Marshall Space Flight Center

A VHDL Core for Intrinsic Evolution of Discrete Time Filters with Signal Feedback

David A. Gwaltney
NASA MSFC
Kenneth Dutton
Jacobs-Sverdrup

2005 NASA/DoD Conference on Evolvable Hardware

Motivation and Background

- The motivation is to provide an autonomous system that can configure itself to enable on-line design, adaptation and self-repair.
  - Implementation of an evolution accessible discrete time computation medium in reconfigurable hardware
  - Targeted for filtering and control applications.
- Hardware evolution for the design of discrete time filters is a topic of much research
  - majority of these investigations use feed-forward structures, esp. in intrinsic work
  - In this work, the capability for also implementing feed-back structures is desired.
    - Required for Controller structures and IIR filters

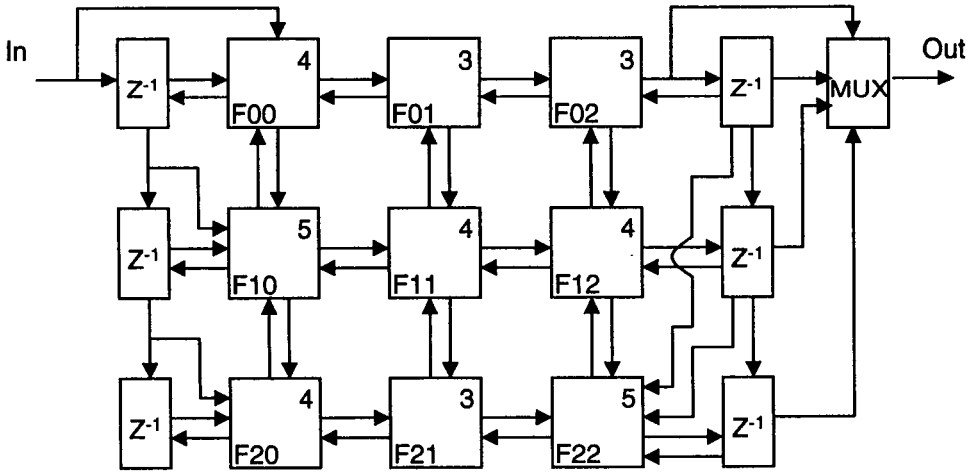
Motivation and Background

Marshall Space Flight Center

- Extrinsic
  - High order filter designs evaluated analytically. (Schnier, et. al., 2001)
  - FIR filters using primitive operator filter (POF) design in simulated device (Hounsell and Arslan,2001, Thomson and Arslan, 2002)
  - Particle Swarm Optimization (PSO) to identify the parameters of fixed IIR structures (Krusienski and Jenkins ,2004)
  - Image Filters using a simulated array of Configurable Function Blocks. Called "Cartesian Genetic Programming at the functional level."(Sekanina and Ruzicka , 2003)
- Intrinsic (FPGA)
  - Adapting the coefficients of an FIR filter on-line in response to changing input signals (Tufte, G. and Haddow 2000)
  - Evolution of image filters using Cartesian Genetic Programming (Zhang, Smith, Tyrell, 2004)
  - Evolving multipliers using boolean functions in "complete evolvable system" configured via inputs to design software. Limited to array of pipelined processing elements at time of submission (Sekanina, L. and Friedl , 2004)

