GPIC: A Genetic Programming Intrinsic Circuit Design Platform

Michael A Terry, Jonathan Marcus, Matthew Farrell ,Una-May O'Reilly

Computer Science and Artificial Intelligence Lab (CSAIL)
Massachusetts Institute of Technology
Cambridge, MA, USA
m_terry@alum.mit.edu, unamay@csail.mit.edu

Abstract. In the realm of analog evolvable hardware, the potential of a coarse-grained, topological search approach which employs intrinsic testing has been unexplored to date. Our evolvable hardware design platform named GPIC (i.e. Genetic Programming of Intrinsic Circuits) is designed for this approach which makes it suitable for adaptive, fault tolerant system design as well as CAD flow applications. On a workstation GPIC evolves circuit designs which are each intrinsically realized and tested on a Commercial Off-The-Shelf (COTS) field programmable device, the Anadigm AN221E04. This intrinsic approach provides speed over software circuit simulators that employ computationally expensive models. To interface with our COTS chip, we have implemented a resource manager within GPIC and configure it via the vendor's design GUI. For circuit search and optimization GPIC uses a cyclic graph representation that explores the topological space of analog circuits. Each circuit is represented using high level functional (i.e. coarse grained) analog components, such as adders and amplifiers. For component value optimization, GPIC uses particle swarm optimization. As demonstration of the platform's capabilities, we describe how GPIC has evolved a simple controller for a linear, third order plant.

1 Introduction

We started specifying GPIC in October of 2004. We decided to investigate whether it is possible to evolve circuits that can be designed efficiently and in a routine manner. Very early in the project's inception, we decided to focus upon the domain of analog circuit design. Our decision stemmed from our recognition of the ubiquity of natural analog processes. Furthermore, as [**?**], (p 1825) states, "despite the trend to replace analog circuit functions with digital computations, (e.g., digital signal processing in place of analog filtering), there are **some typical functions that will** *always* **remain analog**" (emphasis from [**?**]). Thus, analog design is here to stay. It is also apparent that analog design has surrendered less to design automation than digital design. The analog domain is populated by design gurus and libraries of reused designs that are tweaked. The design flow has steps that are interdependent and too time consuming to allow

humans to freely explore novel designs ([**?**]). We want to investigate whether evolvable hardware approaches can contribute in a domain with this level of complexity. The potential payoff is valuable in terms of generating better designs that can be adopted by the industry and providing tools that allow the analog flow to keep pace with its digital counterpart.

Since we have committed to using evolutionary search and optimization as the primary means of deriving circuits, we have to select the means by which candidate circuits are tested. As per Koza [], it is possible to employ a circuit simulator such as Spice, [**?**]. However, as demonstrated in Koza's work, Spice simulations take a lot of time (because of their high fidelity models) and are so narrow or specific that they have to be repeated for many test configurations (e.g. corner conditions). As per McConaghey, [**?**], this may preclude the use of such an approach to actually realize its evolved circuits. There are three alternatives to Spice-like simulation: the use of symbolic analysis tools, simulation with simpler, faster circuit models or intrinsic testing. Each method presents a tradeoff in speed and fidelity. Ultimately we would like GPIC to employ all of them. Simpler models are fast but only go so far because they do not model higher order aspects such as XXX. They are appropriate at an early design stage but as flow progresses to sizing and validation they must be replaced. Symbolic analysis tools yield interpretability and might provide faster circuit evaluation than Spice. It seems that the potential stemming from early work by [**?**] has not been investigated to date. Complementary tools such as ISAAC and Donald ([**?**]) can be used for circuit structure and sizing design. However, their integration into an EHW system seems challenging as they have become commercial products. Instead, as a starting point, we selected intrinsic testing via a reconfigurable device (in this case a chip). This choice is commensurate with our interest in adaptive, robust evolutionary systems. Using a reconfigurable device allows us to investigate the potential of the concepts around evolutionary self-repairing hardware. We find the possibility that evolutionary algorithms could provide the benefits of robust, more life-like systems because of their nature-inspired design process very compelling. In addition, reconfigurable devices offer a definite advantage in speed of circuit evaluation. We do recognize that using a reconfigurable device raises issues concerning whether evolution can produce a useful general circuit. Even if we do not exploit material silicon properties a la Thompson,[**?**], the means by which a device is reconfigurable can bias the solutions evolved with it as their substrate. **Varun: insert here stuff on using COTS FPAA**.

Choosing the granularity of GPIC's circuit elements is another key decision. An evolutionary algorithm must be provided with the right building blocks so that it can find solutions without too much effort within the search space implied by composition of these elements. It is an open question as to whether the building blocks that humans use for circuit design embody fundamental design principles that arise from the physical properties of circuits (i.e. voltage, current, resistance) or whether they are simply human-fashioned abstractions that suit human style cognition (which is biased toward modular, decomposable systems). In the field of EHW, the circuit elements of each reported system reflect where

their architects' decided to weigh in on this question. The cost, availability and control over the configuration process are also factors considered when granularity is decided upon. In Section **??**, we shall elaborate upon how our decision process arrived at the selection of the Anadigm AN221E04 for GPIC. This is a coarse grained device. Its building blocks are functional analog CAMs which include adders, amplifiers, differentiators and integrators.

