A Circuit Based Evolvable Hardware Architecture

by

Delano Christopher Sanchez

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author ....
Department of Electrical Engineering and Computer Science
May 26, 2006

Certified by ....
Karl K. Berggren
Assistant Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

A Circuit Based Evolvable Hardware Architecture

by

Delano Christopher Sanchez

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis presents an Evolvable Hardware Architecture that was developed in the Quantum Nanostructures and Nanofabrication Laboratory. We believe intrinsic evolution is a promising tool that can be used to exploit the physics of complex systems. I present a reconfigurable analog circuit platform that is coupled with a genetic algorithm to evolve circuit functions. The design process is detailed along with the results of three evolved circuits.

Our coarse grained analog system parallels other evolvable hardware platforms that have been developed using the same architecture. I place our platform in the context of other efforts in the field and our intentions for future work. The speed and complexity of our board is discussed with areas for future development outlined.

Thesis Supervisor: Karl K. Berggren
Title: Assistant Professor

Acknowledgments

I have had tremendous support and encouragement from my family and close friends who have inspired me to test unchartered waters and rise up to new challenges. I hope I can provide the same and offer my service to you. A special acknowledgement to the love of my life Candice for coming along on this journey keeping me in check, putting things in perspective and keeping me honest.

Marilyn Pierce has made the department a supportive environment to navigate turning mountains into mole hills and challenges into opportunities to succeed. I have been able to enjoy my research as a result.

The experience of this project is quite unforgettable and I am a better professional and individual for it. Karl Berggren has been an exceptional leader, motivator and mentor beyond what I could have hoped for. The positive research atmosphere and success of the QNN group have been established through his enthusiasm and respect for science, understanding of human nature and his desire to positively impact those around him. Since its genesis our group has grown in number and accomplishments. Thank you all for being good friends and colleagues in an excellent research environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This project is an exploration into a new realm of design methodology for complex reconfigurable physical systems and was funded by Lincoln Laboratories and the Quantum Nanostructures & Nanofabrication Laboratory. We set out to produce an electronic system which can be used to explore the area of Evolvable Hardware (EHW), where Evolutionary Algorithms (EA) are applied to reconfigurable physical systems to engineer solutions to problems. Research in our laboratory is focused on quantum systems and systems with nanometer length features. Engineers working in these areas can expect to face challenges as existing engineering techniques are challenged with ever shrinking systems requiring microscopic models that may be prohibitively complex or non-existent.

Evolutionary design techniques have had promising success as a viable approach even within already established areas such as electronics [8, 7, 20]. For our purposes it is most attractive in areas where tools based on human intelligence, knowledge and intuition are not easily applied. Traditional engineering design often relies on past knowledge about a system or some reasonable ability to predict behavior. When faced with systems which do not lend themselves easily to predicting behavior, challenging problems may arise requiring the engineering of novel solutions often times through trial and error. In the most difficult problems, the solution may be customized for the system and may not allow the formulation of design techniques to approach occurrences of the same problem. Using novel solutions on a trial and error basis is in itself an engineering skill that has been used for centuries. Evolvable systems seek to mimic this methodology by testing solutions, evaluating their performance and most importantly generating new solutions which capture useful design features from previous solutions.

Evolvable hardware is an emerging area of research still expanding to applications in new areas having been established with pioneering work in areas such as robotics and electronics [25, 27] . It uses the process of *intrinsic* evolution, the implementation of an evolutionary optimization algorithm evaluated in real time on a target hardware platform with the goal of achieving some desired function or behavior. *Extrinsic* evolution evaluates systems using models and simulators that require a sufficient amount of computing power to allow the convergence of the algorithm in a reasonable amount of time. Intrinsic and extrinsic evolution are the two branches of evolutionary design each with their own merits and challenges. For many systems there are limitations inherent in using models of expected behavior. Evaluating possible solutions *in situ* not only allows for faster evaluation of solutions in many cases, but also utilizes all the natural behavior of the system [26]. This exciting feature of evolvable hardware, the harnessing of all the reproducible phenomena within the system, predicates its appeal to complex systems with relatively limited models of behavior or that have complex configurations which prove difficult to analyze within a theoretical framework.

The early pioneers for evolutionary approaches gained inspiration from taking a look at its manifestation in the natural world and developed an optimization framework for mimicking the proven processes of natural selection and reproduction [26, 5]. This basic framework is still the basis of evolutionary approaches and represents a top down approach that still has some degree of unexplained success in its use. Though understanding of evolutionary methods is still under study, the success of early experiments and further research has established its acceptance as a viable optimization method for numerous classes of problems. Its suitability as a design tool in unexplored areas is still to be proven and it is our intention to provide evidence that work should be continued within these areas.

Our project closely follows that done by Ozsvald and Thompson at the University of Sussex . Ozsvald's thesis, Short-Circuiting the Design Process [17], demonstrated the use of a Genetic Algorithm (GA) to evolve circuits based on a simulator model and a four chip evaluation board of the Totally Reconfigurable Analog Circuit (TRAC) chip, both produced by Zetex. The TRAC chip provides most of the analog capabilities of the circuit with the addition of the large switching network and passive elements taking the complexity of the system significantly beyond that of the previous work. Taking this work further for evolution on a real complex circuit system would help us develop the knowledge and tools to explore evolvable hardware as a viable approach for the type of systems this architecture is ultimately geared toward [10, 15, 14, 19].

The next chapter introduces evolvable systems as three major components; an Evolutionary Algorithm, Reconfigurable Hardware and a Performance Evaluator(PE). Chapter 3 details the evolvable platform development process. The role each platform played in the overall development process is discussed with the associated goals, components, challenges and progress.

Chapter 4 presents the results of hardware evolution on our platforms. We present and discuss the evolutionary results for three evolved circuits.

Chapter 5 summarizes our work and places it in a broader context of other developments in the field. I discuss my conclusions and outline future work that can build upon what we have done.

Chapter 2

Evolvable Systems

Figure 2-1 shows the functional representation of an Evolvable Hardware system. We have developed our architecture within this framework and discuss the role of the major components here.

2.1 Genetic Algorithm

A Genetic Algorithm(GA) mimics nature's evolutionary process of generating solutions to some problem with measurable performance. A population of candidate solutions is evaluated to measure their fitness according to some objective function. These scores are used to select individuals for the generation of an offspring population using mechanisms such as natural selection, mutation and mating[9, 4].

2.1.1 Solution representation

Individual solutions are represented as chromosomes with their characteristic configuration encoded in the values of the chromosome genes. The chromosome is effectively a string of gene values each of which encodes some information about the candidate configuration. The chromosome representation, genotype, is translated to a unique candidate configuration, phenotype, to which test cases can be applied to produce outputs with measurable performance.

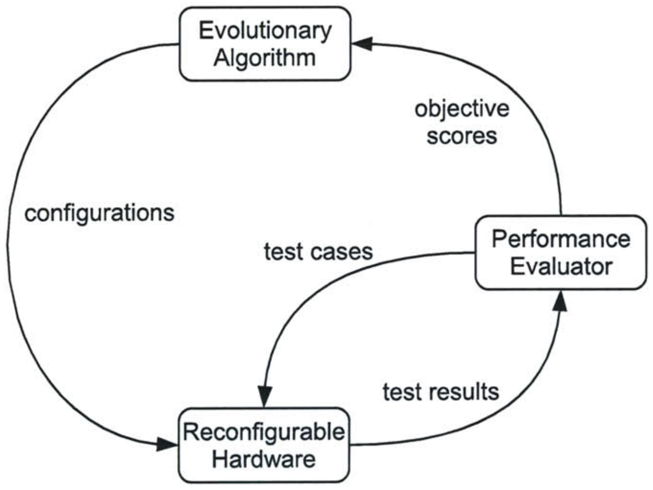

Figure 2-1: Evolvable Hardware System.
The Evolutionary Algorithm generates a population of individuals each representing a hardware configuration. These hardware configurations are sent to the Reconfigurable Hardware to evaluate their performance for some desired behavior. A Performance Evaluator uses test cases and objective function(s) to score the performance of configurations based on the test results. The EA uses the scores and configurations of a population to generate new individuals and the process is repeated with the expectation that performance progressively improves.

Figure 2-2: A gene with chromosome position i and integer value A_i from a set of 8 possible numbers

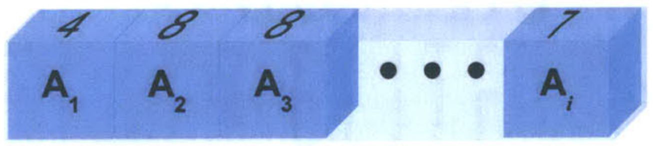

Figure 2-3: A chromosome with integer value genes. Each gene has a unique position in the chromosome with this ordering of genes common to all individuals.

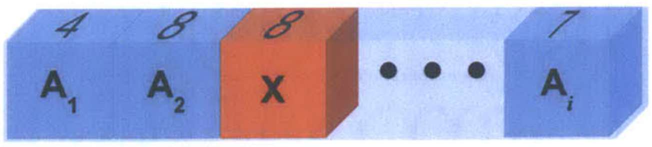

Figure 2-4: One gene mutated from a value of A_3 to a new value of X. The new value X comes from the same set of possible values of this particular gene which in this case is from the 8 possible integer values.

2.1.2 Regeneration

New individuals are created by undergoing some combination/sequence of two main operators and/or retention of some of the previous generations' individuals. The two operators are mutation and crossover.

Mutation

This operator randomly changes gene values in the individuals chromosome to some new value contained within the scope of the gene. The probability of changing any particular gene is used to set the probability of mutation. Higher values will produce an individual that looks likely a completely random individual with little resemblance to the parent, while a lower rate produces some variant of the parent chromosome with gene values that may still be unique from the parent. It is intended to introduce genetic diversity to the population and allow exploration into new areas of the solution space.

Crossover

New individuals are formed by mating individuals in the current population. The new individual inherits gene values from the parent chromosomes. The favorable traits of the parents are hoped to be passed on to the new individuals producing a better individual.

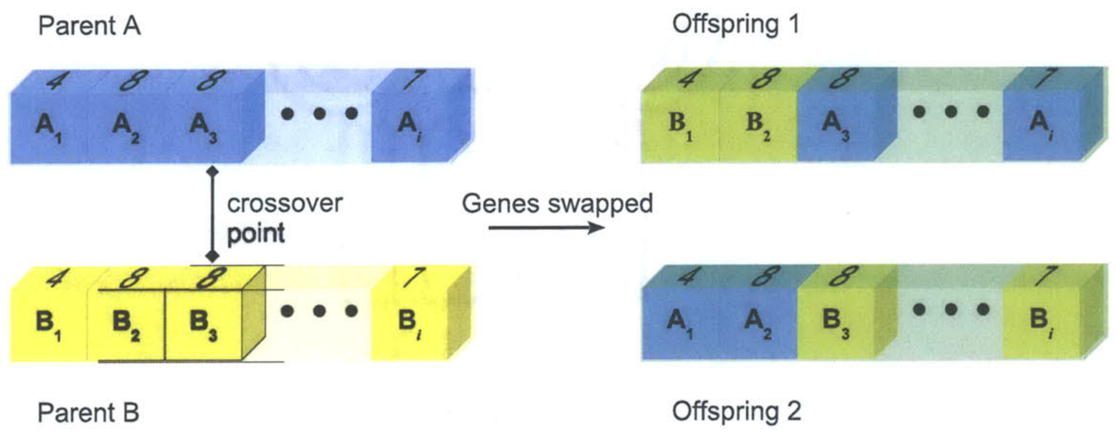

Figure 2-5: Single-point crossover of Parent A & Parent B to produce Offspring 1 & 2. Each gene in the offspring inherits its value from the corresponding gene in either Parent A or B. In single-point crossover one particular gene position is identified as the crossover point in the parent to split the chromosomes of the two parents. These split chromosomes are then swapped and recombined to form the new offspring.