Evolvable Machine VHDL Core

Marshall Space Flight Center

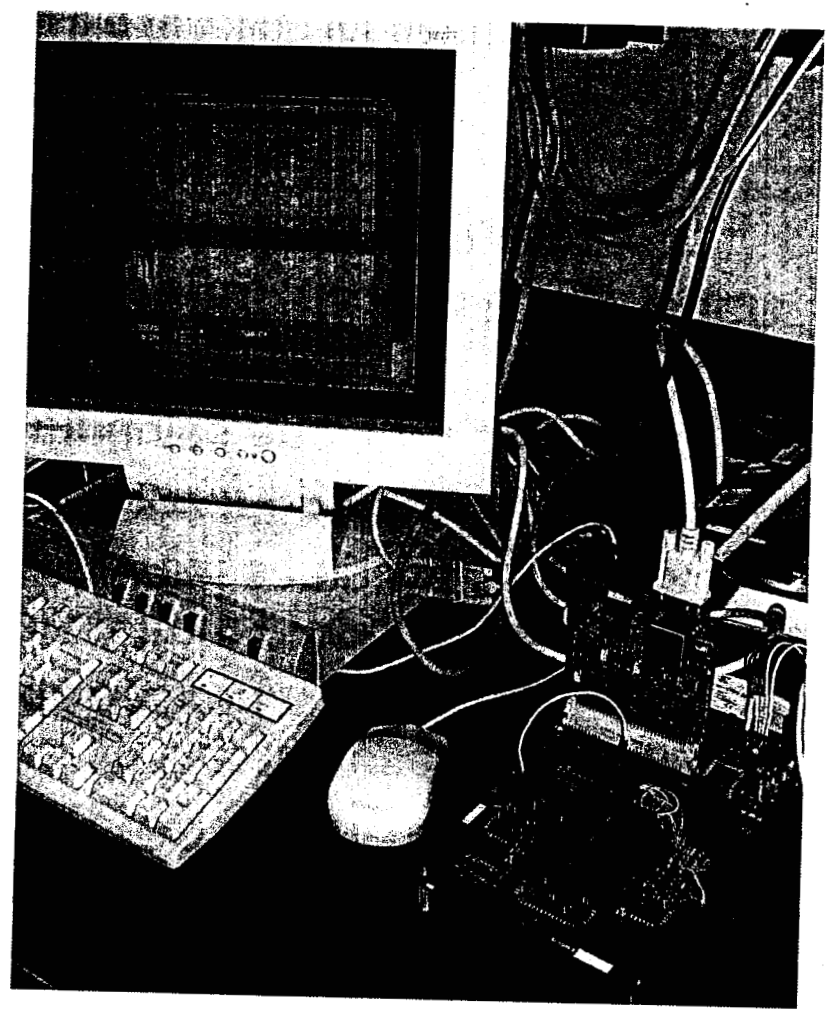


Code	Function
0	out = 0
1	out = in1
2	out = in2
3	out = in1 and in2
4	out = not(in1 and in2)
5	out = in1 or in2
6	out = not(in1 or in2)
7	out = in1 + in2
8	out = in1 - in2
9	out = in1 x in2

- Array of 3 x 3 tiles with sixteen bit inputs and registered outputs
- Tiles configured to perform different Boolean and arithmetic operations
- Tiles interconnected via multiplexed sixteen bit busses
- Input from up to two neighboring tiles, except in case of edge cells
- A tile may also use an internally stored value as a function input
- Delay registers can be cascaded for discrete time filtering or can delay output of neighboring cells
- Programmable clock divider sets data sampling rate in relation to rate of data propagation through the tile array.
- The design consumes 47% of the CLB slices available on a Xilinx Virtex XCV600E FPGA.

Evolvable System Architecture

- FPGA configured with EMVCore installed in a Xilinx AFX prototype board
  - Samples externally applied input data or data provided by DSP SBC
  - Data output to external register and internal test registers
- Custom DSP SBC executes EA
  - TMS320VC33PGE150 floating point device
  - FPGA/EMVCore configured via external memory interface
  - Generates clocks for data flow control
  - Accesses EVMCore I/O for fitness evaluation via memory interface
- PC with GUI
  - Configures EA on DSP
  - Controls execution of EA
  - Display/Records/Loads EA configuration data
  - Serial (RS-232) interface to DSP SBC