Like most powerful EHW systems, GPIC must evolve the topology of a circuit as well as parametric aspects of a given topology. GPIC is also required to be sufficiently general to handle different reconfigurable devices. While the particulars of a specific device ground the lowest level circuit building blocks and constrain the available resources, in GPIC no assumptions about the device level blocks or resources are allowed in the evolutionary algorithm. Further, practicality dictates that GPIC be produced with less than $5K of funding.

By combining the exploitation of coarse grained elements with intrinsic testing, we think GPIC sits in an interesting, novel space. It allows a distinctive foray into on-line adaptive and fault tolerant evolvable hardware circuits because it uses a COTS device and standard components. This should make its results more acceptable to industry. It also allows an economical and time efficient foray into the broad domain of VLSI and CAD with its use of elements that are conversant with human design.

The goal of this paper is to describe how we arrived at GPIC. We believe that sharing how we made our decisions and handled the issues that arose in realizing it will be helpful to others who decide to build an analog, intrinsic system on a modest budget.[1] In Section 2 we describe how we reached our decision on granularity and selected a reconfigurable device. In Section 3 we describe how the Anadigm AN221E04 can be configured from an evolutionary algorithm. In Section 4 we describe our genetic representation of a circuit and explain how we handle resource usage tracking from within the evolutionary algorithm. In Section **??** we describe the particle swarm algorithm that handles component value optimization. After these descriptions, we use a straight forward controller problem to demonstrate that GPIC can evolve circuits. We conclude with a short list of plans for using GPIC in the future.

2 Choosing GPIC's Circuit Elements and Reconfigurable Device

Choosing the granularity of GPIC's circuit elements required that we balance our preferences with availability. Our preferences were slightly *a priori* biased toward including the information that humans use when designing circuits. Humans compose circuits from very intentionally chosen building blocks. The lowest level building blocks may include resistors, capacitors, diodes, inductors and transistors. With the exception of the transistor, one design merit of these components is their linear behavior. Because of their linearity, they are relatively

[1] An anecdotal note: designing GPIC was our entry into the EHW realm.

simple to assemble into systems that have analytically derivable behavior. This allows designs to be generated and tested in conjunction with specifications. This allows designs to be *transparent*, i.e. comprehensible. Often these lowest level building blocks are combined and encapsulated to form a higher level set of building blocks that a designer works with. For example, differentiators, integrators and adders are specific configurations of opamps. These higher level building blocks encapsulate useful functions and establish clear interfaces. Thus, like the lowest level blocks, they facilitate the generation and analysis of modular designs.

Realistically, however, the capabilities of evolvable hardware are limited by the availability of reconfigurable devices. This has been true from the field's inception. Within the scope of digital circuits, a wide variety of FPGA devices are commercially available. This market is very broad, and companies (e.g. Xilinx) have been successfully developing these reconfigurable technologies for a number of years. Unfortunately, for analog and mixed-signal circuit design, field programmable devices are a niche market, and very few devices are easily obtainable. The most recent and informative evaluation of analog options available to us was published in 2001, [**?**] so we initiated our own survey that considered the following devices:

- Two custom designed field programmable transistor arrays: FPTA and FPTA-2. FPTA has been used in U. of Heidelberg experiments (e.g. [**?**]). It is an apparently older version. FPTA-2 is used in JPL experiments. It appears to be a successor of FPTA. It fits into the evolution-oriented reconfigurable architecture (EORA). It is also part of the SABLES system [**?**,**?**].
- PAMA, Programmable Analog Multiplexor Array, developed at Catholic University of Rio de Janeiro([**?**])
- the Lattice Semiconductor ispPAC10 FPAA [**?**]
- the Anadigm FPAA family. [**?**]

In Table 2 we compare these devices on the basis of technology for configuration ('Config'), operational bandwidth, interconnection versatility ('Topo'), device resources, configuration time and relative cost. Configuration technology is important because certain technologies can limit how many times a device can be reconfigured. For example, the Anadigm AN221E04 use of SRAM is advantageous in this respect. Operating bandwidth contributes to application versatility.

Each device in our survey offers a different level of circuit element granularity. In Stoica et al ([]) the authors note that the optimal choice of device resources and *programmable granularity* is task dependent. They state *transparent architectures* as desirable so that analysis is possible. Plus a device should not be open to damage by an evolved configuration.

