3.1 FIELD PROGRAMMABLE ANALOG ARRAY (FPAA)

Field-programmable analog arrays (FPAA) are integrated circuits with a collection of analog building blocks connected through a wire and switch fabric to achieve reconfigurability similar to the FPGAs of the digital domain. Like FPGAs, FPAAAs can help reduce the time and money costs of the integrated circuit design cycle and make analog design much easier. In recent years, several types of FPAAAs have been developed. Among these, FPAAAs that use floating-gate transistors as programming elements have shown great potential in scalability because of the simplicity they provide in configuring the chip. Existing tools for programming FPAAAs tend to be device specific and aimed at specific tasks such as filter design. To move FPAAAs to the next step, more powerful and generic placement and routing tools are necessary. This thesis presents a placement and routing tool for large-scale floating-gate-based FPAAAs. A topology independent routing resource graph (RRG) was used to model the FPAA routing topology, which enables generic description of any FPAA architecture with arbitrary connectivity including possible FPGA support in the future as well. So far, different FPAA architectures have been specified and routed successfully. The tool is already in use in classes and workshops for analog circuit and system design. Efficient ways to describe circuits and user constraints were developed to allow easy integration with other tools. Analog circuit performance was optimized by taking into account the routing parasitic effects on interconnects under various device-related constraints. Parasitic modeling allows simulation and evaluation of circuits routed on FPAA. Finally, a methodology was developed to explore the optimum architecture for a set of circuit classes by evaluating the efficiency of different architectures for each circuit class.
Electronics design processes, which are inherently iterative, must avoid the costly failures whether incurred as money or time loss. Meanwhile, IC manufacturing technology continues to improve, but the very high end processes are often accessible by only the big players. Application Specific Integrated Circuits (ASIC) also require large scale production to be feasible. Considering these factors, using reconfigurable arrays in the design process emerged as an alternative to reduce time and money costs [59]. Figure 3.1 demonstrates the potential to save money, time and energy by inserting the reconfigurable arrays into the design process. Since, Field-Programmable Gate Arrays (FPGA) has been attractive reconfigurable options for digital system design and testing. FPGAs not only give engineers the flexibility of trying their incomplete designs and bring them to perfection, but, can also hit the market early enough with a smaller investment for a limited production allowing the small players into the game as well. Today, FPGAs continue to receive interest from both engineers who use them for product prototyping and CAD researches who try to find ways to get the best out of these versatile devices. However, time to market and design costs are not the only factors that determine the success of the design process. As embedded computing becomes mainstream, a significant market has emerged for feature-rich signal processing devices that consume very little power. While digital processors and FPGAs can perform the desired functions, there are many cases where an analog design can offer the same functionality at a fraction of the power required for the digital solution [11]. For computations that don’t require very high resolutions, an analog solution is less expensive than a digital solution [73].
3.2 ANALOG VS DIGITAL IMPLEMENTATION

The resources required for the same task of multiplying two signals with 16 distinct levels is shown in Figure 3.2. A 4-bit digital multiplier requires 466 transistors, which is 66 times more than the number of transistors in a Gilbert multiplier that can perform as well for this resolution and maybe higher. In cases where a fast, approximate result is the real need rather than a very precise one, analog systems can replace digital systems with huge power and chip area savings. FPAAs are analogous to FPGAs in that they allow designers to rapidly prototype circuits without having to fabricate new ICs [22]. As FPAAs become more capable with their increased sizes and resources, the need for new CAD tools becomes more apparent. On the other hand, the methods that work for the physical design of digital circuits in FPGAs don’t work very well for FPAAs. The criteria for a successful design are different and more complex. Signal integrity of an analog circuit is more difficult to maintain and can severely impact the
functionality of the circuit, therefore, it is of higher priority than achieving the most compact design [24].

Figure 3.2 Analog solution vs digital solution for multiplying two signals at 4 bit resolution. (a) 1-bit Full Adder contains 24 transistors. (b) A 4-bit multiplier contains 466 transistors (15 Full Adders, 12 AND gates, and 5 inverters). (c) A Gilbert multiplier cell contains only 7 transistors and 2 resistors.