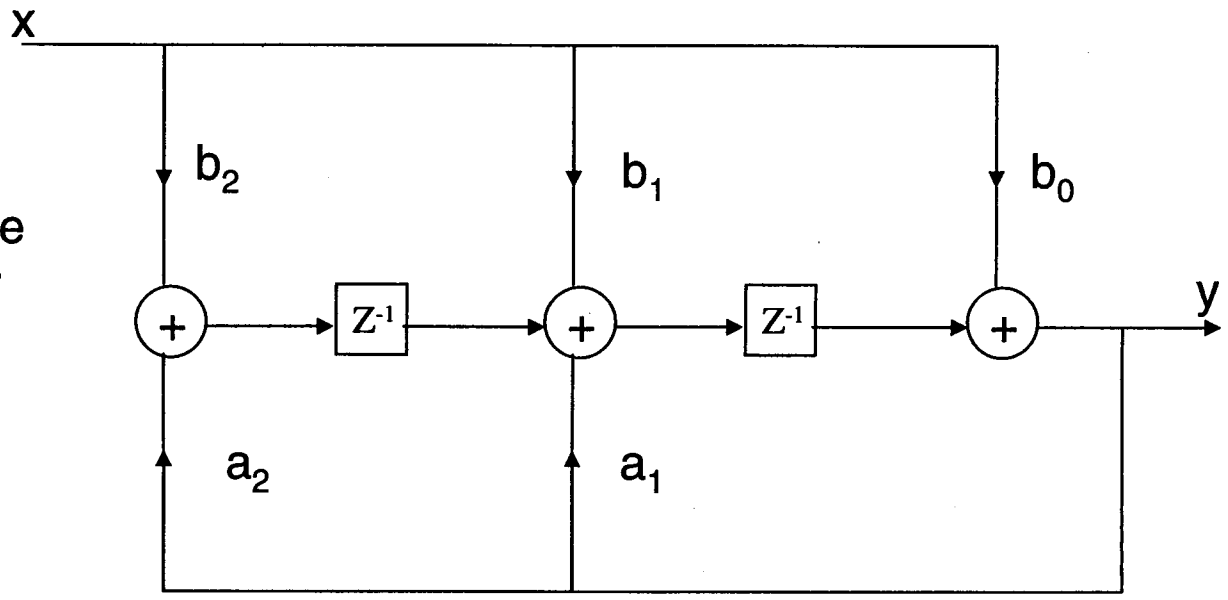
Evolvable System Hardware

Evolutionary Algorithm

- Currently GA with selection, crossover, mutation and fitness operators that are selected by the user.
  - The firmware design is modular so that the user can add, or exchange, operator modules as desired.
    - For instance, users can select from proportional, tournament or stochastic universal sampling selection.
    - A variety of mutation and crossover operators available
  - User defines fitness operator modules and then adds them to a menu for later selection
- Gene consists of a string of configuration bits for each tile containing of 9 groups of three data values
  - One tile has 3 configuration data values;
    - 4 bit value for external signal select
    - 7-bit value for selection of the tile function input and operator
    - 16-bit value for the internal data constant.

Second Order Filter Test Case

Direct form II transpose structure for 2nd order IIR filter



The general equation represented by the diagram is

$$y(k) = \sum_{i=0}^{2} b_i x(k-i) - \sum_{i=1}^{2} a_i y(k-i)$$

Coefficients selected for Lowpass with Fc = 200Hz at a sample rate of 5 KHz

Coefficient	Value
b_0	0.02651057178
b_1	0.02651057178
b_2	0.0
a_1	-1.647848081
a_2	0.7008692244

Sampling Rate and Digital Filters

- A low-pass filter designed to provide a cut-off frequency of 2 KHz at a sampling rate of 10 KHz (sampling period 0.1 milliseconds), will provide a cutoff frequency of 20 KHz at a sampling rate of 100 KHz.
- This feature is utilized in this test case in that the input data is sampled at a specific rate and the filter is designed for that sampling rate, but the filter is tested with a higher sampling rate (230 KHz) using stored data input samples supplied as input at each sample update time.
- This decreases the evaluation time during evolution