2.1.3 Evolutionary Process

An initial population of individuals with pre-assigned and/or random gene values serves as the initial gene pool. Individuals are translated to their phenotype configuration and their performance evaluated by applying test cases and measuring their adherence to expected output performance. These fitness measurements are used to select parent to generate offspring populations. New individuals are created by mutation and crossover of parent individuals while the most fit individuals may be retained by natural selection rules. The cycle of evaluation and regeneration continues for some iteration upper limit or until some performance threshold is met. The mutation rate, proportion of mutation, crossover and retention can evolve over the duration of the evolutionary cycle based on performance and generation number. Figure 2-6 represents the evolutionary process of the GA. The major blocks for the GA are:

1. Initialization
2. Evaluation
3. Termination Check

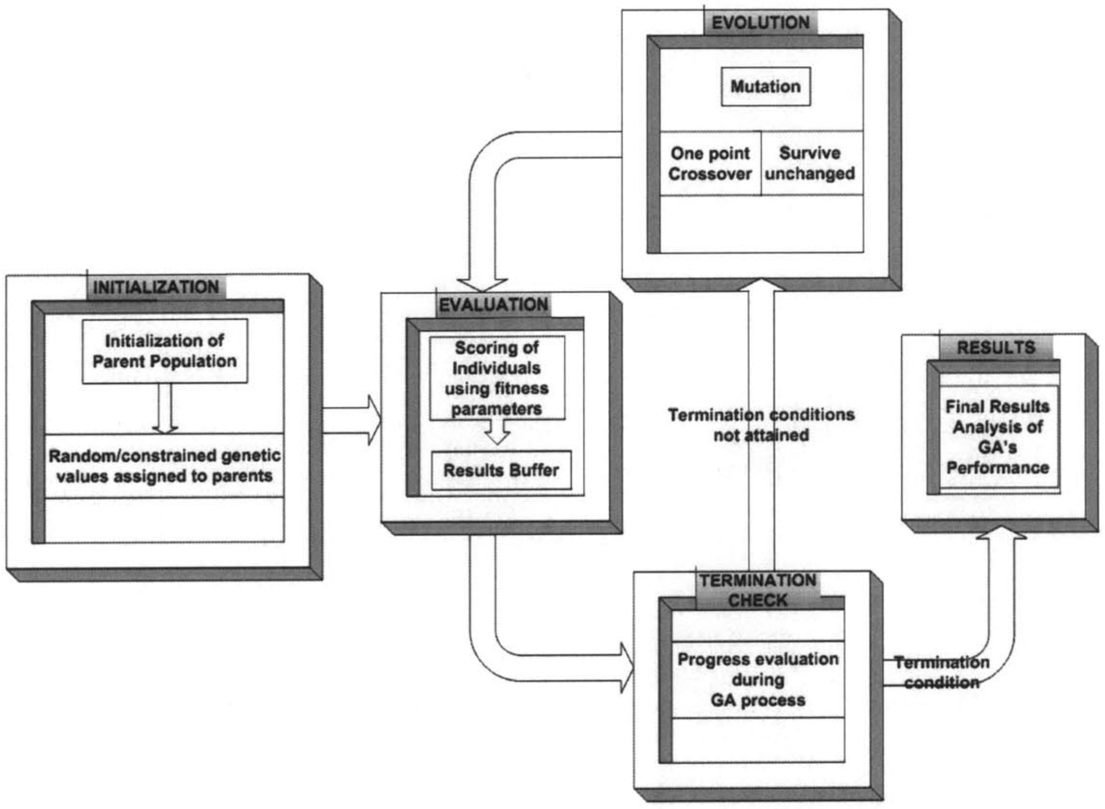
EVOLUTION
Mutation
One point Crossover
Survive unchanged
INITIALIZATION
Initialization of Parent Population
Random/constrained genetic values assigned to parents
EVALUATION
Scoring of Individuals using fitness parameters
Results Buffer
Termination conditions not attained
RESULTS
Final Results Analysis of GA's Performance
TERMINATION CHECK
Progress evaluation during GA process
Termination condition

Figure 2-6: Major Components of a Genetic Algorithm. Initialization generates the first population of candidate solutions. Evaluation uses the scores from the PE's objective function to rank the fitness of individuals. Evolution generates new individuals by applying mutation, crossover and natural selection to the parent population. The evolutionary loop is terminated when performance or process thresholds are exceeded. Results record data accumulated during the process when the evolutionary process is finished

4. Evolution

5. Progress & Results

2.2 Evolvable Hardware

Evolutionary methods can operate on simulated or modeled systems, intrinsic evolution, or on real time systems, extrinsic evolution. Evolvable Hardware falls into the category of extrinsic evolution. The hardware system is reconfigurable and loads each individual's configuration. Performance measures are based on physical measurements of each individual's output response to test cases. The evolutionary algorithm can exploit the measured physical characteristics of the hardware which may not be accurately represented in simulators and models[26]. This can lead to a more efficient use of the resources of the system as one does not rely on models of behavior which are inherently approximations to what actually exist. The higher the degree which characteristics peculiar to the target hardware system are exploited the less likely the evolved solution will produce the same performance on an identically designed hardware system. The tradeoff between resource exploitation and robustness can be weighted by the choice of test cases and performance measures and will be dependent on the intended use of the evolved solution. The efficiency of evolving in situ is dependent on the speed with which individual configurations can be loaded on the hardware and subsequently evaluated. Evolvable hardware eliminates the need for time intensive computations inherent in intrinsic evolution. It can provide performance gains in convergence time if its reconfiguration and evaluation time is faster than what pertains in simulation using available computing resources where memory and processing power is finite. The challenges faced are different from extrinsic evolution. Irreproducible output behavior such as transients can work against the evolutionary process with fitness scores being distorted by behavior that cannot be replicated. Evolving configurations that harness a feature or characteristic that is unique to the host hardware may also prove to be a problem if the configuration is meant to be used in other similar systems. The success of intrinsic evolution will also be tied to the accuracy of the measured output behavior which in some systems can conceivably be worse than simulators or models. Many systems may not lend themselves easily to measurement as with quantum systems where measurement can alter the state of the system.

2.3 Performance Evaluator

To evolve toward some desired behavior there must be some pressure to do so. In real life we can look at the evolution of biological features such as teeth in a particular predator as an example. The pressure to catch prey in order to survive will favor the survival of individuals with teeth better suited to catch their type of prey. The form of the teeth in this example would have some measure of suitability based on the type of prey the predator seeks. In the case of evolvable hardware we have some system behavior to satisfy with many possible configurations available. To discriminate between these configurations we apply a set of test cases and evaluate the response of each configuration based on appropriate criteria. The criteria, measured by an objective function, would apply performance pressure by favorably scoring configurations that are likely to lead to better solutions. The choice of test cases and objective function plays a major role in evolving good configurations and may be adapted to increase the efficiency and success of evolution.

Chapter 3

Evolvable Platforms: EHW and GA Development

This chapter introduces the design platforms used and the work done to develop our EHW system. We were motivated to begin our work in the area of Analog Circuits based on the previous work in the field that has been producing promising results. Standard electronic circuits are static in nature and are not reconfigurable. We have decided to use a Field Programmable Analog Array (FPAA) as the major electronic component in our evolvable system. Some approaches in the field have used complex switching systems to reconfigure the topology and use of static electronic components but the vast majority of work has utilized reconfigurable hardware such as FPTAs, FPGAs and FPAAs [21]. Evolving circuits on the transistor level using FPTAs can be a very time consuming process as the search space for this fine grain element is typically quite large. Our FPAA is a more coarse grained device with functional analog blocks.

3.1 Simple software based curve fitting

To form an understanding of GAs I decided to write code from scratch for a simple curve fitting problem. This provided a better understanding of the concepts and a chance to delve into the mechanisms used by the algorithm. No third party GA toolboxes or functions were used as I developed the code as a learning experience. The GA would evolve the parameters of a polynomial to fit a target curve. The chromosome used for the problem was a grey coded binary string representing pairs of signed real numbers. Each pair was the coefficient and power for a member of the polynomial. Developing a simple extrinsic evolutionary system as well as all the functions provided a good first understanding of the capabilities, constraints and parameters that can be adjusted during an evolutionary run. The concept of genotype and phenotype became clearer and even with this simple problem some gauge of the processing time for our particular PC would be helpful in forming benchmarks for the number of generations that could be evolved, the size of populations and chromosome length. What was not incorporated in this GA was the ability to monitor progress after each generation. This would be developed for subsequent problems to provide more information on the evolutionary run.

3.2 TRAC Evaluation board

After forming some insight about the GA from the previous exercise with curve fitting we wanted to get started by working with a hardware system. Using the TRAC evaluation board as a first step was an obvious choice as we had no previous experience with the device and hoped to gain information quickly from the evaluation board which was operational straight out of the box.

3.2.1 Goals

This was the first attempt at an intrinsic evolutionary system with hardware in the loop. My goal was to get the system running and take notes for areas of improvement or changes that could be made. Developing code to control hardware resources for all subsequent work was a key task.

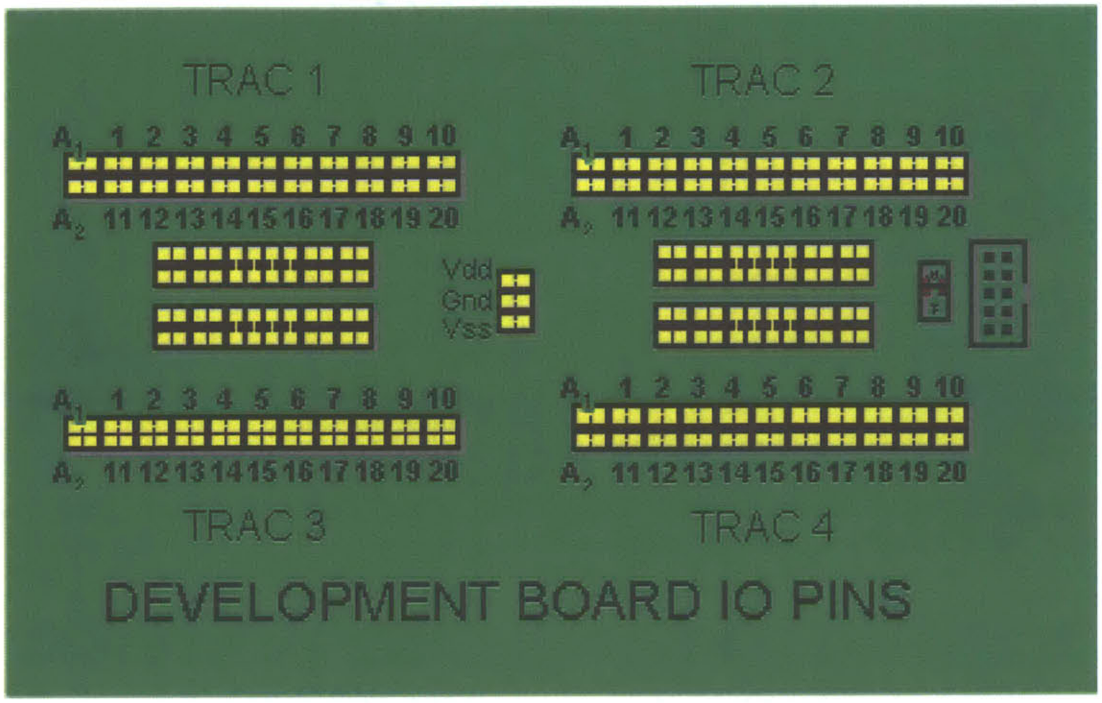
TRAC 1
TRAC 2
A_1 1 2 3 4 5 6 7 8 9 10
A_2 11 12 13 14 15 16 17 18 19 20
Vdd
Gnd
Vss
TRAC 3
TRAC 4
DEVELOPMENT BOARD IO PINS

Figure 3-1: TRAC Evaluation Board
*source:*Analog Devices[29]

3.2.2 EHW Design

TRAC Evaluation Board

Hardware

The TRAC evaluation board was inherited from previous work on this project at Lincoln Labs. It consists of 4 TRAC chips with connections to interface with the IO pins on each TRAC (Figure 3-1). A power adapter provided voltage supplies for VDD 3V and VSS -2V. The chips were programmed by the accompanying software via a parallel port connection.

Software

A Graphical User Interface (GUI) allowed a drag and drop design environment to set the functions of each TRAC cell. The simulator allowed analysis of predicted behavior once parameters such as gain were set. The actual gain measured on the evaluation board depended on the values of the passive elements connected to the IO pins of the TRAC. Various designs were loaded onto the TRAC evaluation board and matched the simulated results.

AD8113 Evaluation Board

Hardware

A cross-point switch was intended to form a switching network to extend the functionality of the TRAC. Cross-connection between TRAC chips, distant pins on the same chip and connections with passive elements would be possible. An evaluation board for the AD8113 was acquired for initial testing of the proposed idea. The switch pins on the chip led to BNC connectors for interfacing to external circuitry. The switch was effectively a demultiplexor with buffered outputs and gain of 2. This meant it was an active switch which would prove unsuitable for creating network connections and would later be replaced by a passive switch.

Software

The evaluation board could be programmed via the serial port or control pins on the board. Software with a GUI configured the switch through the serial port. In addition to this I was able to program the AD8113 using MATLAB by directly addressing the parallel port. This would be the programming method used for a simple input-output matching problem using only the AD8113 evaluation board.