The PAMA device, [**?**], can function as a fine-grained architecture, similar to an FPTA, or as a coarse-grained architecture when a human designer manually configures certain of its switches. It was custom designed with evolution in mind. It can accept random configurations and not sustain any damage. Rather than

Device	Config Tech	BdWth	Topo	Resources	C-Time	Cost
PAMA	Programmable Muxes	undetermined	limited	BJTs, resistors	.08 ms	high
FPTA-2	SRAM	undetermined	intra-cell and inter-cell	MOS transistors	.008 ms	high
Anadigm AN221E04	SRAM	500 kHz	intra-CAB, inter-CAB, inter-chip	4 CABs of 2 opamps each	3.8 ms	low
IsPAC10	EEPROM	200 kHz	limited	opamps, caps, resistors	100 ms	low

Table 1. Comparison of different analog field programmable devices.

using CMOS transistors, it uses bipolar transistors, which tend to be more suited for analog functions. It does not load its configuration into memory but must hold the state as constant signals switched through internal muxes. This would be problematic if it were used in an embedded system.

The U. of Heidelberg's FPTA ([**?**]) is a switched network of 256 (16 X 16) programmable CMOS transistors (half NMOS and half PMOS) arranged in a checkerboard pattern. Typically, with this FPTA, a genetic algorithm with a fixed length bit string genome directly represents a circuit as a vector of routing bits, transistor terminal connections and channel geometry in the network. Like FPTA, FPTA-2, shown in Figure 1, is a 'sea of transistors' interconnected by other transistors that act as signal passing devices. It consists of an 8X8 array of cells. A cell's reconfigurable circuitry consists of 14 transistors connected through 44 switches. Each bit in the genome controls the opening of a switch. A cell also includes 3 fixed capacitors and a small number of directly configurable resistors. In addition, it is suspected that evolution may be able to exploit parasitic capacitances.

The FPTAs are very flexible, fine-grained devices because they use multiple transistors configured by a relatively large number of switches within each cell. With their lowest level elements they are able to express novel and standard functions. As a result of this flexibility, however, it is open to question how interpretable or 'standard' a circuit design is. Clearly, circuits that humans design are not connections of cells of transistors. This mismatch of organization is one aspect that makes the FPTA evolved designs difficult to comprehend. Another aspect of the FPTA architecture is that it omits the design principle of small-signal modeling. Typically, human designers model non-linear transistors as linear by selecting a DC reference voltage to bias the input signal so that the transistor's operating range has linear behavior with respect to the range of a

small input signal. The lack of biasing is an especially relevant problem for MOS devices, since their transconductances are generally lower than that of BJTs throughout their operating regions.[**?**] Working with linear behavior makes constructing approximately linear circuits out of nonlinear components tractable for humans. On one hand the omission of bias in the FPTAs offers radical flexibility. Although it is conceivable that evolution may discover appropriate linear operating points, nothing in the FPTA's direct genome specification ensures this. Instead, evolution is unrestricted and can find arbitrary biases in combination so long as it yields a circuit that is highly fit. On the other hand, it is debatable whether evolved circuits on the FPTAs contain building blocks in any re–useable or modular sense due to this 'flexibility'. As well, circuits which use the devices in a mode that is outside their intended operating envelope are very likely to not be trustworthy at varying temperatures or consistent across different chips.

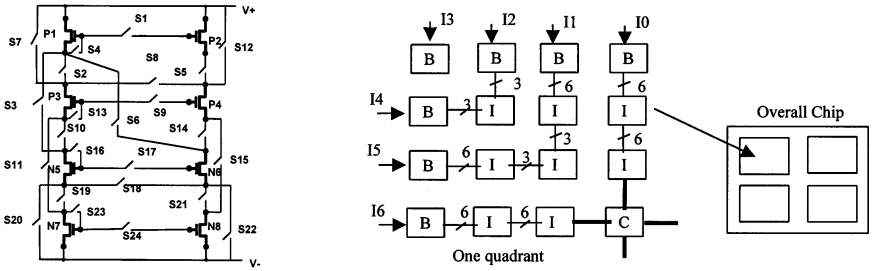

Fig. 1. Left: FPTA-2 cell schematic showing 24 switches and 8 transistors, Right: FPTA-2 upper left quadrant of 16 cells. From [**?**]**.**

The output of an opamp, see Figure 2, is the signal difference of its inputs with some gain. It does not require a bias point for linearity and, in combination with resistors or capacitors, it is very versatile. For example, an opamp in closed loop with a resistor is a linear amplifier. If a capacitor is in series with the resistor in the feedback look, a continuous integrator is obtained. Thus, an array of opamps, capacitors and/or resistors is a viable alternative to a transistor array despite not being quite as fine grained. Such an array can be realized very cheaply with a PCB. We considered designing a PCB-based opamp, RC array with versatile topological interconnect switching. With additional PCBs and switches, this setup would scale well and cost little. However, despite also being transparent to configure, a PCB implementation would have limited bandwidth and require other special purpose hardware to integrate into a testbench.

