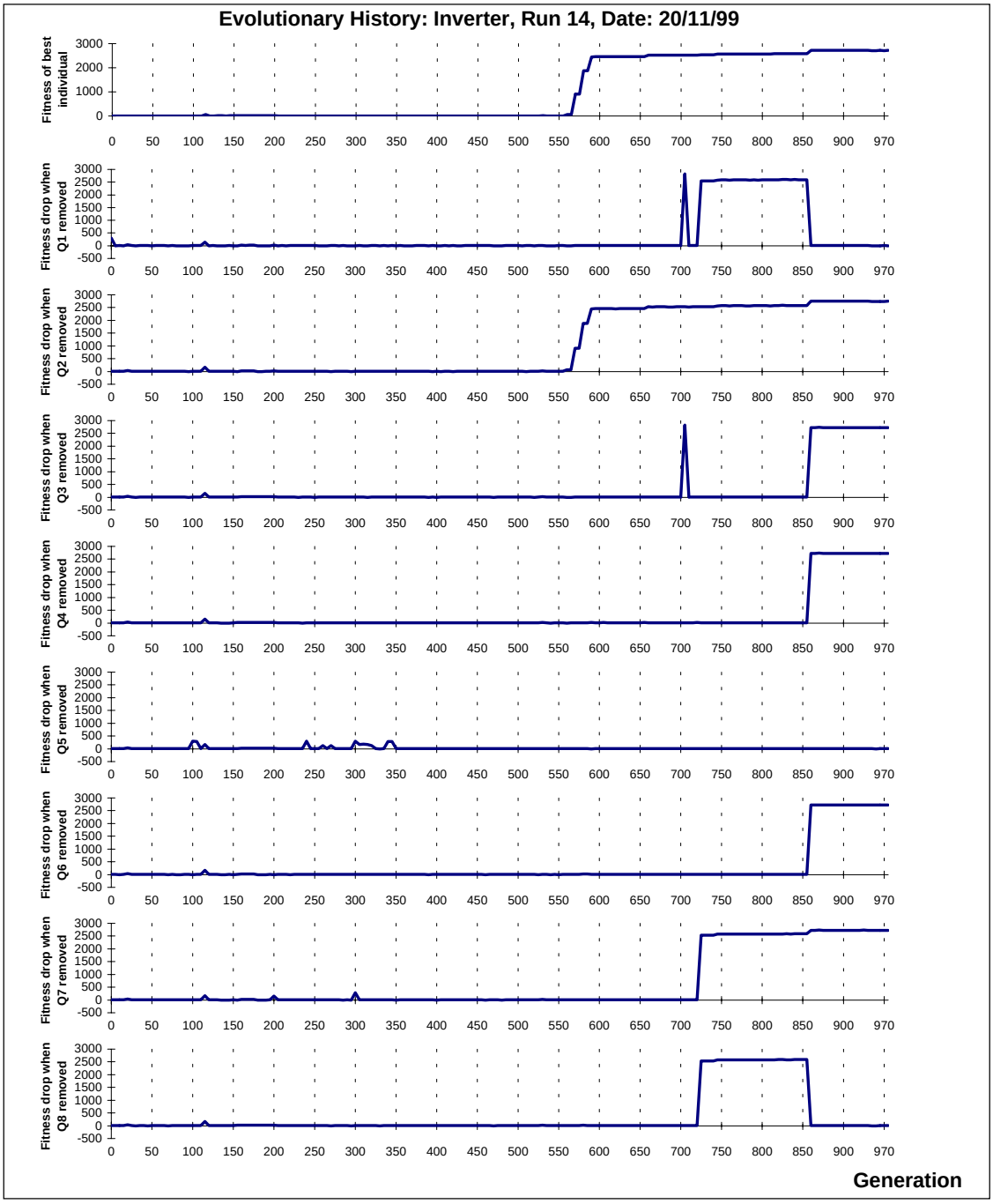


Figure 6.5: Plots indicating the stage of evolutionary history at which constituent transistors were incorporated into the design of an evolving inverter. The top plot is the fitness of the best individual of each generation with all transistors present. Below it are plots of the *drop* in that individual's fitness with respective transistors removed

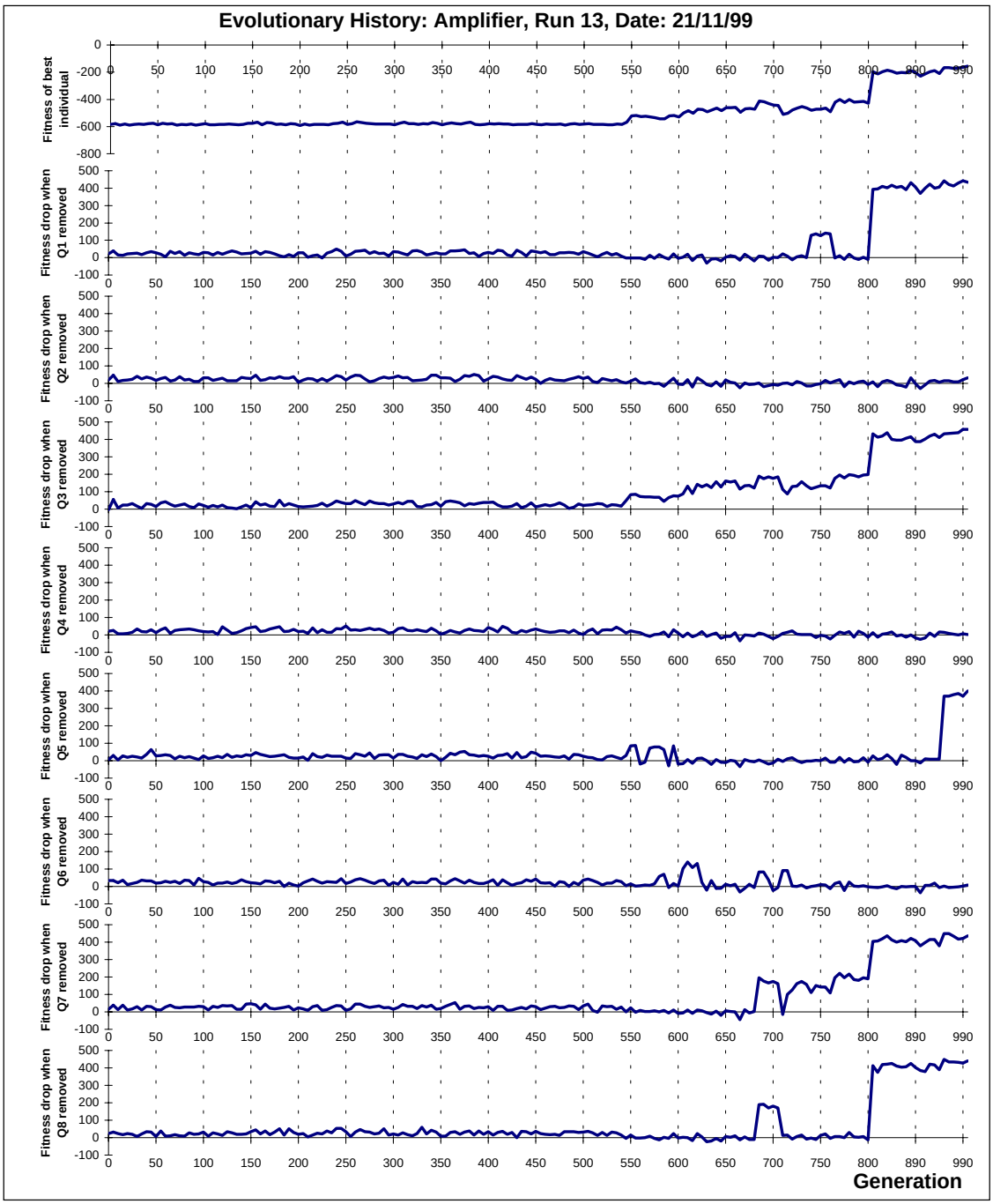


Figure 6.6: Fitness of best individual for each generation and transistor importance profiles for one of the amplifier runs

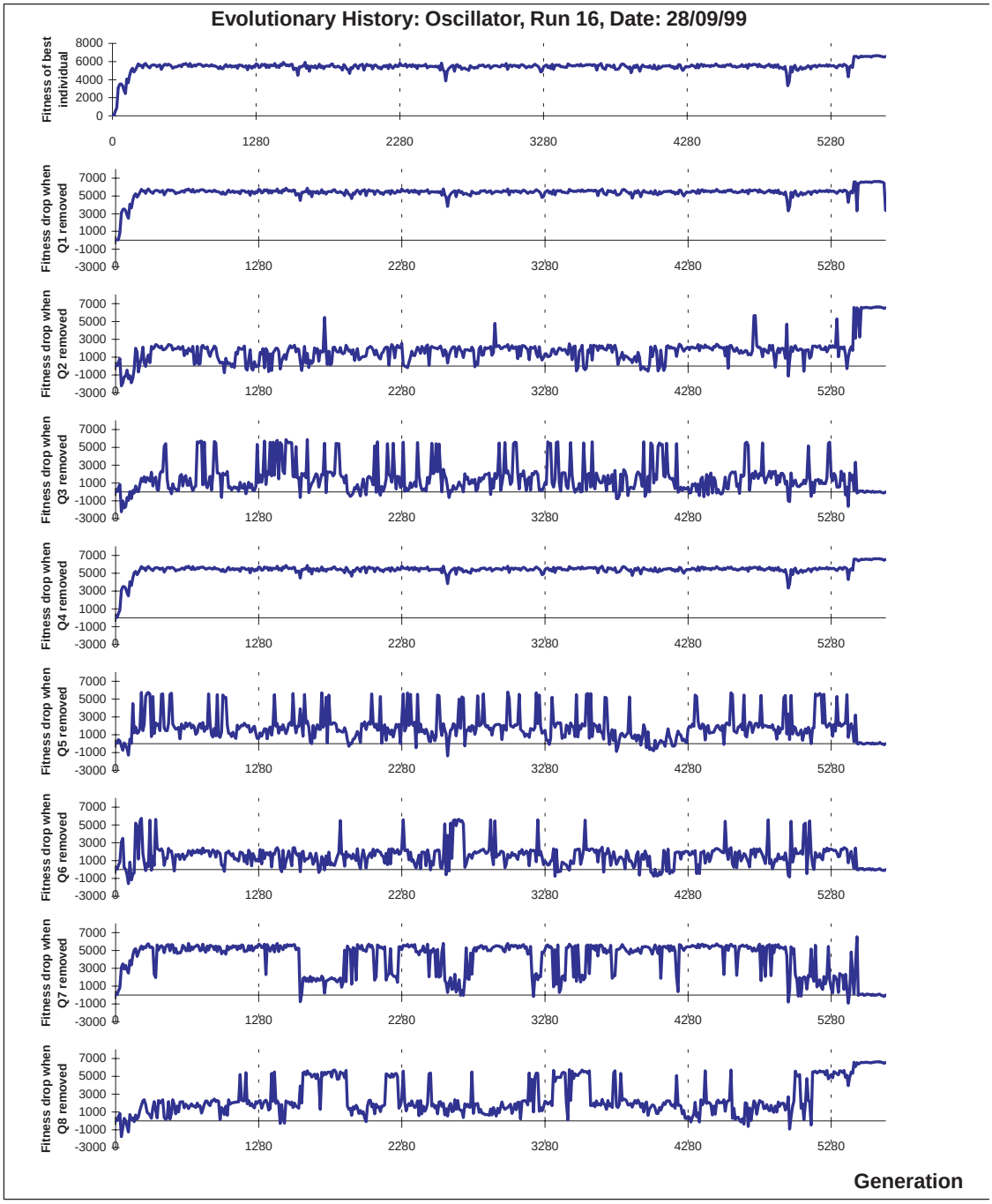


Figure 6.7: Fitness of best individual for each generation and transistor importance profiles for one of the oscillator runs

Q3, Q7 and Q8 at generation 800, and Q5 at generation 850.

The oscillator plots of figure 6.7 are somewhat more difficult to interpret. Q1, Q4 and Q7 seem to be major circuit components at most stages in the evolutionary history while the others appear to have minor, transient roles. Until approximately generation 5500, relatively unstable oscillators were produced. Removing one of the spikey-profiled transistors caused a minor change in the resistance or capacitance at some point in the circuit. For some configurations, the change was sufficient to stop the circuit oscillating altogether, resulting in a severe fitness drop. For others, the oscillation frequency was altered, reducing the fitness less substantially. To alleviate concerns that the profiles were noise, due to external influences (such as radio-frequency signals, humidity, or power-supply variation), the transistor disconnection/re-evaluation process was later repeated for all of the oscillator runs. Identical plots were produced. At generation 5500, a major improvement occurred, resulting in reliable, stable oscillation. After this point, the constituent transistors are clear: Q1, Q2, Q4 and Q8.

For each of the tasks, we observe at least one transistor whose profile is identical (disregarding measurement noise) to the plot of best fitness for that task: Q2 for the inverter, Q3 for the amplifier and Q4 for the oscillator. This was observed in all but one of the 30 runs: One or two of the transistors forming the prototype remain an essential part of the design while it is improved by the gradual addition of other transistors. The sole exception was inverter run 19, for which a low-fitness prototype formed around one transistor was superceded entirely in the early stages of evolution (see appendix D).

We are now in a position to test hypothesis H2. If H2 is correct, then we can predict that transistors whose importance profiles are identical to the plot for best fitness will *not* manifest PFT when faulty since they have been a constituent component of every generation's best circuit from the earliest prototype, while any others (whose failure renders the BBF individual useless) will. This prediction was carried out for each fault represented by a grey or black circle on figure 6.1, runs 11-20. Table 6.1 displays the predicted occurrences of PFT alongside the actual occurrences. Astonishingly, the prediction is correct for 26 of 30 faults (inverter), 26 of 27 faults (amplifier), and 28 of 34 faults (oscillator), or 87%, 96%, and 82% correct, respectively. This is strong evidence in favour of H2, though the other hypotheses may still have some small role. However the correlation is too close to be chance, hence H3 surely cannot be true: PFT is established as an inherent quality.

6.5 Discussion

The evolutionary history of candidate circuit designs showed clearly that once the GA had found a prototype circuit giving significantly better fitness than constant output, all subsequent designs were based on it. The accuracy with which it was possible to make predictions of PFT based on which transistors were essential components from the prototype onwards and which were later additions, is very strong evidence in favour of earlier circuits' configurations remaining largely intact in the final population. PFT is an inherent quality of evolved circuits produced by the evolutionary algorithms and evolvable architecture used for this study. Indeed, it is likely that PFT is inherent in a broader generic class of evolved circuits. Using the knowledge of PFT gained by

INVERTER			
Run	Q	PFT	
		Predicted	Observed
11	5	Y	Y
	6	N	N
	7	Y	Y
	8	Y	Y
12	3	N	N
13	1	Y	Y
	2	N	N
	6	Y	Y
	7	Y	N
	8	Y	Y
14	2	N	N
	3	Y	Y
	4	Y	Y
	6	Y	Y
	7	Y	Y
15	2	N	N
16	2	N	N
17	1	Y	Y
	4	Y	Y
	5	N	N
	6	Y	Y
	7	Y	Y
	8	Y	Y
18	4	N	Y
19	1	Y	Y
	5	Y	Y
	6	Y	Y
20	5	Y	N
	7	Y	N
	8	N	N

AMPLIFIER			
Run	Q	PFT	
		Predicted	Observed
11	1	N	N
	8	Y	Y
12	1	Y	Y
	3	N	N
	7	Y	Y
	8	Y	Y
13	1	N	N
	4	Y	Y
14	5	Y	Y
	6	N	N
15	1	Y	Y
	2	N	N
	3	Y	Y
	5	Y	Y
16	1	Y	Y
	3	Y	Y
	4	Y	Y
	5	N	N
17	2	Y	N
	6	N	N
18	3	Y	Y
	7	Y	Y
	8	N	N
19	5	Y	Y
	6	N	N
20	5	Y	Y
	6	N	N

OSCILLATOR			
Run	Q	PFT	
		Predicted	Observed
11	1	N	N
	4	N	Y
	5	Y	Y
12	2	N	N
	3	Y	Y
	5	Y	Y
	7	Y	N
13	4	N	N
	5	N	N
	7	N	N
	8	N	N
14	1	N	N
	4	N	N
	6	Y	Y
15	3	N	N
	7	Y	Y
	8	N	N
16	1	Y	Y
	2	Y	Y
	4	N	Y
	8	Y	Y
17	1	Y	Y
	3	Y	Y
	5	N	N
	6	Y	N
18	2	N	N
	3	Y	Y
	6	N	Y
	7	Y	Y
19	3	N	N
	5	Y	Y
20	5	Y	Y
	6	N	Y
	7	N	N

Table 6.1: Comparison of predicted and observed occurrences of PFT

this study, we can speculate that such a class might contain circuits designed by many incremental strategies, which gradually incorporate additional components into a basic initial prototype. We can also speculate that PFT would be less likely to occur for on-line evolution in a varying environment: Although later designs may be gradual improvements of a largely unchanged ancestor, the ancestor itself would be of little use were it a solution to a different problem. There is plenty of room for further understanding and improving PFT, and possible directions are indicated in the next chapter. But by applying analytical techniques to simple (but non-trivial) circuits, together with a tool flexible enough to carry out the analysis within a tractable time-scale, a new and complex phenomenon has been revealed and — to a degree — understood. PFT is one of possibly many desirable inherent qualities yet to be yielded by the evolutionary design process. Isolating and understanding others can only increase hardware evolution's effectiveness as an engineering tool.

Summary of Chapter 6

Electronic circuits exhibit inherent qualities, which are not explicitly stated as behavioural or non-behavioural requirements, but arise from the design procedure. Knowledge of the inherent qualities that circuits derived through various strategies are likely to possess, is potentially useful when determining the most appropriate strategy for a given task. In the context of HE, such knowledge helps to determine whether areas problematic for conventional design may be suitable for addressing through artificial evolution. Results from previous work by the author and elsewhere suggest that a population of evolved circuits may include individuals robust to major faults, even if the faults cause the population's fittest circuit to fail entirely. This phenomenon, named Populational Fault Tolerance (PFT), is investigated as a case study for the EM and analysis techniques presented

in this thesis. The study attempts to verify the extent to which PFT can be expected for a set of test cases, and whether it is an inherent quality of evolved circuits.

Multiple runs were conducted for three different tasks. On completing the runs, faults were effected by removing constituent transistors one at a time, and re-evaluating the final populations. Removing a transistor had a number of outcomes, one of which was the fittest individual failing entirely, but another individual still performing the task adequately. This outcome was a 'fault' that manifested PFT, and was observed for a significant proportion of faults in nearly every run.

Having established the existence of PFT, a number of hypotheses were postulated to explain it, and various tests devised to support or disprove them. One hypothesis — that the GA population contained diverse solutions — was evaluated by repeating the process above for test cases evolved with a $(1+1)$ ES. A larger population was then produced containing the ES parent and mutants of it. This experiment gave similar results to the first, and hence strong evidence against the hypothesis. Plots showing the stages in evolutionary history at which each transistor was deployed in the evolving circuit were then constructed. They confirmed that the final evolved circuits were derived from successive modifications to an initial prototype. For every run, the transistor around which the prototype was based remained fundamental to the fittest individual's operation in all subsequent generations. With very few exceptions, the removal of any other transistor either manifested PFT, or did not severely affect the fittest individual, but removing the transistor that was fundamental throughout caused the entire population to fail. These observations strongly support the hypothesis that PFT is the result of ancestor circuit configurations remaining largely intact in later genotypes, and establish PFT as an inherent quality.

Chapter 7

The Way Forward: Extended Experimentation and Future Work

7.1 Extended Study with Diverse Basic Elements

A major part of this thesis has been devoted to investigating analytical techniques and tools for the exploration of issues predominating HE research. Many of the experiments have been conducted to illustrate the capabilities of the techniques and tools, rather than comprehensively to resolve the issues in question. Consequently, there is plenty of scope to extend them. In particular, I have examined how the use of commercial configurable devices places restrictions on the sorts of circuits that can be evolved, with regard to the limited choice of basic elements and interconnection architecture, but have demonstrated only a few alternatives. Concerning basic elements, I have used bipolar transistors exclusively, because their operation is well understood, and because I wished to determine the extent to which evolution could exploit properties they possess — such as resistance and capacitance — without recourse to adding separate resistors and capacitors, which can be difficult to implement in VLSI. It would now be interesting to assess the impact of including them in combination with transistors. The resulting circuits could perhaps evolve faster, or more reliably, or may be easier to analyse if say, a capacitor could be switched into the circuit using one or two mutations, rather than evolution having to derive some complex transistor configuration to produce capacitance.

If resistors of high value compared with the configurable switch resistance were supplied, then the switches' own physical characteristics would be potentially less vital to an evolved circuit's operation. However, I doubt that this is the case. We have seen clearly how evolution will progressively tinker with configurations until precisely the correct value for a given property corresponding to optimal circuit fitness is achieved. In reality we cannot manufacture ideal electronic components; they all have some degree of resistance, capacitance and inductance, and evolution will use any component that possesses the desired property.

Empirical work is required to assess the merits of diversity in the basic elements. This would be easy to implement on the evolvable motherboard. Programmable resistors and capacitors exist, and the general-purpose configuration lines routed to every daughterboard (unused in this thesis) could be used to configure them. Equally, the use of different types of transistor, or more function-

level basic elements such as operational amplifiers has not been investigated here. Nearly all the experiments detailed in this thesis could be carried out using alternative basic elements, although direct intrinsic/extrinsic comparison would be problematic with elements less well modelled in simulation than transistors are.

7.2 Potential Benefits in Combining Extrinsic and Intrinsic HE

In section 5.3.2, a direct comparison between extrinsic and intrinsic modes of HE was conducted to determine whether common notions of the modes' relative merits could always be considered true. The results of the comparison also imply several potential benefits in *combining* the two modes. The first concerns speed of evolution in the presence of noise. Under the noiseless conditions of extrinsic evolution, initial prototype circuits are allowed to propagate even if they are only marginally fitter than constant-output circuits. Intrinsic prototypes must be considerably better at performing the desired function in order to propagate, because their fitness score must exceed that of constant-output circuits, plus a noise threshold. If they do not, their rank within the population will be exceeded by useless, but noisy circuits, and they will not survive further rounds of selection. If evaluation noise cannot be reduced to a negligible level, then a possible solution to the problem is to use extrinsic evolution to find a stable prototype whose fitness exceeds the noise threshold, after which intrinsic evolution could take over. This method is subject to certain conditions however: The extrinsic prototype would have to yield similar fitness scores when transferred to reality, and we have seen how this cannot be relied upon to happen unless the simulation precisely mirrors the physical components and environment. The predicament raises a second potential benefit of combining the two modes.

The experiments prompt an empirical framework by which we could choose *bounds* on the degree to which a simulation model should reflect the true physical characteristics of an electronic device or system. HE could be carried out in both modes simultaneously over a series of runs in which the accuracy of the simulation model is varied. It would be interesting to discover exactly how sophisticated a simulation model would be required to evolve a circuit that performs well in both modes. All we can say at present is that if the simulator does not accurately model *every* physical characteristic encountered, then success is not guaranteed. This observation is entirely unhelpful because such a simulation is currently infeasible and likely to remain so. The author has already conducted a small study wherein an evolving inverter was evaluated concurrently in simulation and physical hardware. Fitness was taken to be the lower of the two evaluation scores. The programmable switches were modelled by resistors as described in chapter 5, and the transistors by the standard library models included with the PSpice simulator. All other experimental parameters were identical to those of section 5.3.2. Results of the two runs are shown in figure 7.1. In one, the extrinsic phenotype was consistently the fitter of the two, resulting in a somewhat jagged fitness plot reflecting the lower, noisy fitness of the intrinsic. The reverse is true for the other run. In both cases however, concurrent evolution failed to achieve more than half of the maximum attainable fitness score, despite one of the runs being extended over 12000 generations. There is a clear implication that the simulation model used in the study was not sufficiently accurate. As mentioned in chapter 2, Stoica et al. (2000) have since reported success for a similar method ('mixtrinsic' evolution), and in a personal communication, conveyed that an accurate simulation

model was required to achieve it.

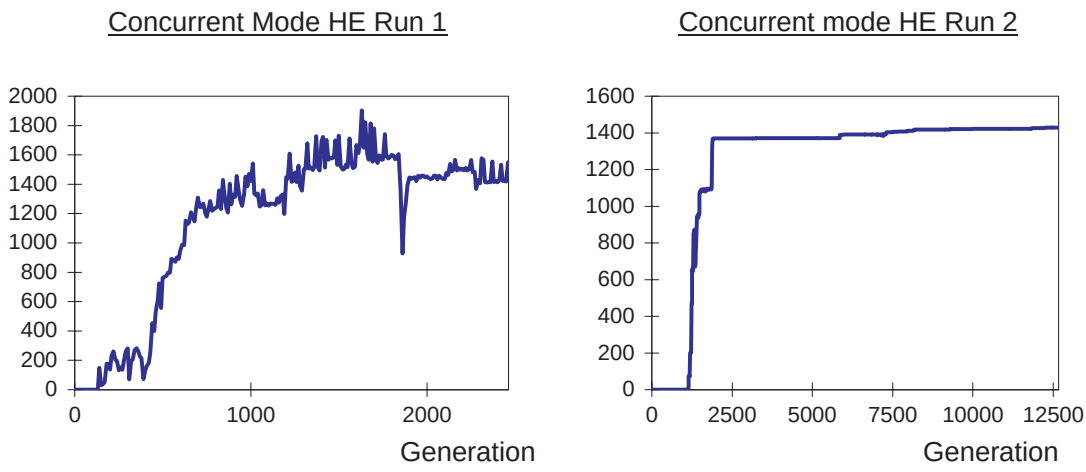


Figure 7.1: Maximum fitness of the population at each generation for inverter runs evolved concurrently in hardware and simulation

7.3 Landscape Neutrality as a Guide for Interconnection Architecture

I have stated at various points that reconfigurable architectures designed for the implementation of conventional circuits are not necessarily ideal for evolution. Only two alternatives have been explored in this thesis, so there is plenty of scope to use the EM's architectural flexibility to explore others. In conceiving the two adopted so far, I have endeavoured not to prejudice their design by knowledge of conventional electronic engineering. The first exhaustive architecture/mapping was eventually rejected because it encouraged the basic element pins to be shorted out, and its successor was chosen specifically to discourage such shorting. It is debatable whether this choice relied on knowledge of engineering circuits, or just common sense. While I have warned that applying too much domain-specific knowledge could result in search spaces inappropriate for artificial evolution, there are no other established approaches to the design of interconnection architecture. Therefore, I propose a new method for the conception of interconnection architectures (or equivalent mappings) that abandons domain-specific knowledge in favour of metrics we believe to result in highly evolvable search spaces.