Both the TRAC and AD8113 evaluation boards used parallel port interfaces with their GUI based software to program the devices. The software did not lend itself easily to integration with the GA and I set out to produce my own code to control the evaluation boards. Having only one parallel port on the PC also meant finding an alternative means to send digital signals to the devices. For this purpose, I obtained a DAQboard 2001 Data Acquisition (DAQ) board with digital IO along with analog input and output capabilities. Using the DAQ board through Matlab the AD8113 was reconfigured and controlled. The Matlab Instrument Control Toolbox would be used to control the Agilent 33220A function generator with the Data Acquisition Toolbox used to interface with the HP 54112D oscilloscope. A considerable amount of effort was spent attempting to control the TRAC evaluation board with Matlab or C++ but this proved futile and led to a cross roads in the project. Alternative Field Programmable Analog Arrays (FPAA) were researched with the Anadigm AN220 & AN221 being considered [23, 24]. The basic building blocks of these FPAAs were at a higher level of abstraction than the TRAC's basic functions and were not pursued further.We made a decision to develop a custom board to program the TRAC. While little to no information was available on programming the TRAC evaluation board, there was sufficient information in the data sheets on programming the TRAC chips allowing us to successfully develop our custom board later on in the project.

3.2.3 GA Development

Most of the design for this part of the project focused on software development and device control. A GA toolbox developed at the University of Sheffield [1] would be used to provide the low level functions for the GA. Each of the 5 blocks in Figure 2-6 were developed using this GA Toolbox. Both binary and integer representations were alternatively used for the chromosome. Subsequent work uses integer representations. The GA was also now able to track the objective scores of the best individuals during an evolutionary run and record the GA parameters and results.

3.2.4 Progress

The architecture developed at this stage consisted of two evaluation boards provided by the manufacturers. I was able to get an evolvable system functional with code written in Matlab using the AD8113 to demonstrate a simple input-output matching problem on the switch matrix. The AD8113 was programmed using the DAQ board IO to interface with the control lines. I used code developed in Matlab to control the DAQ board removing any dependence on vendor supplied software to interface with the reconfigurable hardware. The software developed for the GA and the instrument control would be used for later generations of the evolvable system. It was the first evolvable system we developed with hardware in the loop. Code to capture waveforms and feed the data into the PE was completed using the Instrumental Control and Data Acquisition Toolboxes in Matlab. Attempts to control the TRAC evaluation board with Matlab or C++ proved futile and an alternative approach was attempted. A customized PCB was the proposed solution to interface with the TRAC chips using the DAQ board digital Input/Output (I/O) lines to stream data to the chip registers.

3.3 First Generation TRAC based evolvable board

The difficulty we faced using the TRAC evaluation board which was ill suited for our purposes forced us to design our own custom board. This would lift a burden off my shoulders after many frustrating months trying to reverse engineer a system created for a different purpose with little to no information or support available. This first custom board would be an experiment to get a TRAC operational with our own software which would allow us to finally integrate the TRAC into a functioning evolvable system.

3.3.1 Goals

Designing a custom PC board for the TRAC would allow us new degrees of freedom. We would be able to program the TRAC chip without the restrictions imposed by the manufacturer supplied software. We would have the freedom to design a board with TRACs and other components in a topology that is suited to an evolvable system. The TRAC data sheet provided sufficient information about the control pins and timing diagrams to develop control software that would program the TRAC through the DAQ I/O lines. The custom board would have to meet the signaling requirements of the TRAC as well as allow use of its analog functions. Verifying the streaming of digital data to the TRAC registers along with testing the analog output behavior from the loaded configurations would indicate the TRACs are successfully programmed and operational. The GA would also be develop further to evolve a circuit on this first

Figure 3-2: TRAC Cell Functions
source: Analog Devices[29]

generation board.

3.3.2 TRAC chip

The TRAC is essentially an IC with a few circuit components such as Operational Amplifiers (Op-Amps) and diodes arranged to allow 8 mathematical functions (Figure 3-2). They are:

1. ADD
2. AUXILIARY
3. ANTI-LOG
4. LOG
5. NEGATE
6. NON-INVERTING PASS
7. OFF
8. RECTIFY

The TRAC loads one of these circuits into Configurable Analog Blocks (CABs) or cells. Each TRAC has 20 cells arranged in two rows of 10 with I/O pins before and after each cell. Neighboring cells in the same row are connected to each other.Figure 3-3 shows the GUI representation of a TRAC.

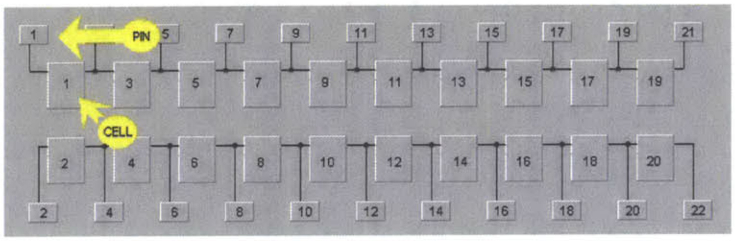

Figure 3-3: TRAC Graphical User Interface
source: Analog Devices[29]

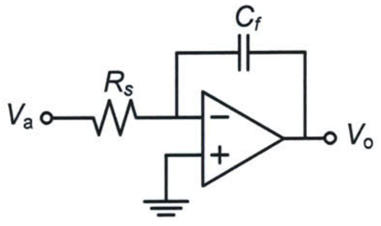

Figure 3-4: Implementation of the Integration function

The AUXILIARY function allows amplification, differentiation and integration with the circuit implemented using Op-Amps and external passive elements in a feedback configuration. Figure 3-4 shows the TRAC implementation of the integration function using an external capacitor and resistor where

$$V_o = -\frac{1}{RC}\int V_a \tag{3.1}$$

With these 8 elemental functions quite a large number of circuits can be synthesized as envisioned by the creators of the TRAC [29]. The ability to manipulate these elemental building blocks with a large scope of functionality makes the TRAC an attractive centerpiece for our evolvable hardware system. For our purposes it is sufficiently robust, easy to operate and has a fast reconfiguration time. The large number of I/O pins is also an attractive feature allowing interfacing to external devices and signals. The TRAC has a documented temperature dependence, particularly for the LOG and ANTI-LOG functions [28, 17]. While this may prove problematic for most circuit designers it is welcomed here as an extra degree of freedom to exploit device physics in a system that does not rely on models but measured behavior.

3.3.3 EHW Design

Eagle PCB was chosen as the layout tool to be used due to its ease of use and large device library database available. The TRAC is not a mainstream Integrated Circuit component and its package and schematic footprints had to be designed using the layout tool. The DAQ I/O sets *logic high* as $5V$ and *logic low* as $0V$. The power supply pins for the TRACs on the evaluation board are operated at $+3V$ and $-2V$. This supply range failed to allow programming of the TRAC due to the mismatch with the DAQ IO static discipline. The power supply pins on the TRAC were adjusted to $5V$ and $0V$ allowing the TRAC to successfully stream digital bits through its registers. The analog functions did not appear to work until the TRAC power supply was switched back to $+3V$ and $-2V$. The AGND pin, which was wired to $0V$, needs to be closer to the middle range of the two supply rails for proper analog functioning without clipping of input signals. To program and operate the TRAC the software would stream bits with the TRAC rails at $5V$ and $0V$, then switch back to $+3V$ and $-2V$ to effectively operate the TRAC. While this continuous switching method was sufficient to realize the task of programming and operating the chips with our own software, it would introduce undesirable effects such as transients and additional delay between evaluating circuits configurations and was noted as an area to be worked on.

3.3.4 GA Development

Each of the twenty cells of the TRAC was defined as 3-bit integer genes in the chromosome for the 8 possible functions of each cell.

3.3.5 Progress

The TRAC was entered into the design library with most of the other components already found in existing libraries. An error was made connecting a data output pin on the TRAC to a digital output line on the DAQ board. This was corrected by physically severing the connection between the TRAC pin and the connector to the DAQ board. The error interfered with the programming of the TRAC with the output pin and the signal line asserting two different voltages. This error was not properly corrected in the layout design and requires steps to sever the connection. The PCB used two signal layers with the board having a low density of signal traces. After all components were soldered onto the board the software developed for controlling the TRAC was tested and modified. Switching the values on the power rails enabled us to program and operate the TRAC but this method would have to be reviewed for future board generations. I was able to successfully program and test circuit configurations from the TRAC manual with the analog output matching expected results. This was a critical juncture in the project with the skies metaphorically opening as we were finally able to tap into the the power of the TRAC. The circuit we tried to evolve was a frequency doubling problem. A design already existed in the TRAC manual using only resistors as external components. This board did not have the ability to reconfigure external components so these were hard wired into the board based on the components used in the pre-existing design. The GA would reconfigure the TRAC with at least one known configuration providing a solution. The large number of possible configurations 8^{20} would gave the GA a large search space to evolve a suitable circuit which was different from the pre-existing solution. The system was able to evolve a circuit which took a sinusoidal input and produced a similarly formed sinusoidal output with twice the frequency. The degree of distortion was not significant.

3.4 Second Generation TRAC Evolvable System

We were able to operate the TRAC and incorporate it into a simple evolvable system using the first generation board. We were satisfied at the progress we made having evolved a circuit which is later described in Section 4.1. We were not able to utilize the full functionality of the TRAC in our previous evolvable system without reconfigurable control of the external passive elements. We began work on a second custom board with reconfigurable control over external passive elements.

3.4.1 Goals

To realize the full capabilities of the TRAC in an evolvable system, reconfigurable passive elements would have to be incorporated into the hardware. The previous generation could at most use passive elements hard wired into the system which limited the functional capabilities of the circuit. The new passive network would have to be designed into the new evolvable system and tested for operability on its own. It would also have to be tested to confirm the expected behavior when used with the TRAC, something that could not be achieved with the system in 3.2. The new components would also have to be incorporated into the GA for chromosome definition and subsequent circuit configuration. Evolving a circuit would have been the last step before proceeding further but no circuit was successfully evolved at this stage before the project shifted to the design of what would be the final evolvable hardware board.

3.4.2 Passive Network

A digital potentiometer, capacitor array and passive switch were the new elements used to incorporate a reconfigurable network of passive components. The AD738/9 is a CMOS switch matrix which acts as a passive multiplexor / demultiplexor effectively forming a short circuit between connected pins. Figure 3-5 shows the functional block diagram of the device. This switch would be used to form a switched capacitor array and network switch for connecting the potentiometer and capacitors to the TRAC IO pins.

Switched Capacitor Array

Figure 3-6 shows the representation of the switched capacitor array while Figure shows one on the board layout. Seven capacitors were used for each array with the range of values in Table 3.1. Their parallel connection gives an equivalent capacitance

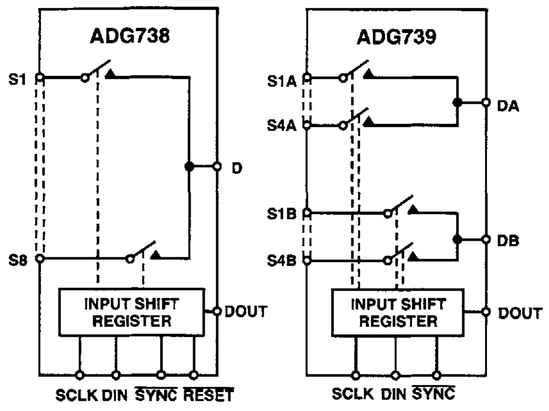

Figure 3-5: ADG738/739 Multiplexor Switch
source: Analog Devices[3]

Index, i	Capacitance (μF)
1	1
2	4.7
3	10
4	22
5	47
6	100
7	220

Table 3.1: Switch Array Capacitors. Each switch in the capacitor array was connected to one of the above listed capacitor values.

for the entire array as:

$$C_{eq} = \sum_{i}^{7} a_i C_i \tag{3.2}$$

where a_i is a binary number representing the switch state of OFF ($a_i = 0$) or ON ($a_1 = 1$). The equivalent capacitance has 128 possible values ranging from 0 - $404.7\mu F$ with the 7 capacitor values in each switch array in Table . The full non-linear range is represented in Figure 3-8.

Digital Potentiometer

The AD5206 is a $10k\Omega$ 8-bit digital potentiometer with 6 variable resistors on chip. Figure 3-9 shows the representation of the device.

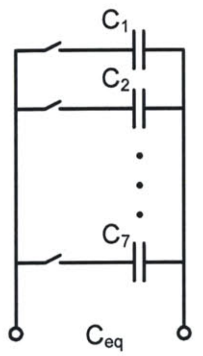

Figure 3-6: A Switched Capacitor Array with 7 different capacitor values connected to each switch in a parallel configuration. The equivalent capacitance across the terminals is the sum of the capacitors connected across the terminals.

