Implementing Evolution of FIR-Filters Efficiently in an FPGA

Knut Arne Vinger and Jim Torresen
Department of Informatics, University of Oslo
P.O. Box 1080 Blindern N-0316 Oslo,Norway
{karnevin,jimtoer}@ifi.uio.no

Abstract

Reconfigurable hardware devices make it possible to change the topology of electronic circuits at runtime. Using reconfigurable devices as a platform for Evolvable hardware (EHW) is well suited for real-time adaptive systems. This paper contains a novel approach on how to evolve the parameters for an adaptive digital filter. Both the filter as well as the evolution is implemented in a single Field programmable gate array (FPGA). The circuit is based on context-switching in FPGA-devices and preliminary results indicate a compact hardware as well as fast adaptation.

1 Introduction

Many of today's digital signal-processing tasks are performed on real-time data. On systems that perform real-time processing of data, performance is often limited by the processing capability of the system. Providing a high processing speed is therefore often a crucial factor to be considered when implementing real-time systems. Also there is a need to have flexible systems that can be changed according to new specifications. Digital Signal Processing (DSP) systems based on software are flexible, but due to the sequential nature of microprocessors, they often suffer from insufficient processing capability. Dedicated hardware on the other hand can provide the highest processing performance, but is less flexible for changes. Reconfigurable hardware devices offer both the flexibility of computer software, and the ability to construct custom high performance computing circuits. Thus, in many cases they make a good compromise between software and hardware solutions. The flexible nature of these devices open up for a new range of circuits that exploit their reconfigurable property. A large variety of real-world applications exist for such hardware [1]. Digital filters are a very important part of DSP, and there is a present need for digital filters that can automatically adapt. There has been some work reported earlier like extrinsic evolution of filters [2] and adaptive filter evolution [3].

This paper shows how a Xilinx Virtex XCV1000 FPGA (a reconfigurable device) can be used to implement a real-time adaptive filter using an approach based on evolvable hardware. The implementation presented is a full on-chip solution where the evolution is performed in a novel way by logic controlled by user defined configuration registers. A set of different circuit configurations are stored on-chip — to represent the population used by evolution. To provide fast context switching among these, a novel architecture is proposed where switching is possible in *one* clock cycle. This is based on earlier work of implementing a *user defined FPGA* inside an ordinary FPGA [4,5] However, it is here extended to include a number of user defined configuration bit-streams used as the population for an hardware-implemented evolutionary algorithm. The next two sections describe evolution of filters and FPGA technology, respectively. Then, section 4 describes how this can be implemented in reconfigurable hardware. Finally, section 5 discusses the preliminary results and draws some conclusions.

2 Digital filters

Digital filters are the digital counterpart to analog filters. Digital filters operate on numbers opposed to analog filters which operates on voltages. The basic operation of a digital filter is to take a sequence of input numbers (e.g. samples), and compute a different sequence of output numbers. There exists a range of different digital filters, where the most common is the finite impulse response filter (FIR). Digital filters are used in a wide range of DSP-applications.

2.1 Adaptive filters

Adaptive filters can be utilized in a variety of applications, both on stored data and on real-time processing tasks. Areas for real-time adaptive filter utilization can be systems of noise-canceling, filters that automatically adapt to changing environment, and identification of unknown systems where the unknown system can be modeled as a digital filter. Adaptability is accomplished by calculating a score for the results obtained by the system on a set of data. This score is then used to manipulate the system in such a way that the system output will converge towards a desired result. Using evolutionary techniques on a digital filter-system is a possible method for obtaining system-adaptability. Normally, the adaptation will have to take place on a secondary filter that is not at the same time operating. However, if the filter is not continuously active, a time sharing between operation and adaptation would be possible.

2.2 FIR-filter

A FIR-filter (Finite input response filter) is a digital filter that is widely used in digital signal processing applications. The FIR-filter computes an output from a set of input-samples. The set of input samples is multiplied by a set of coefficients and then added together to produce the output — see Fig. 1. The filter behavior is determined by the filter-coefficients. A general FIR filter is described in the following equation:

$$Y(n)=k_1x(n) + k_2x(n-1) + k_3x(n-2) \dots k_mx(n-m)$$

Where k_i is the coefficient i. x is the input signal and y is the output signal. m is the number of filter coefficients — called taps, and n is the input sample number.