Increased segmentation of wires may result in additional capacitances that can result in undesired circuit behavior after routing [58]. Another challenge to be faced is the fact that parasitics not only deteriorate the performance as in digital circuits but may also destroy the functionality completely, which requires monitoring the impact of parasitics on performance metrics during the synthesis steps [26]. A placement and routing solution that satisfies all device and net
constraints may not necessarily be a desired one; the performance often has to be optimized as well.

The primary goal of this research is to make large-scale floating-gate-based FPAA technology more accessible and more practical by means of automated placement and routing tools. Our work is targeted at a particular type of large-scale FPAA that uses analog floating-gate elements for programming both routing and configuration of the components, however, it can be extended to support other FPAA technologies as well [71].

3.3 FPGA VS FPAA PHYSICAL DESIGN

Design automation, specifically in the phases consisting of clustering, placement, and routing, is often deeply influenced by the architecture and device model of the implementation medium one is working with. Although there have been general solutions proposed to handle a wide variety of architecture models, usually these design tools are not as efficient as techniques customized specifically for pre-defined architectures. This same idea applies to using FPGA design automation methods for solving FPAA applications. The similarities and the differences between FPGA and FPAA physical design flows can be viewed as depicted in Figure 3.3. A comparison between FPGA and FPAA design decisions can be approached from three fronts: the actual algorithms that are used, the metrics used in the algorithms, and the architectures used in the design. Some of the traditional FPGA clustering algorithms include multi-level clustering based on delay/area minimization to reduce the number of repeatable blocks. These algorithms are highly influenced by the device model one is dealing with and will not work without major changes for the FPAA hardware implementation.
The device and interconnect constraints in the floating-gate-based FPAA make traditional FPGA algorithms unapplicable. During the placement phase, FPGA architectures often use simulated annealing or genetic (evolutionary) algorithms to achieve an optimal solution. Although similar ideas can be applied to FPAA implementations, the cost functions in these algorithms vary\cite{57}. Typically, higher priority is placed on reducing the wire length, net density, and overall delay in placement and routing for FPGA or other digital systems. On the other hand, FPAA and analog architectures need to focus on reducing the loading effects of routing parasitics and distortion of the signals. The device architecture plays a major role in the physical synthesis decisions that are made. For instance, FPGAs traditionally have a very simple interconnect network architecture, where the vertical and horizontal channels have the same number of routing tracks and propagate the same signal down a pipeline. Each horizontal and vertical wire is segmented and is shared among several interconnects. This varies quite significantly from most FPAA, where the interconnect wires are usually not segmented and a single interconnect occupies the entire vertical/horizontal highway. The basic logic element used in FPGAs is called configurable logic block (CLB), and is a combination of programmable lookup tables with universal
functionality and flip-flops in general; whereas a configurable analog block (CAB) in a FPAA consists of many more various programmable analog components with distinct functionality. In addition, there are no sequential elements in a FPAA. Thus, behavioral as well as physical synthesis needs entirely different approaches to handle new cost functions under new types of device constraints. Thus, it is fairly difficult to directly associate automation algorithms between FPGAs and FPAA.

### 3.4 FPAA (FIELD PROGRAMMABLE ANALOG ARRAY) WITH SENSORS AND I/O INTERFACING

A Field-Programmable Analog Array (FPAA) is an integrated device containing configurable analog blocks (CAB) and interconnects between the blocks. Unlike their digital counterpart (the FPGA), the devices tend to be more application driven than general purpose as they may be current mode or voltage mode devices. For voltage mode devices, each block usually contains an operational amplifier in combination with programmable configuration of passive components. FPAAAs usually operate in one of the two modes: (i) continuous time and (ii) discrete time.

FPAAAs help in the algorithmic implementation of the analogue circuit creation policies, provides a very convenient medium in which analog circuits and systems can be designed and implemented in a very short time frame. The ultimate goal is to define a generic FPAA which would be capable of implementing almost any of the analog function. Alternately, it is more pragmatic to categorize the analog circuits into different groups and define optimum circuit primitives for each, thus leading to a family of FPAAAs. In that direction, switched current and switched capacitor based FPAAAs may be utilized for the audio frequency range,
whereas, OTA-C and current mode based FPAA's can be employed at higher frequencies. The architecture of an FPAA is shown in Figure 3.4. Field Programmable Analog Arrays (FPAA's) are devices used in applications where real time is considered the main feature. The emerging field of FPAA reprogrammable devices appears as a new challenge when dealing with real-time control.