We knew of at least two COTS device families, the IsPAC10 and Anadigm FPAA, that have circuit elements in the opamp family. The IsPAC10 , see Figure 3, consists of 4 programmable analog modules (4 opamps, and 8 input amplifiers total) interconnected with programmable switching networks. This is very

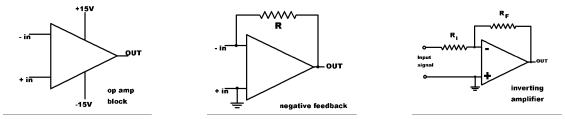

Fig. 2. Left: opamp, Middle: with negative feedback, Right: as inverter

limited interconnect between a small quantity of resources. In [**?**] a non-minimal (and non-intuitive) implementation of a lead compensator was configured on the IsPAC10 that used 3 analog modules. The parameters (i.e. gains) of the compensator were optimized using simple parameter search via a GA. Despite a relatively simple purpose, Greenwood reported in [**?**] that he found the range of available capacitors constraining. Configuration of the IsPAC10 is a proprietary process. Through personal communication, [**?**] we learned that a partnership with a Lattice Semiconductor staff member enabled the Greenwood team to write a simple conversion program to map their genome to an appropriate bit-stream format. Because their genome was a list of component values that were applied to a topology that was configured beforehand, this did not help us to determine whether topological reconfiguration from an evolutionary algorithm could be achieved. We assume the configuration is proprietary.

Ultimately we chose to use the Anadigm AN221E04 over the IsPAC10. Our discussion and description of our choice is detailed in the following subsection.

2.1 The Anadigm FPAA

Anadigm is a spinoff from Motorola. It currently offers a second generation product line of commercial FPAAs called the Anadigmvortex. For detailed description of these devices, see [**?**]. They are available in an industry standard 44 lead Quad Flat-Pack(QFP) package.

Resources: Each device is an array of configurable analog blocks(CABs), each of which contain two opamps, 8 capacitors, a comparator, and a Successive Approximation Register (SAR) that performs 8-bit analog-to-digital conversion of signals. The device also contains one programmable lookup table that can be used to store information about the generation of arbitrary waveforms, and is shared amongst the CABs. The architecture is illustrated in the left hand block diagram of Figure 3. The FPAAs within the product line share a common general architecture, although they vary in their I/O capabilities, their reconfiguration capacity, and the number of CABs on a chip. All but one of the FPAAs in this line contain a 2x2 matrix of CABs that can be freely connected to one another. **INSERT Sample and Hold, Clock phase.** We used the Anadigm AN221E04 which is the entry level device. Any signal can be routed to the I/O pins of the device through 4 programmable I/O interface blocks and two dedicated outputs, each of which can also act as filters or amplifiers. The option for expanding

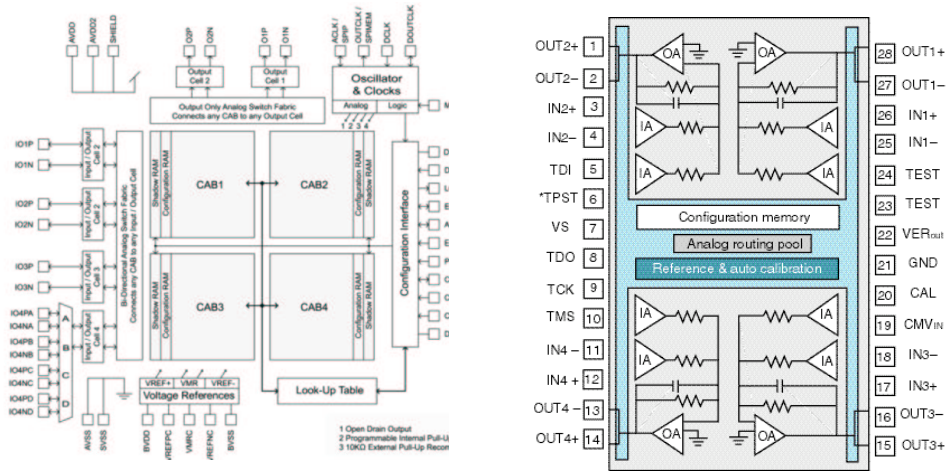

Fig. 3. Reconfigurable FPAA Architectures: Anadigm(Left), Lattice ispPAC10(right)

the number of resources is to daisy chain multiple devices. The accompanying AnadigmDesigner software supports this.

Configurable Elements: Despite the existence of opamps and switched capacitors, the Anadigm AN221E04 does not support circuit design at this level of granularity. [2] Instead, with the exception of the SAR, a circuit must be specified at the abstraction of coarser grained building blocks termed Configurable Analog Modules (CAMs) that are interconnected by wires. CAMs come predefined by Anadigm. Their documentation states they are open to requests for custom CAMs. See Table 2 for the set of available CAMs. Among the set is a selection of filters, amplifiers and rectifiers.This is an extensive set in terms of capability and flexibility. It includes analog building blocks that humans frequently use such as an amplifier, a filter, adder, and integrator. During human design, each time a CAM is added to the current circuit design in the AnadigmDesigner GUI, the GUI displays how the resources that implement it are allocated from a CAB. This is illustrated in Figure 2.1. To insert a CAM the GUI must be able to fully allocate its resources from one CAB.