The method attempts to represent a portion of the search space graphically, in a manner similar to figure 7.2, a hypothetical search space in which individuals above a certain fitness threshold are represented as vertices on a connected graph. Vertices are connected if the individuals they represent are genetically separated from each other by one unit of hamming distance. The vertices are also colour-coded according to a pre-determined fitness range to which the corresponding individuals belong. In this example, low-fitness individuals are black, suboptimal ones are grey, and optimal ones are white. A clustering algorithm is applied to the graph to create spatially distinct regions of highly connected vertex sets. The fitness threshold is necessary to ensure that the graph has structure. Clearly if all individuals were included in the graph, then each vertex would have an equal number of one-hamming-distance companions, and the clustering algorithm

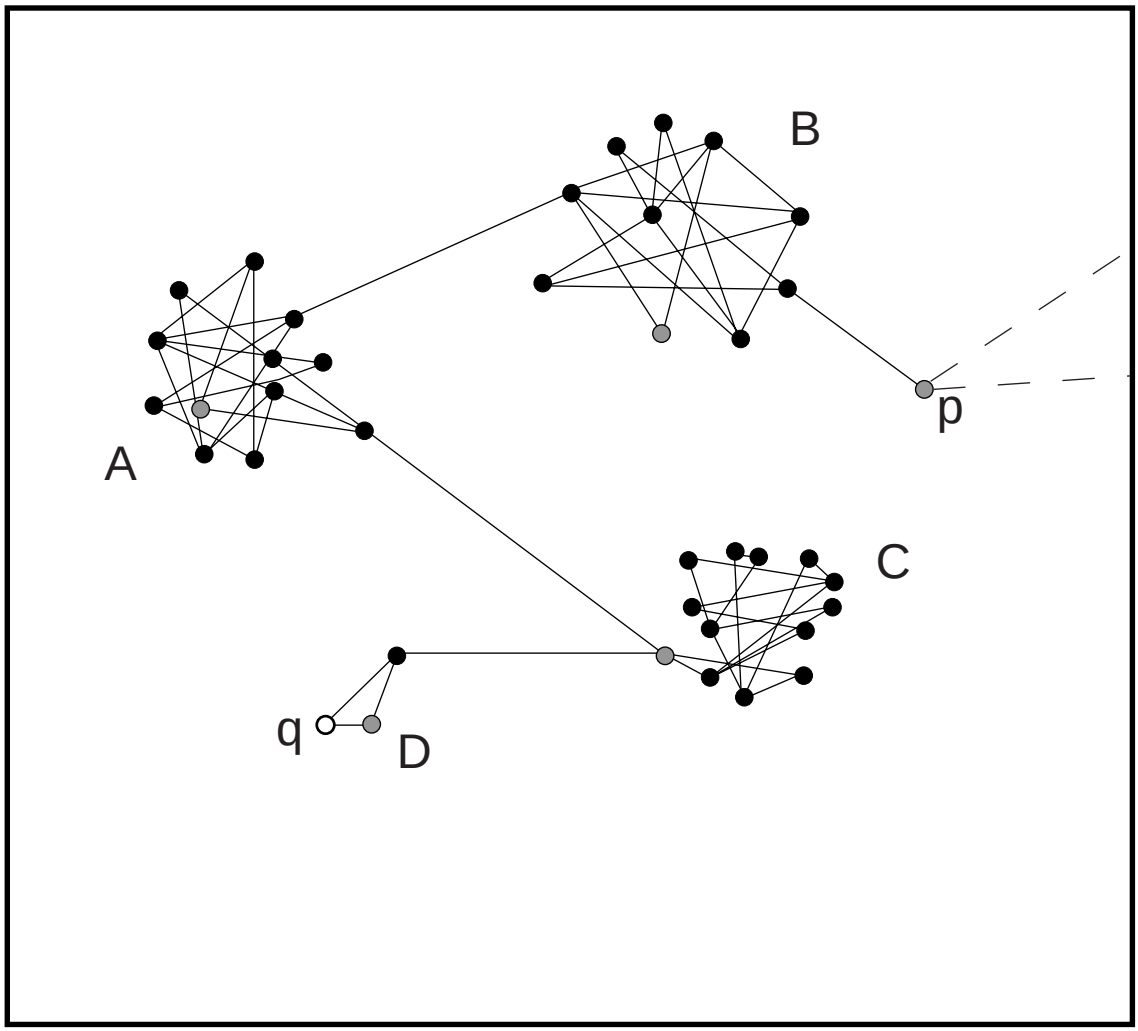


Figure 7.2: Hypothetical search space represented as a connected graph. Vertices on the graph represent individual genotypes, shaded according to their fitness (see text). Where two genotypes are separated by hamming-distance one, their vertices are connected

would fail. The threshold is set low enough that individuals of that fitness or above should be found reasonably quickly by random walk.

The basic idea is that if a genetically converged evolutionary search contains an individual lying in a cluster, then other individuals in the same cluster should be easily located by the search due to their low hamming distance. The cluster can hence be regarded as a single individual. We would like to determine the likelihood of evolution progressing to another cluster — one which contains individuals of higher fitness, but which are on average of high hamming distance from the original one. Even though the average hamming distance between individuals in one cluster and those in another is high, there may be one or more pairs — one individual in each cluster — separated by a single unit, as shown in figure 7.2. Where this is the case, the evolutionary search is likely to extend to the new cluster, but if it is not, a more fortuitous double mutation is required. The purpose of representing the search space in this way is therefore to render such inter-cluster pairs of individuals as apparent as possible, as a path that the search can follow. Of course there are other factors affecting the search's direction, such as suboptimal individuals existing within

a cluster of poor ones, but this does not hinder the *exploration* of new regions; if new clusters accessible through inter-cluster paths contain similar suboptima, then the search is likely to find them.

To illustrate, consider a small evolutionary population lying on cluster **A** in the figure. The target is the optimum individual labelled **q**, located in cluster **D**. There exist two inter-cluster paths by which members of the population can travel with a single-bit mutation either to cluster **B** or cluster **C**. Clearly **C** is the preferred choice, from which the optimum is easily reachable. If the population migrates to **B**, a path still exists back to **A**, so all is not lost. However it may migrate further to the suboptimum **p**, lying in a cluster disconnected from **A**, **C**, or **D**. The probability of backtracking to **A** is now reduced, especially if **p** is of significantly higher fitness than any suboptima yet encountered. If this search space had contained more inter-cluster paths, **D** may have been reached by another route, implying a space more amenable to evolutionary search.

This method of representing electronic search spaces is inspired by the concept of neutrality and its possible association with redundant, or 'junk' information contained within the genotype (see section 2.4). It may be that interconnection architectures/mappings that correspond to genotypes containing a high proportion of junk yield more inter-cluster paths, and are hence more evolvable. On the other hand, they may also yield much larger clusters, decreasing the likelihood of traversing the paths. The intention is to apply the representation to several arbitrarily chosen interconnection architectures, to determine the structure of their search space and any correlation with junk or evolvability. Note that while inspired by work on neutrality, the method utilises *ranges* of fitness rather than individuals of *equal* fitness, hence it does not attempt to reveal truly neutral networks.

The method has been applied to inverter search spaces in a preliminary study. The study applies prior work for revealing structure in unweighted, connected graphs (Kuntz et al., 1997; Kuntz, Snyers, & Layzell, 1999). In this work, a distributed algorithm displays spatially separate clusters of highly connected vertex subsets on a two-dimensional grid. Variants of direct and multiplex genotype⇒phenotype mapping were chosen for which genotype length was kept to an absolute minimum, so that the representation could be applied to the whole search space.

Figure 7.3 illustrates the mappings and corresponding genotype structure. In both mappings, the input, output and power are connected permanently to the rows shown. The areas of the wiring grid configurable by the genotype are highlighted. The direct mapping assigns each switch in this area to a unique genotype bit whereas in the multiplex mapping, pairs of bits address a single column to be connected to a given row. Only one transistor is used. An exhaustive search was conducted for each of the mappings, using the same fitness function as for the inverter circuits in section 5.3.1. Evaluation took place in simulation so that the results would not be influenced by evaluation noise. Figure 7.4 shows the search space for the direct mapping, and figure 7.5, the space for the multiplex mapping. The same shapes and colours are used in both of the figures to distinguish individuals of fitness f_{IND} lying within a given range.

Only individuals of zero fitness (constant-output circuits) or better are displayed. The vertices shown as green circles represent exclusively individuals of zero fitness, hence sets of connected ones are true neutral networks.

Referring to figure 7.4 (direct mapping), we see several isolated clusters and few inter-cluster

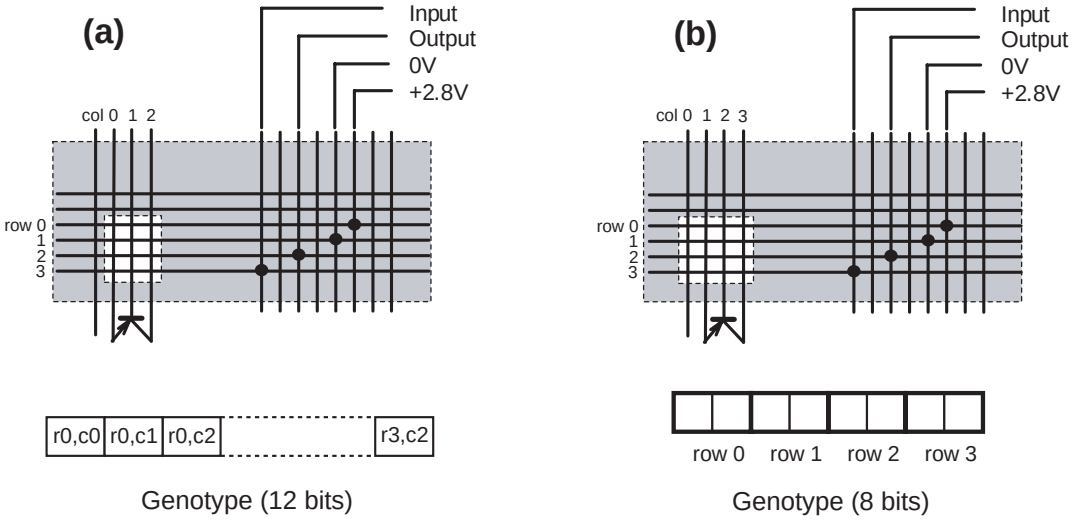


Figure 7.3: Mapping strategies adopted for a preliminary study on search-space structure (see text for details).

pathways. Some vertices — a few of which are suboptimal — are completely isolated, meaning that they can only be reached by single mutations of individuals with fitness less than zero. The single optimum (red square, fitness 1205) lies in a distinct cluster centre-right in the diagram, which has *no* inter-cluster pathways. We can therefore speculate that an evolutionary search is unlikely to find it, being prone to wander around the suboptima (orange triangles) that appear in a few connected clusters.

The search space created by multiplex mapping has an entirely different structure. Here we see two main clusters, predominantly individuals of zero fitness, with multiple inter-cluster paths. The optimum individual is at the top of the right-hand cluster (yellow star, fitness 2274). Although the other cluster contains sub-optima, the abundance of inter-cluster paths, and the lack of isolated vertices and clusters implies far less difficulty in locating the optimum.

The graphs tell us that the two search spaces are very different in structure, but there is little that can be generally inferred from these simple runs. Applying the technique to the vast search spaces generated by more typical genotype lengths is currently impossible. It could be applied to sample portions, but some degree of homogeneity in the space would have to be assumed and this is unlikely to be the case. Also, it is important to state that the arguments I have used to conceive the method and interpret the observed structures are highly speculative. There are many factors that influence evolvability in addition to those highlighted here. A great deal of further work is necessary to test the validity of this approach. Nonetheless, it represents at least one potential alternative to the use of conventional engineering knowledge for the design of evolvable architectures.

7.4 Improving the Understanding of PFT

Recall that the experiments of chapter 6 showed a close correlation between the point in evolutionary history at which a transistor became a functional part of the circuit — in the sense that its removal affected the best individual's fitness — and whether or not the final population as a whole

was robust to its 'failure' when it was removed at the end of evolution. So far we have a binary result: PFT is not expected for faulty transistors constituent from the earliest prototype onwards, but for the rest, it is. I would like to know if there is anything more we can say about these latter, and in considering this, I introduce two new terms. When evolution was complete, a transistor was removed and the population re-evaluated. Let us take the best fitness after re-evaluating to be the *degree* of PFT; the better the fitness, the better the degree of PFT exhibited for a given fault. Take also the generation at which a transistor became a functional part of the evolving circuit and remained so until evolution was stopped, to be its *birthday*. We know that PFT is due to the incremental nature of the evolutionary process. Therefore it is reasonable to expect that on average, the degree of PFT for a given transistor fault is some function of that transistor's birthday; the later the birthday, the better the expected degree of PFT. No such relationship was observed in the experiments, but this is not surprising given the small population size of 50. On evaluating final populations (either before or after transistor removal), only around 10 to 20% of individuals exhibited significant fitness scores. It was mostly these individuals that displayed PFT, hence to seek a qualitative relationship among 5 to 10 cases per run is statistically unjustifiable. The experiments could be repeated using much larger population sizes, but this is unnecessary. We have seen from the ES experiment (section 6.4.2) that PFT is present in a population made by mutating only the final base individual. Therefore, I propose taking the best individuals of the GA experiments, and making very large populations of mutants of them. With this approach, final population sizes of 10,000 are feasible, easily large enough to reveal any function relating the degree of PFT to a given transistor's birthday.

7.4.1 Use of Genotype Information to Predict PFT

Now consider the logistics of PFT prediction. To determine the periods during evolutionary history for which a given transistor was a functional part of the circuit required an automated process, which amounted to unplugging the transistor and re-evaluating the population every few generations. This could be carried out during evolution, or where the evolutionary history was stored on disk, it could be carried out *a posteriori*. Either way, the process adds a significant computational overhead, because the transistor-removal and re-evaluation process must be carried out many times. An alternative method that does not require either component disconnection or additional fitness evaluations is possible. If the relationship between a given component's configuration and the genotype bits that specify that configuration is clearly defined, then it may be possible to determine the role of the component in the evolving circuit by analysing information contained in the genotype. The bits specifying the configuration of non-constituent transistors should be more likely to change over several generations without affecting fitness, than those specifying functional configurations. There are two major factors influencing the viability of this approach. The first is the genotype⇒phenotype mapping employed. As an example, consider evolving a circuit with an FPGA. Circuit configurations are commonly represented in the genotype by a sequence of bit-strings, each specifying the function of a particular cell and its nearest-neighbour routing. Since we know exactly where to look in the genotype for information pertaining to a given cell, then making predictions of that cell's contribution to the circuit by observing how the bits change over time is relatively easy. By contrast, the experiments of chapter 6 used a mapping that specified

programmable switch addresses. Information pertaining to a given transistor could exist anywhere in the genotype, and its location could change over the course of evolution. It would not be impossible to conduct the same prediction; the genotype could be scanned for re-occurring switch addresses, and the relation between the *addresses* and the transistors is known. But it would be a more computationally complex process, perhaps as expensive as disconnecting the transistors automatically and re-evaluating. The second factor might be termed 'configuration redundancy'. In explaining the implied cause of PFT in chapter 6, I have been careful to state that the 'configurations of ancestor circuits' remained largely unchanged in later generations. This is not the same as genotypes remaining largely unchanged: many different genotypes could specify exactly the same circuit topologies. For example two transistor pins could be connected using different EM rows. In this case the topology would be the same but the genotypes would be different. Some method of taking this into account when using genotype information to assess a component's role would have to be found.

Despite these difficulties, the approach is worth investigating. It may not be as accurate as the original one, but may still be sufficiently accurate to be useful. If the factors discussed above are taken into account when designing the evolutionary framework, then it may represent a more economical way to predict PFT. Even if this does not prove to be the case, investigating the approach is also likely to yield valuable data for further understanding PFT and evolvability, since it would reveal the extent of 'configuration redundancy' for a given mapping.

7.4.2 An Information Theory Perspective on PFT

The underlying mechanism of PFT and its potential scalability could be explored from a theoretical perspective. Where finite-length genotypes are used, information theory (see Ash (1965) for details) can be applied to derive a limit on the quantity of significantly different ancestor circuit configurations that a genotype may contain. We can use the transistor importance profiles to estimate how many significantly different configurations were produced over the course of each evolutionary run in chapter 6. In the profiles, it is easy to discern generations at which transistors previously constituent in the evolving circuit were no longer required, and *vice versa* (for example, generation 850 in figure 6.5). These major transitions in transistor importance occurred on average only two to three times for each run. We can infer that the best circuits just before and after each transition are significantly different, while different configurations appearing between transitions are only minor modifications of essentially the same circuit. If this is so, then the quantity of significantly different circuits produced over each run in chapter 6 is very small.

The theoretical limit on the quantity of different circuit configurations a given genotype can contain affects our prediction method for faults that manifest PFT. Consider a large-scale evolutionary run, over the course of which 15 significantly different circuits are produced (after the initial prototype is found). In this hypothetical example, let information theory define the limit of largely unchanged, significantly different ancestral circuits that the final genotype can contain to be 5. It is reasonable to suppose that these 5 circuits are produced in later, rather than earlier generations. Hence, transistors which were constituent to each of the last 5 different circuits are *not* likely to manifest PFT when faulty, even if they were unused by earlier circuits. This is clearly different to the PFT prediction method used in chapter 6, where any transistor that was not

constituent at every generation following the prototype was expected to manifest PFT when faulty.

The argument above could be validated by experiment without necessarily having to evolve large-scale circuits. Instead, a program based on NK and NKp fitness landscapes will be commenced in the near future. The PFT phenomenon should be relevant to fields outside of HE, where the characteristics of an evolutionary medium are subject to alteration over time. Hence it is hoped that the experimental program will produce results pertinent to the inter-field relevance of PFT, as well as advancing our understanding of the mechanism responsible for it.

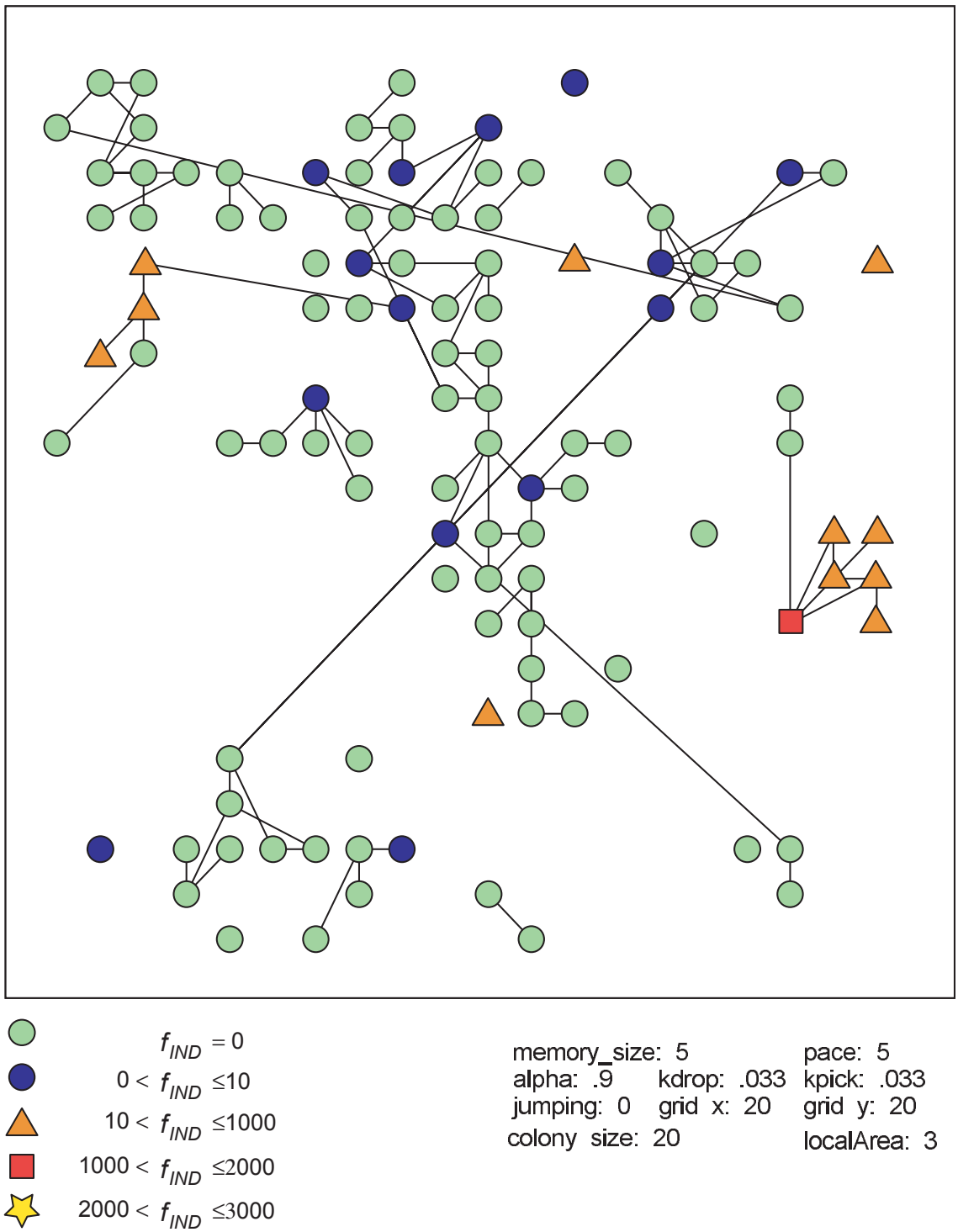


Figure 7.4: Connected graph representing the direct-mapped search space of inverters. Directly below the graph are the parameter values used for the graph algorithm (see the publications cited in the text for an explanation of their use).

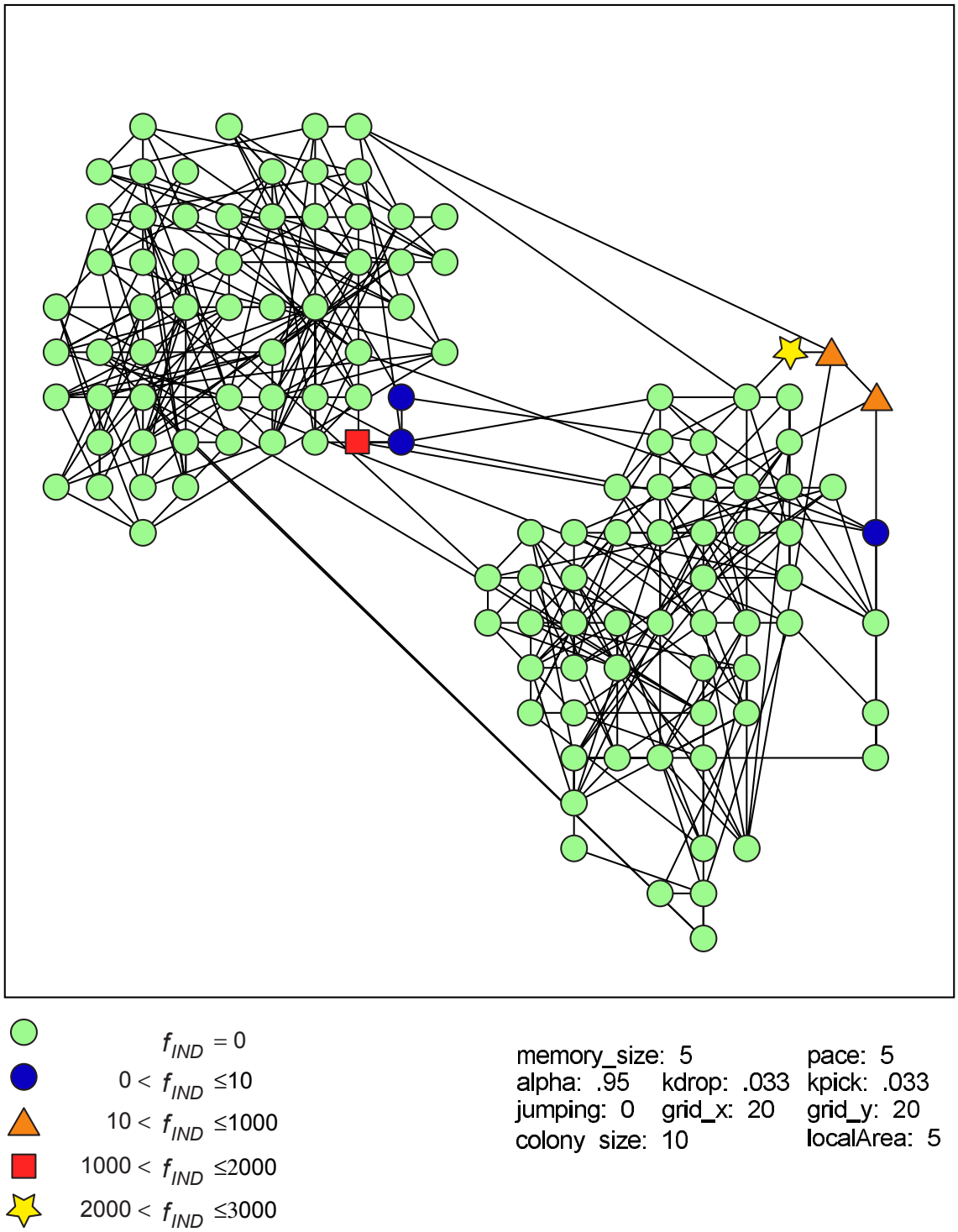


Figure 7.5: Connected graph representing the multiplex-mapped inverter search space.