Figure 3.4 Generic FPAA with programmable components
A typical FPAA circuit with the source and sensors interfaced to the FPAA inputs is shown in figure 3.5.

![Figure 3.5 FPAA with source and sensor interface](image)

Since FPAA integrates the idea of combining the advantages of both discrete and analogue circuits into one chip, an increased number of prototypes and applications can be met. With the increasing degree of configurability, new concepts like evolvable hardware and mixed reconfigurable analog-digital architectures also appear.
A Field Programmable Analog Array (FPAA), built in CMOS technology, contains uncommitted (i.e. functionality not yet defined) operational amplifiers, switches, and banks of programmable switched capacitors (S/C) and can be used to build filters for analog signals as well as a large number of diverse analog applications. The parameters of an application, such as a filter, are functions of the capacitor values. The chip is divided into ‘N’ identical, configurable analog blocks (CABs) (for ex: N may be 20), each composed of an operational amplifier, five capacitor banks, and switches that can be used to interconnect the cell components and determine their operation. There are both static and dynamic CMOS switches. The static switches are used to determine the configuration of cell components and inter-cell connections. These switch settings are determined once during the programming phase of an application after which they remain unchanged. The dynamic switches are associated with capacitors and are switched periodically during the circuit operation changing the effective function of capacitors as typically exploited in switched capacitor (S/C) circuits. Both static and dynamic switches are electronically controlled and thus the functionality of each CAB, the capacitor sizes, and the interconnections between CABs are programmable. As a result many diverse circuit architectures can be implemented.

The FPAA technology is suitable for numerous engineering applications like (i) electrical signal filtering, (ii) construction of controllers and phase correctors for continuous and sampled data feedback systems,(iii) conditioning of sensor signals and signal generation. The power of the FPAA is that it can be reconfigured “on the fly” to implement different device or parameter settings and that’s why the chip is as suitable for dynamic reconfiguration as it is demonstrated in the experimental investigations.
3.5. INTERCONNECTION OF FPAAS

The concept of reconfigurable hardware is uses two parallel FPAAs being switched by a multiplexer. A digital system controls the switching and provides communication to the FPAAs in control of reconfiguration. At the moment all the control is done by a PC and the software, but implementations use a FPGA. It includes digitization of the analog signal input and output to the FPGA in order to implement a digital feedback loop for comparison and control of the filter behavior. The control signal for the multiplexer is generated on the FPAA during the programming phase. So, when the FPAA starts to work after reconfiguration a high peak appears that enters into an adjustable time delay circuit on the control board and then to the multiplexer in order to interchange the analog outputs of the FPAAs. The block diagram of the system is shown in Figure 3.6.

![Figure 3.6 Block diagram of the system composed of Two parallel FPAAs](image)

3.6 FPAA DEVICES AND THE EXISTING TECHNOLOGIES

A Field-programmable analog array (FPAA) is an integrated circuit which can be configured to implement various analog functions. The most important
elements in a FPAA are the Configurable Analogue Blocks (CAB) which manipulates the signals and the interconnecting routing network. The analogue functions to be implemented are defined by a set of configuration bits loaded into an on-board shift register. The analogue blocks have parameters that can be programmed to accommodate the application. Moreover, the routing network has programmable switching facilities to connect the signals and the blocks. Each CAB can implement a number of analog signal processing functions such as amplification, integration, differentiation, addition, subtraction, multiplication, comparison, log and exponential. The interconnection network routes the signals from one CAB to another, and to and from the I/O blocks. The structure of the CAB’s components depends on the technologies used. FPAAs are designed to operate in both continuous-time and discrete-time domains. A discrete-time FPAA is designed with switched-capacitor or switched-current technology. The idea is to obtain a variable resistance using a different frequency for commutation of the interrupter. The advantages of this technology can be appreciated in terms of programmability and insensitivity to resistance in programming switches.