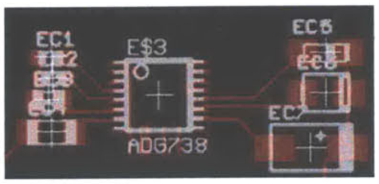

Figure 3-7: PCB layout of a Switched Capacitor Array

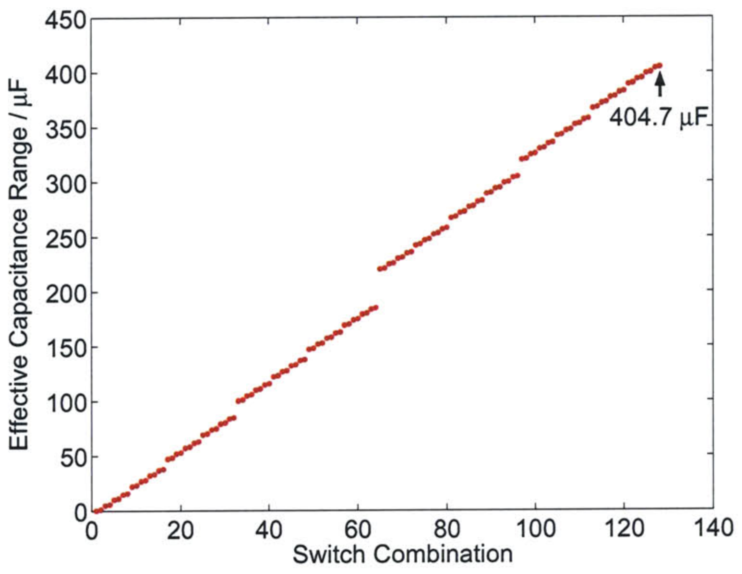

Figure 3-8: Capacitance Range of the Switched Capacitor Array. There are 2^7 or 128 possible combinations of switches in the ON state. Each of these is a unique combination of capacitor values connected in parallel with the values represented above.

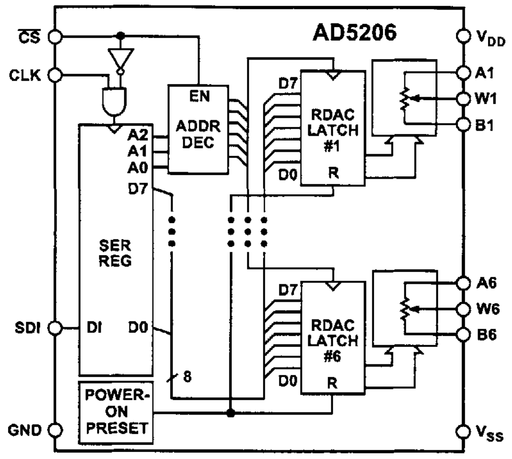

Figure 3-9: AD5206 Digital Potentiometer with 6 variable resistors on chip. Each variable resistor is represented by an 8 bit integer gene for the 256 possible values between 0 - $10k\Omega$

source: Analog Devices[2]

3.4.3 EHW Design

Passive elements are recommended to be used with the TRAC in a feedback mode as is typical for Op-Amp circuits for amplification, differentiation and integration as shown previously in Figure 3-4. The capacitors and potentiometers would be connected to the TRAC I/O pins as shown in Figure 3-11. This configuration allowed passive elements to be connected between non-neighboring TRAC cells, a departure from the normal operation of the TRAC and beyond the simulation capabilities of the Zetex evaluation software. This would perhaps extend the circuit capabilities of the TRAC though there was also the possibility this new capability could prove fatal to the device and would have to be monitored. The GA for this board would have to developed to incorporate the new devices and the circuit configurations now possible. Each variable resistor had 256 possible values, each capacitor array 128 and the network switches 256. Each device could be directly mapped into the chromosome with a number representing each possible configuration. This was done for the TRAC cells, variable resistors and the capacitor array. The switch network was treated differently with the gene representing the possible two-switch combination of a resistor and capacitor connected in parallel as shown in Figure 3-10.

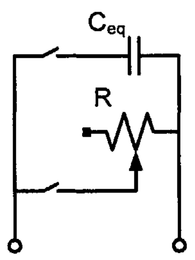

Figure 3-10: Passive Element Parallel Network. The switched capacitor array uses 7 of the 8 switches on the ADG738. The last switch is connected to a variable resistor as shown above. The capacitor switch array and variable resistor are both represented in the chromosome by their respective genes. An additional gene determines the 4 possible combinations of these two passive elements which appear across the nodes of the switch.

3.4.4 GA Development

The TRAC cells were represented as 3-bit integers as described in 3.3.4 with each capacitor array represented as 7-bit integer genes for all possible combinations of the 7 capacitors in the array. Each switch node and variable resistor was represented as 8-bit integer genes. A 2-bit integer gene determined the passive element parallel network shows in Figure 3-10 with the following four configurations across the output terminals:

1. Resistor
2. Capacitor
3. Resistor || Capacitor
4. Open Circuit

3.4.5 Progress

The initial pads for the TRAC were too small making the soldering process difficult for this board. The error was corrected for the later generation evolvable boards. Programming of the new chips required some troubleshooting beyond the usual verification of connections. The chips were daisy chained together to stream data for serial programming of the registers. The omission of pull up resistors as recommended by the Analog Devices data sheets for both of their devices was found to be the cause of data not getting past the first chip in the chain. Resistors were soldered in place to fix the problem and would be noted to be included in future work. The capacitor array and potentiometer were tested and found to be fully functional with the TRACs. The board configuration was fully specified in the chromosome with every possible circuit configuration available to the GA. Replication of the frequency doubling problem from the previous system in 3.3 was attempted with the passive elements now reconfigurable. The hard wired DC voltage inputs sources used in the previous attempt were not included. No suitable circuit was evolved for this problem. The search space for this system was much larger than the previous system. With additional work on the GA a circuit may have been evolved but we decided to move forward having used this board to successfully incorporate the new components into a working system. Work on the GA was put off for the next system and work proceeded to developing a larger board with scalable complexity.

3.5 Final Generation TRAC Evolvable System

At the outset of this project we envisioned evolvable hardware on systems with high complexity. This is in part due to the work done in our laboratory on quantum systems. The evolvable system with the second custom board was operational but we felt more insight could be gained from a board with more functionality and a huge search space to explore. The increased complexity would also challenge our understanding of the systems behavior and hopefully carry us into the domain where simulation and modeling could not easily be used as a suitable replacement. This unfamiliarity with the system would force us to rely on the GA to learn about our system with all the limitations and characteristics inherent in it.

3.5.1 Goals

This final board would have significantly more circuit resources available for evolution. The components of 3.4 would be used for this board in larger numbers along with a more complex switch network. The system would be scalable allowing the circuit complexity to grow by stacking boards together. Each board would be designed taking into account the expected time to populate, hardware resources available to program and evaluate a single board. The increase in the number of devices made the programming and evaluation time of the board of greater consequence requiring work to optimize performance. The control signal sequence to be developed would also need to optimize programming time while ensuring accurate and reliable circuit configuration. The GA would have to be developed to evolve circuits on this board with particular attention paid to an Analog-Digital Converter (ADC) circuit. Other circuit problems would be attempted. Successfully populating and testing a board of larger complexity would be a non-trivial task requiring troubleshooting and testing.

3.5.2 Switch Network

The complexity of our system would depend in part on the interconnections possible in the system. I designed a switch network with two layers; one with a large number of switched nodes connecting TRAC I/O pins in close proximity; the second layer with fewer number of super nodes connecting distant local nodes. Figure 3-11 shows a representation of the switch network.

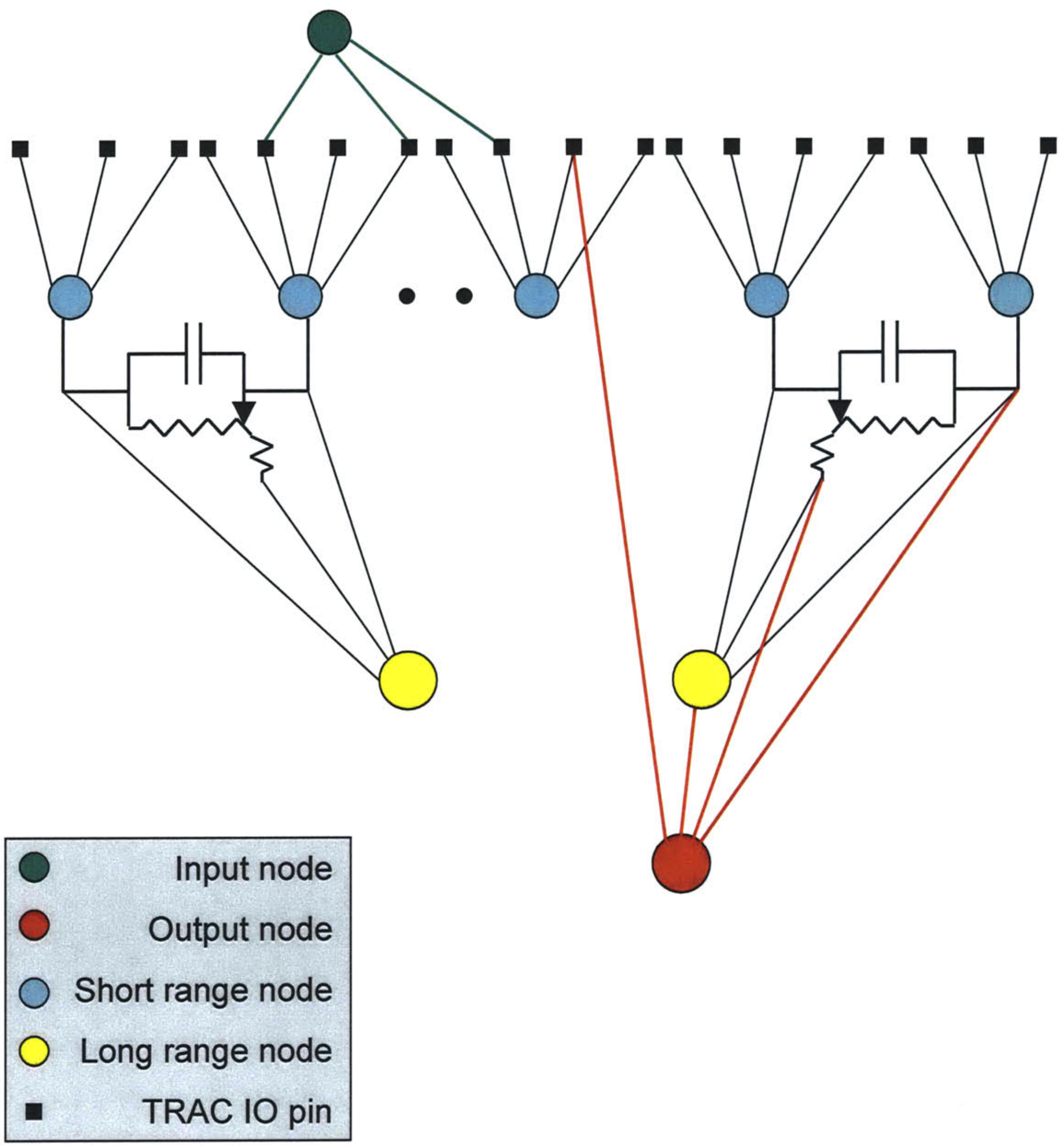

Figure 3-11: Circuit Switch Network. Network connections were arranged in layers. The input and output nodes provide access to the test signal and test output respectively. Local nodes are short range interconnects of neighbouring TRAC pins. Super nodes are interconnects of local nodes to allow routing between distant points of the board.

3.5.3 EHW Design

The placement of devices would have to allow high interconnectivity between circuit components while minimizing the amount of routing needed. The board would also have to allow some scalability to the system. The process of populating the board would have to be methodical and organized to minimize troubleshooting time and errors. Troubleshooting would depend on data streaming through the devices. A control signal sequence would have to be designed specific to the board layout, control signal connections and data programming connections. The DAQ board is the hardware interface between the reconfigurable circuit and the GA. Matlab was previously used for controlling the DAQ but the length of time it took to stream data bits was determined to be too long so work began to test operation of the DAQ using C++. Increasing the communication speed with the DAQ by using C++, the GA would be able to evaluate more circuit configurations. Doing most of my work in Visual Studio, I worked on transitioning from the Matlab DAQ Toolbox to C++ based Matlab Executables (MEXs) to control the DAQ board. I was able to control the following three basic functions of the DAQ board which are integral parts of the evolvable system:

1. Digital Output

2. Analog Output

3. Analog Input

Figure 3-12 shows the role of these functions in the evolvable system with all three used between the hardware interface (DAQ board) and the reconfigurable circuit. The digital output sends a stream of bits to the registers and control ports of the circuit components, loading the circuit configuration sent from the GA. Analog Output sends the test case waveforms to the circuit inputs while Analog Input digitizes the output response of the circuit. Op-amp buffer circuits were also used with the Analog inputs.