Chapter 8

Conclusion

The flexibility and observability of the EM makes it ideal as a general-purpose tool for investigating the majority of issues arising in HE research. Since it was first presented in Layzell (1998a), it has been adopted by a number of institutions from high schools to university departments (for example, Flockton and Sheehan (1999)). It has also been instrumental in laying the foundations for more recent developments in reconfigurable devices intended specifically for HE, for example Langeheine et al.'s VLSI Transistor Array, which incorporates comprehensive internal probing points for measurement of voltage, current, and temperature (Langeheine, Folling, Meier, & Schemmel, 2000).

The engineering usefulness of hardware evolution and the circuits it derives is improved with increased understanding gained through analysis. Analysis of the circuits evolved in the previous chapters has been greatly facilitated by the EM's flexibility and comprehensive interconnection architecture. The signals present at the basic elements have been revealed by direct probing (chapter 5), and determining whether or not a given basic element is constituent in an evolved circuit has been a simple matter of unplugging it or re-routing its connections (chapters 5 and 6). Direct measurement and basic element removal are not possible with commercial FPGAs, and the re-routing of connection nodes requires larger alterations of signal pathways than is necessary with the EM (see chapter 3). We have seen examples in chapters 3 and 5 of unconstrained evolution's tendency to exploit whatever physical characteristics are available to it, therefore it is essential when analysing an evolved circuit to keep such alterations to a minimum. We have also seen the first examples of intrinsically evolved transistor circuits. There are no commercial FPGAs for which this could have been achieved. The first point of the thesis has thus been shown to be true.

Through the experiments of chapters 4 and 5, insights have been gained into portability, choice of interconnection architecture, intrinsic versus extrinsic approaches, as well as inherent properties of evolved circuits such as component and environment exploitation:

- We have observed disadvantages of a directly mapped, comprehensive interconnection architecture, with regard to its evolvability.
- Our existing understanding of the conditions under which extrinsically evolved circuits may be considered as portable as those evolved intrinsically has been extended.

- Additional factors influencing the speed of intrinsic versus extrinsic hardware evolution, such as evaluation noise, have been demonstrated. We have seen one example for which extrinsic HE proved the faster.

- Unconstrained evolution is likely to exploit any of the physical characteristics possessed by the medium. Here, it has consistently exploited those of the configuration circuitry and in some cases even those of the printed circuit board. This strongly influences the topology of evolved circuits (although there are many other factors).

These are the major insights revealed by the experiments. I do not wish to overstate their importance by using any stronger term, nonetheless they are useful contributions to our understanding of the issues I have set out in chapter 2. There are also other minor contributions such as choice of basic evolutionary element. While transistors have been used throughout, we have for the first time been unrestricted by the limited variety available in commercial reconfigurable devices, and are thereby permitted the freedom to consider other reasons for their adoption (in this case, the numerous simulation models available) in intrinsic HE.

These insights have been achieved through a practical analytical framework combining established techniques with novel ones — such as the use of evolutionary history and the evolvable motherboard — applied to small evolved circuits. Hence, point 2 of the thesis is shown to be true.

While the program of chapter 5 was devised primarily to demonstrate the validity of the EM and the proposed analytical techniques, they have been applied in earnest in chapter 6 to investigate a phenomenon for which only fragmentary evidence was available initially. We now know that PFT is a real phenomenon whose discovery is directly due to my analytical toolset. Through methodical examination of the constituent components in ancestor circuits, it was established as an inherent quality, and the correlation observed between the transistors' importance profiles and their likelihood of manifesting PFT when faulty strongly supports one of the hypotheses offered to explain it — that configurations of ancestor circuits (which did not make use of the transistors whose removal resulted in PFT) still exist relatively undisturbed in the final genotypes. Point 3 of the thesis has thus been shown to be true, and points 1 and 2 are reinforced.

The majority of analytical techniques adopted in this thesis have long been employed in circuit design, reverse-engineering, and evolutionary computation. I am not the first to use evolutionary history to aid understanding of the evolutionary process, however I have successfully applied it in novel ways further to understand the operation of the tone-discriminator, the influence of noise in evolving stable prototypes (section 5.3.1), and the mechanisms underlying populational fault tolerance.

I have not set out to assess the viability of PFT as a practical methodology for the design of fault tolerant systems, but since the concept was first published (Layzell, 1999a), a group at JPL (Jet Propulsion Laboratory, Pasadena) has carried out a series of experiments comparing it to another technique in which known faults were introduced during the evolution of an exclusive NOR gate and an analogue multiplier (Keymeulen, Stoica, & Zebulum, 2000). The authors conclude that "in both experiments the population approach designs fault-tolerant circuits with a better performance and in less time than fitness based approaches.". Their work is also further evidence that generic classes of evolved circuit can be expected to manifest PFT; both the reconfigurable platform and the GA used were different to those employed in this thesis.

Research documented here and by other authors has revealed positive and negative properties inherent in circuits evolved using various different approaches. Whatever benefits they offer, a principal characteristic shared by circuits evolved using the unconstrained method is that they are difficult to analyse, due to the unusual ways in which components are exploited and numerous feedback loops. More techniques must be developed before we can tackle the analysis of large complex circuits evolved using this method. However, a methodical approach must be taken if the field is to progress as a credible engineering tool. Thus research on small or simple circuits continues to play a major role in the development of HE. Analysis of such circuits is far from impractical, and is likely to contribute further to understanding the properties that evolution can and cannot exploit, and why.

Bibliography

Anderson, T. (Ed.). (1985). *Resilient Computing Systems*. Collins, London.

Ash, R. B. (1965). *Information theory*. John Wiley, New York.

A.Thompson, Layzell, P., & Zebulum, R. (1999). Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE transactions on Evolutionary Computation*, *3*(3), 167–196.

Avizienis, A. (1997). *Toward Systematic Design of Fault-Tolerant Systems*. IEEE Computer Society Press.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *1st International Conference on Genetic Algorithms and Their Applications*, pp. 101–111.

Barnett, L. (1997). Tangled webs. evolutionary dynamics on fitness landscapes with neutrality. Master's thesis, School of Cognitive and Computing Sciences (COGS), University of Sussex, UK.

Barnett, L. (1998). Ruggedness and neutrality - the NKp familiy of fitness landscapes. In *Proc. 6th Int. Conf. on Artificial Life*, pp. 18–27. MIT Press.

Beckers, R., Holland, O., & Deneubourg, J. (1994). From local actions to global tasks : stigmergy and collective robotics. In *Artificial Life IV*, pp. 181–189. MIT Press.

Bennet, F., Koza, J., Andre, D., M., & Keane (1996). Evolution of a 60 decibel op amp using genetic programming. In Higichi, T., Iwata, M., & Lui, W. (Eds.), *1st International Conference on Evolvable Systesm: From Biology to Hardware*, Vol. 1259 of *LNCS*, pp. 455–469. Springer-Verlag.

Bennett, F., Koza, J., Wu, J., & Mydlowec, W. (2000). Automatic synthesis, placement, and routing of an amplifier circuit by means of genetic programming. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 1–10. Springer.

Benten, M. S. T., & Sait, S. M. (1994). GAP: a genetic algorithm approach to optimize two-bit decoder PLAs. *Int. J. Electronics*, *76*(1), 99–106.

Booker, L. B. (1988). Classifier systems that learn internal world models. *Machine Learning*, *2-3*(3), 161–192.

Bradley, D. W., & Tyrell, A. M. (2000). Immunotronics: Hardware fault tolerance inspired by the immune system. In Miller, J., Thompson, A., Thompson, P., & Fogarty, T. (Eds.), *3rd Intenational Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 11–20. Springer.

Bratt, A., & Macbeth, I. (1996). Design and implementation of a field programmable analogue array. In *Proc. ACM/SIGDA 4th Int. Symp. on Field-Programmable Gate Arrays (FPGA'96)*, pp. 88–93.

Burton, T. D. (1994). *Introduction to Dynamic Systems Analysis*. McGraw-Hill.

Chean, M., & Fortes, J. (1990). A taxonomy of reconfiguration techniques for fault-tolerant processor arrays. *Computer, 4*, 55–69.

Christodoulou, A. (1999). *Performance Modelling of Large Scale Systems*. Ph.D. thesis, University of Leeds.

de Garis, H. (1993). Growing an artificial brain with a million neural net modules inside a trillion cell cellular automaton machine. In *Proc. 4th Int. Symp. on Micro Machine and Human Science*, pp. 211–214.

de Garis, H. (1995). The CAM-BRAIN project: The genetic programming of a billion neuron artificial brain by 2001 which grows/evolves at electronic speeds inside a cellular automaton machine. In *Proc. Int. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA95)* Ales, France.

de Jong, K. A. (1975). An analysis of the behaviour of a class of genetic adaptive systems.(doctoral dissertation, university of michigan. *Dissertation Abstracts International*, *36*(10).

Dittrich, P., Bürgel, A., & Banzhaf, W. (1998). Learning to move a robot with random morphology. In Husbands, P., & Meyer, J.-A. (Eds.), *Proc. 1st European Workshop on Evolutionary Robotics (EvoRobot98)*, Vol. 1468 of *LNCS*, pp. 165–176. Springer.

Eigen, M. (1987). New concepts for dealing with the evolution of nucleic acids. In *Cold Spring Harbor Symposia on Quantitative Biology*, Vol. LII. Cold Spring Harbor Laboratory.

Flockton, S., & Sheehan, K. (1998). Intrinsic circuit evolution using programmable analogue arrays. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd Int. Conf. on Evolvable Systems: From Biology to Hardware (ICES 98)*, Vol. 1478 of *LNCS*, pp. 144–153. Springer-Verlag.

Flockton, S., & Sheehan, K. (1999). A system for intrinsic evolution if linear and non-linear filters. In Stoica, A., Keymeulen, D., & Lohn, J. (Eds.), *1st NASA/DoD Workshop on Evolvable Hardware*, pp. 93–100. IEEE Computer Society.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. John Wiley & Sons, Inc.

Forrest, S. (Ed.). (1990). *Emergent Computation: Self-organizing, Collective and Cooperative Phenomena in Natural and Artificial Computing Networks*, Emergent Computation.

Franks, N. R. (1989). Army ants: A collective intelligence. *American Scientist, 77*(2).

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison Wesley.

Grefenstette, J. J. (1996). Genetic learning for adaptation in autonomous robots. *Robotics and Manufacturing: Recent Trends in Research and Applications, 6*.

Guccione, S., & Levi, D. (1998). A java-based interface to FPGA hardware. In Schewel, J. (Ed.), *Configurable Computing: Technology and Applications*, pp. 97–102. International Society for Optical Engineering.

Hamilton, A., Papathanasiou, K., Tamplin, M., & Brandtner, T. (1998). Palmo: Field programmable analogue and mixed-signal VLSI for evolvable hardware. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES 98)*, Vol. 1478 of *LNCS*, pp. 335–344. Springer.

Hamilton, A., Thomson, P., & Tamplin, M. (2000). Experiments in evolvable filter design using pulse based programmable analogue VLSI models. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd International Conference on Evolvable Systems: From Biology to Hardware (ICES 2000)*, Vol. 1801 of *LNCS*, pp. 61–71. Springer.

Harvey, I. (1992a). The SAGA cross: The mechanics of recombination for species with variable length genotypes. In Männer, R., & Manderick, B. (Eds.), *Proc. Parallel Problem Solving from Nature (PPSN) II*, pp. 269–278. North-Holland.

Harvey, I. (1992b). Species Adaptation Genetic Algorithms: A basis for a continuing SAGA. In Varela, F. J., & Bourgine, P. (Eds.), *Towards a Practice of Autonomous Systems: Proc. 1st Eur. Conf. on Artificial Life*, pp. 346–354. MIT Press.

Harvey, I., Husbands, P., & Cliff, D. (1994). Seeing the light : Artificial evolution, real vision. In Cliff, D., et al. (Eds.), *From Animals to Animats 3: Proc. 3rd Int. Conf. on simulation of adaptive behaviour*, pp. 392–401. MIT Press.

Harvey, I., & Thompson, A. (1997). Through the labyrinth evolution finds a way: A silicon ridge. In Higuchi, T., & Iwata, M. (Eds.), *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, Vol. 1259 of *LNCS*, pp. 406–422. Springer-Verlag.

Hemmi, H., Mizoguchi, J., & Shimohara, K. (1996). Development and evolution of hardware behaviours. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware: The evolutionary engineering approach*, Vol. 1062 of *LNCS*, pp. 250–265. Springer-Verlag.

Higuchi, T., & Kajihara, N. (1999). Evolvable hardware chips for industrial alpplications. *Communications of the ACM*, *42*(4), 60–66.

Higuchi, T., & Hirao, Y. (1995). Evolvable hardware with genetic learning. In *Proc. 2nd Workshop on Synthetic Worlds - Modelizations, Practices, Societal Impacts* Paris.

Higuchi, T., Iba, H., & Manderick, B. (1994). Applying evolvable hardware to autonomous agents. In Davidor, Y., Schwefel, H.-P., & Männer, R. (Eds.), *Proc. Parallel Problem Solving from Nature (PPSN) III*, Vol. 866 of *LNCS*, pp. 524–533. Springer-Verlag.

Higuchi, T., Iwata, M., Kajitani, I., et al. (1996a). Evolvable hardware and its application to pattern recognition and fault-tolerant systems. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware*, Vol. 1062 of *LNCS*, pp. 118–135. Springer-Verlag.

Higuchi, T., Iwata, M., Kajitani, I., et al. (1996b). Evolvable hardware with genetic learning. In *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS96)* Atlanta, USA.

Hikage, T., Hemmi, H., & Shimohara, K. (1998). Comparison of evolutionary methods for smoother evolution. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1478 of *LNCS*, pp. 114–124. Springer.

Hillis, W. D. (1992). Co-evolving parasites improve simulated evolution as an optimization procedure. In Langton, C., et al. (Eds.), *Artificial Life II*, pp. 313–324. Addison-Wesley.

Hirst, A. J. (1996). Evolving adaptive computer systems. In *Proc. 1st On-line Workshop on Soft Computing (WSC1)*. See http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Horowitz, P., & Hill, W. (1989). *The Art of Electronics* (2nd edition). Cambridge University Press.

Horrocks, D. H., & Khalifa, Y. M. A. (1994). Genetically derived filter circuits using preferred value components. In *Proc. IEE Colloq. on Analogue Signal Processing*, pp. 4/1 – 4/5 Oxford, UK.

Horrocks, D. H., & Khalifa, Y. M. A. (1996). Genetic algorithm design of electronic analogue circuits including parasitic effects. In *Proc. 1st On-line Workshop on Soft Computing (WSC1)*. See http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/.

Huelsbergen, L., Rietman, E., & Slous, R. (1999). Evolution of astable multivibrators *in silico*. *IEEE Transactions on Evolutionary Computation*, *3*(3), 197–204.

Huynen, M. A., & Hogeweg, P. (1994). Pattern generation in molecular evolution: Exploitation of the variation in RNA landscapes. *J. Mol. Evol.*, *39*, 71–79.

Intersil Corp. (1997). *CD22M3494 datasheet*. Melbourne, FL 32902, N. America.

ITTC (1993). *Progressive Bi-level Image Compression*. International Telegraph and Telephone Consultative Commitee.

Iwata, M., Kajitani, I., Yamada, H., et al. (1996). A pattern recognition system using evolvable hardware. In Voigt, H.-M., et al. (Eds.), *Parallel Problem Solving from Nature IV: Proc. of the Int. Conf. on Evolutionary Computation*, Vol. 1141 of *LNCS*. Heidelberg: Springer-Verlag. In press.

Jakobi, N. (1998). *Minimal Simulations for Evolutionary Robotics*. Ph.D. thesis, School of Cognitive and Computing Sciences (COGS), University of Sussex,UK.

Jakobi, N. (1996). Facing the facts: Necessary requirements for the artificial evolution of *complex* behaviour. CSRP 422, COGS, University of Sussex, UK.

Jeffries, D. J. (1998). Crisis-induced intermittency in simple electronic circuits: Experimental results and simulations. In *1998 International symposium on nonlinear theory and its applications (NOLTA 98)*.

Kajitani, I., Hoshino, T., Nishikawa, D., et al. (1998). A gate-level EHW chip: Implementing GA operations and reconfigurable hardware on a single LSI. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Evolvable Systems: From Biology to Hardware, Proc. of 2nd Int. Conf. on Evolvable systems (ICES 98)*, Vol. 1478 of *LNCS*, pp. 1–12. Springer.

Kandel, E. R. (1976). *Cellular basis of behaviour: An introduction to behavioural neurobiology*. W.H.Freeman and Co.

Kauffman, S. A. (1993). *The Origins of Order*. Oxford University Press.

Keymeulen, D., Iwata, M., Kuniyoshi, M., & Higuchi, T. (1998). Comparison between an offline model-free and an on-line model-based evolution applied to a robotics navigation system using evolvable hardware. In Adami, C., Belew, R. K., Kitano, H., & Taylor, C. E. (Eds.), *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, pp. 109–209. MIT Press.

Keymeulen, D., Stoica, A., & Zebulum, R. (2000). Fault-tolerant evolvable hardware using field programmable transistor arrays. *IEEE Transactions on Reliability, Special Issue on Fault-Tolerant VLSI Sytsems*, *49*(3). IEEE Press. In press.

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press, New York.

Kitano, H. (1998). Building complex systems using developmental process: An engineering approach. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1478.

Kitano, H. (1996a). Challenges of evolvable systems: Analysis and future directions. In Higuchi, T., Iwata, M., & Lui, W. (Eds.), *1st International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1259 of *LNCS*. Springer-Verlag.

Kitano, H. (1996b). Morphogenesis for evolvable systems. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware: The evolutionary engineering approach*, Vol. 1062 of *LNCS*, pp. 99–117. Springer-Verlag.

Kodjabachian, J., & Meyer, J. (1998). Evolution and development of neural networks controlling locomotion, gradient following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, *4*(234-249).

Korkin, M., Nawa, N. E., & de Garis, H. (1998). A "spike interval information codeing" representation for ATR's CAM-brain machine (CBM). In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES 98)*, Vol. 1478 of *LNCS*, pp. 256–267. Springer.

Koza, J., Bennet, F., Andre, D., & Keane, M. (1996). Reuse, parameterized reuse, and hierarchical reuse of substructures in evolving electrical circuits using genetic programming. In Higuchi, T., Iwata, M., & Lui, W. (Eds.), *1st International conference on Evolvable Systems*, Vol. 1259 of *LNCS*. Springer-Verlag.

Koza, J. R. (1992). *Genetic Programming: On the programming of computers by means of natural selection*. MIT Press, Cambridge, Mass.

Koza, J. R., Andre, D., Bennett III, F. H., et al. (1996). Evolution of a low-distortion, low-bias 60 decibel op amp with good frequency generalization using genetic programming. In Koza, J. R. (Ed.), *Late Breaking Papers at the Genetic Programming 1996 Conference*, pp. 94–100. Stanford, CA: Stanford University Bookstore.

Kruiskamp, W., & Leenaerts, D. (1995). Darwin: Cmos opamp synthesis by means of a genetic algorithm. In *32nd Design Automation Conference*, pp. 433–438. Association for Computing Machinery.

Kuntz, P., Layzell, P., & Snyers, D. (1997). A colony of ant-like agents for partitioning in VLSI technology. In Husbands, P., & Harvey, I. (Eds.), *Proc. 4th European Conf. on Artificial Life (ECAL'97)*, pp. 417–424. MIT Press.

Kuntz, P., Snyers, D., & Layzell, P. (1999). A stochastic heuristic for visualising graph clusters in a bi-dimensional space. *Journal of Heuristics*, *5*(3), 327–352.

Langeheine, J., Folling, S., Meier, K., & Schemmel, J. (2000). Towards a silicon primordial soup: A fast approach to hardware evolution with a VLSI transistor array. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *1st International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 123–122. Springer.

Layzell, P. (1998a). The 'Evolvable Motherboard': A test platform for research of intrinsic hardware evolution. CSRP 479, School of Cognitive and Computing Sciences, University of Sussex.

Layzell, P. (1998b). A new research tool for intrinsic hardware evolution. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98)*, Vol. 1478 of *LNCS*, pp. 47–56. Springer-Verlag.

Layzell, P. (1999a). Inherent qualities of circuits designed by artificial evolution: A preliminary study of populational fault tolerance. In Stoica, A., Keymeulen, D., & Lohn, J. (Eds.), *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*, pp. 85–86. IEEE Computer Society, Los Alamitos, California.

Layzell, P. (1999b). Reducing hardware evolution's dependency on FPGAs. In *Proc. 7th Int. Conf. on microelectronics for neural, fuzzy and bio-inspired systems*. IEEE Comp. Soc. Press.

Lienig, J., & Brandt, H. (1994). An evolutionary algorithm for the routing of multi-chip modules. In Davidor, Y., Schwefel, H.-P., & Männer, R. (Eds.), *Proc. Parallel Problem Solving from Nature (PPSN) III*, Vol. 866 of *LNCS*, pp. 588–597. Springer-Verlag.

Lin, L. J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning, 3-4*(8), 297–321.

Lohn, J., Haith, G., Colombano, S., & Stassinopoulos, D. (1999). A comparison of dynamic fitness schedules for evolutionary design of amplifiers. In Stoica, A., Keymeulen, D., & Lohn, J. (Eds.), *1st NASA/DoD Workshop on Evolvable Hardware*, pp. 87–92. IEEE Computer Society.

Lund, H. H., Webb, B., & Hallam, J. (1997). A robot attracted to the cricket species gryllus bimaculatus. In Husbands, P., & Harvey, I. (Eds.), *4th European Conference on Artificial Life*, pp. 246–255. MIT Press.

M. Goeke, M. Sipper, D. M., et al. (1996). Online autonomous evolware. In Higuchi, T., & Iwata, M. (Eds.), *1st International Conference on Evolvable Systems: from Biology to Hardware*, Vol. 1259 of *LNCS*. Springer.

Mange, D., Goeke, M., Madon, D., et al. (1996). Embryonics: A new family of coarse-grained field-programmable gate array with self-repair and self-reproducing properties. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware: The evolutionary engineering approach*, Vol. 1062 of *LNCS*, pp. 197–220. Springer-Verlag.

Marino, L. R. (1981). General theory of metastable operation. *IEEE Transactions on Computers, C-30*(2), 107–115.

McCaskill, J. S., Tangen, U., & Ackermann, J. (1997). VLSE: Very large scale evolution in hardware. In Husbands, P., & Harvey, I. (Eds.), *4th European Conference on Artificial Life*, pp. 398–406. MIT Press.

MicroSim (2000). *PSpice Product information, MicroSim, North America*.

Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.). (2000). *3rd International Conference on Evolvable Systems: From Biology to Hardware (ICES 2000)*, Vol. 1801 of *LNCS*. Springer.

Miller, J. F., & Thomson, P. (1995). Combinational and sequential logic optimisation using genetic algorithms. In *Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pp. 34–38. IEE Conf. Publication No. 414.

Miller, J. F., & Thomson, P. (1998). Aspects of digital evolution: Geometry and learning. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98)*, Vol. 1478 of *LNCS*, pp. 25–35. Springer-Verlag.

Milner, R. (1989). *Communication and Conccurency*. Prentice-Hall.

Moreno, J. M., Madrenas, J., Faura, J., et al. (1998). Feasible evolutionary and self-repairing hardware by means of the dynamic reconfiguration capabilities of the FIPSOC devices. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: From Biology to Hardware (ICES 98)*, Vol. 1478 of *LNCS*, pp. 345–355. Springer.

Nimwegen, E. V., Crutchfield, J., & Mitchell, M. (1999). Statistical dynamics of the royal road genetic algorithm. *Special Issue on Evolutionary Computation, Theoretical Computer Science, 229*, 41–102.

Nonaneau, E., Dorigo, M., & Theraulaz, T. (1999). *From Natural to Artificial Swarm Intelligence*. Oxford University Press.

OrCad (1997). *PSpice product information, Orcad, North America*.

Ortega, C., & Tyrrell, A. (1999). Reliablilty analysis in self-repairing embryonic systems. In Stoica, A., Keymeulen, D., & Lohn, J. (Eds.), *1st NASA/DoD workshop on Evolvable Hardware*, pp. 120–128. IEEE Computer Society.

Ozsvald, I. (1998). Short-circuiting the design process: Evolutionary algorithms for circuit design using reconfigurable analogue hardware. MSc thesis, KBS, School of Cognitive and Computing Sciences, University of Sussex.

Paredis, J. (1994). Steps towards co-evolutionary classification neural networks. In Brooks, R., & Maes, P. (Eds.), *Artificial Life IV: Proc. 4th Int. Workshop on the Synthesis and Simulation of Living Systems*, pp. 102–108. MIT Press.