A continuous-time FPAA, is usually designed using transconductor technology. The basic cell consists of an op-amp and programmable capacitors linked by a transconductor-based array. These devices have advantages in terms of bandwidth, but, have narrow programming range for their parameters. It seems obvious that the ICs should integrate as many cells as possible. A very fine grain architecture is not desired, Since the numerous switches will cause a significant signal degradation. Therefore, typically 4x5 CABs matrix (which users can configure and interconnect) are used. The technologies used in building the basic cell provide a list of bandwidths for some of the FPAAs. Nowadays, commercially available FPAAs include AN10E40 (designed by Anadigm) and TRAC020 FPAA from (Zetex Semiconductors). The difference between two manufacturers mainly consists in the internal structure of the basic cell: discrete-time domain (Anadigm)
and continuous-time based technology which increases the bandwidth to 4MHz. Moreover, in the case of TRAC, the cells are placed in two parallel row configuration, the output of one cell goes to the inputs of the following two cells as well as to an external pin. In case of AN10E40, a 4x5 matrix architecture is present and a cell can interconnect with its 8 neighbors and an I/O interface.

3.7 FPAA ARCHITECTURE

FPAA’s are based on switched capacitor technology, 2x2 matrix of fully Configurable Analog Blocks (CABs), surrounded by a fabric of programmable interconnect resources. Configuration data is stored in an on-chip SRAM configuration memory and can also accommodate nonlinear functions. The FPAA features an advanced input/output structure that allows programming up to six outputs and has four configurable input/output cells and two dedicated output cells. The FPAA circuits can be programmed in many different ways such as with the help of a series E²PROM or with a connected microcontroller through the I²c line created as an inner structure.

The FPAA architecture with four CABs and Input-output interface blocks is shown in figure 3.7. The architecture has a bidirectional analog switch Fabric through which any CAB can be connected to any Input-output cell. The configuration RAM and shadow RAM are also present in the architecture.
3.8 SWITCHED CAPACITOR TECHNOLOGY

The switched capacitor technology is the technique by which an “equivalent resistance” can be implemented by alternatively switching the inputs of a capacitor. Switched capacitor is a circuit design technique for discrete time signal processing. It works by moving charges between different capacitors when switches are opened (OFF) and closed (ON). Usually, non-overlapping signals are used to control the switches, so that not all switches are ‘ON’ simultaneously.
Voltage amplification can be achieved by repeatedly switching capacitors from a parallel arrangement (with regard to the supply) to a series arrangement with regards to the load. This arrangement is called a charge pump. The simplest switched capacitor (SC) circuit is made of one capacitor and two switches which connect the capacitor with a given frequency alternately to the input and output of the SC. This simulates the behaviour of a resistor and facilitates the use of SCs are used in integrated circuits instead of resistors. This is shown in Figure 3.8. A parasitic insensitive integrator with a simple switched capacitor is shown in figure 3.9.

![Figure 3.8. Simple switched capacitor circuit](image)

![Figure 3.9 Simple Switched Capacitor (Parasitic Insensitive Integrator)](image)

### 3.8.1 Working of Switched Capacitor in FPAA

A voltage mode switched-capacitor Field Programmable Analog Array (FPAA) is used to implement various analog circuits. The FPAA consists of
uniform configurable analog blocks (CABs) allowing implementation of different functions. Each CAB consists of two back-to-back connected inverting and noninverting strays-insensitive switched-capacitor integrators. The interconnection between CABs is implemented by switched and unswitched capacitor networks. The internal structure of CABs and the interconnection between different CABs are configured by user-programmable digital control signals. The functionality of the FPAA includes embedding of different types of filters, programmable amplifiers, biquads, modulators and signal generators.