Second Order Filter Test Case

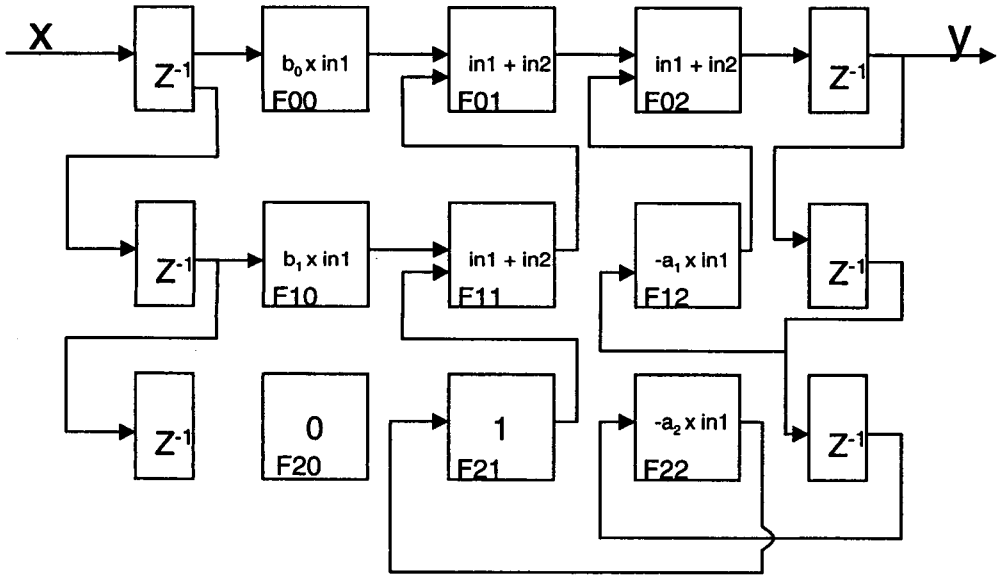
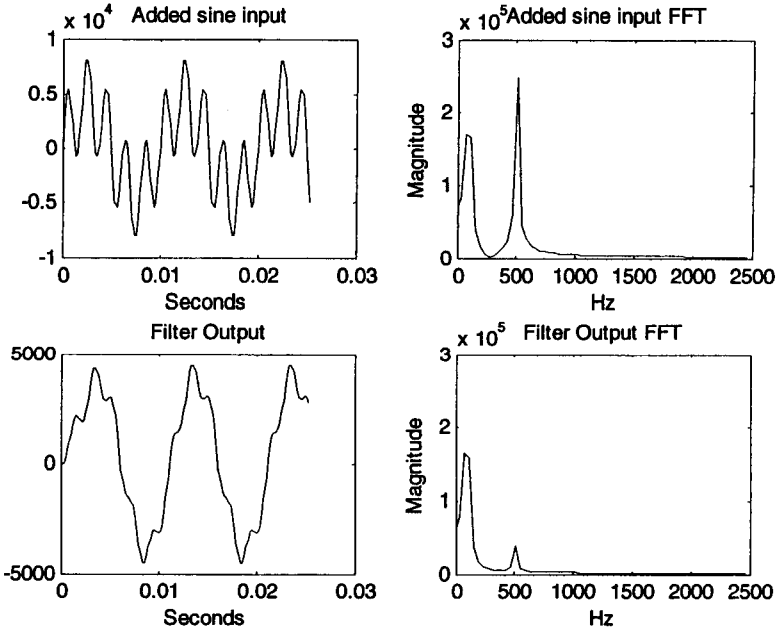


Diagram of the low pass filter implementation on the EMVCore



Time and frequency response for low pass filter implemented on the EMVCore

Input to the filter is an added sine wave with two frequencies, one at 100 Hz and the other at 500 Hz.

Evolved Filter

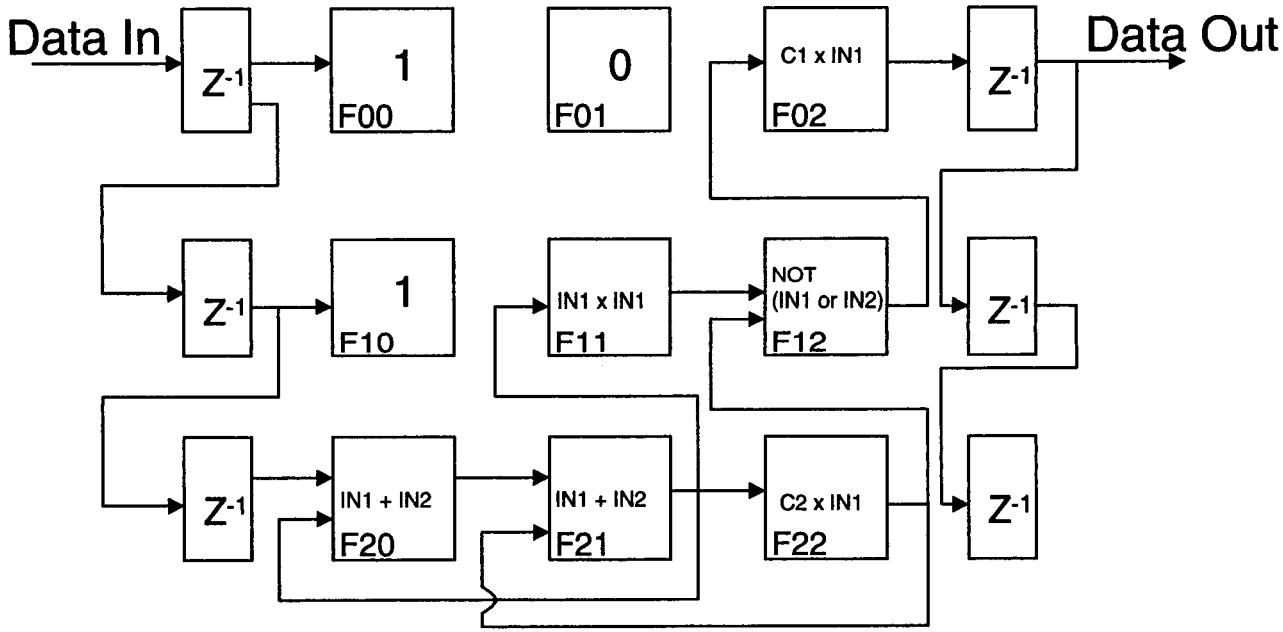
- Tournament selection with a size of 2 is used
- Two point crossover operator, with points chosen at random.
- An adaptive mutation operator is used that is based on the mutation operator found in [Zhang, Smith, Tyrell, 2004]. Here, a mutation probability is used to determine if mutation will occur. If mutation occurs, the number of bits that are mutated is governed by