Pomeau, Y., & Manneville, P. (1980). Intermittent transition to turbulence in dissipative dynamical systems. *Communications in Mathematical Physics, 74*, 189–197.

Prieto, A. (Ed.). (1999). *Seventh International Conference on Microelectronics for Neural, Fuzzy, and Bio-inspired Systems*. IEEE Computer Society.

Sanchez, E. (1996). Field programmable gate array (FPGA) circuits. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware: The Evolutionary Engineering Approach*, pp. 1–18. Springer-Verlag.

Sanchez, E., Mange, D., Sipper, M., et al. (1996). Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In Higuch, T., Iwata, M., & Lui, W. (Eds.), *1st International Conference on Evolvable Systems: From Biology to Hardware (ICES 96)*, Vol. 1259 of *LNCS*. Spinger-Verlag.

Schnecke, V., & Vornberger, O. (1995). Genetic design of VLSI-layouts. In *Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pp. 430–435. IEE Conf. Publication No. 414.

Schoonderwoerd, R., Holland, O., Bruten, J., & Rothkrantz, L. (1997). Ant-based load balancing in telecommunications networks. *Adaptive Behaviour, 2*(5), 169–207.

Schwefel, H.-P., & Rudolph, G. (1995). Contemporary evolution strategies. In Morán, F., et al. (Eds.), *Advances in Artificial Life: Proc. 3rd Eur. Conf. on Artificial Life*, Vol. 929 of *LNAI*, pp. 893–907. Springer-Verlag.

Simon, H. A. (1996). *The Sciences of the Artificial* (3rd edition). MIT Press.

Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.). (1998). *Proc. 2nd Int. Conf. on Evolvable Systems (ICES98)*, Vol. 1478 of *LNCS*. Springer-Verlag.

Slorach, C., & Sharman, K. (2000). The design and implementation of custom architectures for evolvable hardware using off-the-shelf programmable devices. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 197–206. Springer.

Stoica, A., Keymeulen, D., & Lohn, J. (Eds.). (1999). *First NASA/DoD Workshop on Evolvable Hardware*. IEE Computer Society.

Stoica, A. (1999). Towards evolvable hardware chips: Experiments with a programmable transistor array. In *7th International Conference on Microelectronics for Neural, Fuzzy, and Bio-Inspired Systems*, pp. 156–162. IEEE Computer Society.

Stoica, A., Fukunaga, A., Hayworth, K., & Lazaro, C. S. (1998). Evolvable hardware for space applications. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES98)*, Vol. 1478 of *LNCS*, pp. 166–173. Springer.

Stoica, A., Zebulum, R., & Keymeulen, D. (2000). Mixtrinsic evolution. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *Proc. 3nd Int. Conf. on Evolvable Systems (ICES2000)*, Vol. 1801 of *LNCS*, pp. 208–217. Springer.

Tanaka, M., Sakanashi, H., Salami, M., et al. (1998). Data compression for digital color electrographic printer with evolvable hardware. In Sipper, M., Mange, D., & Perez-Uribe, A. (Eds.), *2nd International Conference on Evolvable Systems: from Biology to Hardware*, Vol. 1478 of *LNCS*, pp. 106–114. Springer.

Teich, J., Blickle, T., & Thiele, L. (1996). An evolutionary approach to system-level synthesis. In *Proc. 1st On-line Workshop on Soft Computing (WSC1)*. See http://www.bioele.nuee.nagoya-u.ac.jp/wsc1/.

T.Higuchi, Iwata, M., et al. (1999). Real-world applications of analog and digital evolvable hardware. *IEEE Transactions on Evoultionary Computation, 3*(3).

Thompson, A. (1995a). Evolving electronic robot controllers that exploit hardware resources. In Morán, F., et al. (Eds.), *Advances in Artificial Life: Proc. 3rd Eur. Conf. on Artificial Life (ECAL95)*, Vol. 929 of *LNAI*, pp. 640–656. Springer-Verlag.

Thompson, A. (1995b). Evolving fault tolerant systems. In *Proc. 1st IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA'95)*, pp. 524–529. IEE Conf. Publication No. 414.

Thompson, A. (1996a). Evolutionary techniques for fault tolerance. In *Proc. UKACC Int. Conf. on Control 1996 (CONTROL'96)*, pp. 693–698. IEE Conference Publication No. 427.

Thompson, A. (1996b). Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution. DPhil thesis, School of Cognitive and Computing Sciences, University of Sussex. Also published in: Distinguished Dissertations Series, Lecture Notes in Computer Science, Springer-Verlag 1998.

Thompson, A. (1996c). Silicon evolution. In Koza, J. R., et al. (Eds.), *Genetic Programming 1996: Proc. 1st Annual Conf. (GP96)*, pp. 444–452. Cambridge, MA: MIT Press.

Thompson, A. (1997a). An evolved circuit, intrinsic in silicon, entwined with physics. In Higuchi, T., & Iwata, M. (Eds.), *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, Vol. 1259 of *LNCS*, pp. 390–405. Springer-Verlag.

Thompson, A. (1997b). Evolving inherently fault-tolerant systems. *Proc Instn Mechanical Engrs, Part I, 211*, 365–371.

Thompson, A. (1998a). *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Distinguished dissertation series. Springer-Verlag.

Thompson, A. (1998b). On the automatic design of robust electronics through artificial evolution. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES'98)*, Vol. 1478 of *LNCS*, pp. 13–24. Springer-Verlag.

Thompson, A., Harvey, I., & Husbands, P. (1996). Unconstrained evolution and hard consequences. In Sanchez, E., & Tomassini, M. (Eds.), *Towards Evolvable Hardware: The evolutionary engineering approach*, Vol. 1062 of *LNCS*, pp. 136–165. Springer-Verlag.

Thompson, A., & Layzell, P. (1999). Analysis of unconventional evolved electronics. *Communications of the ACM*, *42*(4), 71–79.

Thompson, A., & Layzell, P. (2000). Evolution of robustness in an electronics design. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 218–228. Springer.

Tofts, C. (1994). Processes with probability, priority and time. *Formal Approaches in Compter Science*, *6*(5), 536–564.

Uchida, M., et al. (1993). Control of a robot arm by myoelectric potential. *Journal of Robotics and Mechatronics*, *5*(3), 259–265.

Vassilev, V., & Miller, J. (2000). The advantages of landscape neutrality in digital circuit evolution. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *Proc. 3nd Int. Conf. on Evolvable Systems (ICES2000)*, Vol. 1801 of *LNCS*, pp. 252–263. Springer.

Vassilev, V., Miller, J., & Fogarty, T. (1999). Digital circuit evolution and fitness landscapes. In *Proc. Congress on Evolutionary Computation*. Piscataway, N.J. IEEE Press.

Xilinx, Inc. (1996). XC6200 Advanced product specification V1.0, June 1996. In *The Programmable Logic Data Book*. See http://www.xilinx.com.

Xilinx, Inc. (1997). *XC6200 Field Programmable Gate Arrays*. Data Sheet, www.xilinx.com. Version 1.10.

Xilinx, Incorporated (1999). *Virtex 2.5V Field Programmable Gate Arrays Datasheet V1.7, San Jose, CA*.

Yang, E. S. (1988). *Microelectronic devices* (Int. edition). Elelectronics and electronic circuits. McGraw-Hill.

Yao, X., & Higuchi, T. (1996). Promises and challenges of evolvable hardware. In Higuch, T., Iwata, M., & Lui, W. (Eds.), *1st International Conference on Evolvable Systems: From Biology to Hardware (ICES 96)*, Vol. 1259 of *LNCS*. Springer-Verlag.

Zebulum, R., Stoica, A., & Keymeulen, D. (2000). A flexible model of a CMOS field programmable transistor array targeted for hardware evolution. In Miller, J., Thompson, A., Thomson, P., & Fogarty, T. (Eds.), *3rd International Conference on Evolvable Systems: From Biology to Hardware*, Vol. 1801 of *LNCS*, pp. 274–283. Springer.

Zebulum, R. S., Pacheco, M. A., & Vellasco, M. (1996). Evolvable systems in hardware design: taxonomy, survey and applications. In Higuchi, T., & Iwata, M. (Eds.), *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, Vol. 1259 of *LNCS*, pp. 344–358. Springer-Verlag.

Zebulum, R. S., Vellasco, M., & Pacheco, M. A. (1998). Analog circuits evolution in extrinsic and intrinsic modes. In Sipper, M., Mange, D., & Pérez-Uribe, A. (Eds.), *Proc. 2nd Int. Conf. on Evolvable Systems (ICES98)*, Vol. 1478 of *LNCS*, pp. 154–165. Springer-Verlag.

Zetex plc (1996). TRAC20 totally reconfigurable analog circuit. Provisional Datasheet Issue 2. See http://www.zetex.com.

Appendix A

Acronyms

A/D	Analogue to Digital
AC	Alternating Current
APR	Automatic Placement and Routing
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuit
BBF	Best Before Fault
CAD	Computer Aided Design
CCS	Calculus of Communicating Systems
CLB	Control Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CSEM	Centre Suisse d'Electronique et de Microtechnique SA
D/A	Digital to Analogue
DNA	DeoxyriboNucleic Acid
DC	Direct Current
DSM	Dynamic State Machine
EA	Evolutionary Algorithm
EHW	Evolvable HardWare
EM	Evolvable Motherboard
EMI	Electro-Magnetic Interference
EPFL	Ecole Polytechnique Fédérale de Lausanne
ES	Evolutionary Strategy
ETL	Electrotechnical Laboratory
FET	Field-Effect Transistor
FIPSOC	FIeld-Programmable System On a Chip
FPAA	Field-Programmable Analogue Array
FPGA	Field-Programmable Gate Array
FPTA	Field-Programmable Transistor Array
FSM	Finite State Machine

GA	Genetic Algorithm
GP	Genetic Programming
HE	Hardware Evolution
IC	Integrated Circuit
IOB	Input/Output Block
ISA	Industry Standard Architecture
LSI	Large Scale Integration
NEWS	North, East, South, West
PCB	Printed Circuit Board
PCI	Peripheral Component Interface
PLD	Programmable Logic Device
PFT	Populational Fault Tolerance
RAM	Random Access Memory
RC	Resistor-Capacitor
ROM	Read Only Memory
SAGA	Species Adaptation Genetic Algorithm
SCCS	Stochastic Calculus of Communicating Systems
SSA	Single Stuck-At
VGA	Variable-Length Chromosome GA
VLSI	Very Large Scale Integration
WSCCS	Weighted Stochastic Calculus of Communicating Systems
XOR	Exclusive OR

Appendix B

Tone Discriminator: Simulation Schematic and Waveforms

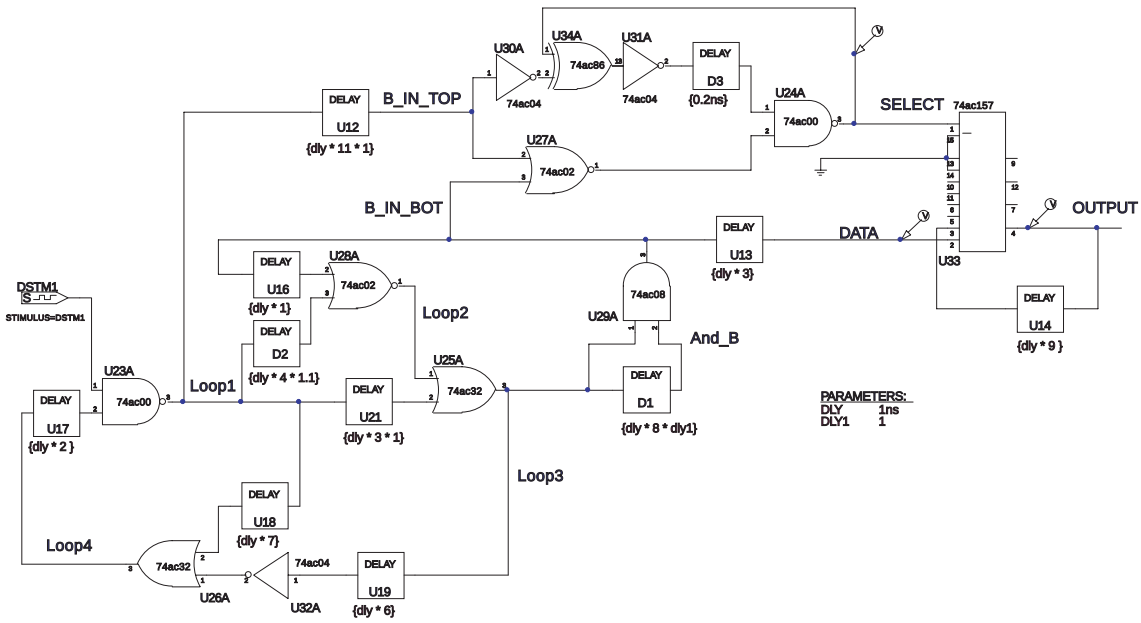


Figure B.1: Circuit schematic used for simulating the tone discriminator. The hexagonal boxes of the schematic from chapter 3 are replaced here by parametric simulator delays.

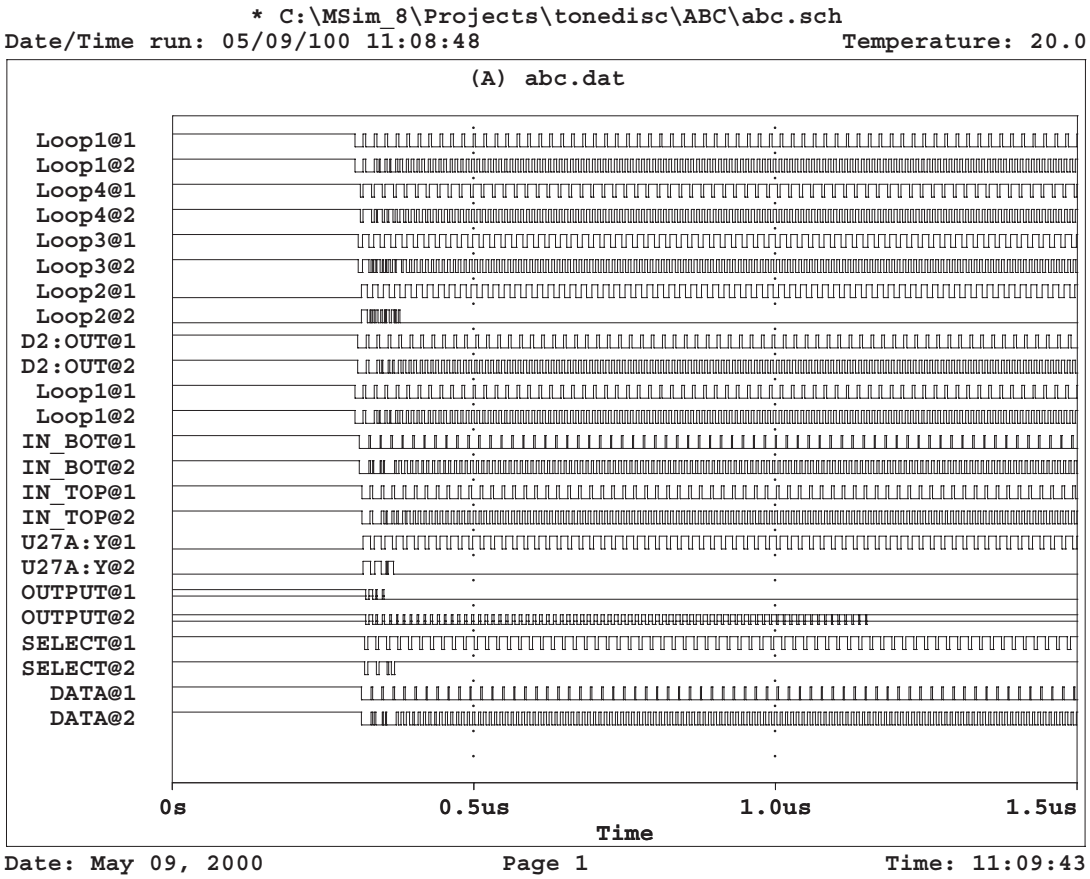


Figure B.2: Simulated waveforms at all nodes of the tone discriminator circuit. Two waveforms are shown for each node; the first for delay D1 = 8.1ns and the second for D1 = 8.2ns. The loops are marked on the schematic of the previous page. D2:OUT is the connection from delay D2 to U28a, IN_BOT and IN_TOP are marked as B_IN_BOT and B_IN_TOP respectively.

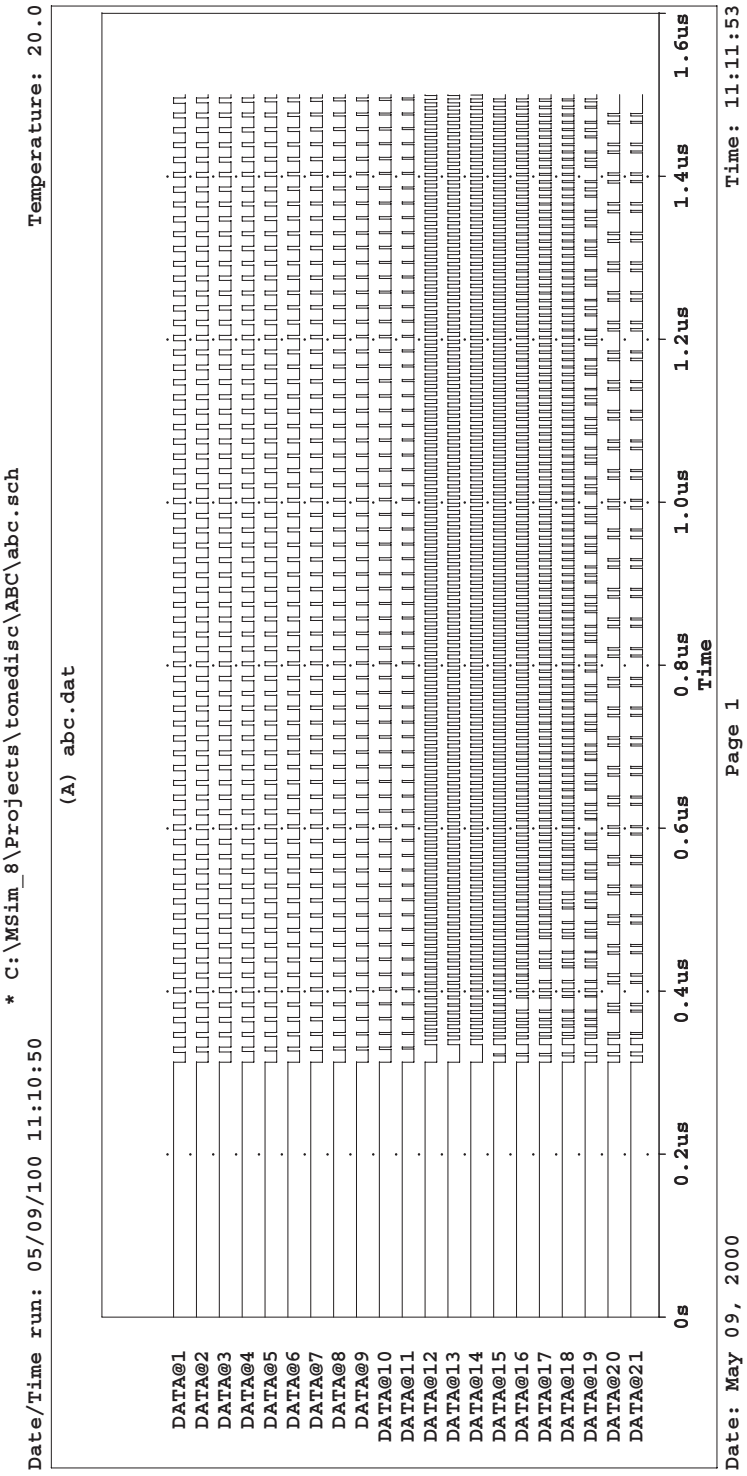


Figure B.3: Simulated waveforms showing three distinct oscillation modes present at the DATA input to multiplexer U33, depending on the value of delay D1. Here, waveforms are displayed for D1 = 4ns to 12ns in 400ps intervals

Appendix C

Evolvable Motherboard: Sample Code and Printed Circuit Foil Patterns

C.1 Sample Code for Configuring the EM

The following C routines are examples of how the EM can be configured using the ISA Interface described in chapter 4. Routines are provided for setting switches either by using the crosspoint chip's own address format, or by specifying its EM row and column.

```
////////////////////////////////////////////////////////////////////////

#define ADDRESS   0x0300
#define HANDSHAKE 0x0301
#define PORTNUM   0

////////////////////////////////////////////////////////////////////////

void reset_motherboard()
// resets all crosspoint switches to open by toggling the RESET line
{
    outportb(HANDSHAKE,0x80);

    delay(1);
    outportb(HANDSHAKE,0x00);
}
////////////////////////////////////////////////////////////////////////

void setswitch(int state, unsigned char chip, unsigned int
switchaddress)
// sets crosspoint switch to state at address switchaddress, chip no. chip
{
    unsigned char byte;
    // first set chip select latch
    byte = chip;               //   delay(1);
    byte &= 0x0F;
    outportb(HANDSHAKE,byte);  //  delay(1);
    byte |= 0x10;
    outportb(HANDSHAKE,byte);  //  delay(1);
    byte &= 0x0F;
    outportb(HANDSHAKE,byte);  //  delay(1);

    // then set the correct switch address
    outportb(ADDRESS,switchaddress);

    // determine open or closed
    if (state == 1)    // if switch to be closed...
        byte |= 0x40;   // .. DATA line high

    // now strobe into relevant chip
    byte |= 0x20;
    outportb(HANDSHAKE,byte);  //   delay(1);
    byte &= 0xDF;
    outportb(HANDSHAKE,byte);  //   delay(1);
}
////////////////////////////////////////////////////////////////////////
```

```
bool setrowcol(int state, int row, int col)
// sets up motherboard switches according to global row and column.
// row 0, col 0 are top left of the motherboard
// if there is no switch at the specified row/col, FALSE is returned
{
    int chip;
    unsigned int  address;
    int x = 7 - col%8;
    int y;
    int offset = 0;

    // First determine correct chip
    if (!chip_no(row,col,&chip)) return FALSE;

    if ((col/8)%2) offset = 8;
    y = (row-offset)%16;

    // Now convert appropriate rows & cols to addresses for this chip
    // using a lookup table
    address = _rotl(x,4);
    switch (y)
    {
        case 0 : address |= 0x0;break; case 8 : address |= 0xa;break;
        case 1 : address |= 0x1;break; case 9 : address |= 0xb;break;
        case 2 : address |= 0x2;break; case 10: address |= 0xc;break;
        case 3 : address |= 0x3;break; case 11: address |= 0xd;break;
        case 4 : address |= 0x4;break; case 12: address |= 0x6;break;
        case 5 : address |= 0x5;break; case 13: address |= 0x7;break;
        case 6 : address |= 0x8;break; case 14: address |= 0xe;break;
        case 7 : address |= 0x9;break; case 15: address |= 0xf;break;
    }
    setswitch (state, chip, address);
    return TRUE;
}
////////////////////////////////////////////////////////////////////////
// CONVERSION ROUTINES
////////////////////////////////////////////////////////////////////////

bool chip_no(int row, int col, int *chipno)
// gives chip no corresponding to a global row/col,
// returning FALSE if there is no chip.
{
    int chip=0;
    int xchip = col/8;
    int ychip;
    int offset = 0;
    // check for obvious illegal values;
    if ((row <0) || (col <0) || (row > 55) || (col > 47)) return FALSE;

    // Find correct chip
    if (xchip%2) offset = 8;
    ychip = (row-offset)/16;

    switch (xchip)
    {
        case (0):   chip = (2 - ychip);
                    if ((chip < 0) || (chip > 2))       return FALSE;
                    break;
        case (1):   chip = (5 - ychip);
                    if ((chip < 3) || (chip > 5) || (row <8))   return FALSE;
                    break;
        case (2):   chip = (8 - ychip);
                    if ((chip < 6) || (chip > 7))       return FALSE;
                    break;
        case (3):   chip = (10 - ychip);
                    if ((chip < 8) || (chip > 9))       return FALSE;
                    break;
        case (4):   chip = (12 - ychip);
                    if (chip != 10)             return FALSE;
                    break;
        case (5):   chip = (13 - ychip);
                    if (chip != 11)             return FALSE;
                    break;
    }
    *chipno = chip;
    return TRUE;
}
////////////////////////////////////////////////////////////////////////

bool localrowcol_chipno(int row, int col,   int *lrow, int *lcol,
int *chipno)
// gives chip no, and its local row & col nos for a given
// global row & col.
{
    int chip=0;
    int x = col%8;
    int y;
    int offset = 0;
```

```
    // First find correct chip
    if (!chip_no(row,col,&chip)) return FALSE;

    if ((col/8)%2) offset = 8;
    y = (row-offset)%16;

    *chipno = chip;*lrow = y; *lcol = x;
    return TRUE;
}
////////////////////////////////////////////////////////////////////////

bool address_chipno(int row, int col, int *addr, int *chipno)
// gives chip no & address for a global row & col.
// row 0, col 0 are top left of diagram as drawn by routine in DRAWSTAT.C
// if there is no switch at the specified row/col, FALSE is returned
{
    unsigned int address=0;
    int chip=0;
    int xchip = col/8;
    int x = col%8;
    int ychip, y;
    int offset = 0;

    // First find correct chip
    if (!chip_no(row,col,&chip)) return FALSE;

    if ((col/8)%2) offset = 8;
    y = (row-offset)%16;


    // Now convert appropriate rows & cols to addresses for this chip
    address = _rotl(x,4);
    switch (y)

        case 0 : address |= 0x0;break; case 8 : address |= 0xa;break;
        case 1 : address |= 0x1;break; case 9 : address |= 0xb;break;
        case 2 : address |= 0x2;break; case 10: address |= 0xc;break;
        case 3 : address |= 0x3;break; case 11: address |= 0xd;break;
        case 4 : address |= 0x4;break; case 12: address |= 0x6;break;
        case 5 : address |= 0x5;break; case 13: address |= 0x7;break;
        case 6 : address |= 0x8;break; case 14: address |= 0xe;break;
        case 7 : address |= 0x9;break; case 15: address |= 0xf;break;

    *addr = address;*chipno = chip;
    return TRUE;
}
////////////////////////////////////////////////////////////////////////
```

C.2 Printed Circuit Foil Patterns

Printed circuit board foil patterns are shown here for the 48×48 evolvable motherboard used throughout this thesis. The printed circuit board is double sided, through-plated. If through-plating facilities are unavailable pins can be soldered through to join the two sides. Note that these patterns are not to scale.