[2] Interestingly, this was supported in an earlier version before Anadigm spun off. Our hunch is that**XX** proprietary in nature.

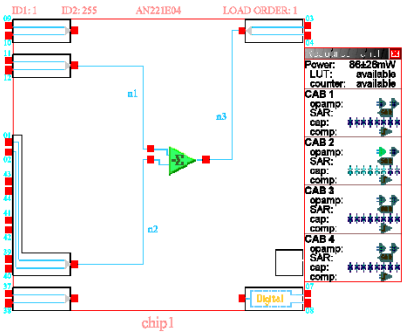

Fig. 4. Configuration of CAMs via low-level resource allocation(From: AnadigmDesigner2 GUI)

Each CAM has programmable *options* and parameters. For example, the SumDiff CAM, see Figure **??**, has a set of 4 options which decide upon clock phase, optional use of inputs 3 and 4, and inversion of each input. Its parameters are its two gains. The range of gains can be quite wide or narrow and differs from CAM to CAM.

Configuration Technology: As illustrated in Figure 2.1, the Anadigm FPAAs achieves dynamic reconfigurability of routing and CAM parameters through the control of switches. The Anadigm FPAAs are apparently unique among FPAAs in their use of 'switched capacitor' configuration technology. A switched capacitor functon implements an equivalent resistance by alternately opening and closing the inputs of a capacitor. Macroscopic resistance is controlled by the frequency of switching. This frequency, of course, is limited to the maximum clock frequency. Microscopic resistance is tuned by selecting capacitance. The capacitance of each internal capacitor in the Anadigm AN221E04 is drawn from a switched bank of capacitors. Although the software allows for the generation and routing of signals between CAMs at design time, the software only allows *dynamic* reconfiguration of the options and parameters of a circuit, not the reconfiguration of a new circuit topology. The actual configuration process and mapping of the configuration bitstream is proprietary. The configuration bitstream is stored in SRAM, which is more reliable than other FPAAs based on EEPROM technology. The disadvantage of switched capacitor technology is that it restricts operation to the discrete time domain. Because its a sample and hold system, the device can not operate at high frequencies. It also requires anti-aliasing and reconstructions filters.

Why we chose the Anadigm AN221E04: We have justified in Section 2 our bias toward human oriented circuit elements. Obviously the Anadigm AN221E04 provides the level of abstraction in building blocks that we desire. Plus, the Anadigm AN221E04 is inexpensive and easily obtained. Compared to the IsPAC10, it has more resources and flexibility. There are still additional signif-

icant advantages. Given our desire to integrate evolutionary design techniques into CAD flow, the Anadigm AN221E04 allows us to work with industry standards. Anadigm targets the device to help shorten design cycles. By intent, Anadigm FPAA designs require less tuning and tweaking in their transfer to breadboard, PCB or silicon. This intention probably drove the choice to use the same switched capacitor technology that is is industry standard for silicon. (Industry prefers switched capacitors because of the natural precision they delivers and how they replace resistors which are typically bulky in silicon.) It also likely motivated a device implementation that makes circuits insensitive to parasitics. This intent is validated by a teamed service called "Frame Freeze" which Anadigm offers. They will transfer a design to silicon to meet ASIC/AAAP level production and silicon level power usage. While we are restricted to the discrete time domain at low to medium frequency response, we nonetheless are content. An industry segment also works in this domain so there an industry target for whom evolutionary techniques may be useful exists.

CAM	CAM	CAM
Voltage Transfer Function	Inverting Differentiator	Divider
Half cycle inverting Gain Stage	Biquadratic Filter	Half Cycle Gain Stage
Half Cycle Sum/Difference Stage	DC Voltage Source	Inverting Gain Stage
Gain Stage: Switchable inputs	Bilinear Filter	Integrator
Gain Stage: Polarity Control	Half Cycle Rectifier	Half Cycle Gain Stage
Gain Stage - Output V Limiting	Inverting Sum Stage	Multiplier
Rectifier w/ Low Pass Filter	Sample & Hold	Sinewave Oscillator
Transimpedance Amplifier		Waveform Generator

Table 2. Anadigm AN221E04 CAMs

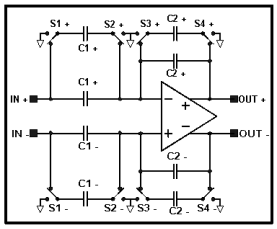

Fig. 5. Sample Anadigm CAM, Inverting Gain block illustrating 'switched capacitor' configuration