In spite of rapid advances in digital programmable gate arrays, namely field programmable gate arrays (FPGAs), there has been little improvement on their analog counterparts. There exists a significant need for an architecture of programmable analog circuit array which provides flexibility for fast prototyping and short turnaround times. Previously, three attempts have been made to solve this problem. In the first approach the programmability feature of configurable analog blocks (CABs) was achieved by user programmable switches placed into the signal paths. In order to reduce the nonlinear and distortion effects of the switches, MOS transistors were operated in the sub-threshold regions. In the second approach differential MOS transconductances were used as programmable resistors and the programmability was achieved by varying the values of the transconductances. In the third approach continuous-time current-mode circuits were used to achieve higher performance. The programmability was achieved by changing the bias currents in the trans linear bipolar topology. In this thesis, switched-capacitor (SC) techniques are employed for the following reasons:

(i) SC circuits are readily suited for programming and reconfiguration because of the existing switches which can be used for programming and reconfiguration. Switches used in SC networks must be distinguished from the switches used for programming. Programming switches either connect or disconnect additional
element to be programmed. Switches in SC circuits form an integral part of the circuit and they act as sampling elements activated by non-overlapping clock phases. They transfer charges on a particular set of capacitors to other capacitors. Even though these switches are in series connection with capacitors, they only affect the settling time, i.e. the time needed for transferred charges to reach their final values. Consequently, the programming and reconfiguration of capacitors switched by non-overlapping clock phases, can be achieved without using any additional switches. In order to program the values of unswitched capacitors, it uses the programmable capacitor array (PCA).

(ii) By altering the clock frequency, effective capacitor values can be changed without changing their physical dimensions.

Figure 3.10 illustrates an fPAA cell with OP-Amp, switched-capacitor Feedback Network and routing resources.

Figure 3.10 Each FPAA cell comprises an Op-Amp Surrounded by a Switched-Capacitor Feedback Network and Routing resources
In addition to cell circuitry, the FPAA integrates configuration logic, 8-bit programmable band gap voltage references (one per cell), and op amps, which can be used for buffering or signal conditioning of the chip's 13 I/O pins. Connections within and between the cells are configured via an RS-232 interface to a PC or an EPROM. An FPAA based on a switched-circuit topology (CMOS based) is shown in figure 3.11. Each cell consists of a switched-capacitor OP-Amp, a comparator, capacitor arrays and SRAM.

![Figure 3.11 FPAA based on a switched-capacitor circuit topology produced in CMOS.](image)

### 3.9 CONFIGURABLE ANALOG MODULE (CAM)

The analog designs that can be implemented in an FPAA are done through CAM’s (configurable analog modules). These CAMs uses the resources of CAB. The various CAMs are amplifiers, filters, comparators, multipliers, differentiators, integrators, dividers etc. These CAM’s can be connected together to form the desired analog systems. CAM’s parameters can be configured in real time to offer the capability of dynamic configuration.
3.9.1 The Functional Platform – the CAM

For programmable analog circuits, the library of virtual components takes the form of configurable analog modules (CAMs) that insulate the designer from the underlying circuit design and silicon by transforming analog design know-how into functional software elements. These elements abstract the physical implementation to a functional representation – the analog equivalent of the logic gate. Examples of such elements are filter stages, gain stages, summing/difference stages, voltage multiplication, phase/voltage comparators, rectifiers, oscillators, and references. Additional customizable functions such arbitrary waveform synthesis or arbitrary voltage transfer functions further exploit the programmable nature of the FPAA.

As a functional abstraction, the CAM in fact offers much higher complexity and flexibility than a logic gate. The main component parts of a CAM are

(i) a circuit net list or net lists,
(ii) algorithms to modify the sub-circuit according to high level parameter settings, and
(iii) a simulation model. These are shown in figure 3.12.
Figure 3.12 CAM User Interface

The design parameters like corner frequency, passband gain and Q-factor, might modify component values, or the circuit structure itself. Factors such as clock frequency, parameter settings, resource availability and performance of the silicon components (e.g. op-amp gain bandwidth product), are all known to the CAM and will influence the “operational envelope”, which is continuously re-evaluated and reported. The CAM is thus entirely self-contained and is a full representation, in software, of a physical analog signal processing function. As such, it can be seen as an analog “standard cell”, with all the associated compatibilities with AHDLs and module-based system development tools such as MatLab.
3.10. DYNAMIC PROGRAMMING OF FPAA

Dynamic programming of FPAA is possible using dynamic configuration provided by Anadigm Designer2 software. Dynamic configuration can be done through Algorithm method and State Driven method. Both these method provides ‘C’ code of the current design. These codes can be called by any application compatible and using the program the parameters of the design can be changed. The changed parameter can be communicated back to the FPAA through I/O port with the help of boot kernel. VC++ prototyping provides the user with a project environment in VC++ platform to work with both Algorithm method and State driven method codes of the current design where it is possible to manipulate the parameters of the design.