$$N = c \times l \times \left(\frac{F_P}{F_A} \right)$$

Where *N* is the number of mutations, *c* is scaling factor, *l* is the limit based on the length of the chromosome, *FP* is the max fitness of the parents and *FA* is the max worst case fitness.

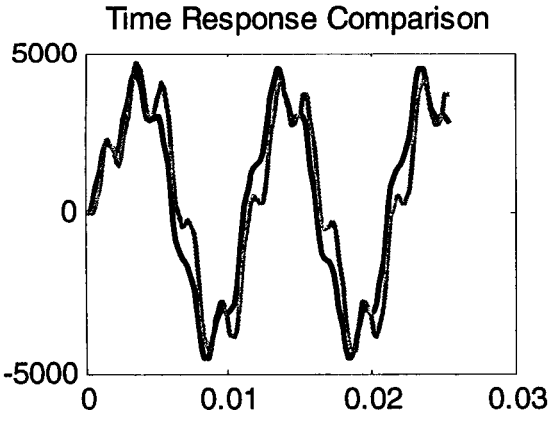
Evolved Filter

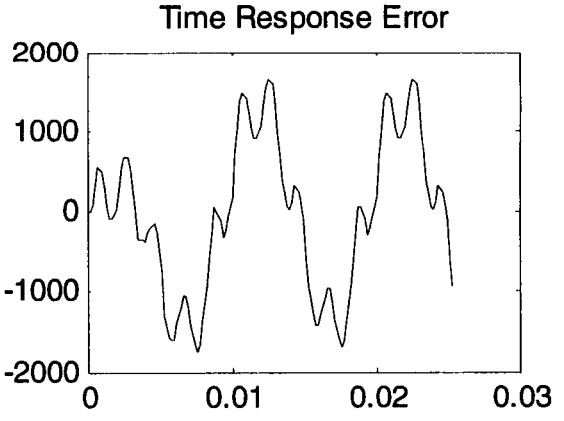
- Obtained the following filter after 543,640 generations