3 Configuring the FPAA with GPIC

One of the problems in choosing COTS devices is that a systems designer must rely on the product vendor for documentation and support. We purchased a Anadigm AN221E04 mounted on a development board for $299 though the advertised price today is $500. This package came with Windows based development software. We connected the development board to our desktop workstation with a serial interface. We took the risk of buying without knowing if we would be able to configure circuits from our software. A human circuit designer uses Anadigm's GUI to graphically place components and connect up a circuit before configuring the device with a simple command over the serial port. We knew the device was not 'evolutionary safe'. That is, random configurations could damage it. The straw-man alternative we considered was to ignore vendor software, mount the device on a custom PCB and interface to our reverse-engineered configuration software. This was infeasible because it would take the same effort a company like Anadigm had already invested and we had much less effort available.

We then investigated configuring the device with our own software which would drive the serial port on the development board. Ultimately, although the product specification provided us with a high-level picture of the device architecture, we were unable to determine the all important mapping of configuration bits to control the internal Configurable Analog Blocks(CABs).

This implied we pursue the option of using the software provided by the vendor. Anadigm sells the Anadigm Designer-2 EDA tool which offers a simple drag-and-drop interface to design. This software has a 'intuitive' graphical interface and links up to a built-in simulator. A C-based software API allows a programmatic approach to generating and tuning circuits. Unfortunately, this ability to put the device under 'Algorithmic Control' did not support features that were sufficient to perform evolutionary search on circuit topologies.

Finally, through a special agreement with Anadigm, we obtained a package called AutomationDoc that included software and an API description. This package had been developed to test the GUI during product development. It is not a product release. As a result of installing the package, the GUI is registered as an Active-X object. With this package and a Microsoft C++ compiler, we were able to send designs from our EA module to the Designer-2 GUI by translating them in the EA subsystem to a series of "build commands" dispatched to the GUI. A subsequent "configure" command downloads the configuration to the device. This takes about a second which is not ideal but not prohibitive either. This solution also proved to be conducive to demonstration, since the GUI provides the option of displaying a design that it is passed. We can call this option as the Evolutionary Algorithm generates a circuit and sends it to the GUI. One disadvantage of this working solution is that it required us to port our development environment over from the GNU C++ compiler to Microsoft development products. Each configuration takes about 1 second.

4 Choosing a Genetic Representation

Evolutionary search that intrinsically realizes and evaluates circuits on reconfigurable hardware is performing search in the space of all possible device configurations. However, not all evolutionary algorithms are the same. One way in which they differ is by the genetic representation they use, i.e. the genome. This distinction is very important because evolutionary crossover and mutation are applied to a genome which is only subsequently mapped to a configuration. Thus, the search trajectory across the same configuration search space differs depending on a genetic representation. The genetic representations of the evolvable hardware community have ranged from directly expressing the configuration bitstream to expressing a circuit level representation. A prominent example of the first extreme are the projects by A. Thompson ([**?**] and others who used the Xilinx 6216. Because the 6216 had no configurations that could possibly short, a bit vector genome directly encoded the configuration vector. For each configuration bit, the genome had a bit. Although devices that have this robustness in configuration have reemerged recently, their complexity and gate count make direct genomes for the configuration bitstream unwieldy because they define an intractably large search space. A genome is also a bit vector with the FPTA devices,. The genome bits, rather than expressing configuration bits express the open or closed state of the device's switches. A subset of the switches determine topoloy – the interconnection between cells. The complementary subset determines intra-cell connections in something akin to component selection since the gates affect transistor function. The bit vector is fixed in length. Its length matches the number of switches on the device.

Koza has demonstrated many equivalent genetic representations for a circuit, [**?**]. When expressing coarse grained components and their topology in the genome, he uses a program tree (often with automatically defined functions). The numerical parameters of the components are handled in an arithmetic result producing branch and searched in tandem with the topology. Koza does not use intrinsic evaluation. Because this work does not consider device instantiation, there is no limitation on how many components can be used in a circuit. The genomes are variable length .

We feel that in order for evolution to perform efficient and routine design of entire systems, an evolutionary algorithm must, like humans, be able to target multiple levels of abstraction and handle mapping between levels of representation. The genome must be able to sufficiently express abstract solutions. The genetic crossover and mutation operators must adequately and effectively search the space in coordination with selection. The genome must also be chosen keeping in mind how it determines the neighborhood relationship in the fitness landscape of the search space. In GPIC a subset of the Anadigm CAM's are chosen to comprise the primitive set (functions and terminals) in the sense of genetic programming. The GPIC genome is a cyclic graph in which each node is an instance of a CAM and directional links define the topology. The graph can vary in its number of nodes and links that connect inputs to output but we implement the graph as a fixed length vector. Each element of the vector

is a structure which specifies a CAM and points to the CAM's input source(s). Because some subgraphs in the vector may not connect inputs to outputs, they are effectively not part of the circuit though they are part of the genome. The genome is algorithmically translated into a sequence of "build" commands sent to the Anadigm GUI (see Section 3). After the translation, a "configure" command is dispatched which downloads the configuration to the device.