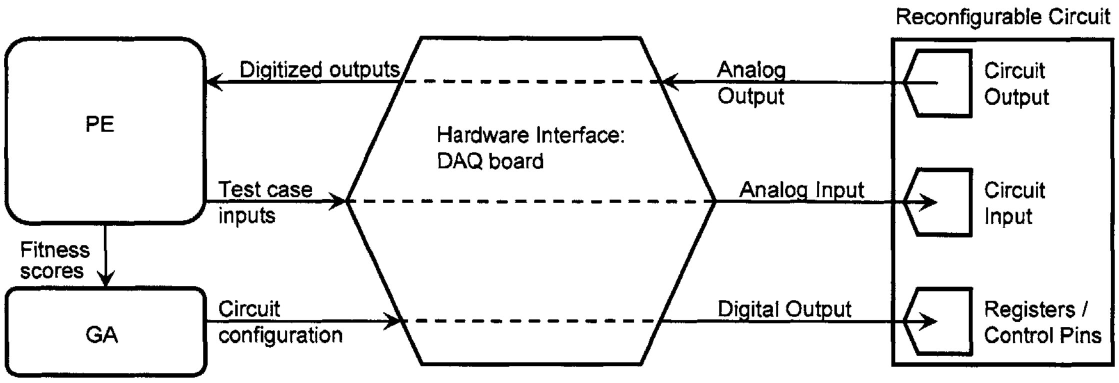

Figure 3-12: Evolvable hardware system. The Evolutionary Algorithm generates a population of individuals each representing a hardware configuration. These hardware configurations are sent to the Reconfigurable Hardware to evaluate their performance for some desired behavior. A Performance Evaluator uses test cases and objective function(s) to score the performance of configurations based on the test results. The EA uses the scores and configurations of a population to generate new individuals and the process is repeated with the expectation that performance progressively improves.

Gene Type	Alphabet Size[1]	Occurences
TRAC	8	160
Variable Resistor	256	18
Capacitor Array	128	18
RC Selector	4	18
Nodes	256	24
Super Nodes	256	8
Outputs	256	8
Input	256	1

Table 3.2: Chromosome characteristics
[1]The alphabet refers to the set of all possible values of the gene.

3.5.4 GA Development

Table 3.2 shows the 255 genes of the chromosome used to specify the circuit configuration of the evolvable board. The chosen representation has a very large search space covering close to every possible setting on each device. The position of the genes in the chromosome was selected based on the relative position of their respective devices. Neighboring devices had their genes closer to each other. The exact chromosome definition is given in Table A.1.

This complete specification along the lines of device settings is not the only approach to representing the circuit but it is the only one developed thus far. An alternate representation could parameterize some features of the circuit such as interconnect density, RC constants or other such descriptions which may lead to a reduction in the size of the search space and perhaps different results.

3.5.5 Progress

The board layout would place devices on both the top and bottom of the board. A ring configuration was chosen for the board layout. This would allow access to the TRAC pins for the switch network and make it possible to place the other devices within close proximity. The configuration also allowed devices to share control and data signals with a reasonable amount of routing due to their close proximity and similar ring configuration. The number of devices and interconnects required four layer signal routing with adjustments made to the routing parameters such as trace width and via diameter to achieve 100% routing. Switches were used to form a network of interconnects. Switch inputs were connected to various circuit pins within the system with the switch outputs representing nodes. A higher level network of nodes connected to super nodes would allow interconnect of distant circuit points. To allow scalability, mechanical inter-board connectors would allow routing of the switch network between stacked boards through the super-nodes. To program the board devices a control signal system had to be designed. Two clocks were used, one for the TRACs and switches, the other for the potentiometers. The potentiometer chips clocked in data on the rising edge of the clock while both the TRAC and switches clock data in on the falling edge. There were 8 daisy chain loops with data going back to the DAQ board. I used data streaming to verify that a chip is operational along with testing of all the pin connections and possible shorts. This method proved effective as each device was mounted on the board then its operation verified. The number of registers for each daisy chain was different and required redundant bits to be put in the control sequence. Figure 3-15 shows a representation of the control signal matrix. The Matlab code was optimized to increase the streaming speed from $12.5 bits/s$ to $554 bits/s$. This was well below the maximum $100 kHz$ clocking of the board. The transition to C++ based MEXs increased the speed of data streaming to $70 kbits/s$, 100 times faster than the Matlab software. The GA was developed to incorporate all the software components for controlling and evaluating the evolvable board. Objective functions for ADC and 2-tone discriminator were developed and implemented to attempt evolution for these two problems.

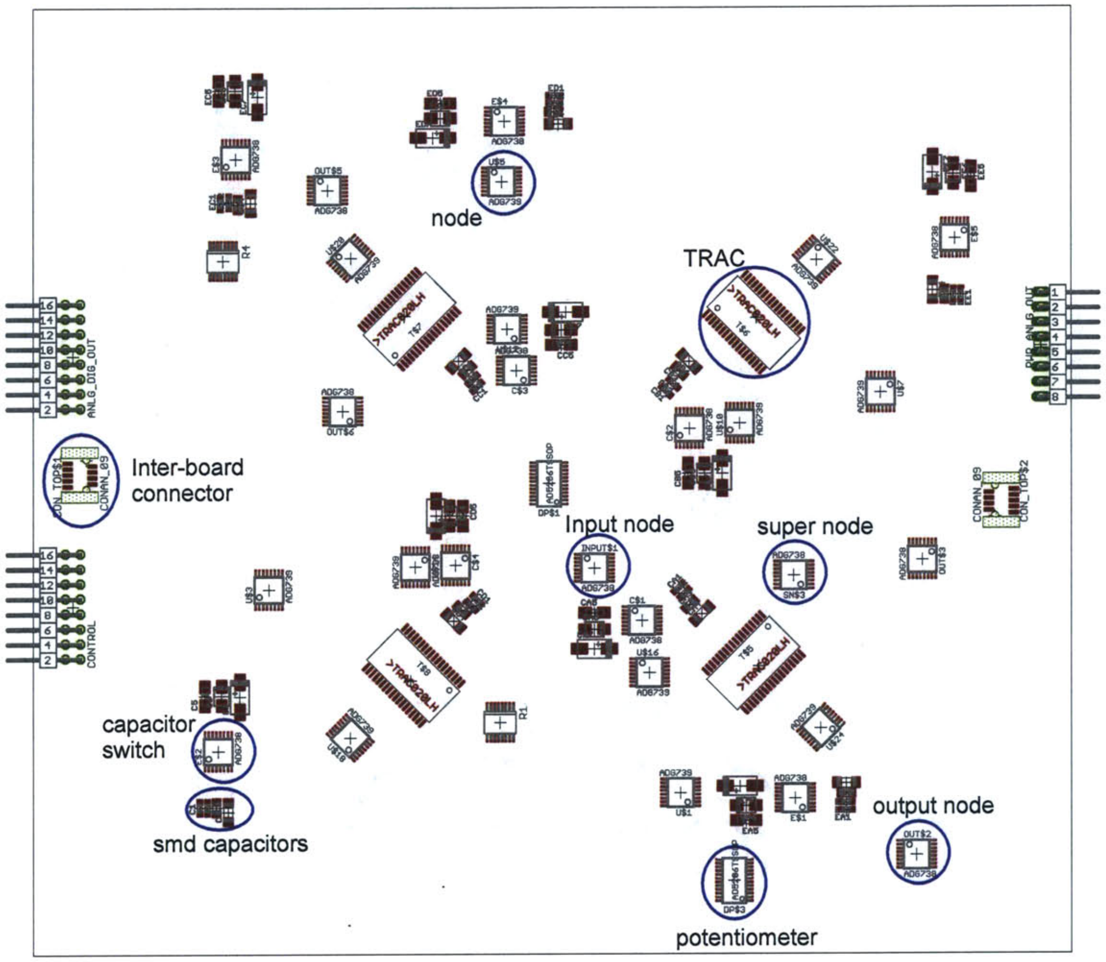

Figure 3-13: Top view of Third Generation TRAC-based Evolvable Board showing the reconfigurable components of the analog circuit.

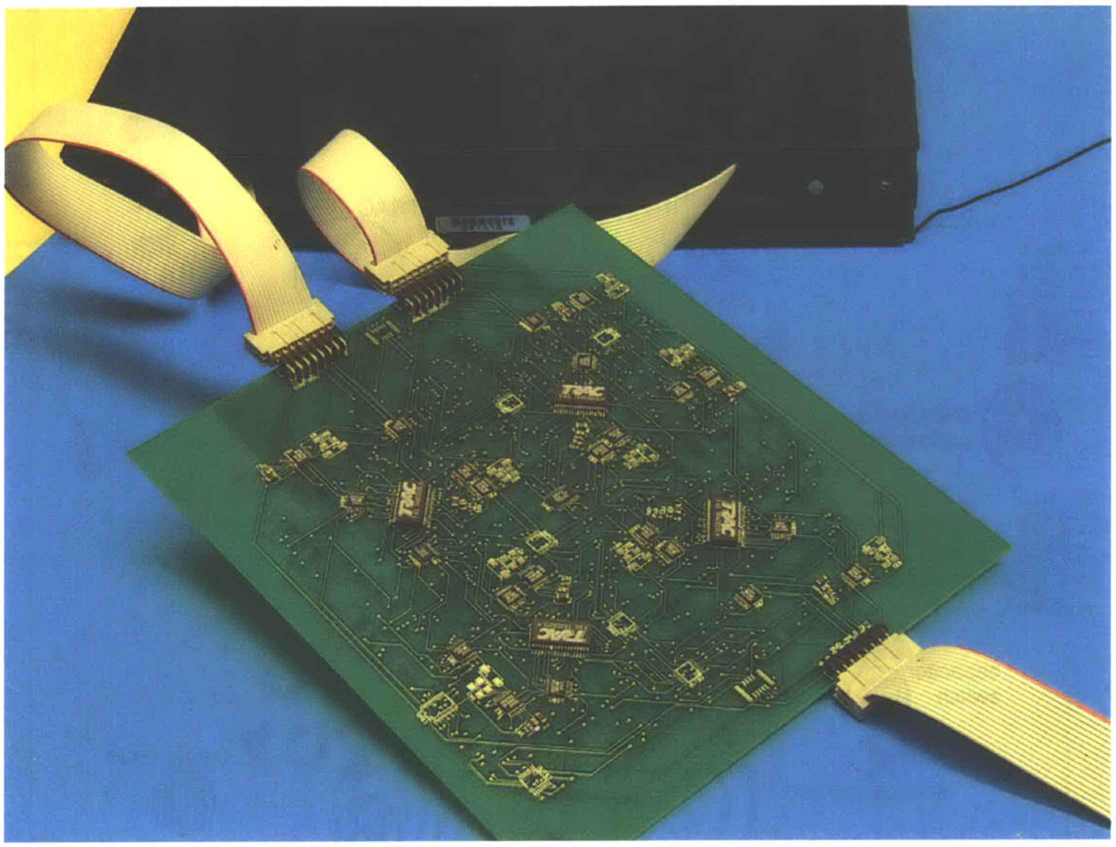

Figure 3-14: PCB layout of the Third Generation TRAC-based Evolvable Board.