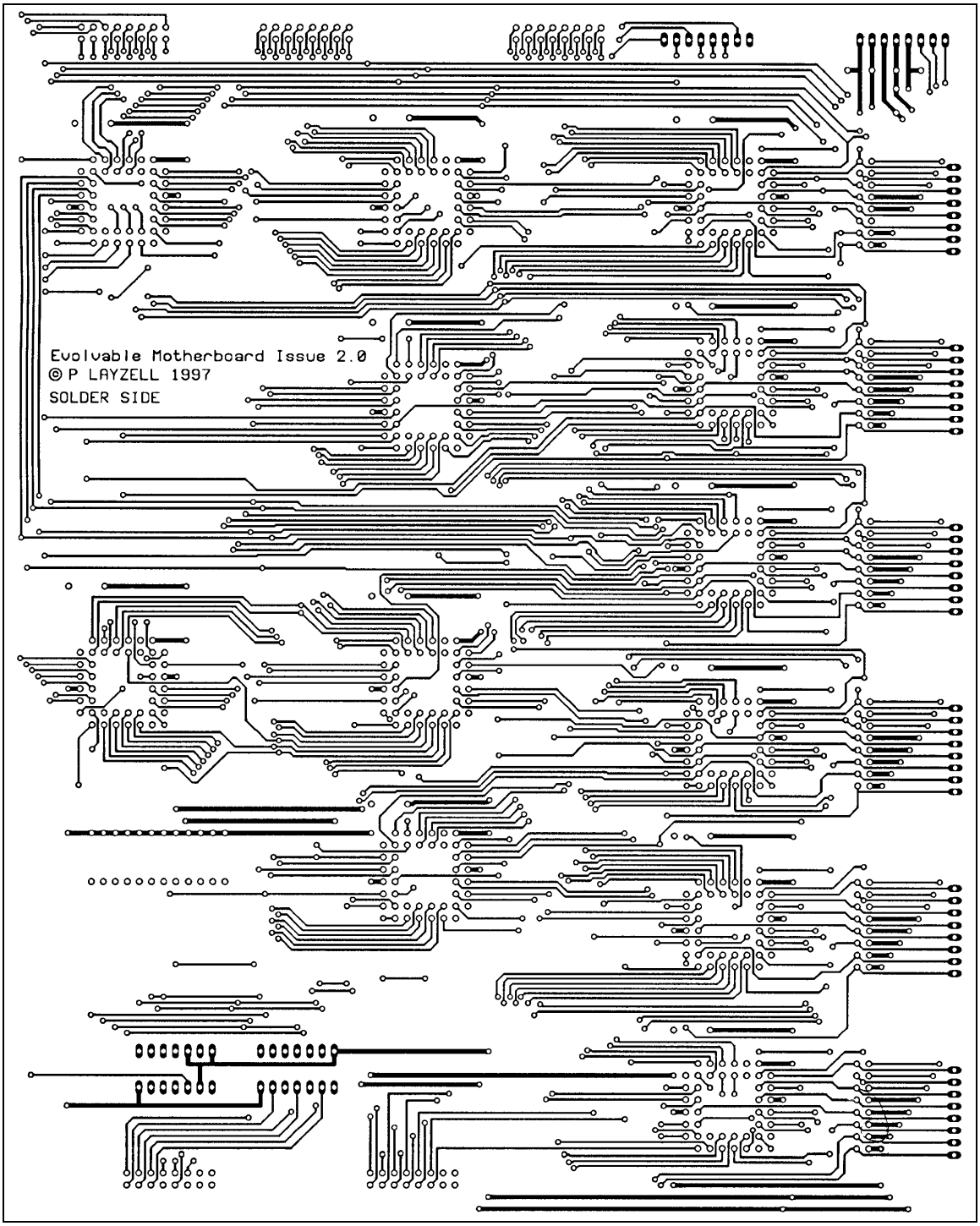


Figure C.1: Printed Circuit Foil Pattern for the 48×48 Evolvable Motherboard: Solder Side.

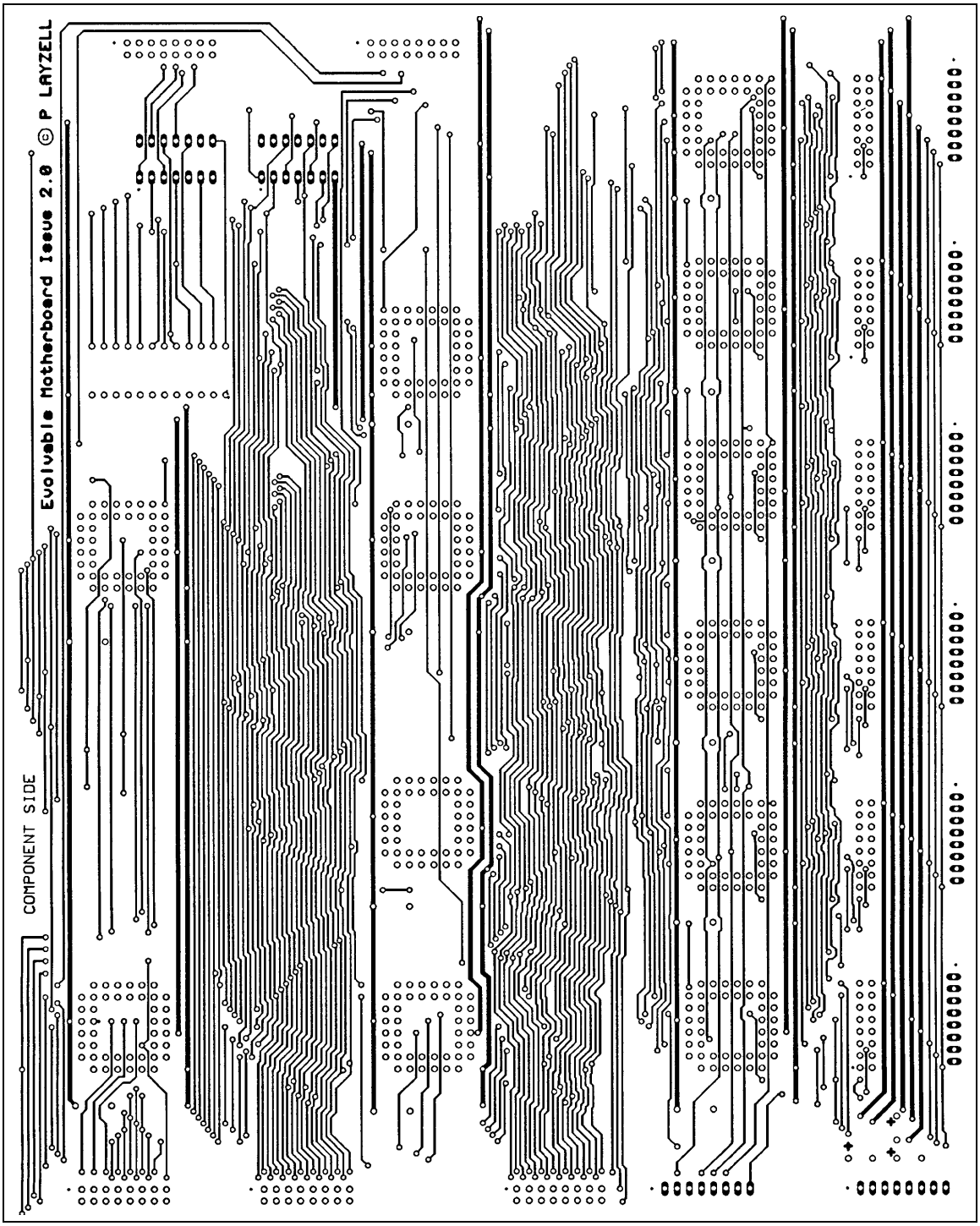


Figure C.2: Printed Circuit Foil Pattern for the 48×48 Evolvable Motherboard: Component Side.

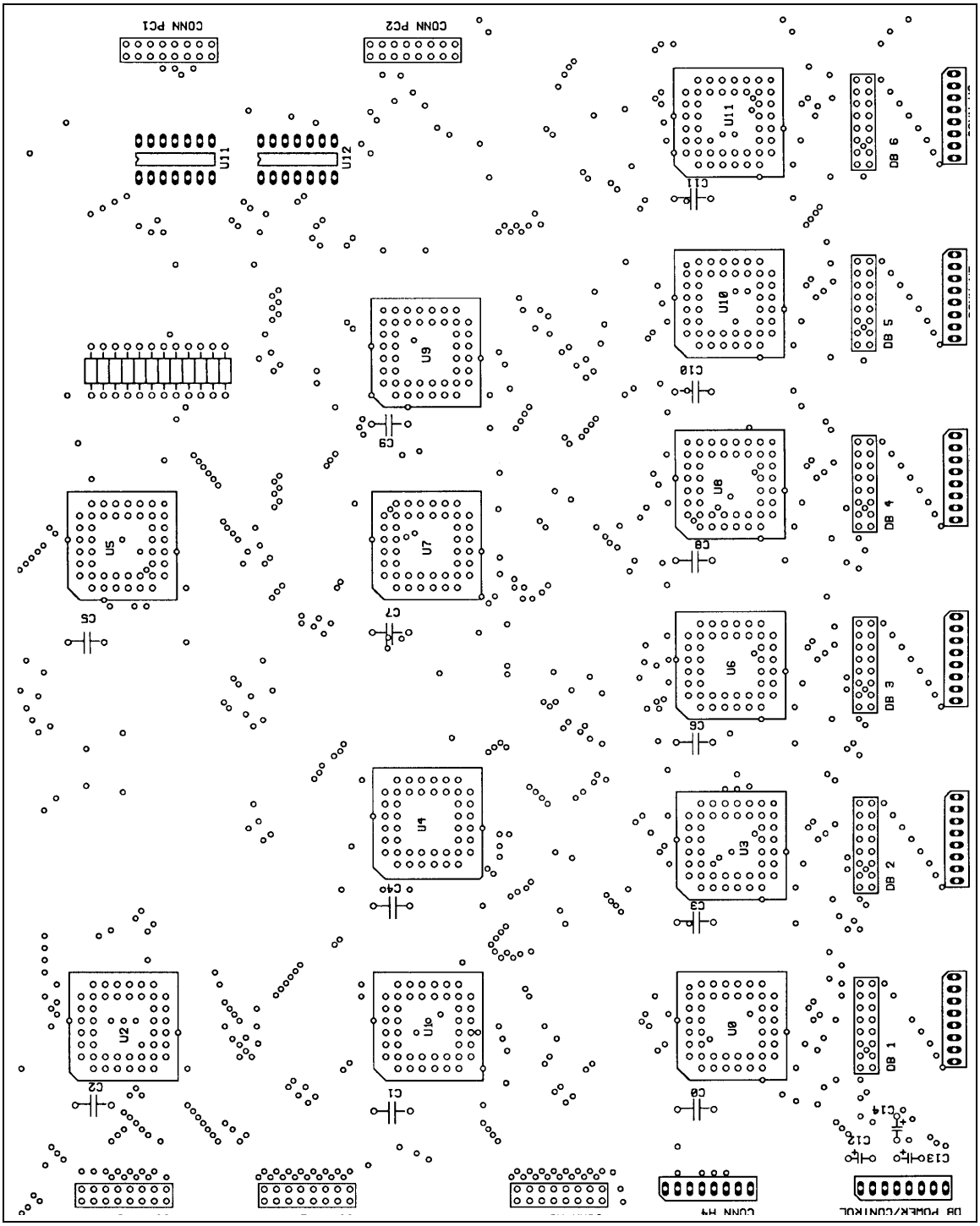
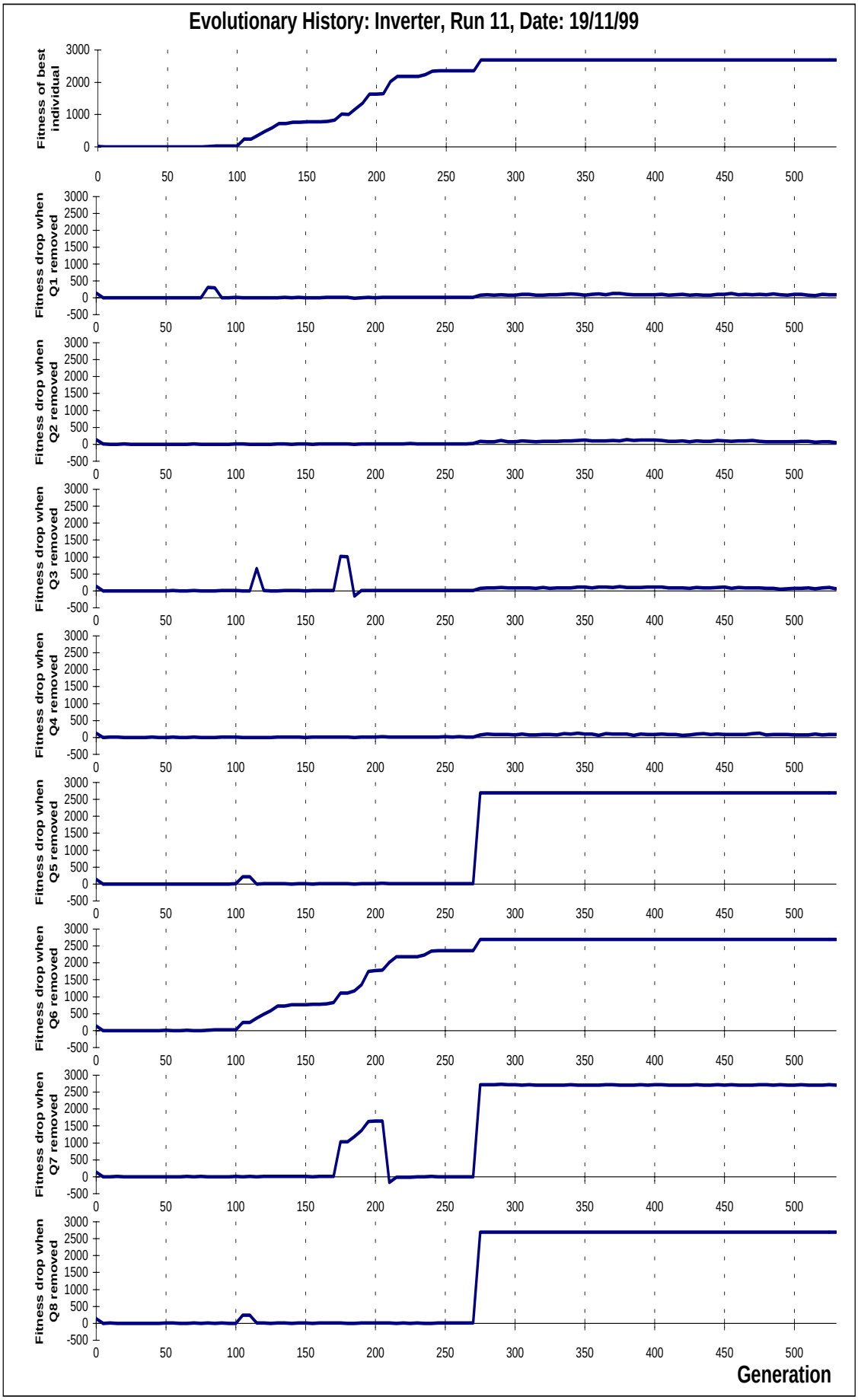


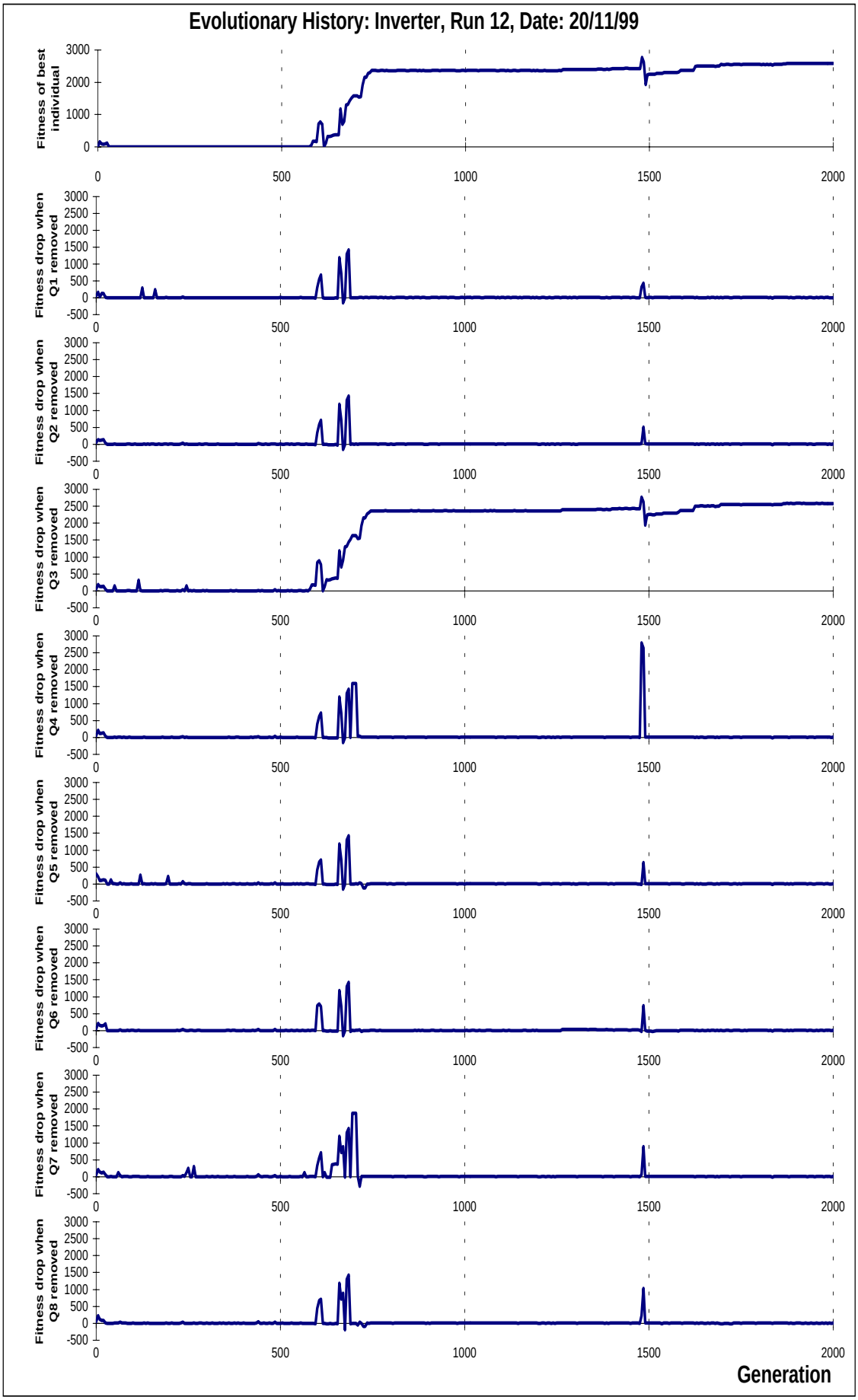
Figure C.3: Component legend for the the 48×48 Evolvable Motherboard.