Implementation of FIR-filters can be undertaken in either hardware or software. A software implementation will require sequential execution of the filter-functions. Hardware implementation of FIR-filters allows the filter-functions to be executed in a parallel manner which makes improved filter processing speed possible.

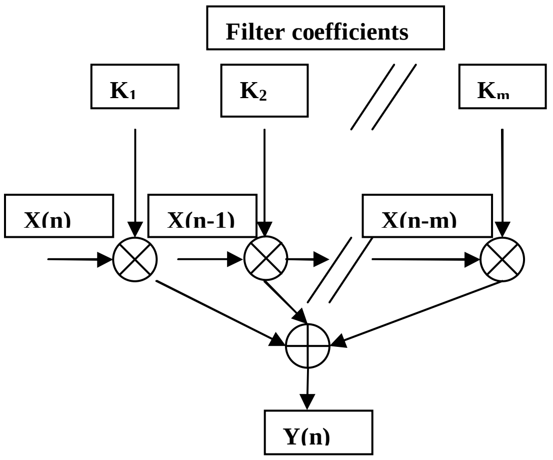

Fig. 1. FIR-filter

3 Virtex FPGA

The Xilinx Virtex XCV1000-6 device is to be applied in this work. The device contains a 64 by 96 array of Configurable Logic Blocks (CLBs) interconnected through a general routing matrix. Each CLB contains a routing matrix and two slices. Each slice consists of two look-up tables (LUTs) and two flip-flops (together with some control logic). The combinational logic is implemented in look-up tables. However, the look-up tables can also be configured as 16-bit shift-registers which is useful to us.

4 Implementation

This section describes the proposed circuit that applies LUTs in shift-register mode to implement context switching. The architecture is designed to include evolutionary operators to make an adaptive real-time FIR-filter. The filter-coefficients of a FIR-filter are determined by evolution, so that the FIR-filter is able to minimize the difference between the filtered signal (given by the evolved filter) and the reference-signal. A fitness-value is calculated for each set of filter-coefficients for each sample. After a fitness value is computed, evolutionary operators are applied to manipulate the corresponding set of filter-coefficients. When the difference between the filtered signal and the reference is minimized, the FIR-filter — determined by the filter-coefficients, resembles the unknown system.

The whole system is implemented on a Virtex XCV1000 FPGA device. On adaptive systems (real-time systems in particular) a limiting factor for the overall system performance is often the speed of which the system is able to adapt to perform a certain task. The amount of time a system uses to adapt is determined by the complexity of the task it is adapting to perform, and the efficiency of the adaptation scheme used. If we consider that the complexity of the task to be performed is fixed, optimization of the adaptation scheme is necessary to decrease the adaptation time of the system. Using an evolutionary approach for the adaptation part of an adaptive filter-system gives the following tasks to be performed:

- Compute a result with the set of current filter-coefficients on the incoming data
- Compute the fitness-values
- Perform evolutionary operators on the filter-coefficients based on the current fitness-values

4.1 Context switching on FPGA

To be able to undertake rapid switching between a set of different circuits, a novel architecture has been developed [6]. A *user defined FPGA* is here defined inside an ordinary FPGA. However, it is extended from an ordinary FPGA to include *a number of* user defined configuration bit-streams. Thus, in this scheme the normal FPGA Configuration Register is *fixed* at runtime. The design consists of a set of User Defined Configuration Registers (UDCRs) — each storing one configuration bit-string. Only *one* is active at a time, and switching between them is possible (by the scheme presented below) in one clock cycle. Each UDCR contains *one* set of filter coefficients. The drawback of this approach is that a certain amount of logic is required for storing and controlling multiple filter configurations. Thus, the amount of logic left as reconfigurable logic for the filter itself will be limited compared to the original amount of logic available. However, the number of logic blocks in the Virtex is large and each block can be configured in a set of ways.