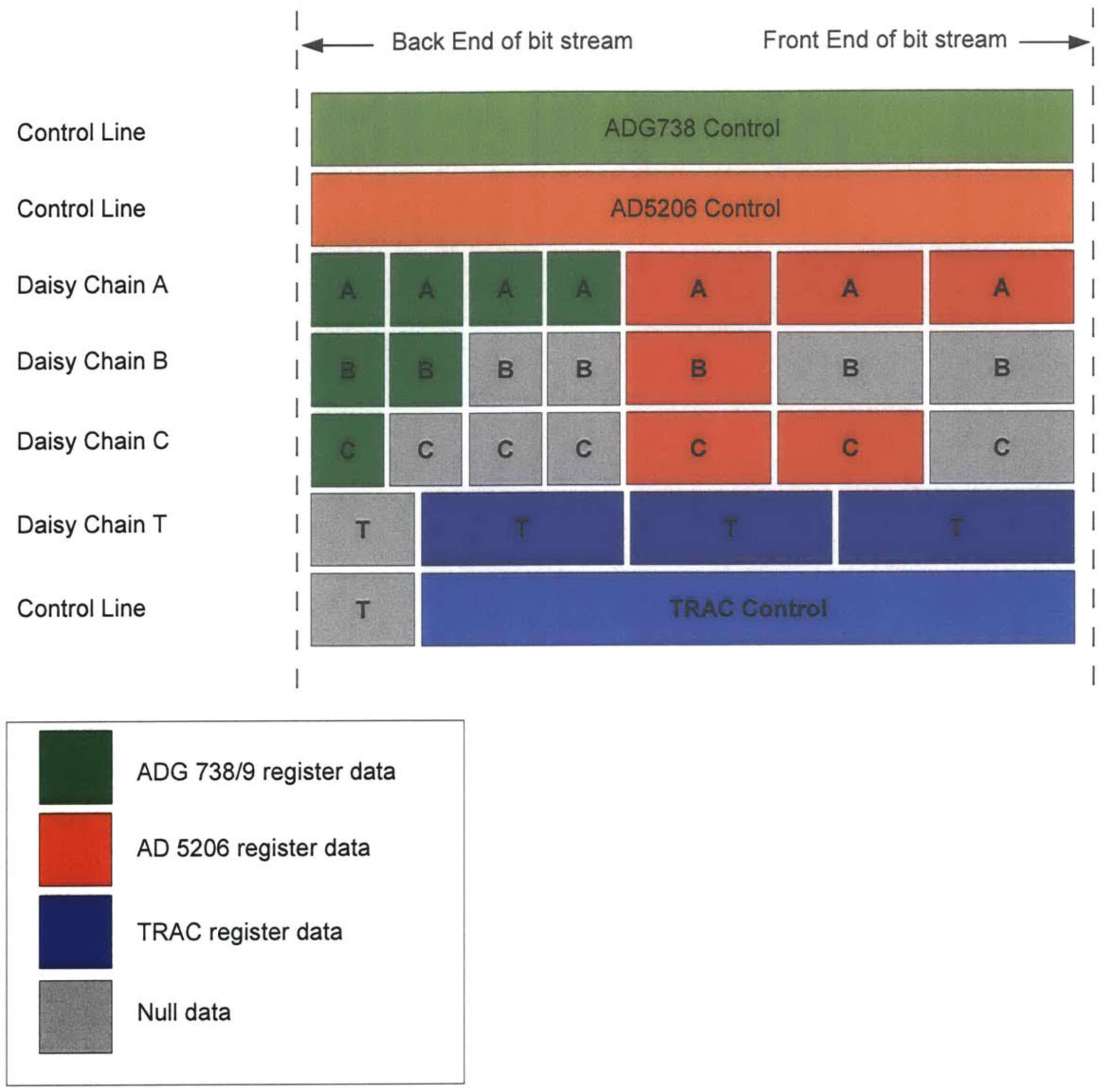

Figure 3-15: Digital Signal Matrix for the TRAC-based Evolvable Board. The control signal matrix represents the sequence of digital signals sent to program the devices on the custom board. The configuration data was streamed through daisy chained devices with 3 such data streams represented above. The signals in the data lines were matched to the corresponding sequence of signals in the control lines to ensure correct programming of the devices. Three control lines are shown for the TRAC, digital potentiometer (AD 5206) and switch (ADG 738/9). The daisy chains are chip registers connected serially. They are of different length but share the same control signals making it important that data arrives at the registers at the right time. Null data is streamed through some of the daisy chains to effectively line up the data in all the daisy chains.

Chapter 4

Evolved Circuits

The utility of an evolvable system is to produce system configurations which have some useful or desired behavior. Three evolved circuits that were used to aid development and characterization of our platforms are presented.

4.1 Frequency Doubling

4.1.1 Circuit Function

The circuit would take a sinusoidal input and produce a sinusoidal output with twice the frequency. The phase shift and amplification were not specified.

4.1.2 Hardware Platform

The evolvable platform was the first generation board in 3.3. Two voltage source inputs were fed into the TRAC as well as two resistors as used in the Frequency doubling design specified in the TRAC manual. The HP oscilloscope was used to measure the output signal, an Agilent 33220A function generator provided the signal input to the board and the HP 6624A provided power both to the evolvable board and to the TRAC pins.

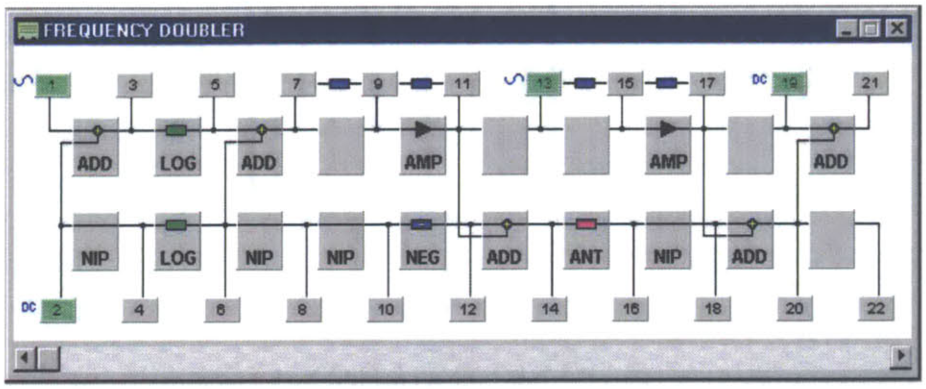

Figure 4-1: Zetex frequency doubling circuit

4.1.3 Chromosome

The chromosome was defined as in 3.3.4.

4.1.4 Objective Function

The oscilloscope was used to measure the frequency of the output signal. The cost function took the form:

$$Cost = |measured\ frequency - 2 \times input\ frequency| \tag{4.1}$$

A previous cost function which measured the difference between the normalized output and ideal output did not produce meaningful results.

4.1.5 Results

The circuit evolved was able to produce a non-ideal sinusoidal output that was twice the frequency of the sinusoidal input. Figure 4-2 shows the output produced from a $100Hz$ sinusoidal input. Every other bottom peak of the output was partially clipped.

4.1.6 Discussion

The search space for this problem was large with 8^{20} possible configurations for the TRAC. The evolved circuit used a configuration that is illegal in the Zetex provided

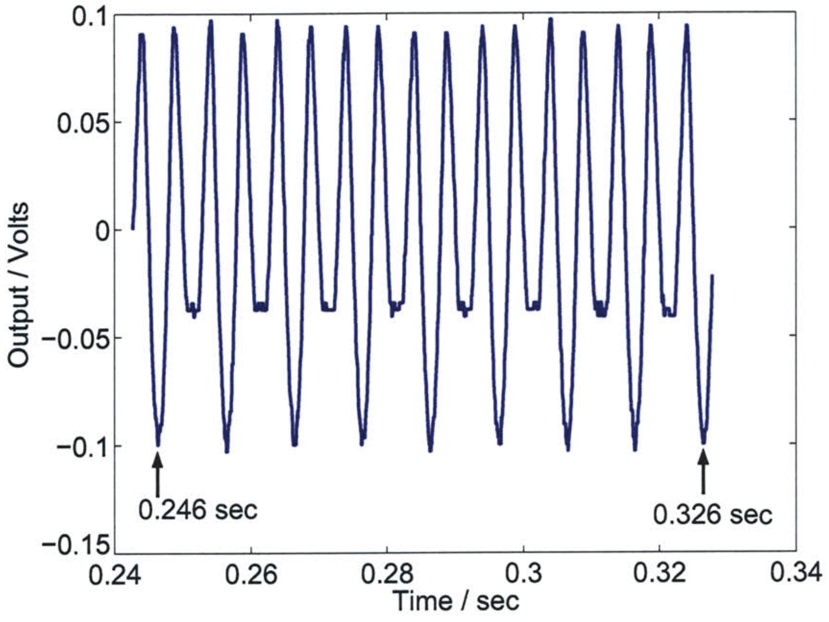

Figure 4-2: Circuit Output $\sim 200Hz$ from a $100Hz$ sinusoidal input

software for modeling and programming the TRAC Evaluation Board. The GA was able to exploit a feature not recommended by the manufacturer. The chip was not damaged by this configuration and though it is not clear if the violated rule was vital to the output behavior, it still highlights the ability of the GA to go beyond the limitations of traditional design which attempt to minimize or eliminate situations which do not allow a model to accurately predict circuit behavior. In this particular case the GA would evaluate the fitness of the illegal configuration based on measured performance. The eventual output was not a perfect sinusoid but we can conclude a frequency doubling circuit of reasonable quality had been evolved.

4.2 Two Tone Discriminator

4.2.1 Circuit Function

The output of the board is a digital signal that gives high for an input sinusoid signal of one frequency and a low output for another frequency.

4.2.2 Hardware Platform

The DAQ board provided both the analog input and analog output channels to the third generation evolvable board described in 3.5.

4.2.3 Chromosome

The chromosome was defined as in 3.5.4.

4.2.4 Objective Function

Two sinusoids of different frequencies are input into the system one after the other and the output signal is measured. The first half of the output corresponds to the first input frequency and the second half of the output to the second input frequency. The ideal output is a binary digital signal which has either a *logic high* or *low* value based on which of the two frequencies is present at the input. The assignment of which frequency is *logic high* or *low* is arbitrary once the behavior is consistent to allow us to distinguish between the two frequencies. The ideal *logic high* is $1V$ and *logic low* $0V$. The analog output points are digitized by assigning high or low based on a selected threshold voltage, V_T, chosen as the mid-point of the output signal range. The ideal *logic high*, $1V$, and *low*, $0V$, are assigned based on which half of the measured output signal has more data points above V_T. The measured output is normalized to a range between 0 - $1V$. The log difference of the measured output from this ideal output determines the fitness of the circuit.

$$Cost = \log \sum |measured\ output\ -\ ideal\ output| \tag{4.2}$$

The log function also provides a steep continuous slope encouraging incremental migration toward better performance scores.

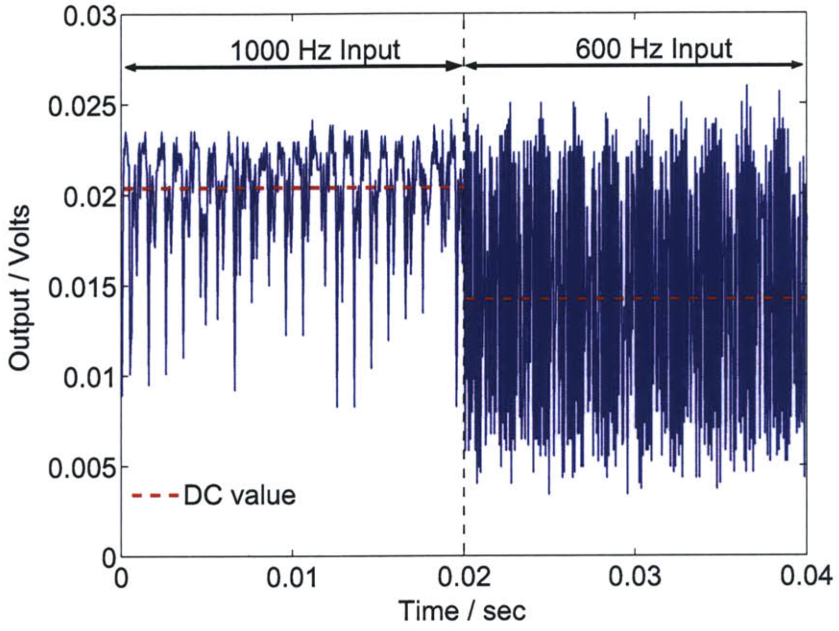

Figure 4-3: Output from Sinusoidal Input signal changing from $1000Hz$ to $600Hz$

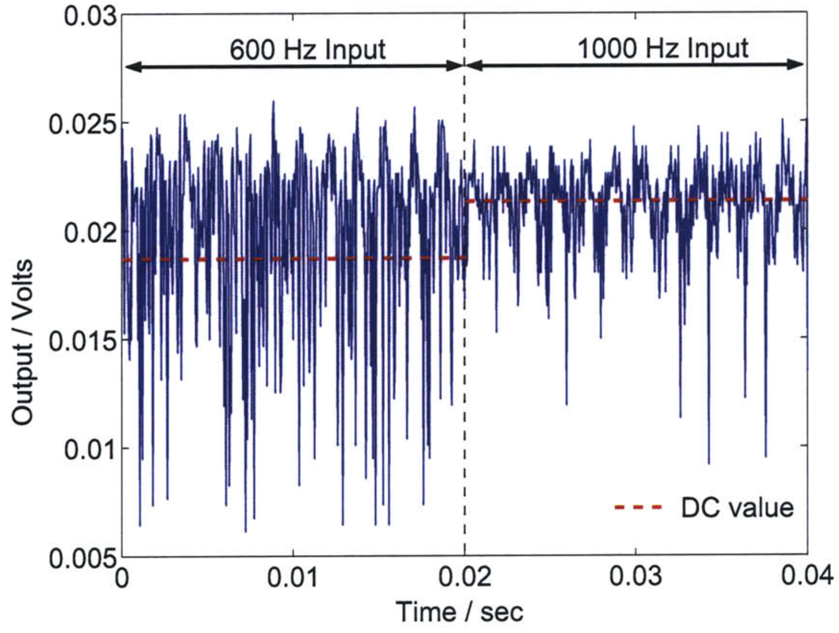

Figure 4-4: Output from Sinusoidal Input signal changing from $600Hz$ to $1000Hz$

4.2.5 Results

The output was quite noisy and non-ideal for detecting a frequency change but there was a discernable DC offset change when switching between input frequencies. The evolved circuit's performance was dependent on the order the input signals were sent to the board. In the first case where the frequency is first $1000Hz$ followed by $600Hz$ we get the change in DC value shown in Figure 4-3 on a consistent basis with $1000Hz$ as *logic high* and $600Hz$ as *logic low*. When the frequency is first $600Hz$ followed by $1000Hz$ we are able to duplicate the logic assignment as shown in Figure 4-4 but this output result was reproduced inconsistently. When the circuit is being tested for this order of frequency change the output would alternate between what is shown in Figure 4-4, to no difference in DC value between the two input frequencies. The system was trained on the input going from a high frequency to a lower frequency with the evolved circuit producing the result in Figure 4-3 on a consistent basis. The solution had better performance for the order of frequency change used during evolution.