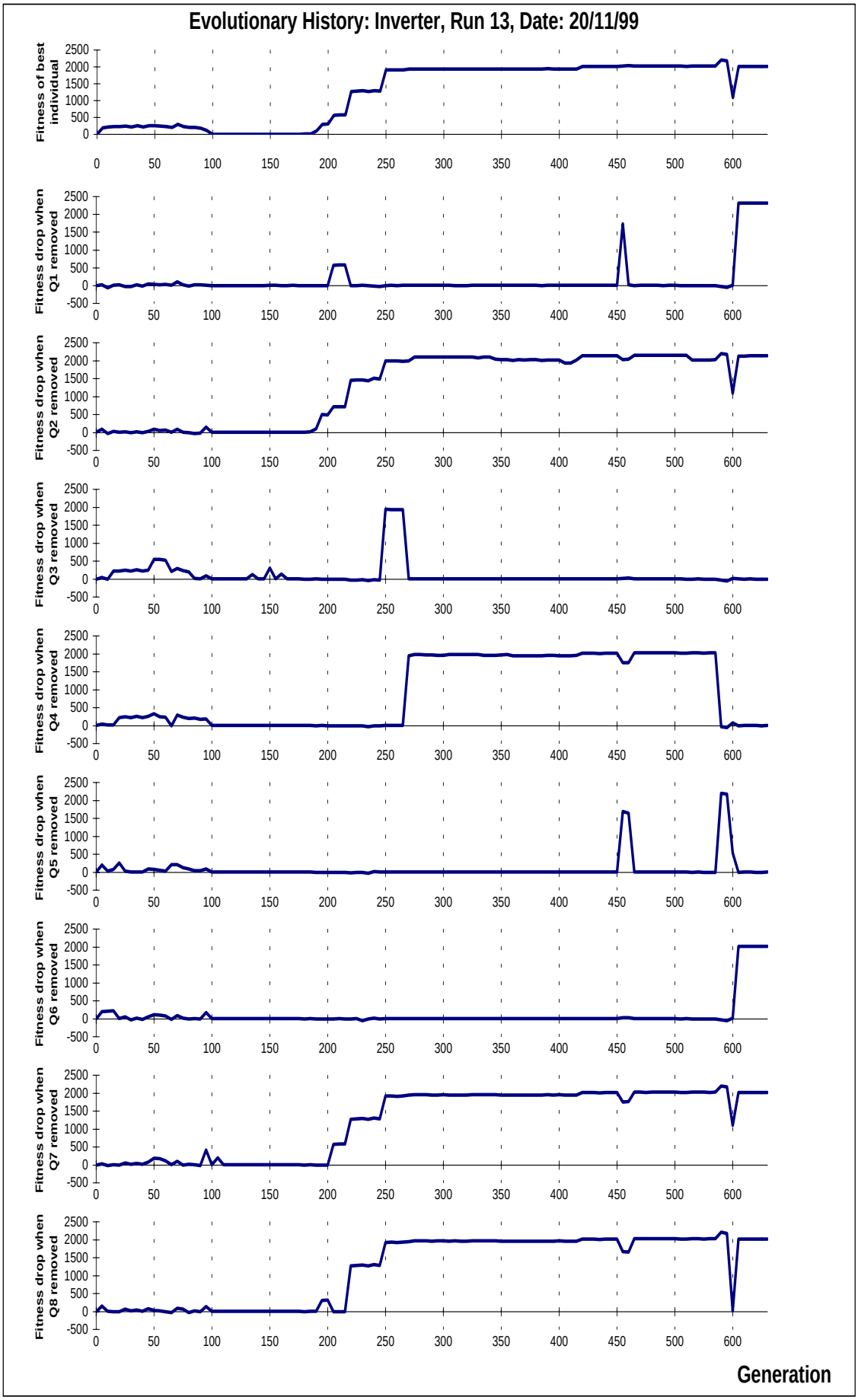
Appendix D

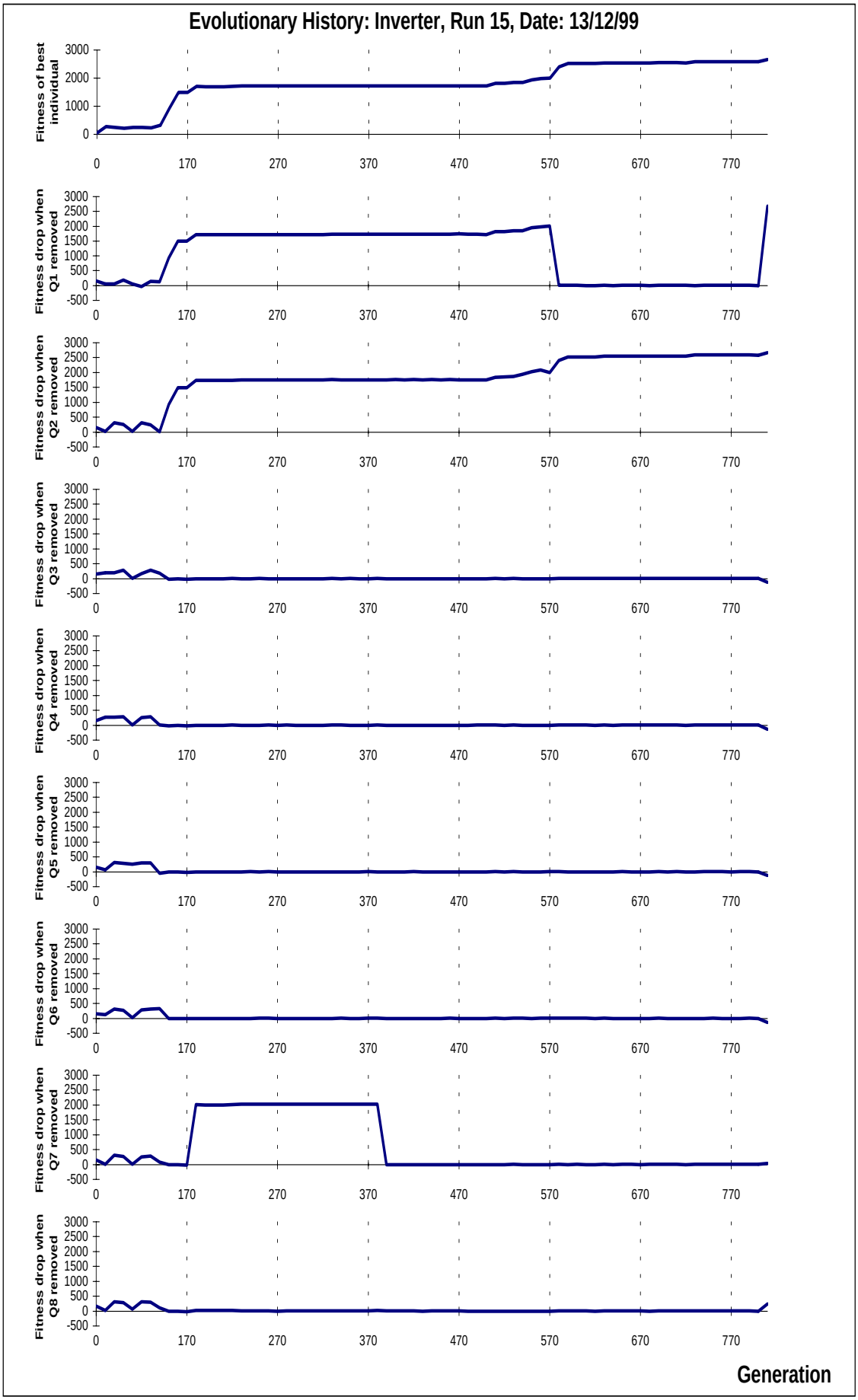
Importance Profiles

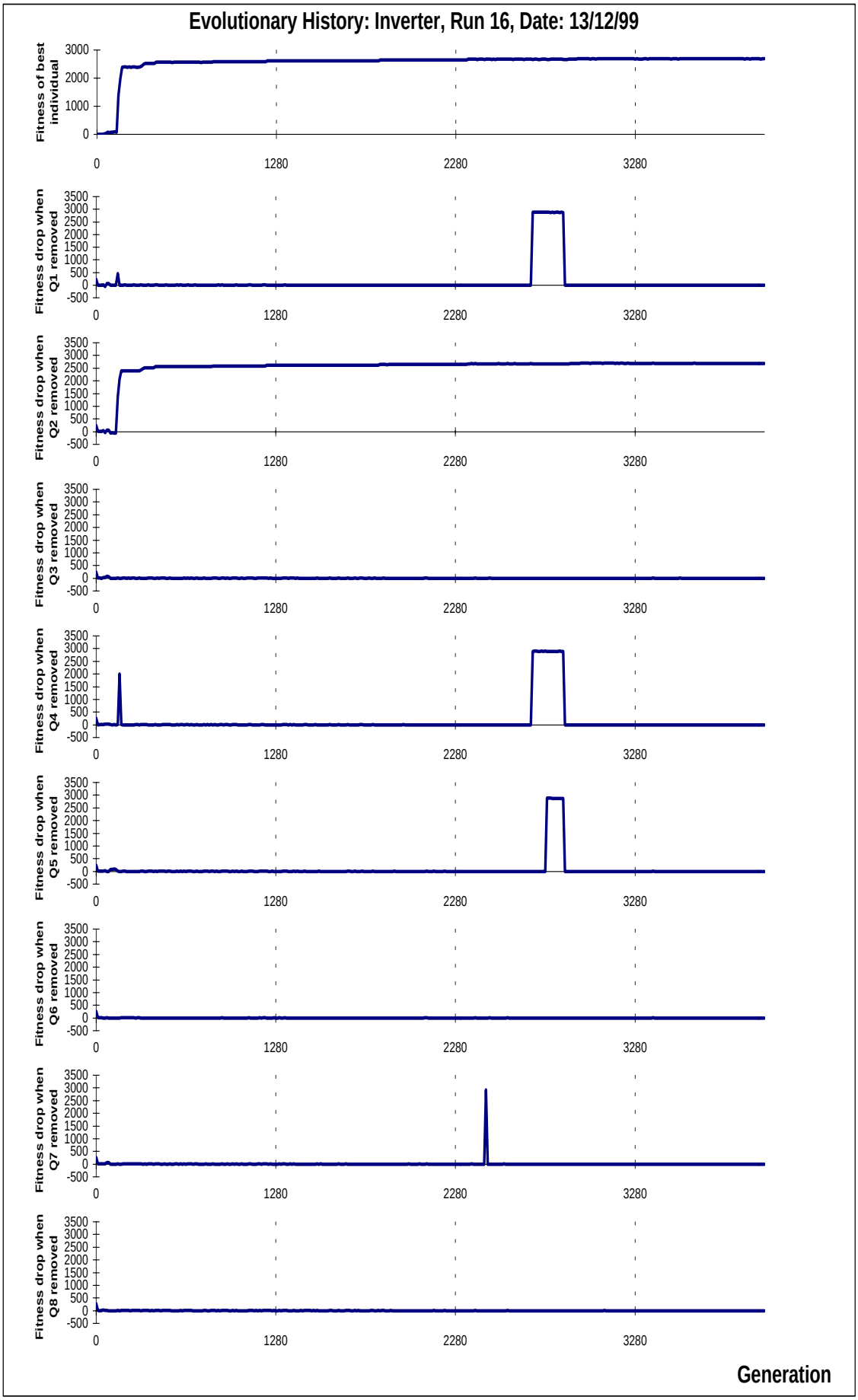
Importance profiles for most of the runs of section 6.4.3 are included here. Recall that they were conducted for 10 runs per task. The data for these runs are of considerable size, and were stored on CD-ROM. Unfortunately, the CD-ROM on which the oscillator profiles were stored refuses to yield its data, and it has only been possible to reproduce about half of them here. Otherwise, profiles not already shown in section 6.4.3 are provided here for all runs of every task.

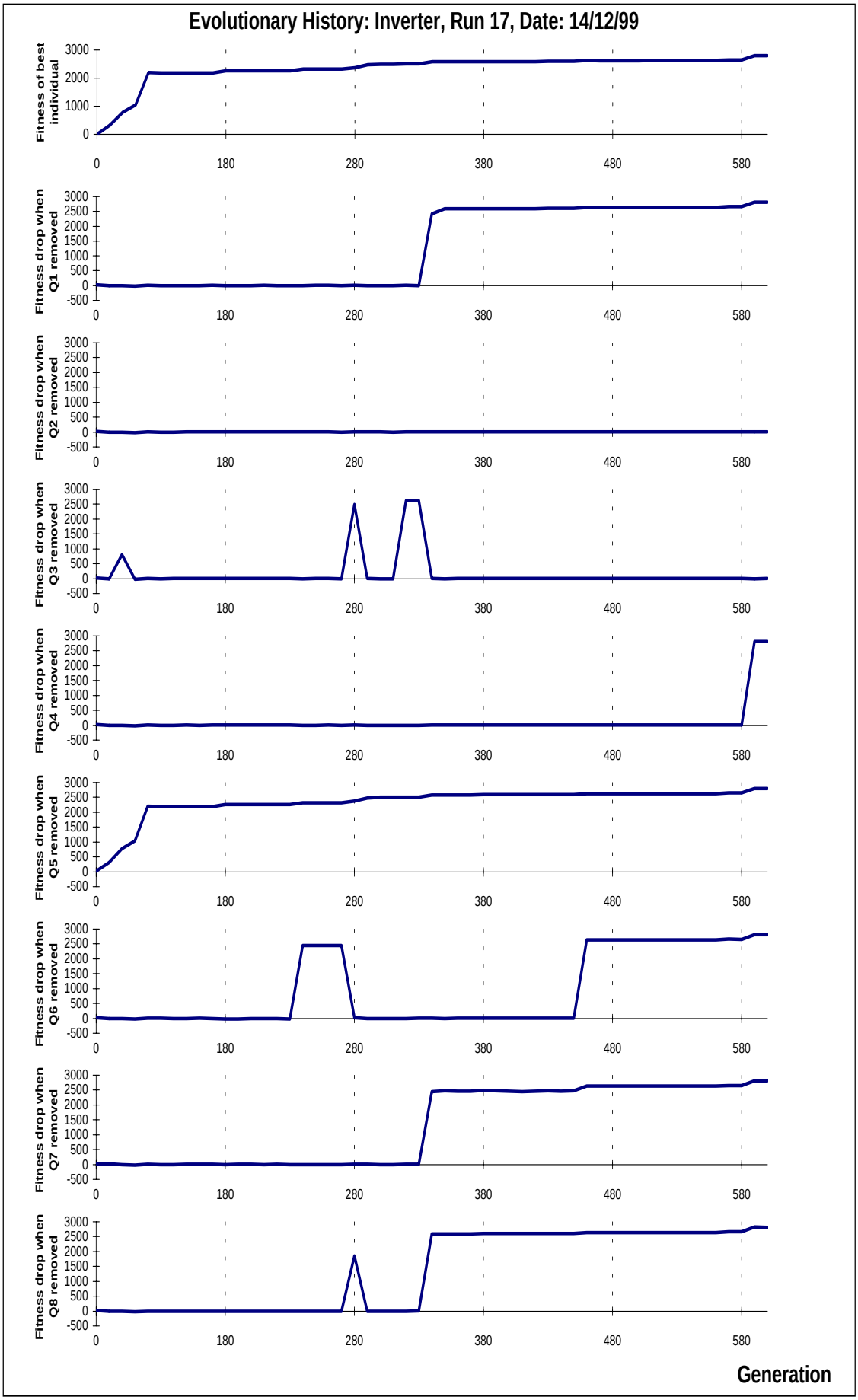

Evolutionary History: Inverter, Run 11, Date: 19/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

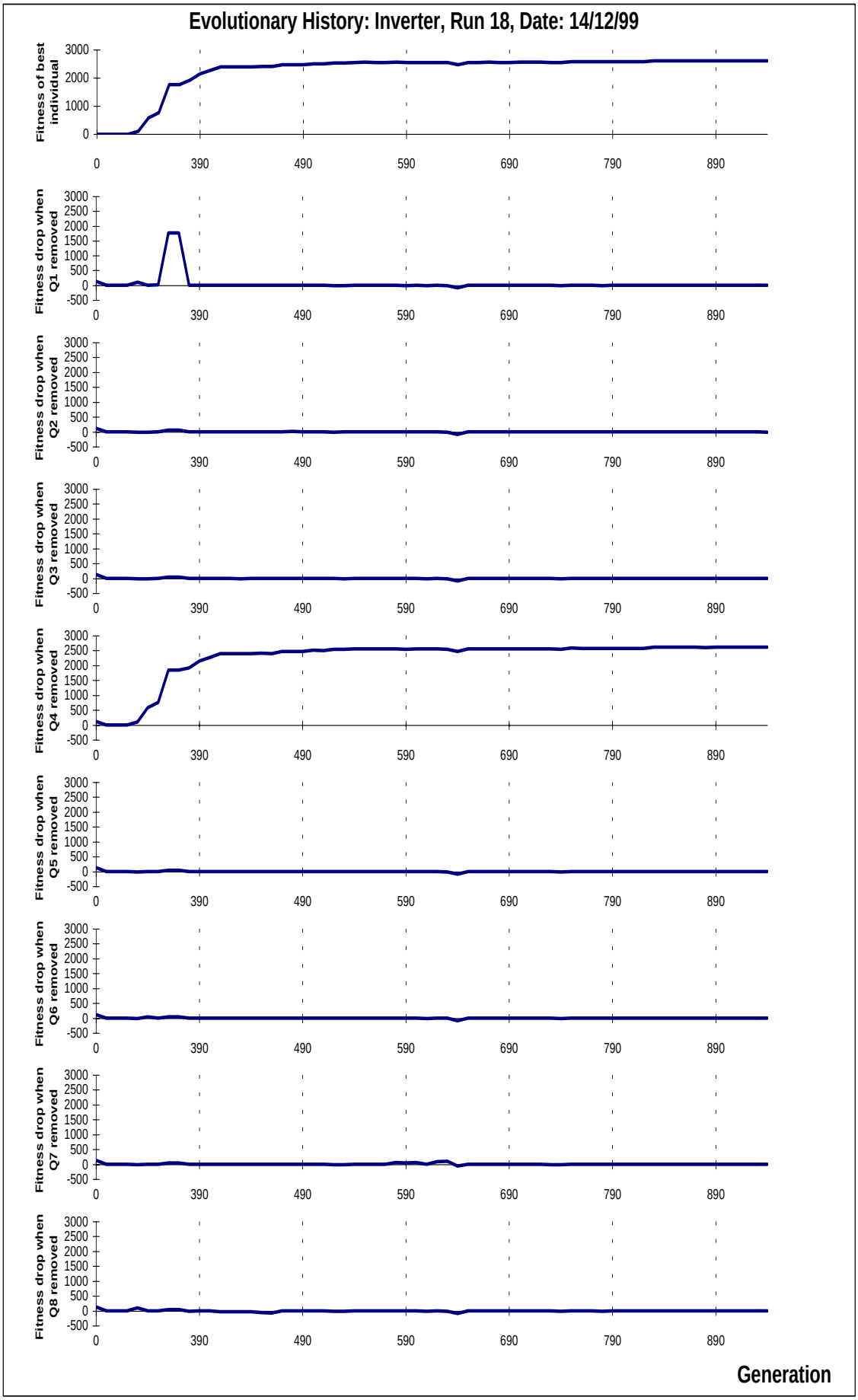

Evolutionary History: Inverter, Run 12, Date: 20/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

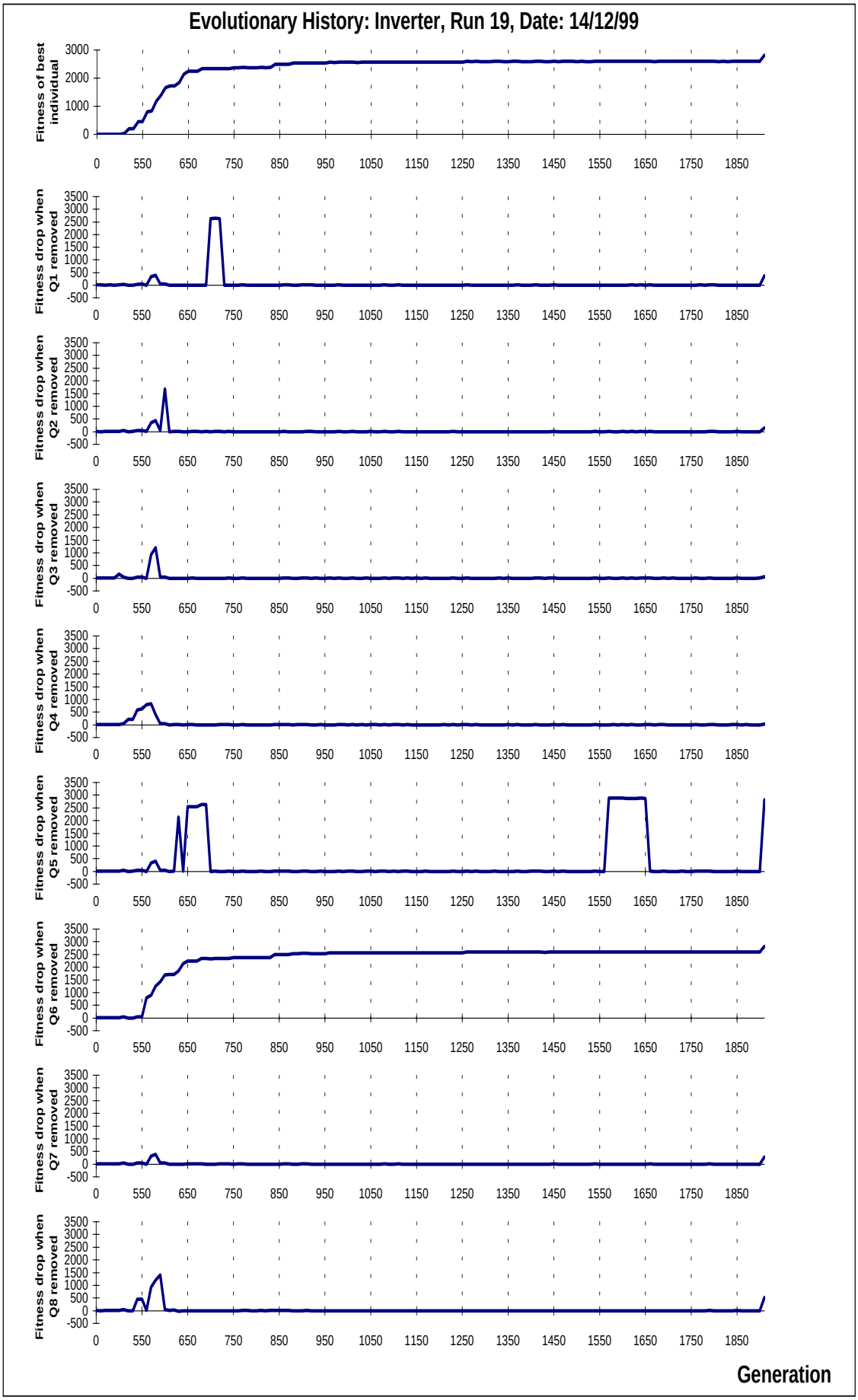

Evolutionary History: Inverter, Run 13, Date: 20/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
2500
2000
1500
1000
500
0
-500
0
50
100
150
200
250
300
350
400
450
500
600
Generation

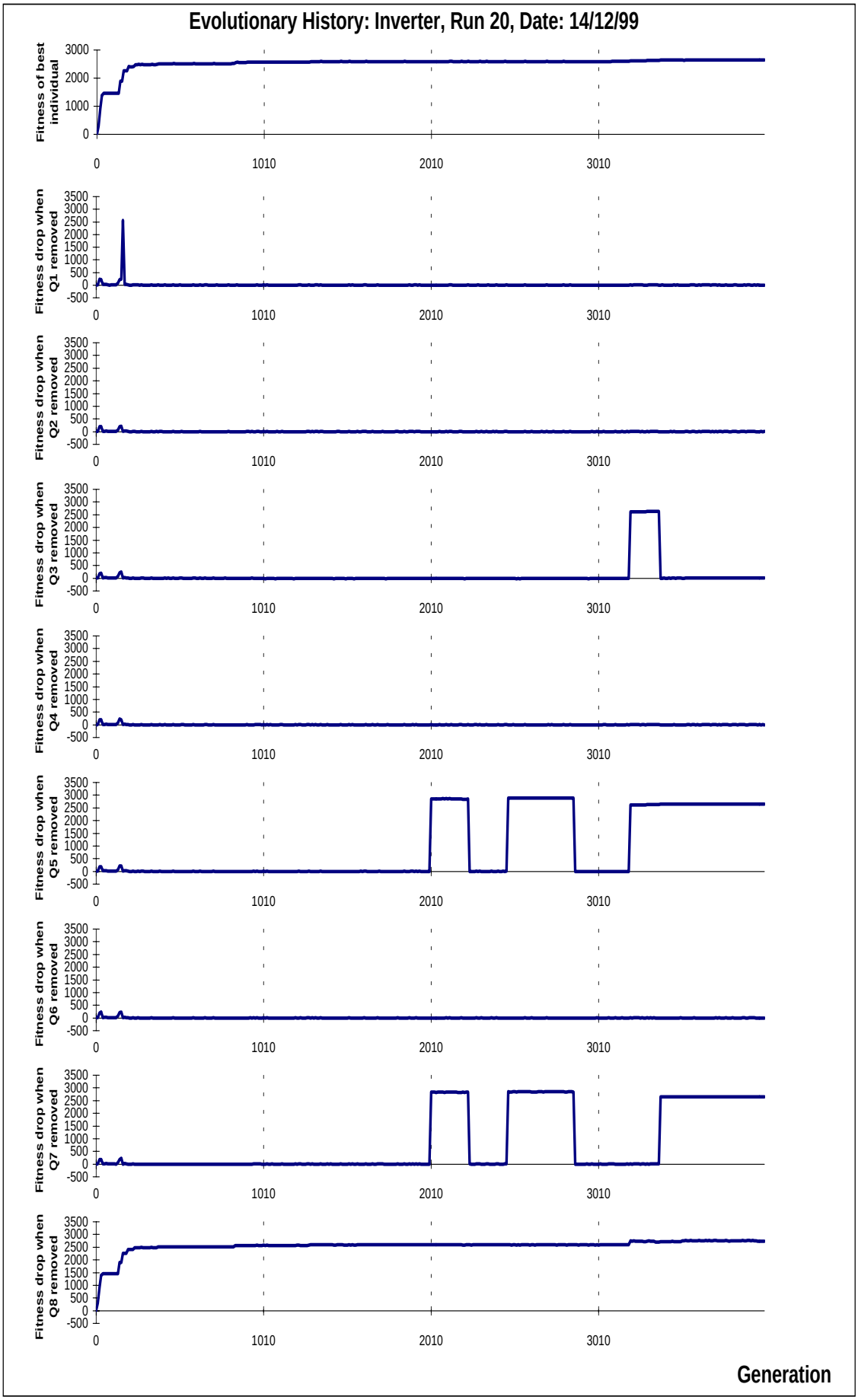

Evolutionary History: Inverter, Run 15, Date: 13/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
170
270
370
470
570
670
770
Generation

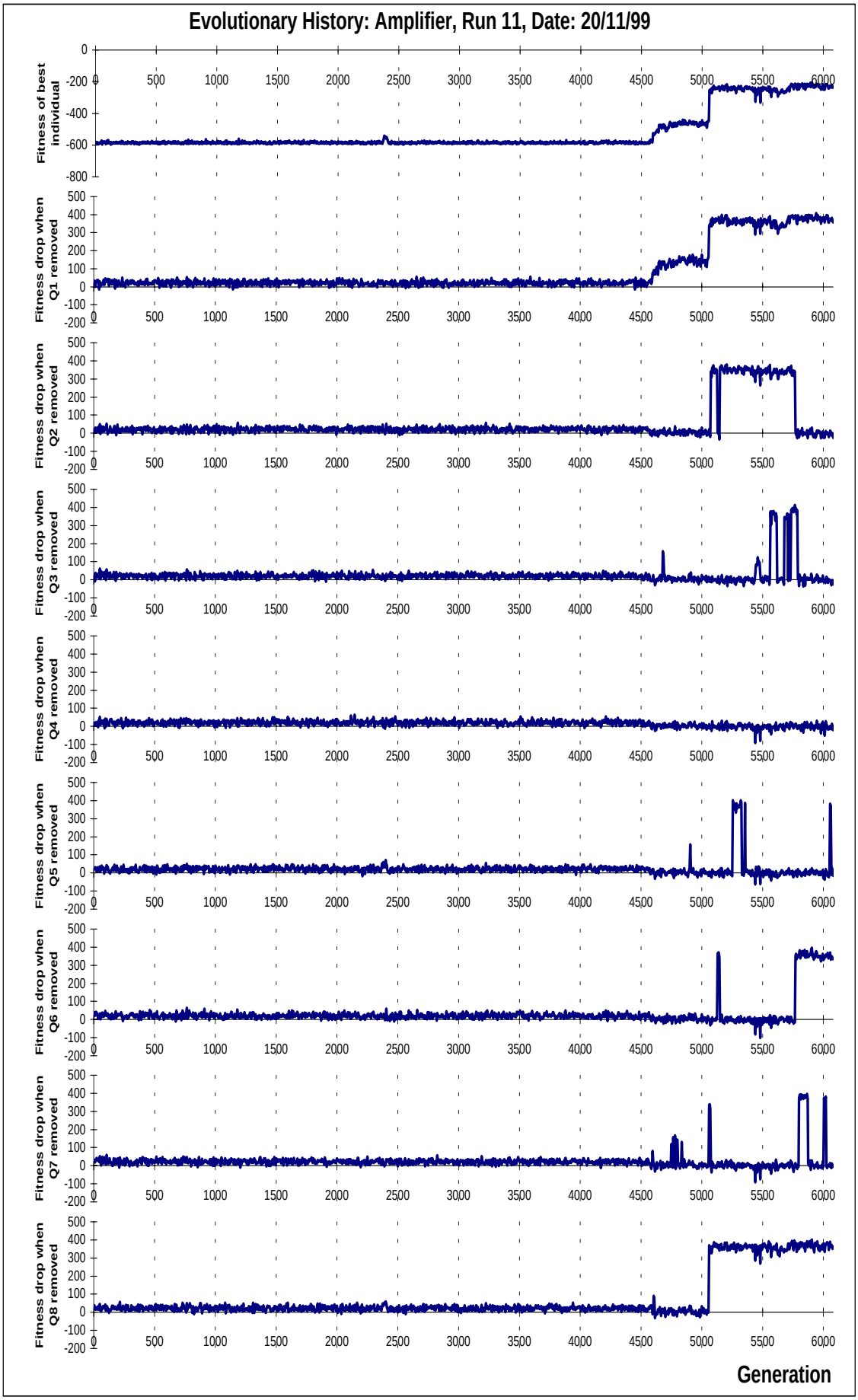

Evolutionary History: Inverter, Run 16, Date: 13/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
1280
2280
3280
Generation