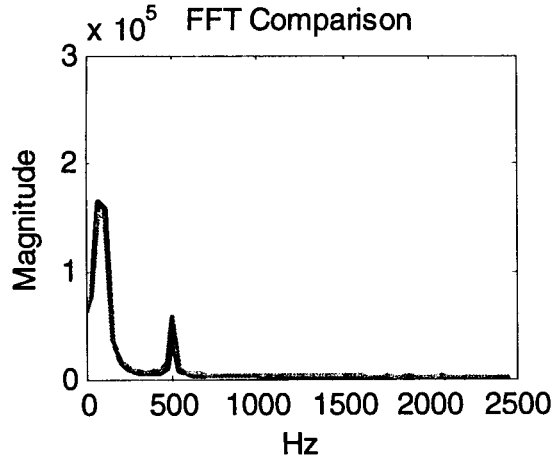


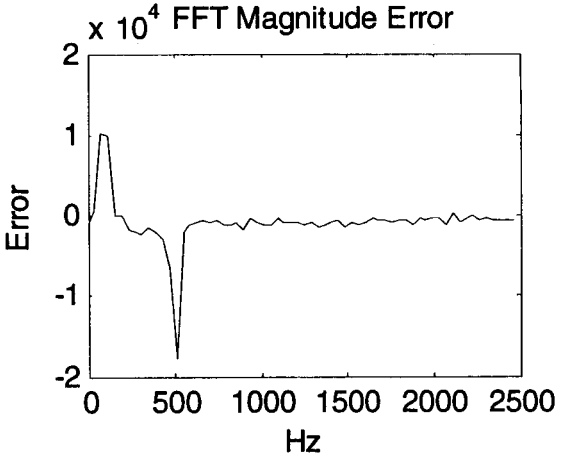
1. The evolved filter uses one of the bitwise Boolean operations in tile F12
2. The input of tile F11 is squared.
3. Two feedback loops present.
4. Does not truly require delayed sampled input or output data
5. It is a non-linear mathematical function that is dependent on the data circulating in tiles F20, F21 and F22. These tiles act somewhat like cascaded integrators, due to the registered output of the tiles.
6. Uses six tiles rather than eight and one less multiplier.

Evolved Filter









Test case – black

Evolved filter - gray

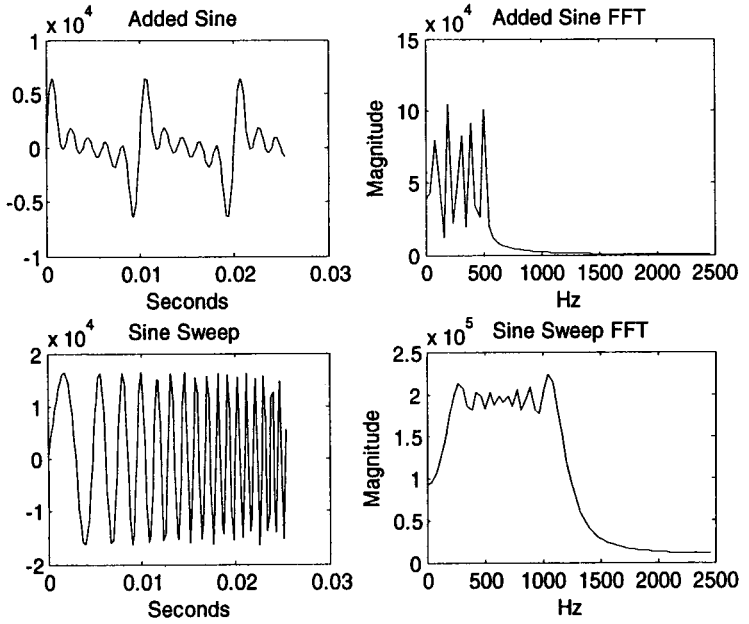
Comparison of time and frequency responses between evolved filter and test case filter.

Evolved Filter

- Placeholder for movie of evolution

Testing with Other Inputs

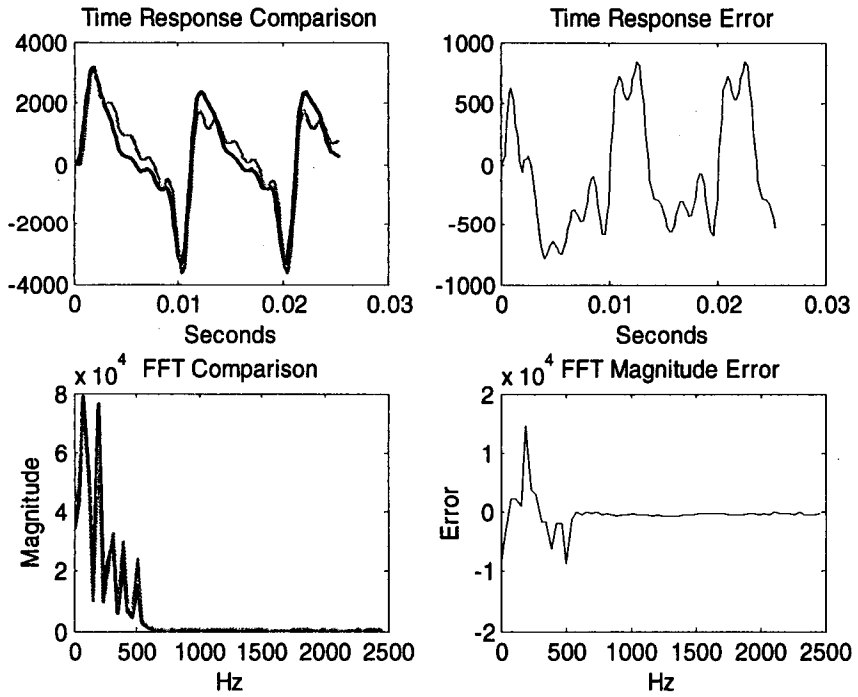
- In order to test the robustness of the evolved design, two other input signals are used.
  - Added sine, with sinusoids at 100 Hz, 200Hz, 300Hz, 400Hz and 500 Hz.
  - Sinusoidal sweep frequency content from DC to 1200 Hz.