4.2.6 Discussion

This experiment shows that evolution may produce circuits whose behavior is specific to the training data and does not generalize the way it is intended. Modifications to the training set and the method of evaluation are options which will likely overcome such behavior. If the DAQ board is contributing to the change in DC values it illustrates that the GA is blind to the source of output behavior. The coupling between reconfigurable hardware and the measuring instrument is something that can either be useful or problematic depending on the intended use of the hardware and is an important consideration for systems where this is likely to occur.

4.3 N-bit ADC

4.3.1 Circuit Function

A circuit which takes an analog signal input and produces N output binary digital signals to form a discrete representation of the input signal over a specified input range.

4.3.2 Hardware Platform

The same platform was used as in 4.2.2.

4.3.3 Chromosome

The chromosome was defined as in 3.5.4.

4.3.4 Objective Function

This is essentially a multi-objective problem to optimize the performance of each of the N bit outputs. There is also the problem of selecting which outputs are assigned to each bit. Each analog output signal is given a binary digital representation based on its measured voltage range and a threshold voltage V_T at the center of this range. A correlation matrix between each of these digitized outputs and the expected digital output is used to sequentially assign outputs to each bit based on the highest observed correlation. This is done until all N bits have been assigned to an output. Each output's cost is then evaluated using 4.2 . A single objective cost is a weighted sum of each channels cost. The weights on each channel could be adjusted based on the observed evolution and performance of the circuit.

$$Cost = \sum_{i}^{N} b_i O_i \tag{4.3}$$

$b_i = bit\ weight$, $O_i = objective\ cost$ of the i^{th} bit in an N-bit ADC.

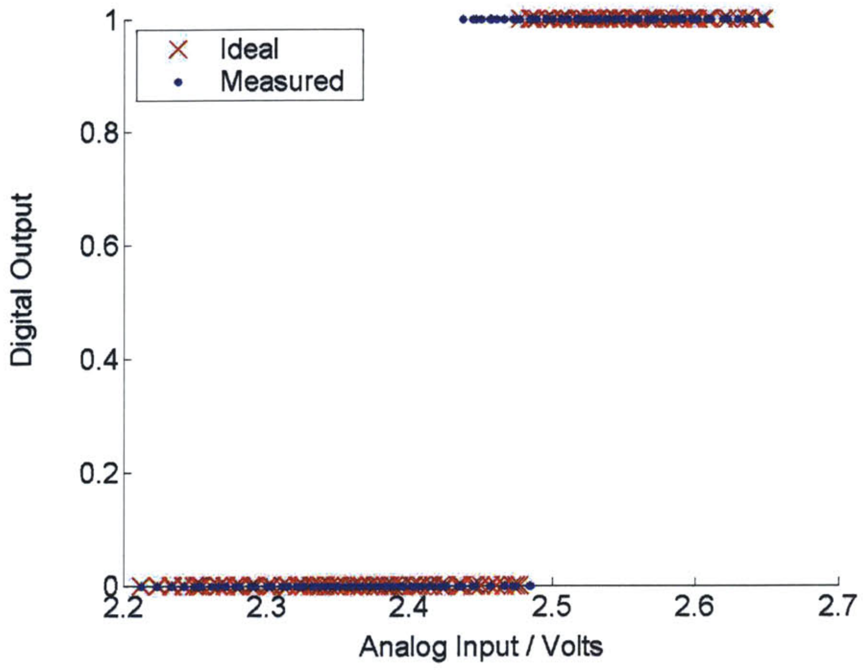

Figure 4-5: Ideal and measured Transfer Functions for the 1-bit ADC

4.3.5 Results

Circuits were evolved for 1,2,4 and 8 bit ADC. The transfer characteristics of the evolved circuits for 1, 2 and 8-bit ADC are shown in Figures 4-5, 4-6 and 4-7.

4.3.6 Discussion

The Most Significant Bit (MSB) was easily evolved for each of the ADC. This is essentially the output of a comparator. The 2-bit ADC showed partial success as the LSB was able to produce the correct output for 3 out of the 4 voltage ranges. This ADC was able to produce 3 of the 4 discrete values. The remaining higher bit ADCs could at best be described as noisy approximations of the ideal. The weighting was adjusted to give the higher bits more significance in the evaluation. The non-linear ideal behavior required to produce the higher bits proving a task that could not be achieved easily. The limits of the evolvable system were tested as the population size was as large as the PCs memory would allow with the evolutionary process being