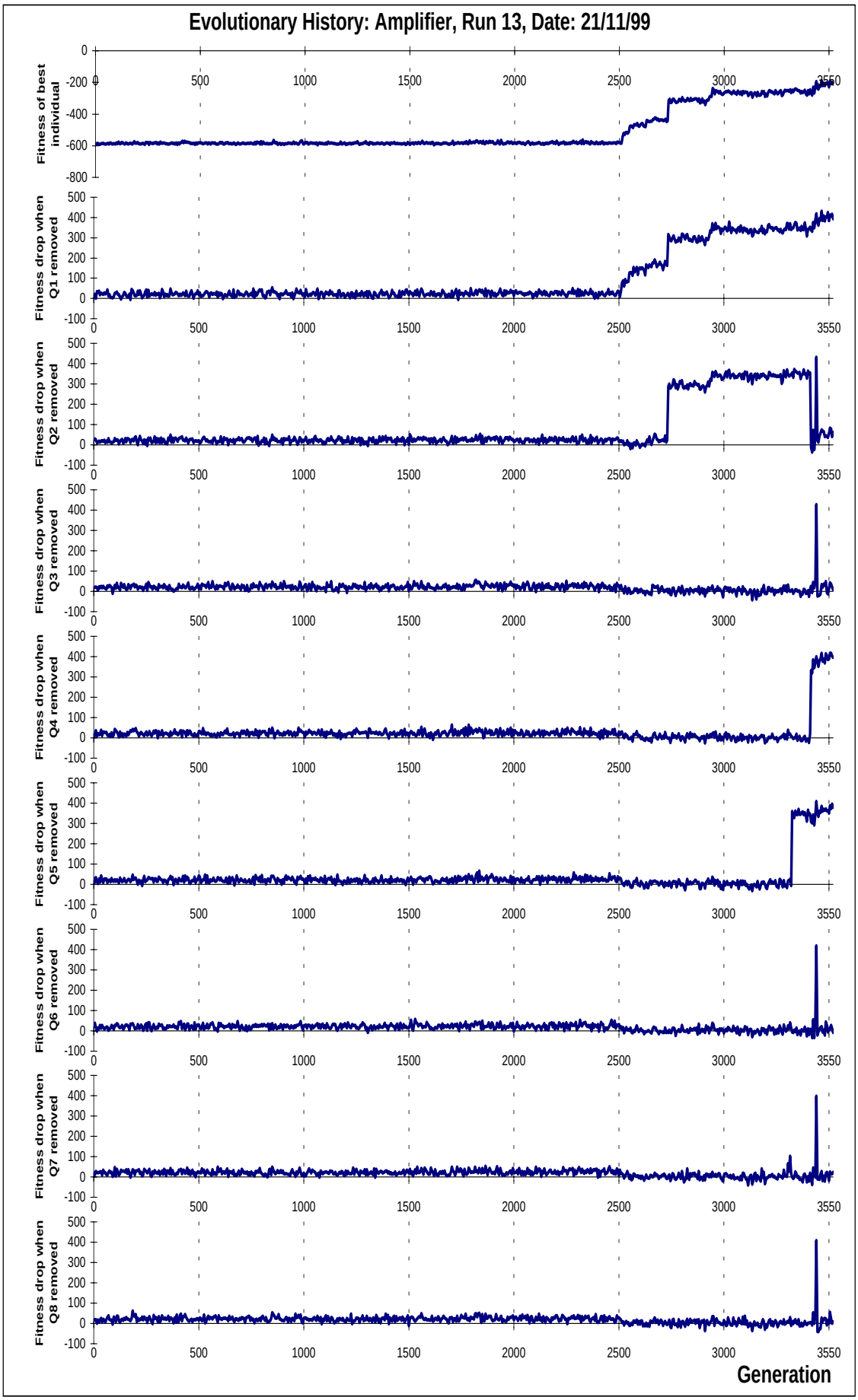

Evolutionary History: Inverter, Run 17, Date: 14/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
180
280
380
480
580
Generation

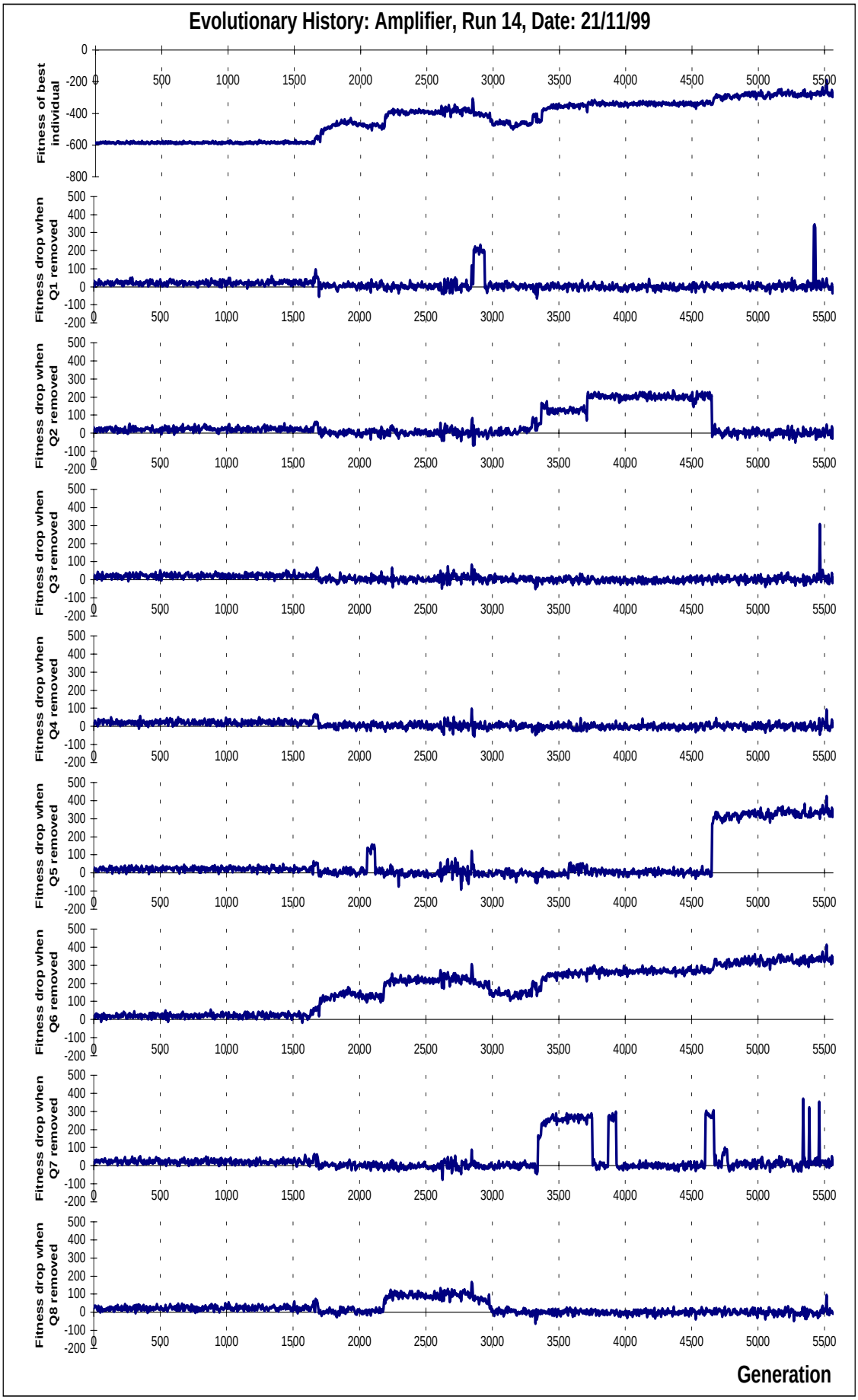

Evolutionary History: Inverter, Run 18, Date: 14/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
390
490
590
690
790
890
Generation

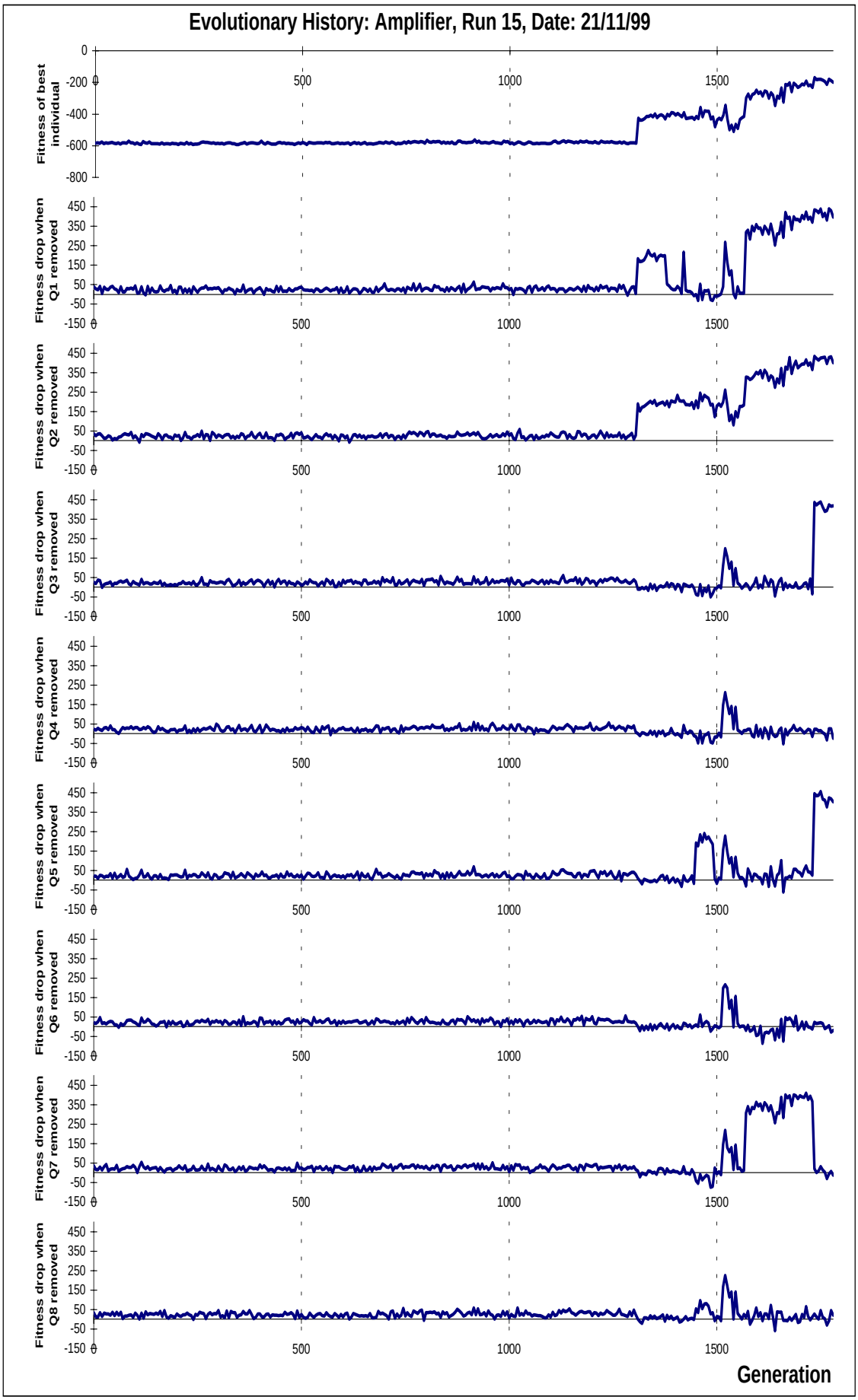

Evolutionary History: Inverter, Run 19, Date: 14/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0 550 650 750 850 950 1050 1150 1250 1350 1450 1550 1650 1750 1850
Generation

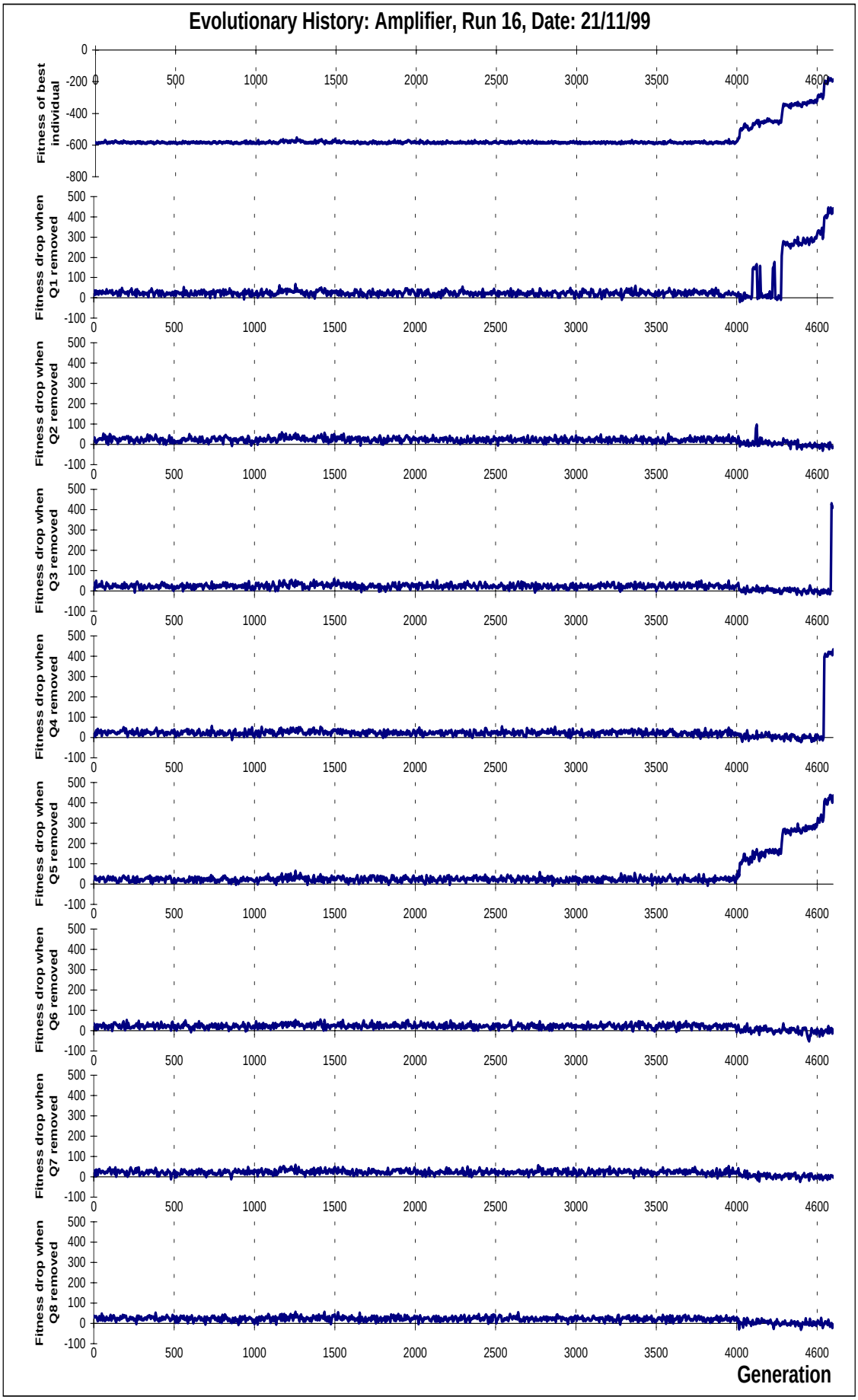

Evolutionary History: Inverter, Run 20, Date: 14/12/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
1010
2010
3010
Generation

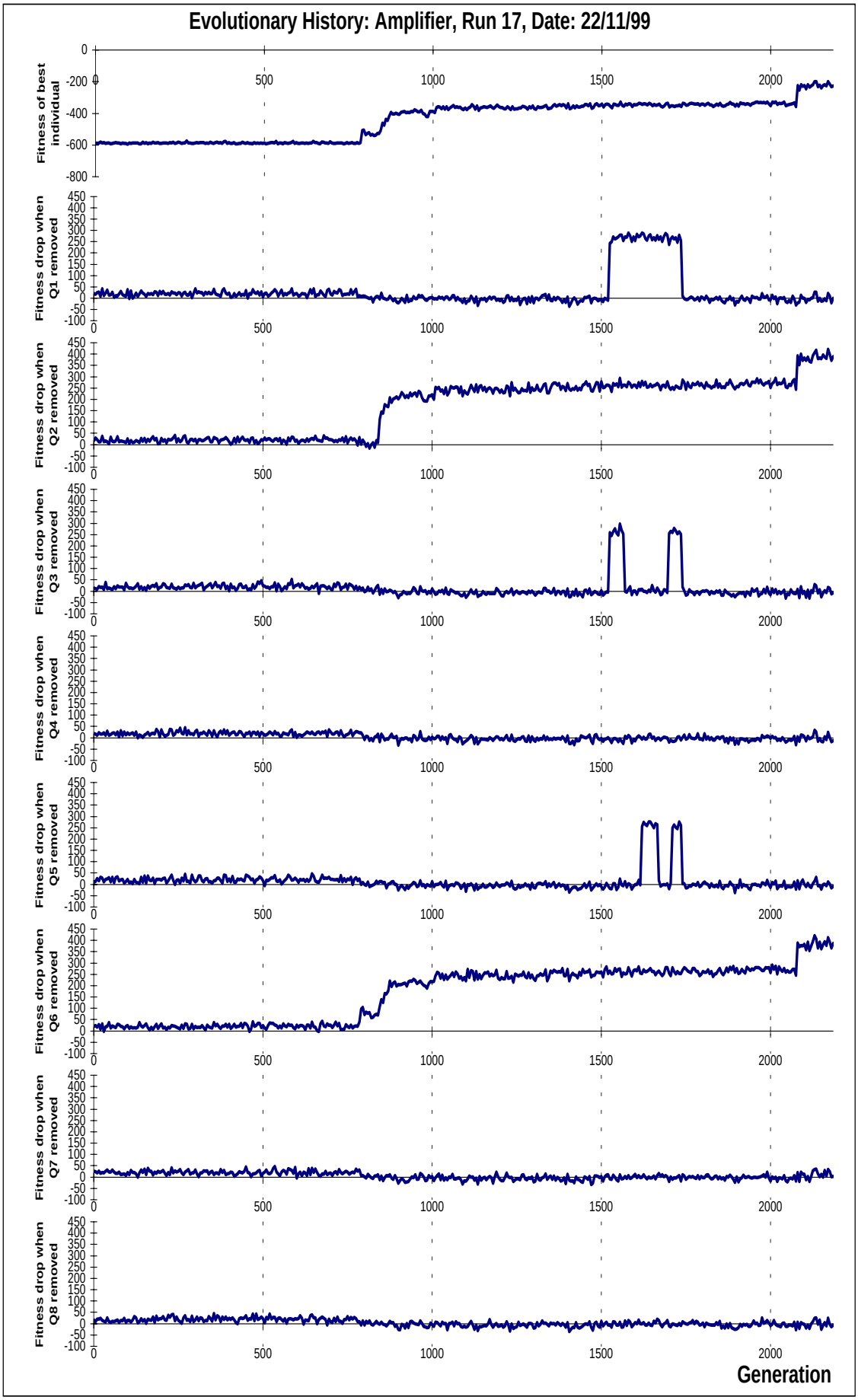

Evolutionary History: Amplifier, Run 11, Date: 20/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

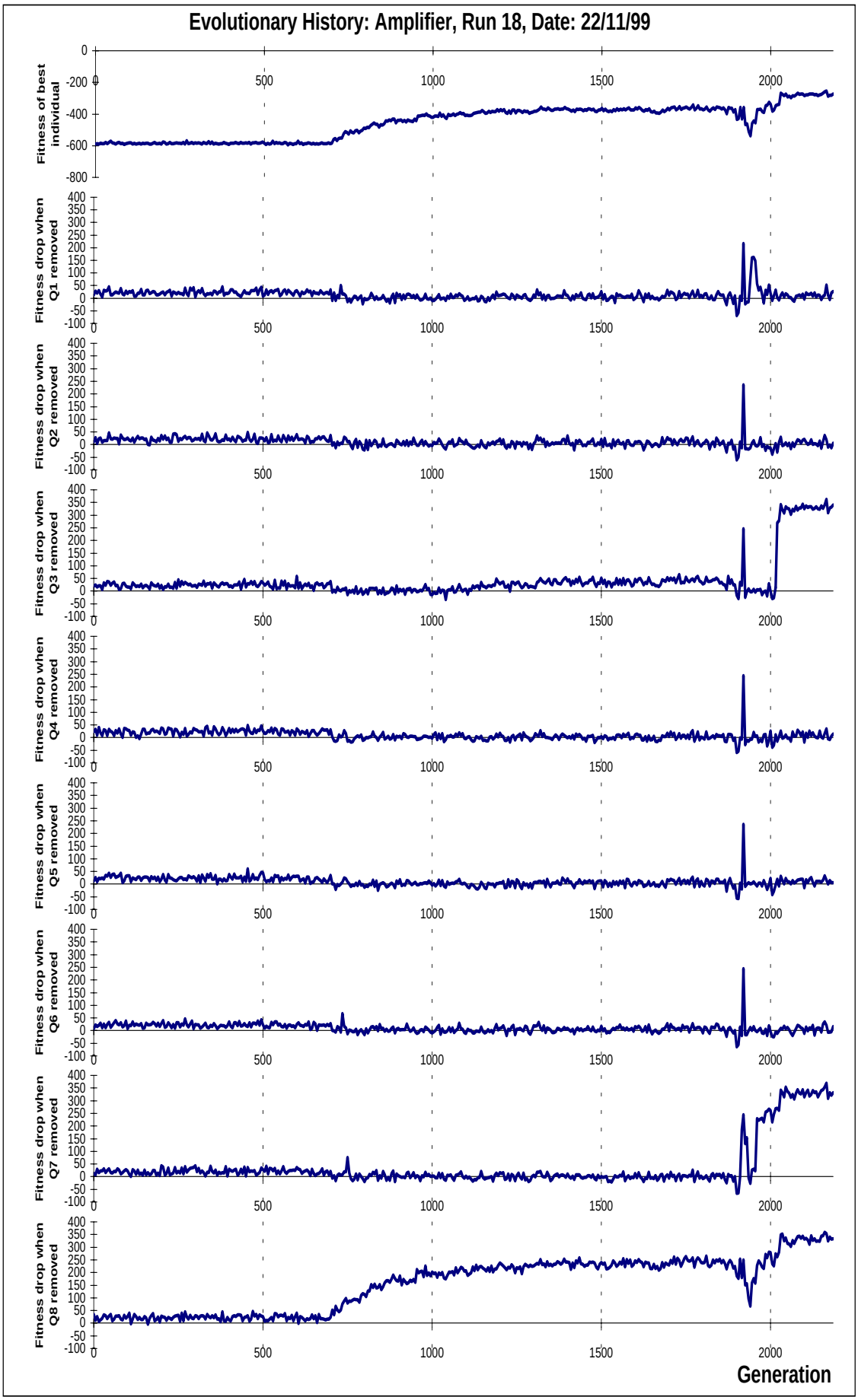

Evolutionary History: Amplifier, Run 13, Date: 21/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
500
1000
1500
2000
2500
3000
3550
Generation

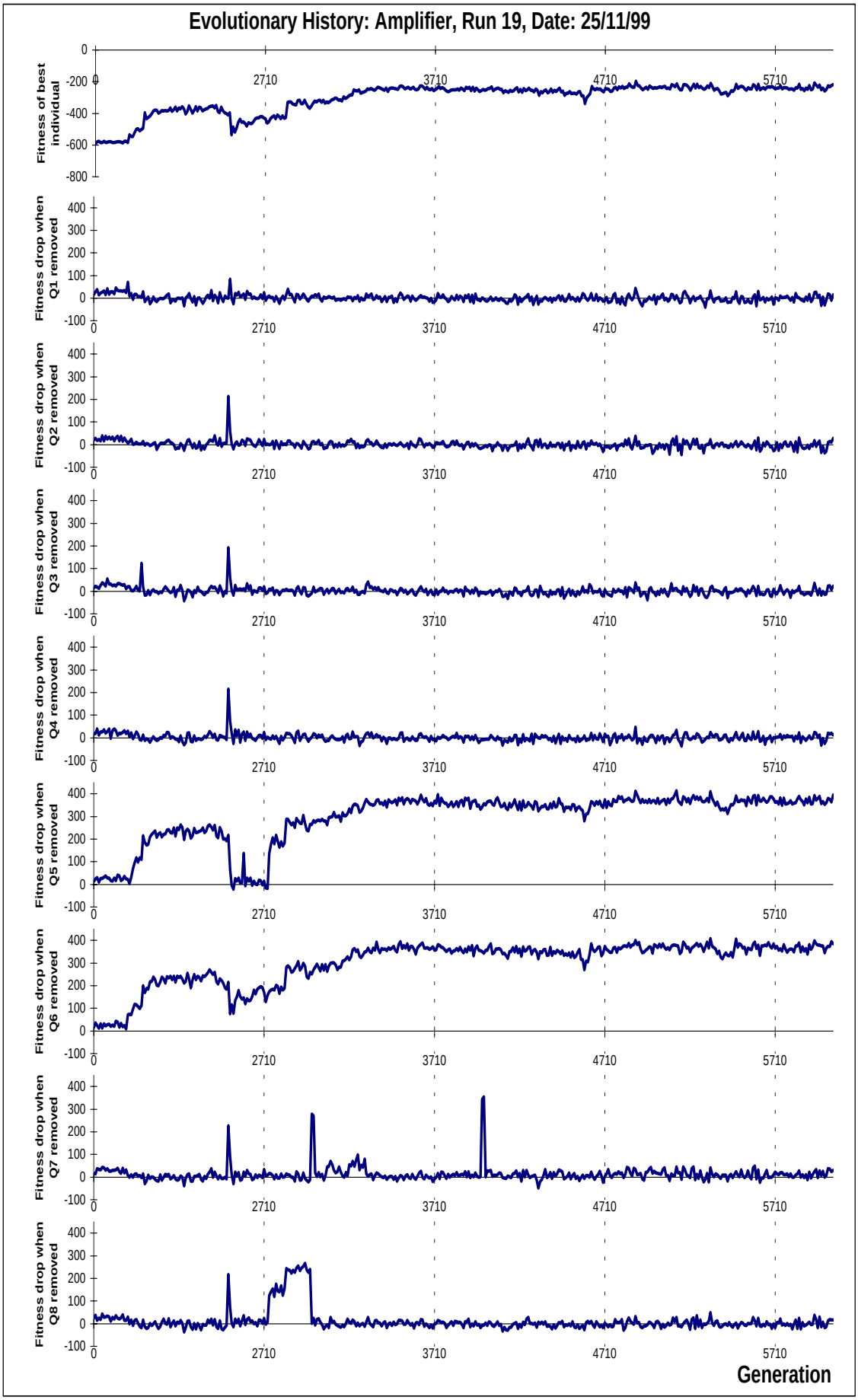

Evolutionary History: Amplifier, Run 14, Date: 21/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