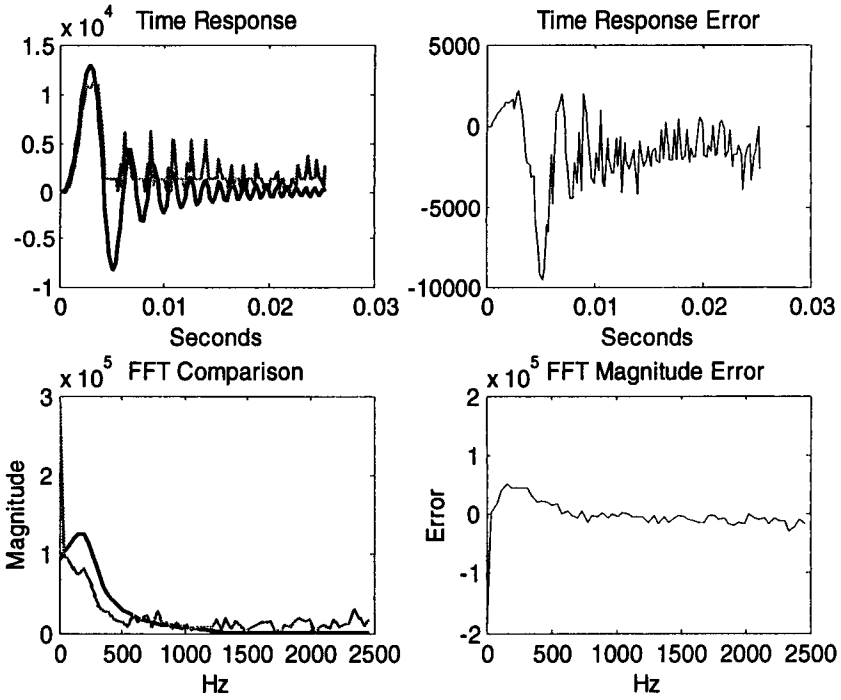
Testing with Other Inputs

Comparison of time and frequency responses between evolved filter and test case filter.

Added Sine Input



Sine Sweep Input



Test case – black
Evolved filter - gray

Summary

- Evolution of a novel low-pass filter design has been presented, along with an assessment of its capabilities.
  - performed well with the two added sines used as input during the evolution.
  - Also performs well, when the input includes more sines at frequencies between the two used during evolution.
  - Fails to perform when input is a sine sweep with wider bandwidth.
  - This illustrates the importance of designing the evolutionary process to be representative of the environment that will be seen by the evolved design during deployment.
- The use of non-standard operators and fewer resources should allow the EMVCore to implement more compact representations of digital filters and to provide fault tolerance by implementing a new solution in the remaining tiles after some are damaged.
- The EMVCore can be used to implement standard discrete time filters in addition to evolved components

Future Work

- In addition to investigating the design of innovative digital filters, future work will include
  - implementation of controllers
  - repair of damaged configurations due to faults in the electronic device.
- The EMVcore can be used to implement functions other than digital filters and controllers.
  - Mathematical functions other than difference equations.
  - Using operators other than those presented here, it is possible to implement fuzzy logic processing and neural networks in an FPGA.
- Implementation on larger FPGAs designed for applications requiring parallel computation
  - Xilinx and Altera have devices available with extensive computational resources including multipliers, adders and accumulators