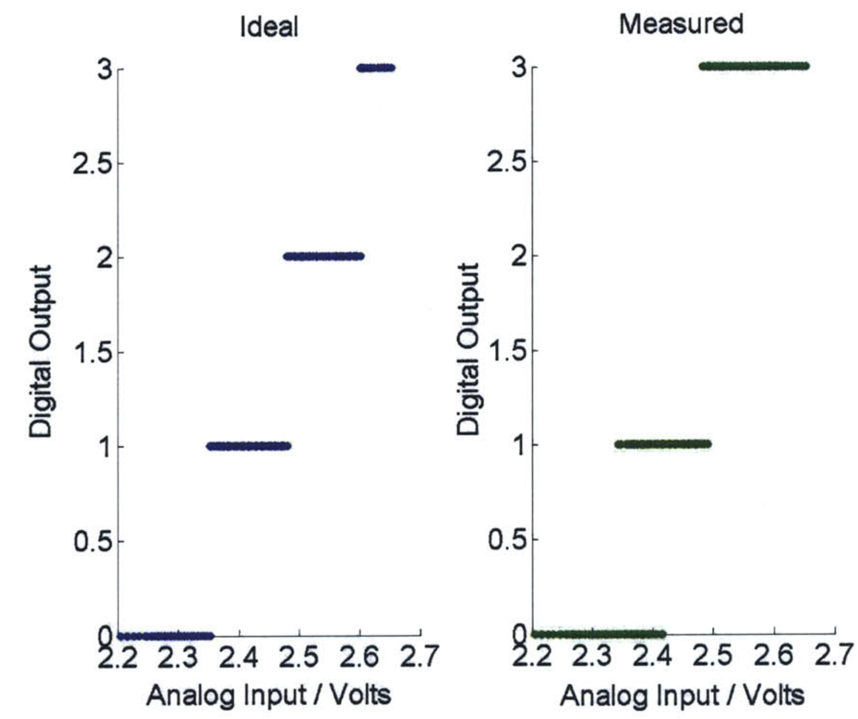

Figure 4-6: Ideal and measured Transfer Functions for the 2-bit ADC

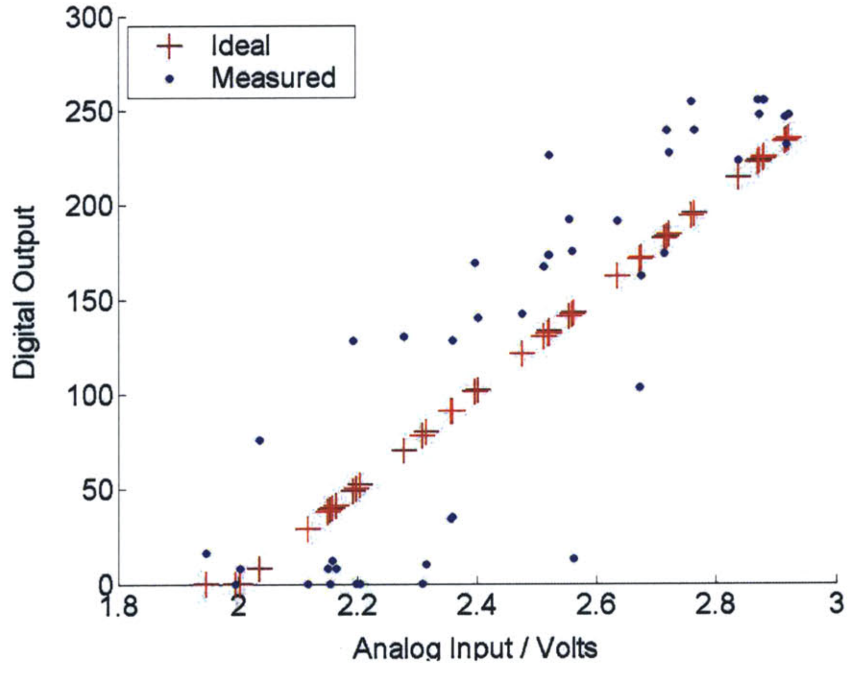

Figure 4-7: Ideal and measured Transfer Functions for the 8-bit ADC

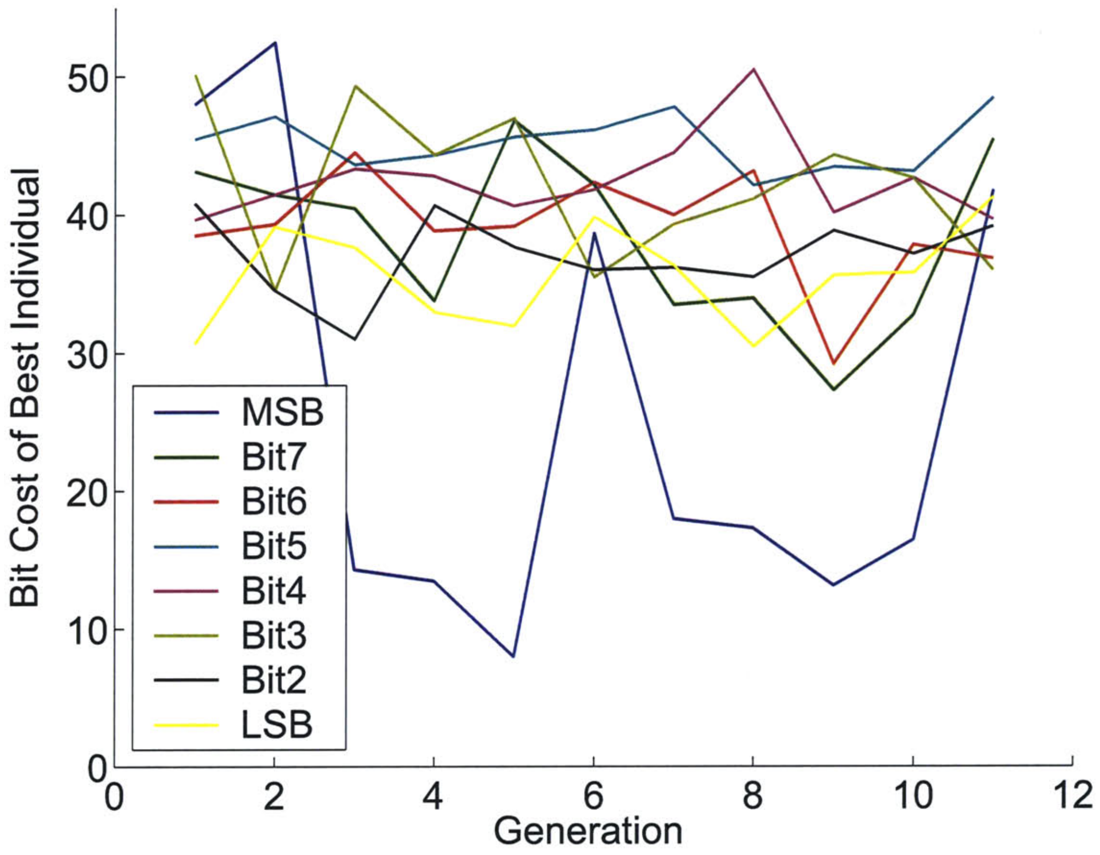

Figure 4-8: Multiobjective Progress of 8-bit ADC

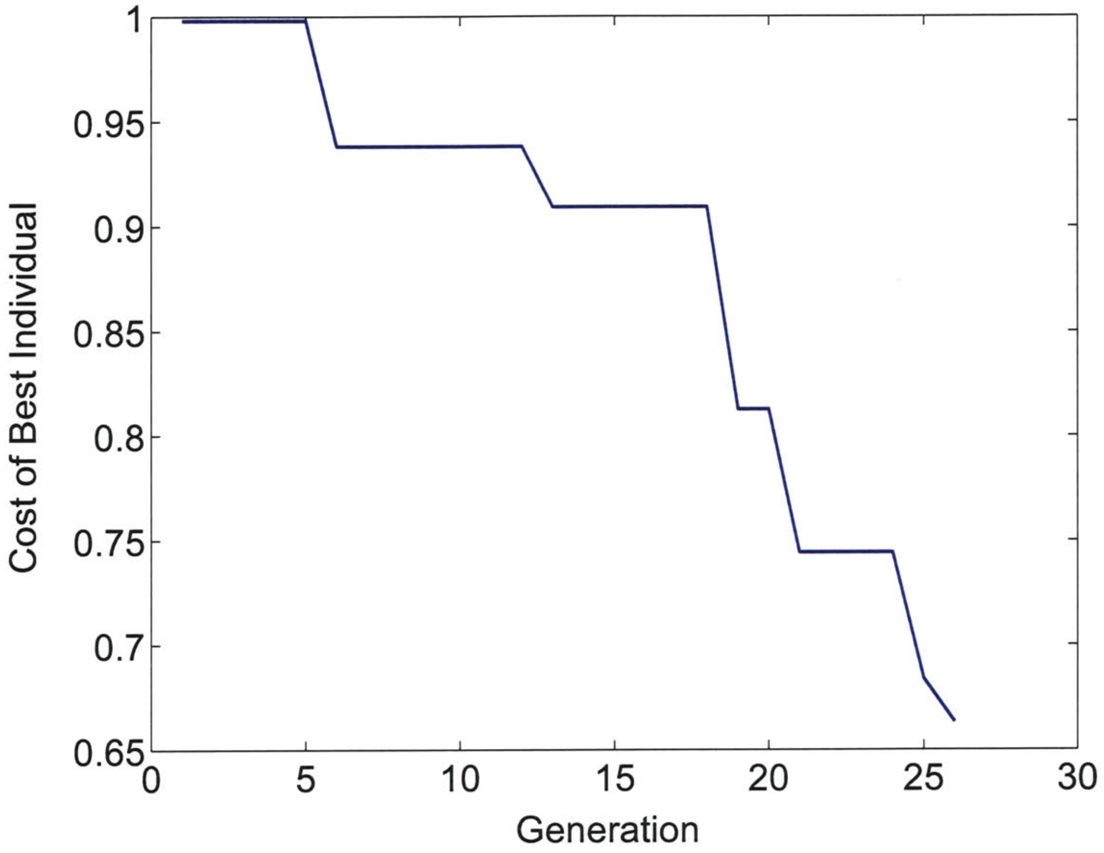

Figure 4-9: Progress of 8-bit ADC

allowed to run for typically 3 hours at a time both keeping previous populations and using new populations. Chromosome definition with the use of real numbers to specify every device can be revisited to allow a larger population, and hence search space, to be evaluated.

Chapter 5

Discussion

The work we have done developing an evolvable platform follows much progress in the field using *intrinsic* evolution [18]. An Evolvable Motherboard was developed by Paul Layzell as a tool for EHW [13]. It uses a reprogrammable switch matrix to interface with external components allowing researches to investigate evolution on a variety of physical media. Miller and Harding have proposed using this architecture to investigate evolution on media such as irradiated FPGAs and Liquid Crystal Displays[6]. Evolvable platforms are also being used in the Digital Circuit domain with Cell Matrix Corp. producing a stackable board made of fine grained logic cells with programmable truth tables[16]. These platforms aim to further explore EHW on complex systems and show promise for future progress.

We have had promising results evolving simple circuits and believe there is much scope to further explore the capabilities of our evolvable platform. The board currently uploads circuits in $\sim 30ms$ constrained by the maximum clocking frequency of our digital I/O equipment. We believe this time can be substantially reduced to $< 1ms$ using a clock speed closer to the $10MHz$ maximum of the TRAC. Our system still has much room to improve speed with much of the evaluation time for a circuit configuration due to the GA and waveform capture. The time for waveform capture is dependent on the problem specification and the requirements of the testing set from the PE[22]. The amount of data required by the PE to properly evaluate performance is the sole determinant of the length of time needed for data capture.

The results we have observed indicate there is potential to test and develop this system even further. We have used a simple genetic algorithm and a few novel benchmark problems thus far. Any EA can be used with the hardware we have developed with a host of more efficient and novel EAs being used in the field. The EA can continue along our current method of topology search, or the optimization of a user-defined design. The chromosome definition used thus far has been restricted to genes representing elemental circuit components such as individual switches. This need not be so as it leads to a large search space and may have a lower yield in passing on useful information through crossover. Genetic programming may also be a be a promising candidate with its tree structure well suited to manipulating the mathematical functions that describe the elemental blocks of the TRAC[11, 12]. The complexity of the platform and the speed of its reconfiguration make it a powerful platform to develop or benchmark EAs solving a host of circuit problems.

Appendix A

Tables

Table A.1: Final Board's Chromosome Assignment

Gene Position	1	2:4	5:7	8:27	28	29	30
Gene Indicant	U15	D4-dp14	F1-dp32	T4	U17	U1	U16

Gene Position	31:33	34:36	37	38	39:58	59:61	62:64
Gene Indicant	E1-dp35	C1-dp24	U24	U8	T5	F5-dp36	D1-dp22

Gene Position	65	66	67:86	87	88	89:91	92
Gene Indicant	U9	U7	T3	U23	U10	C2-dp21	U22

Gene Position	93	94:113	114:116	117:119	120	121:123	124
Gene Indicant	U6	T6	E5-dp43	F4-dp42	U11	D2-dp23	U21

Gene Position	125:144	145	146:148	149	150:152	153	154
Gene Indicant	T2	U5	E4-dp44	U12	C3-dp13	U4	U20

Gene Position	155:174	175:177	178	179:181	182:184	185	186:205
Gene Indicant	T7	E3-dp45	U13	D3-dp11	F3-dp46	U19	T1

Gene Position	206	207:209	210:212	213	214	215:234	235
Gene Indicant	U3	C4-dp12	E2-dp34	U14	U18	T8	U2

Gene Position	236:238	239	240	241	242	243	244
Gene Indicant	F2-dp31	OUT1	OUT2	OUT3	OUT4	OUT8	OUT5

Gene Position	245	246	247	248	249	250	251
Gene Indicant	OUT6	OUT7	INPUT1	SN1	SN2	SN3	SN4

Gene Position	252	253	254	255
Gene Indicant	SN5	SN8	SN6	SN7

U	Node
C,D,E,F	RC Switch
dp	Digital Potentiometer
IN	Input Node
OUT	Output node
SN	Super node

Bibliography

[1] A.J. Chipperfield and P.J. Fleming. The matlab genetic algorithm toolbox. In *Applied Control Techniques Using MATLAB, IEE Colloquium on*, pages 10/1–10/4, 1995. TY - CONF.

[2] Analog Devices. *AD5206 - 6-Channel, 256-Position Digital Potentiometer.* Data Sheet REV. 0. 1999.

[3] Analog Devices. *ADG738 - CMOS, Low Voltage, Single 8 to 1 Multiplexer, Serially Controlled Matrix Switch.* Data Sheet REV. 0. 2000.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[5] David E. Goldberg. Genetic and evolutionary algorithms come of age. *Commun. ACM*, 37(3):113–119, 1994.

[6] Simon Harding and Julian Francis Miller. A scalable platform for intrinsic hardware and in materio evolution. In *EH '03: Proceedings of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 221–224, Washington, DC, USA, 2003. IEEE Computer Society.

[7] T. Higuchi, M. Iwata, D. Keymeulen, H. Sakanashi, M. Murakawa, I. Kajitani, E. Takahashi, K. Toda, N. Salami, N. Kajihara, and N. Otsu. Real-world applications of analog and digital evolvable hardware. *Evolutionary Computation, IEEE Transactions on*, 3(3):220–235, 1999.

[8] Tetsuya Higuchi and Nobuki Kajihara. Evolvable hardware chips for industrial applications. *Commun. ACM*, 42(4):60–66, 1999.

[9] John H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.

[10] C.P. Husband, S.M. Husband, J.S. Daniels, and J.M. Tour. Logic and memory with nanocell circuits. *Electron Devices, IEEE Transactions on*, 50(9):1865–1875, 2003.

[11] John R. Koza. *Genetic programming IV : routine human-competitive machine intelligence.* Genetic programming series ; GPEM 5. Kluwer Academic Publishers, Norwell, Mass., 2003.

[12] J.R. Koza, S.H. Al-Sakran, and L.W. Jones. Cross-domain features of runs of genetic programming used to evolve designs for analog circuits, optical lens systems, controllers, antennas, mechanical systems, and quantum computing circuits. In *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, pages 205–212, 2005. TY - CONF.

[13] Paul J. Layzell. A new research tool for intrinsic hardware evolution. In *ICES '98: Proceedings of the Second International Conference on Evolvable Systems*, pages 47–56, London, UK, 1998. Springer-Verlag.

[14] Martin Lukac and Marek Perkowski. Evolving quantum circuits using genetic algorithm. In *EH '02: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH'02)*, page 177, Washington, DC, USA, 2002. IEEE Computer Society.

[15] Martin Lukac and Marek Perkowski. Evolutionary approach to quantum andreversible circuits synthesis. *Artif. Intell. Rev.*, 20(3-4):361–417, 2003.

[16] N.J. Macias and L.J.K. Durbeck. A hardware implementation of the cell matrix self-configurable architecture: the cell matrix mod 88/spl trade/. In *Evolvable*

Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on, pages 103–106, 2005. TY - CONF.

[17] Ian Ozsvald. *Short-Circuiting the Design Process: Evolutionary Algorithms for Circuit Design Using Reconfigurable Analogue Hardware.* Kbs msc, University of Sussex, 1998.

[18] E. Ramsden, G.W. Greenwood, and D. Hunter. Earp-1 - an evolvable analog research platform. In *Evolvable Hardware, 2005. Proceedings. 2005 NASA/DoD Conference on*, pages 20–25, 2005. TY - CONF.

[19] B.I.P. Rubinstein. Evolving quantum circuits using genetic programming. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 144–151 vol. 1, 2001. TY - CONF.

[20] Luk Sekanina. *Evolvable components : from theory to hardware implementations.* Natural computing series,. Springer, Berlin ; New York, 2004.

[21] A. Stoica. Evolution of analog circuits on field programmable transistor arrays. In *Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on*, pages 99–108, 2000. TY - CONF.

[22] A. Stoica, R.S. Zebulum, M.I. Ferguson, D. Keymeulen, and Vu Dong. Evolving circuits in seconds: experiments with a stand-alone board-level evolvable system. In *Evolvable Hardware, 2002. Proceedings. NASA/DoD Conference on*, pages 67–74, 2002. TY - CONF.

[23] Michael A. Terry. *Evolving circuits on a field programmable analog array using genetic programming.* M. eng., Massachusetts Institute of Technology, 2005.

[24] Michael A. Terry, Jonathan Marcus, Matthew Farrell, Varun Aggarwal, and Una-May O'Reilly. Grace: Generative robust analog circuit exploration. In *EvoWorkshops*, pages 332–343, 2006.

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

[26] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In Tetsuya Higuchi, Masaya Iwata, and L. Weixin, editors, *Proc. 1st Int. Conf. on Evolvable Systems (ICES'96)*, volume 1259 of *LNCS*, pages 390–405. Springer-Verlag, 1997.

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

[28] Zetex. *Totally Re-configurable Analog Circuit - TRAC.* Data Sheet. 1999.

[29] Zetex. *TRAC User Guide.* Zetex, 2.x edition, 1999.

4649-S1