The encoding of coarse grained components in the genome makes GPIC reminiscent of Koza's genetic programming tree representation ([**?**] as just described. The obvious difference is the cyclic graph versus the tree. We implement crossover not as a subtree swap but as a swap of vector values. Another difference is the genome length – fixed in GPIC's case and variable in Koza's. The physical limitation of resource quantities on the device demand that GPIC not evolve a genome that requires more resources than on the device. This is ensured by the fixed length genome and by the decoding algorithm that maps the genome to a series of build commands. The decoding algorithm makes use of a resource manager to account for resources that will be used on the device as it translates the genome into "build" commands. If it ever encounters a CAM (ie. node) for which the resources cannot be allocated, it replaces this node with a wire. GPIC's genome is also influenced by Miller's Cartesian Genetic Programming (CGP), [**?**]. We consulted that description in designing ours. The CGP genome is also a graph mapped to a matrix of varying component with links between and among columns. CGP's genome is not cyclic in contrast to GPIC's.

Each instance of a CAM has a variable number of programmable options and parameters. For example, the SumDiff CAM of Figure **??** has 4 options and two gain parameters while the simple "Half cycle Gain Stage" has only 2 options and 1 gain parameter. The genome stores in each structure another two vectors of data that the genome to circuit translation process interprets as parameters and options. Each vector is a fixed length. If the parameters and attributes of the CAM are fewer than the vector length, the extra values are ignored. Like the redundant nodes and links of the circuit which do not connect input to output, this redundant information is maintained in the genome. It may have some function as memory. At this point, we have not examined its dynamics closely.

4.1 The search algorithms:

We use the standard generation based processing loop of GP and GAs to conduct topology search. At initialization, a population of random genomes is created. Each genome is mapped to a circuit topology with each instance of a CAM specified using its options and parameter values. Serially each genome is configured on the device and given a test signal. The resulting output signal is captured and evaluated in comparison to a desired output signal. The error is mapped to a genome fitness. After the entire current generation is tested, tournament selection supplies parents for the next generation. Each parent is copied to create an offspring in the next generation. Offspring are mutated before being added to the population of the next generation. Mutation can be applied in two ways

to the genome: to a CAM instance by changing its type and, to the input(s) of a CAM by changing a link in the graph. We do not mutate the options and parameters of a CAM. Instead of being evolved, they are optimized via Particle Swarm Optimization [**?**]. Pseudocode of the PSO is in Figure **??**.

Note from UM to reader: This section needs

- a figure of a circuit
- a figure of the same circuit represented as a cyclic graph
- a figure of the vector with one structure element shown
- a figure of the crossover?

4.2 Tracking Resource Allocation on the Anadigm AN221E04

Although GPIC's genetic representation uses the Anadigm CAMs , we took care to ensure that a circuit element of a genome is an implemented abstraction that hides the details of the Anadigm CAM implementations (or of any device elements in general). We want the evolutionary search and PSO algorithms of our software to be totally portable to devices with different sets of resources and interconnects. One property specific to a device that needs to be tracked is the quantity of available device resources. Tracking this is not an issue with intrinsic digital evolvable hardware. In the digital domain [**?**], a Hardware Description Language can conveniently be used to describe high-level structures and functions independent of any particular hardware platform. Translation down to an FPGA is handled by the design compiler and synthesis tools. In contrast, we have not used an analog hardware description language. To solve the problem in GPIC, we define a hardware resource abstraction that expresses the constituent elements of the circuit elements. These resources form a program tree description of the circuit, implemented in C++ as a directed graph. Because constraints are also placed on the signal routing of most programmable devices, we represent wires as resources as well. We have reverse engineered the resource allocation strategy of the Anadigm software and re-implemented its logic as a set of device specific resources that specialize the hardware resource abstraction. Abstract resource are tracked by a resource manager. Figure **??** describes this diagrammatically.

5 GPIC in Action: Evolving a Controller

We have initially used GPIC to evolve a controller for the simple 3rd order plant shown in Figure **??**. A table of the CAMs in the primitive set can be found in Table 5.2 along with their respective parameters and options. Table **??** provides the parameter settings we used for population size, swarm size etc.