The analog subsystem within an application has normally stood alone-difficult to design, even harder to implement and outside the purview of the system microcontroller. The advent of programmable analog devices allows for the first time an abstraction in the analog design process, since the implementation of the analog system is done within a precise, drift-free and correct-by-construction silicon fabric. Not only does this technology promise a quantum leap in designer productivity and rapid development cycles, it also allows the analog functionality to be controlled by merely editing the configuration bit-stream-akin to the digital FPGAs. The SRAM-based programmable analog technology goes even further by allowing on-the-fly (or dynamic) update of the configuration bit-stream, creating in effect a dynamically reconfigurable analog subsystem. Once the various components of this technology are in place, it is easy for the system processor to update and even completely change the analog functions implemented in the programmable analog device without the need for system reset or power down.

In this research, a dynamically reprogrammable FPAA is constructed, the device is designed so that partial configuration can occur without affecting
continuity of operation of the device. Since the behavior of the analog system (implemented in a FPAA) is completely defined by the configuration bitstream, this means that the device must be capable of operating while the configuration bitstream is being updated. To facilitate this newer FPAA devices such as the Anadigm vortex family (from Anadigm) that implement dynamic reconfigurability in silicon by architecting a dual-stage, SRAM-based configuration memory structure was chosen for implementation in this research. Such a structure allows the new configuration bit-stream to be loaded into the Shadow SRAM without affecting the device operation. The data can then be transferred to the main configuration memory in a single clock cycle where it immediately impacts device operation-effecting a real-time update of the device functionality.

The deployment of dynamic reconfiguration within a design relies on "correct-by-construction" design methodology, and hence abstraction from the silicon. The latest generations of FPAAs embody this fresh approach by capturing analog designs as abstracted functional blocks. These are not a panacea for all analog, but are the most significant development in providing a programmable platform for analog functionality that can be controlled by the embedded processor.

3.10.1 Software methodology for dynamic reconfiguration

To affect static reconfiguration, design tools classically deliver passive, "pre-calculated" configuration data sets (and this is all the host needs). Dynamic configuration is a much more complex proposition. For real-time control, the software must achieve four things:

(i) It must know or calculate the new configuration data segment to be applied-this is function-dependent, where structure and
component values may be algorithmically derived from high level parameters.

(ii) It must know where to apply the data within the memory map-this is always unique to each design.

(iii) It must assemble the necessary data sequence to deliver to the configuration logic of the selected FPAA device-this may include device address information and functions such as CRC16 error-checking.

(iv) It must control the timing of the update event-a critical capability for real-time systems.

It is clear that significant information must be passed to the host processor beyond raw configuration data. Some of this information relates to structural change, some may be algorithmic (e.g. re-setting a filter's frequency response) and clearly some must relate to decisions that the FPAA tools made during the design phase about where in the device it chose to implement specific functions: so that the host processor knows which section of the memory map to change.

3.11 MEMORY UNITS IN FPAA

There are mainly two types of RAM used in the FPAA namely

(i) Shadow RAM

(ii) Configuration RAM

3.11.1 Shadow RAM

In this research, the contents of a ROM chip are copied to SRAM or DRAM to allow shorter access times (as ROM may be slower). The ROM chip is then disabled while the initialized memory locations are switched in on the same
block of addresses (often write-protected). This process, sometimes called shadowing, is fairly common in both computers and embedded systems.

### 3.11.2 Configuration RAM

Each cell in the configuration RAM has two output modes: a digital continuous output, which is provided as a continuous control signal to various peripheral circuits a selectable analog output which is used to read the information stored in the configuration RAM.