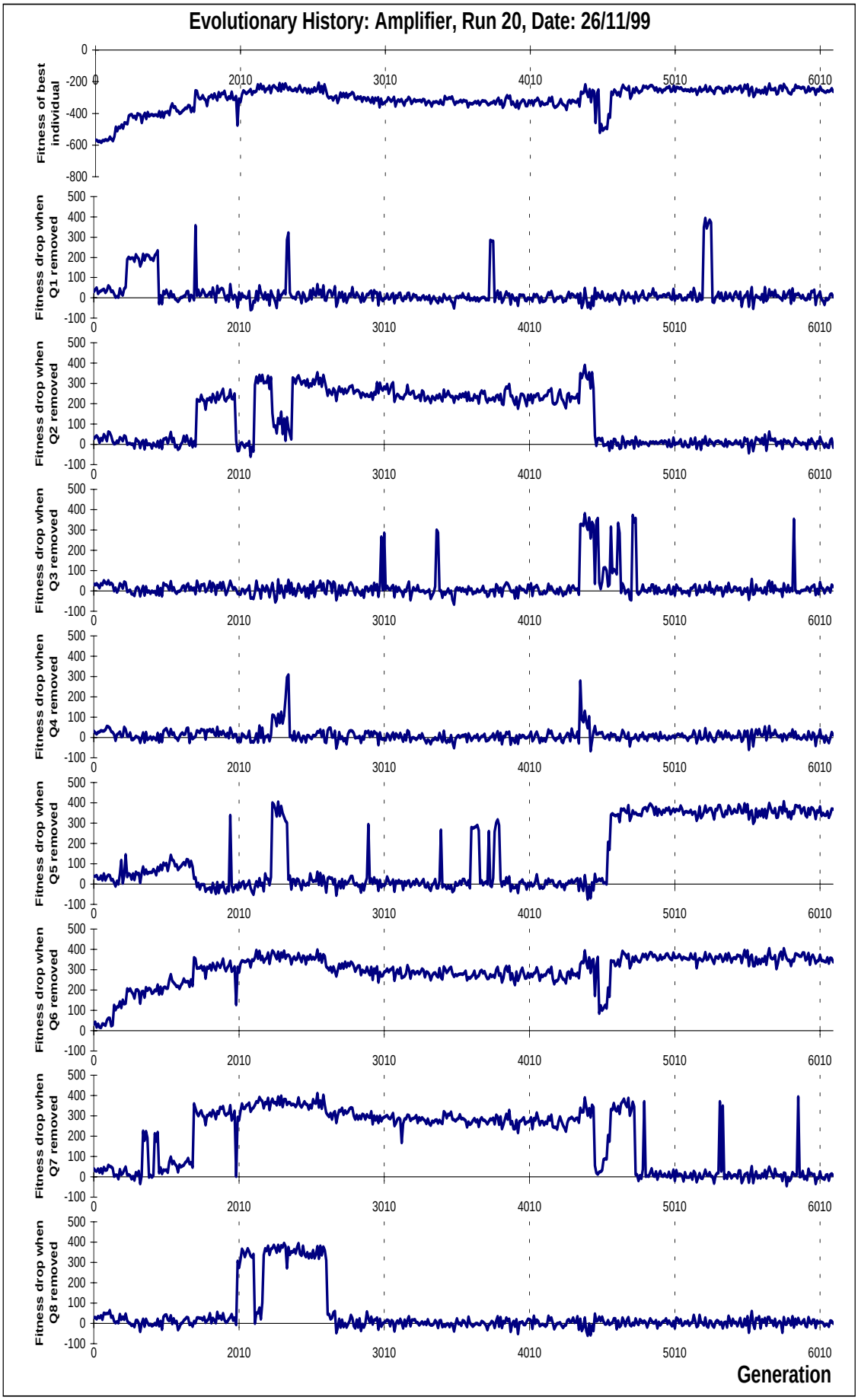

Evolutionary History: Amplifier, Run 15, Date: 21/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
-200
-400
-600
-800
450
350
250
150
50
-50
-150
0
500
1000
1500
Generation

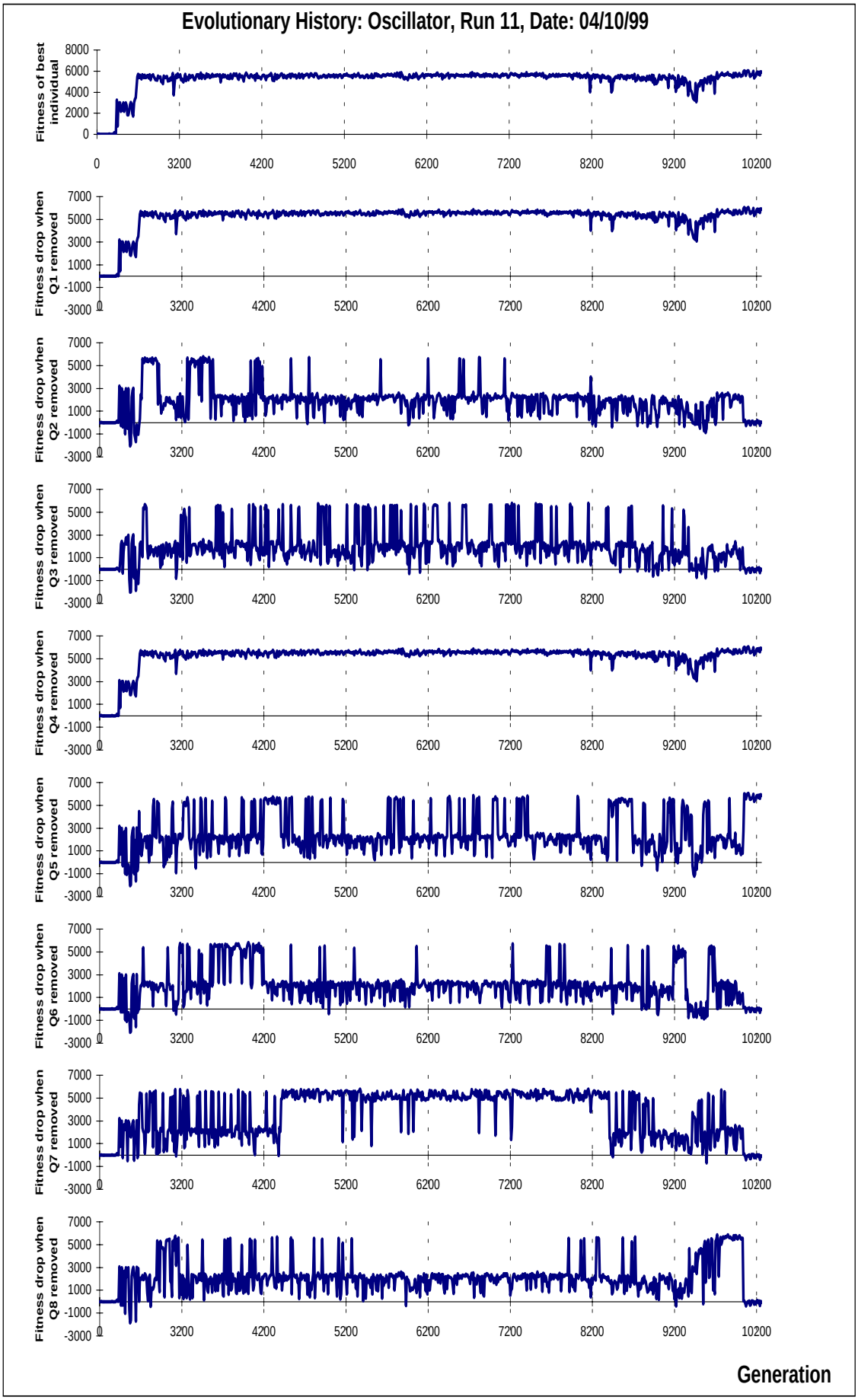

Evolutionary History: Amplifier, Run 16, Date: 21/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

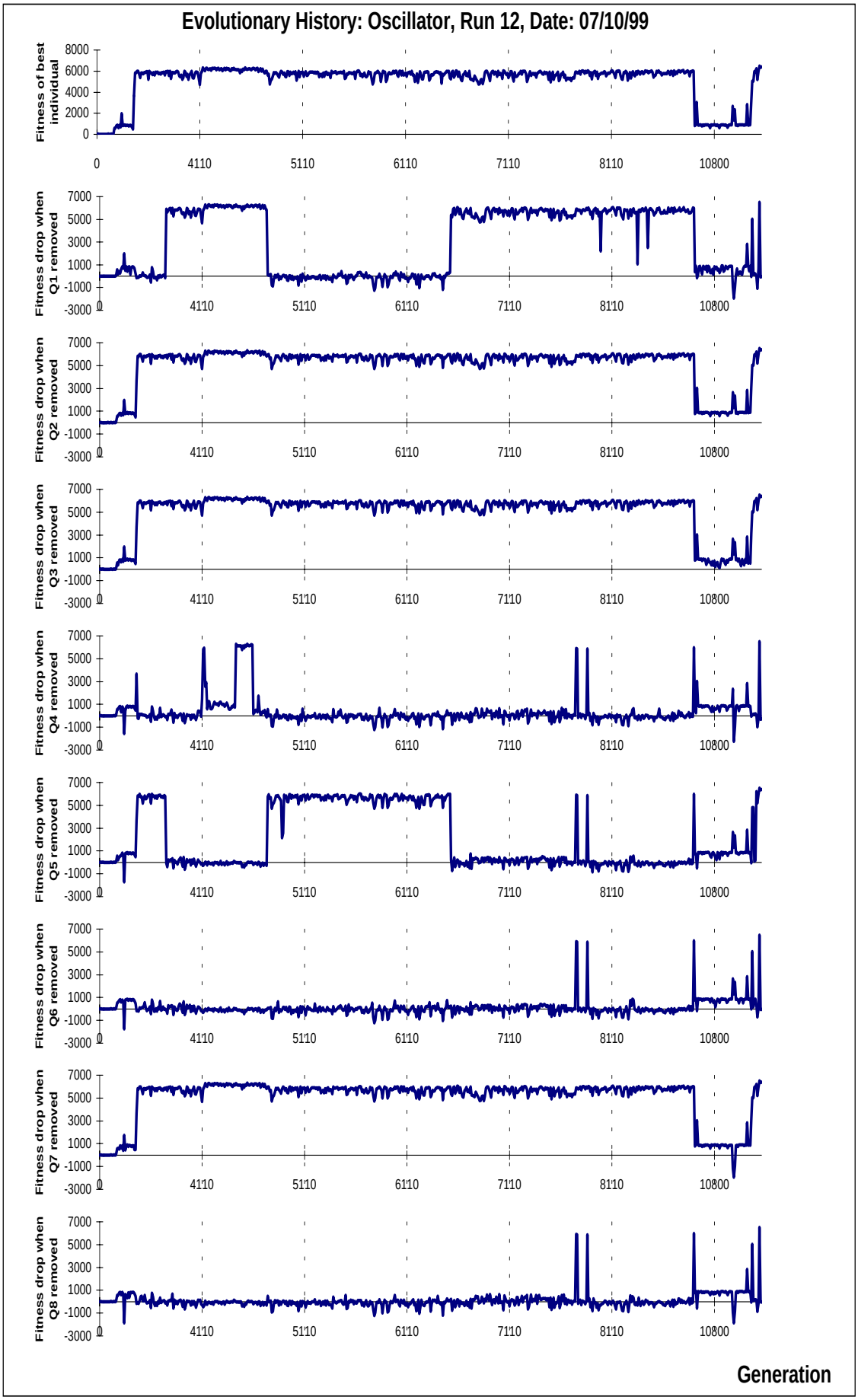

Evolutionary History: Amplifier, Run 17, Date: 22/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

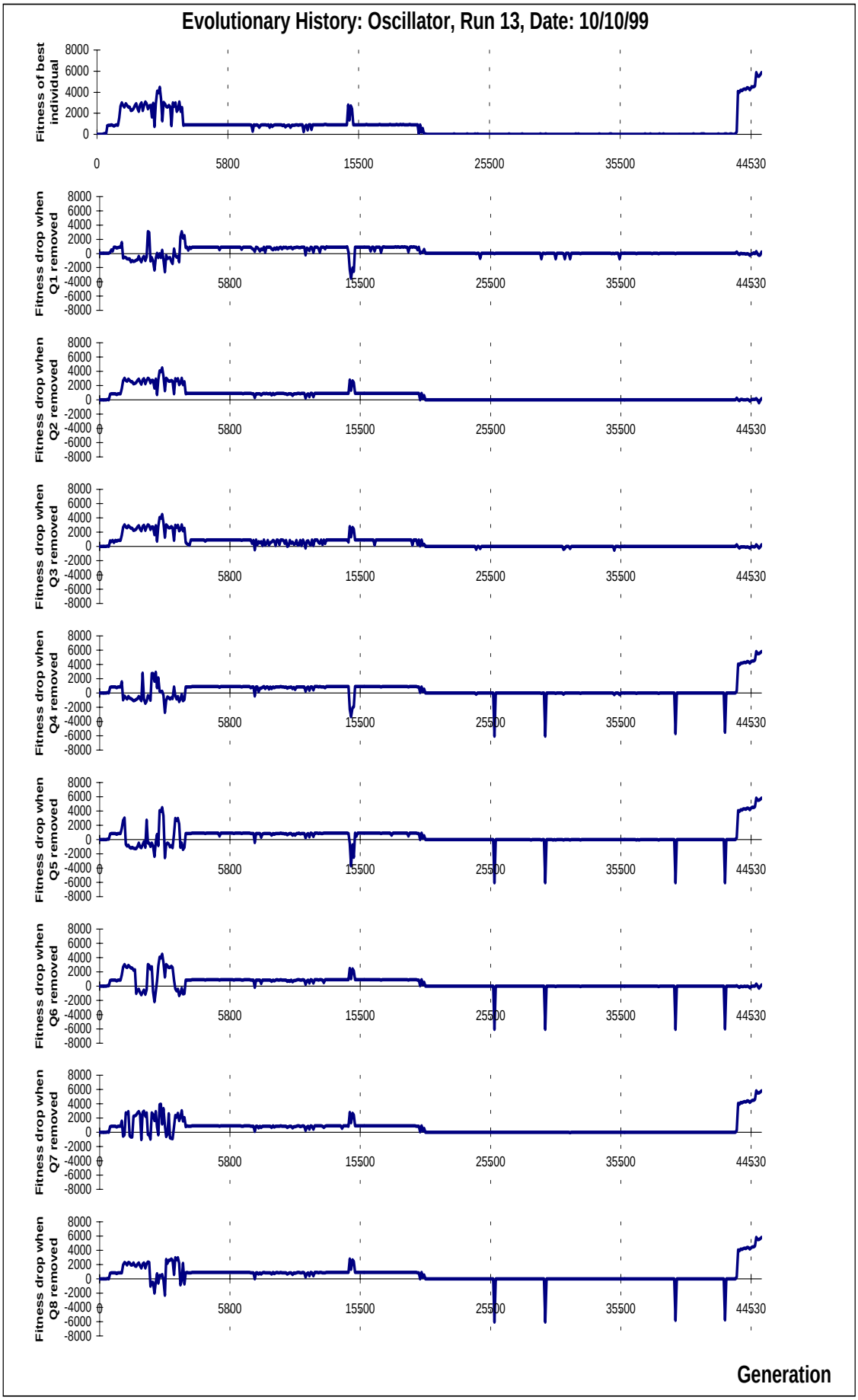

Evolutionary History: Amplifier, Run 18, Date: 22/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
Generation

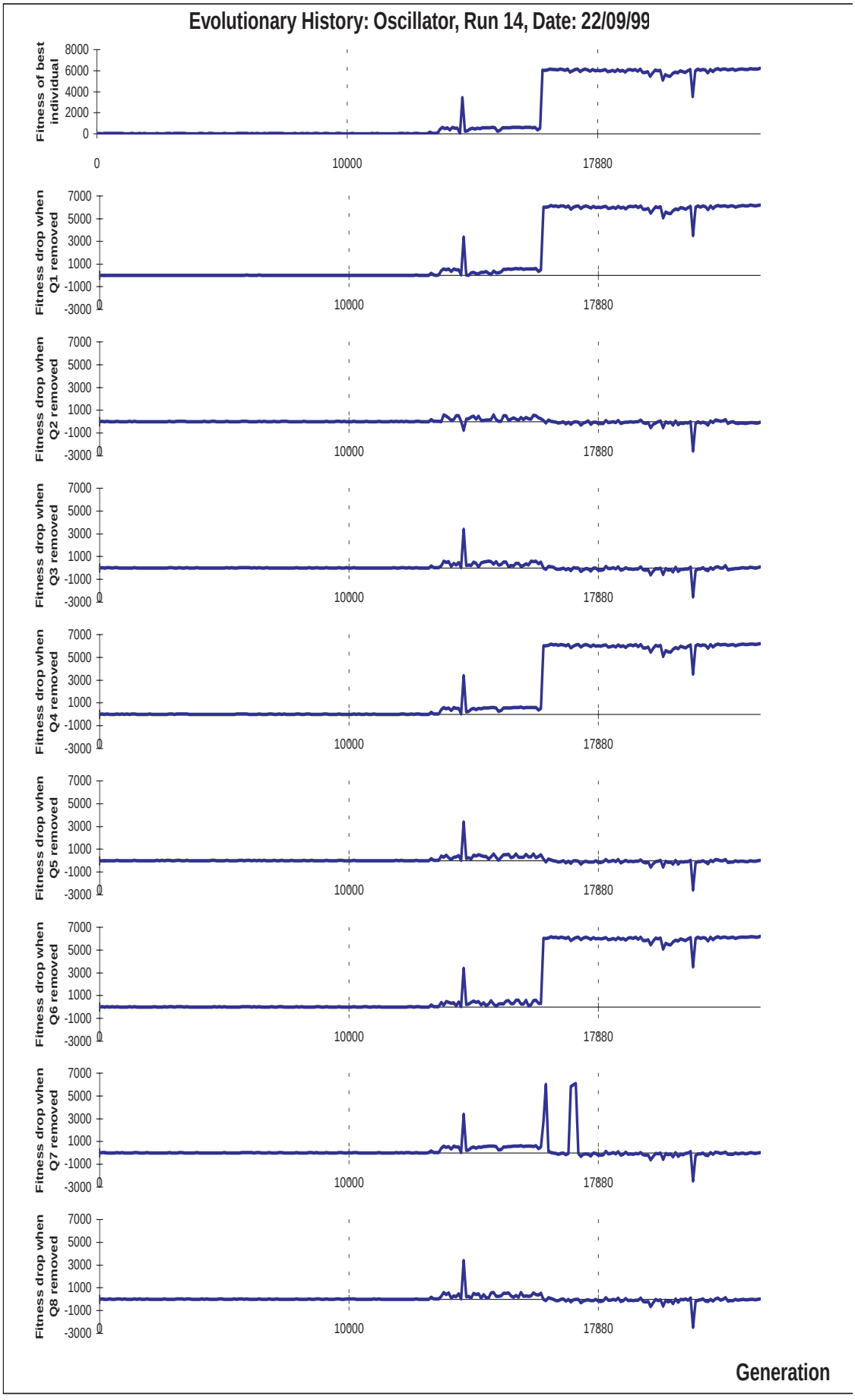

Evolutionary History: Amplifier, Run 19, Date: 25/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
2710
3710
4710
5710
Generation

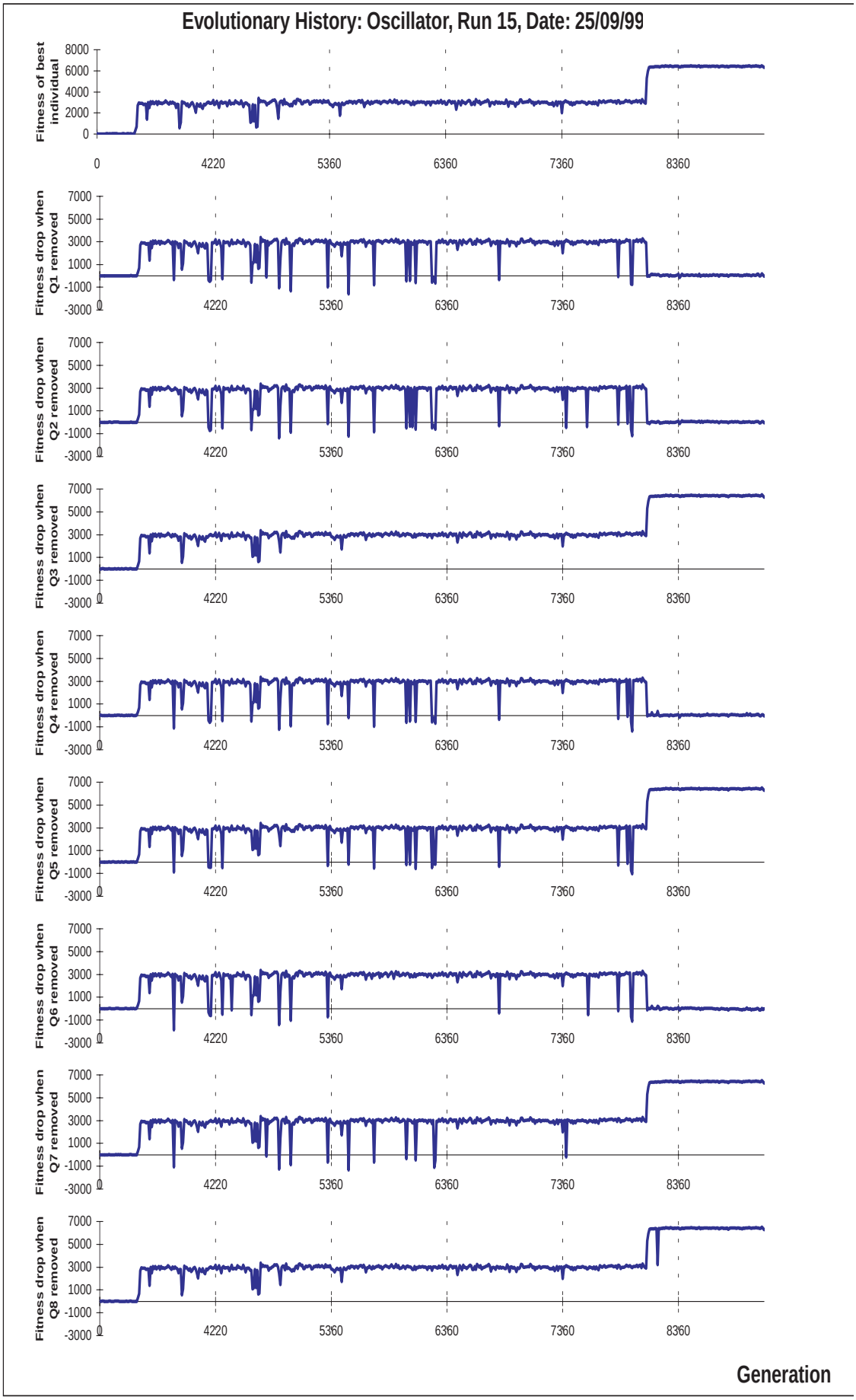

Evolutionary History: Amplifier, Run 20, Date: 26/11/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
2010
3010
4010
5010
6010
Generation


Evolutionary History: Oscillator, Run 11, Date: 04/10/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
3200
4200
5200
6200
7200
8200
9200
10200
Generation


Evolutionary History: Oscillator, Run 12, Date: 07/10/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
0
4110
5110
6110
7110
8110
10800
Generation


Evolutionary History: Oscillator, Run 13, Date: 10/10/99
Fitness of best individual
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
8000
6000
4000
2000
0
-2000
-4000
-6000
-8000
0
5800
15500
25500
35500
44530
Generation


Evolutionary History: Oscillator, Run 14, Date: 22/09/99
Fitness of best individual
8000
6000
4000
2000
0
0
10000
17880
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
7000
5000
3000
1000
-1000
-3000
Generation


Evolutionary History: Oscillator, Run 15, Date: 25/09/99
Fitness of best individual
8000
6000
4000
2000
0
Fitness drop when Q1 removed
Fitness drop when Q2 removed
Fitness drop when Q3 removed
Fitness drop when Q4 removed
Fitness drop when Q5 removed
Fitness drop when Q6 removed
Fitness drop when Q7 removed
Fitness drop when Q8 removed
7000
5000
3000
1000
-1000
-3000
0
4220
5360
6360
7360
8360
Generation