5.1 Fitness Function

Though a simple step function would seem to be all that is required to evaluate a controller, we used a more complex signal to ensure that GPIC did not evolve

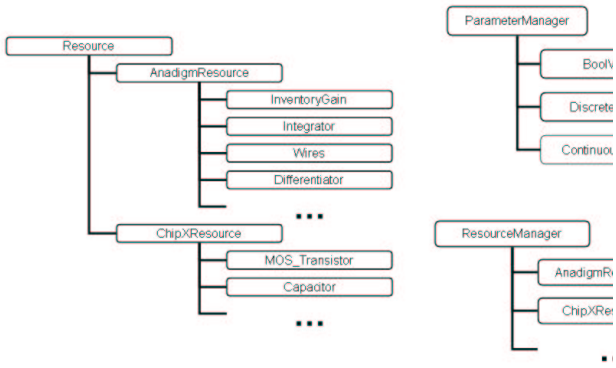

Fig. 6. Device Independence through Object-Oriented Resource Abstraction

a signal generator regardless of the input. The signal is shown in Figure 7. The fitness of a circuit is the sum of squared errors between the circuit's output signal and the test signal. We sampled the signal at XX Hz.

5.2 Evolved Solutions

In every run we found a fit individual. Here are a few examples....

CAM	Options	Parameter(s)
Inverting Gain Block		gain value
Integrator		integration constant, ref. voltage
Bilinear Filter		type: {lowpass, highpass, allpass }
Comparator		reference voltage

Table 3. CAMs used in the GPICFunction Set for Controller Evolution

Fig. 7. Test Signal for Controller Experiment GPIC

6 Summary

By combining the exploitation of coarse grained elements with intrinsic testing on a COTS device, we think GPIC comprises a distinctive approach to analog EHW. To put it in the context of extant EHW systems, Table 6 shows GPIC's characteristics using a framework offered by Torresen in [**?**]. GPIC's application domain is analog circuits, it uses a genetic programming-like representation and targets analog technology. It evolves a complete circuit design where building block functions and their interconnections are evolved. The building blocks of GPIC are at a functional level. They are analog blocks, which for a controller problem, include an adder, amplifier, differentiator and integrator. The target hardware is a COTS FPAA, the Anadigm AN221E04 and the fitness computation in hardware is online, i.e. the hardware device gets configured fore each GP population member each generation. The GP module runs off-chip on a standard Intel-based Windows workstation.

This paper's goal has been to elucidate our decision process in designing GPIC. We feel our decision to use the Anadigm AN221E04 forges GPIC's identity. We chose a COTs device over a custom device. The COTS device is much cheaper. The Anadigm AN221E04 only cost about $300. The price of the full

Application	EA	TE	AR	BB	THW	FC	EV
Analog circuits	GP	Analog	CCD	Analog Function blocks e.g. adder, differentiator	COTS FPAA of op-amps, resistors and capacitors, AN221E04	online	off-chip

Table 4. The Genetic Programming Intrinsic Circuit (GPIC) design platform summarized.EA = Evolutionary Algorithm, TE=Technology for the target hardware, AR = evolved architecture, CCD = complete circuit design, BB = Building Block, THW = Target Hardware, FC = Fitness Computation, Ev = Evolution.

SABLES system which includes FPTA-2 is $16000, [**?**]. [3] One usual hesitation with COTS hardware regards access to the configuration process so that an evolutionary algorithm can direct it. The Anadigm AN221E04 does not have an open configuration process. But, with software from the vendor, a reasonable solution was worked out: GPICuses the Anadigm design GUI as an intermediary between the evolutionary algorithm and the configuration process. Another hesitation is ensuring that the device is never configured with a circuit that will short it. GPIC sidesteps this problem with its genetic representation of circuits that ensures correctness. GPIC gets a double-check from the Anadigm GUI too, during configuration. The Anadigm AN221E04 also uses SRAM to hold a configuration which is a more robust method for an adaptive, fault tolerant system. The PAMA device, for example, does not use non-memory based configuration technology. The Anadigm AN221E04 is superior to the IsPAC10 in terms of quantity of op-amps and flexibility.

Among COTs FPAAs, we find the Anadigm AN221E04 to be superior to the IsPAC10. It has more resources and better interconnect flexibility. Importantly, the switched capacitor configuration technology of the Anadigm AN221E04 is standard with industry. It implies that evolved circuits when taken to silicon or breadboard will meet specifications within broader tolerances because there are no parasitic interconnects influencing the evolved circuits. This will facilitate the ultimate placement of evolved circuits in the field.

Finally, in using the Anadigm AN221E04, we opted for coarse grained elements. Coarse granularity makes GPIC contrast with FPTA approaches. We think that GPIC enables a parallel set of investigations that will provide interesting comparisons between the non-linear design space of the FPTA and the human oriented, conventional design space. We believe our choices additionally provide us with traction into both adaptive, robust hardware evolution and the more traditional pursuit of analog CAD.

[3] Obtaining simply the FPTA-2 device alone was also not a viable option because some licensing issues could not be resolved. This would have additionally required work to fabricate a custom-designed printed circuit board to mount the device which would have slowed us down.

Acknowledgements

We would like to thank Eduardo Torres-Jara, Dimitri Berensen, Adrian Stoica, Didier Keymeulen, Garrison Greenwood, David Hunter, and Anadigm for their contributions to the development of GPIC.