### 3.12 FPAA BASED APPLICATIONS

An evaluation of electronic filters implemented by using FPAA devices, which has the advantage of the FPAA implementation consists not only in the variety of the possible circuits which can be implemented on-chip but also the accuracy of its operation confirmed by the measurements presented in this approach. Laboratory applications have been reported in simulation, control and circuit design. The uses of the FPAA devices are chosen due to their functionality and rapid prototyping characteristics. In conclusion, the performances achieved, demonstrate that the analog devices are a viable target for control implementation. The functionality of an individual cell like: summing of input signals, multiplication of two signals (squaring of one signal or multiplication of two independent signals) made possible the implementation of various logics. One of the first FPAA developed had as target, the implementation of a neural network. The functions required to implement most neural networks are addition, threshold operation and coefficient multiplication. In order to implement these functions in CMOS technology models, these models were used in simulation of the neural network. Based on the prototype and simulation results, the FPAA concept appears to be a promising candidate for neural network applications. Analog processing can be an attractive alternative to digital processing for low-level vision algorithms in which the accuracy requirement can be low and a large
amount of data must be rapidly processed. The inherent drawback of analog processing was, until now, the lack of programmability. A prototype for a programmable mixed array processor which combines the programmability of a digital processor with the low power of an analog circuit was lacking.

A mixed signal technology as well as evolution on hardware can be considered new trends in this domain. A dynamically reconfigurable mixed signal circuit using the technology of Field Programmable Analog Array (FPAA) combined with existing well known technology of FPGAs. Their architecture takes advantage of performance and programmability of two FPAA for filtering analogue signals controlled by a digital system. FPAA can filter the analog input signal while the other is in a stand-by mode ready to be reprogrammed under the control of the digital system. An architecture suitable for research of behavior modeling of mixed signal systems and experimental verification. Reconfigurable systems based on FPGAs and FPAA offer the ability to adapt the hardware in real-time to the changing requirements of the system and its environment. The automatic reconfiguration potentially can be driven by Evolutionary Computation namely Genetic Algorithms. The circuit is divided into three parts: discrete components, analogue multiplexers and analogue bus. The discrete part contains components from low level building blocks like transistors, resistors, capacitors to higher level such as op-amp, comparators etc. Each component is connected to a certain line of the analogue bus through an analogue multiplexer. The connection is made depending on a chromosome bit generated by the computer that selects each multiplexer.

3.13. FPAA IMPLEMENTATION OF A NONLINEAR AMPLIFIER (NLA)

Nonlinear Amplifier have been intrinsically evolved on FPAA as an illustration. This could lead to a "reconfigurable structure" where all elements, from the low-level building blocks, could be programmed hierarchically. In a NLA the output is proportional to the logarithm of the input signal,
\[ v_0 = -V_T \ln \frac{v_i}{I_0 R_1} \] \hspace{1cm} \cdots (3.1).

An initial synthesis of a NLA with no gain has been based on a conventional logarithmic amplifier's design is shown in Figure 3.13. The synthesis of a Nonlinear Amplifier with fewer discrete components, no restrictions in the internal connections and with a gain of two are presented.

Figure 3.13. Basic Logarithmic Amplifier

The GA parameters used are:
- Generations 300
- Population of 400 circuits,
- Crossover rate 0.7,
- Mutation rate 0.15,
- Steady state 10 and
- Exponential normalization c parameter 0.9.

Discrete components used were:
- Two amplifiers from IC TL075 (supply voltages: +5V and -5V),
- Matched transistors from IC3086, 1 PNP transistor,
- One diode, one zener 3.3V and
- Resistors (1K, 22K, 47K, 3K3, 2K2, 330K, 100K).
The input signal is a saw-tooth waveform of 100Hz with an amplitude range of 150mV to 500mV. The fitness function used for evaluation of the configured circuits is,

\[ desired \_ output = -2 \times \ln(Vin) \] ……(3.2)
\[ fitness = | desired \_ output - real \_ output | 5 \] ……(3.3)

Figure 3.14 (a) shows the schematic diagram of the best circuit evolved. The 0.2k resistances refer to FPAAs fixed multiplexers' Ron. Figure 3.14 (b) presents the input samples and the actual and desired outputs for the best circuit. About 200 discrete measurements are made to evaluate each circuit configured. This result shows that the evolved circuit performs satisfactorily, since only a small mapping error takes place. Figure 3.14 (c) depicts the evolutionary process of this error minimization problem,
(a)
Figure 3.14. (a) Schematic diagram of the best log amplifier evolved, (b) Circuit's response, (c) Evolutionary process