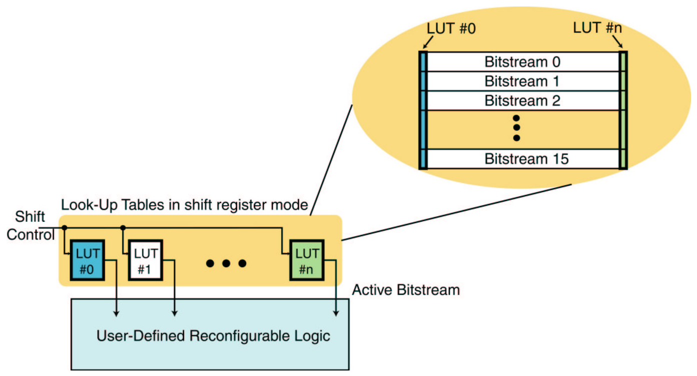

Fig. 2. Virtex LUTs used to store a set of configurations

The proposed system — see Fig. 2, applies a set of LUTs as shift registers to store a *set* of user defined configuration bit-streams. Each stream represents one set of filter coefficients. As illustrated in the figure, each LUT stores *one bit* of each bit-stream and the maximum number of bit-streams is 16. The shift control is common for all the LUTs. Thus, when context switching between two bit-streams is to be undertaken, the LUTs shift one position and the next bit-stream is presented on the LUT outputs. Only a single clock cycle is needed. The LUT outputs define the values of the filter-coefficients.

4.2 Storage of filter-coefficients

To be able to change the filter-coefficients in one clock-cycle, Virtex LUTs set to operate in shift-register mode are used to design user-defined configuration registers (UDCR) for storage of the filter-coefficients. Each output bit of the UDCRs is implemented by a pair of two LUTs where one LUT is configured as a 16-bit shift register (SRLUT) and the other LUT is utilized as control logic as depicted in Fig. 3. By making a feedback it is possible to use the LUT as a strobe for continuously repeating the same 16 bit sequence. This is very useful to quickly evaluate 16 different sets of coefficients. To be able to control and manipulate the bit sequence, a second LUT is applied in the feedback loop. The *control LUT* is configured to perform one of three tasks: enable loading of new data (i.e. configuration), passing the feedback directly through and back to the shift-register or inverting the currently stored bit (to be used for mutation).

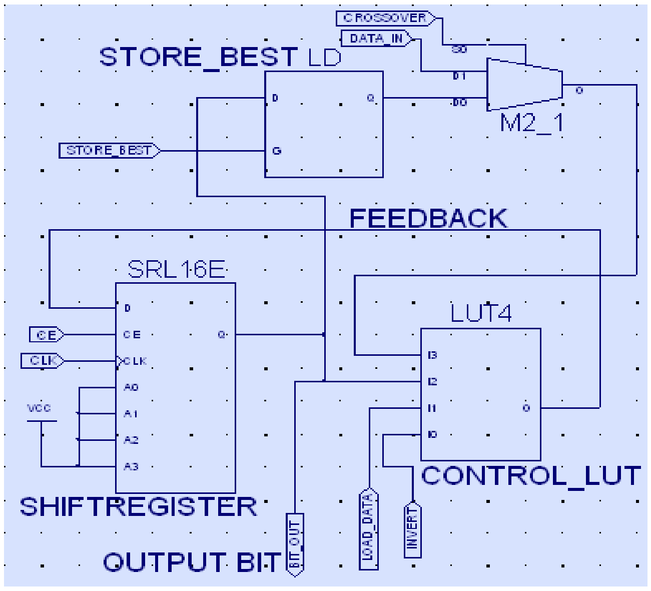

Fig. 3. Configuration storage unit with crossover

The UDCRs both reduce the logic necessary to store data, and decreases the time necessary to change the filter-coefficients to one clock-cycle. This is possible because the UDCRs are connected directly to the full-multipliers in the FIR-filter. In addition to making the fast reconfiguring of the filter-coefficients possible, the UDCRs are well suited to integrate evolutionary operators. Both mutation and crossover has been implemented with very little additional logic as outlined below. The evolutionary operators that are integrated in the UDCRs are also performed in one clock-cycle.

4.3 Computing the fitness-value

Each time an input sample is loaded, a fitness-value is calculated for each set of filter-coefficients. Switching between filter-coefficients takes less time than loading new samples. This enables computation of a fitness-value for each set of filter-coefficients every time a sample is loaded. The fitness value for a set of filter-coefficients is simply the difference between the output of the FIR-filter and a reference signal. When this difference is small, the FIR-filter has efficiently adapted to remove the difference between the input-signal to the FIR-filter and the reference signal. To compensate for the phase shift of the filter, the reference signal is delayed by the same amount of clock-cycles as the length of the FIR-filter. It is expected that the evolved filters will have a linear phase-shift.

4.4 Performing evolutionary operators

The UDCRs have been modified to integrate the evolutionary operators, mutation and crossover. The mutation operator is an extended variant of the ordinary totally random mutation. Each set of bits which makes up *one* filter-coefficient is connected to a mutation-signal that is calculated from the current fitness-value. The mutation is based on inverting *one* bit in one randomly chosen coefficient. A low fitness-value will result in a more significant bit being mutated than for a high fitness-value. This is done to increase the efficiency of the mutation operator.

The crossover operator is also a slightly modified version of the normal crossover operation. The circuit has a crossover-control unit that holds the current population's best and the worst fitness-value. Every time a new fitness-value is calculated, the control unit compares the new fitness-value to the ones it has stored. In the case that the new fitness-value is better than the current best, it stores the new value as the best, and gives a signal to the UDCR to store the current filter-coefficient. The UDCRs have been added one D-latch for each output bit to store one set of filter-coefficients *separate* from the ones stored in the shift-register (shown in Fig. 3). These store the *current best* filter-coefficients. If the new fitness-value is worse or equal to the current worst, the control-unit stores this value as the new worst, and sends a signal to the UDCR to do a crossover with the filter-coefficients stored in the separate D-latches. The crossover randomly writes the even or odd numbered *half* of the current best coefficients over the current worst filter-configuration. The crossover operation would probably have been more efficient with doing a crossover where the two best configurations totally replaced the worst. The modified version of crossover was chosen to fit better to the architecture of the Virtex FPGA.

5 Preliminary results and conclusions

The choice of adaptation method to use in adaptive systems is usually a trade-off between area efficiency and adaptation time. With the use of UDCRs it has been possible to implement a genetic algorithms-style adaptation technique with a minimum of extra logic. Utilizing full multipliers in the FIR-filter enables the computation of a large amount of fitness-data, it is expected that this will result in very fast adaptation time for the filter.

An 8-tap 12-bit filter has been implemented and simulated with the Xilinx ISE 4.2 design package. The target hardware is a Celoxica RC-1000 board equipped with Xilinx Virtex XCV1000 FPGA. Approximately 8 percent of the resources available on the FPGA were used, hence larger filters can be implemented. Current timing results indicate a clock speed at around 60 MHz.

References

[2] J.F. Miller, "Evolution of Digital Filters using a Gate Array Model", Proceedings of the First EvoIASP'99 Workshop on Image Analysis and Signal Processing, R. Poli , H.M. Voigt, S. Cagnoni, D. Corne, G.D. Smith, eds., Goteborg, Sweden, 26-29 May 1999. Lecture Notes in Computer Science Vol. 1596, pp. 17-30, 1999.

[5] L. Sekanina, and R. Ruzicka, "Design of the special fast reconfigurable chip using common FPGA". In Proc. of Design and Diagnostics of Electronic Circuits and Systems - IEEE DDECS'2000, pages 161-168.

[4] J. Torresen, "A divide-and-conquer approach to evolvable hardware", In Sipper, M. et al., editors, Evolvable Systems: From Biology to Hardware. Second Int. Conf., ICES 98, pages 57--65. Springer-Verlag. Lecture Notes in Computer Science, vol. 1478.

[6] J. Torresen and K. A. Vinger, "High Performance Computing by Context Switching Reconfigurable Logic", In proc. of 16th European Simulation Multiconference (ESM-2002), pp. 207-210, June 2002, Darmstadt, Germany.

[3] G. Tufte and P. Haddow, "Evolving an Adaptive Digital Filter" EH'00, pp 143-150.

[1] J. Villasenor and W. Mangione-Smith, "Configurable computing",
Scientific American, (6), 1997.