Design and Implementation of High-Performance FPGA Signal Processing Datapaths for Software Defined Radios

Chris H. Dick
Xilinx Inc.
2100 Logic Drive
San Jose
CA 95124

Henrik M. Pedersen
Center for Fast Ultrasound Imaging,
Dept. of Information Technology, Techn. Univ. Denmark

1 Introduction

The communications infrastructure that has become so much a part of daily life is expanding at an exponential rate. Figure 1 illustrates the diverse range of communication technologies used virtually on a daily basis: wireless cellular (high and low mobility users), satellite, and microwave links. To meet consumer, business and life-style demands infrastructure suppliers must build sophisticated systems that no longer simply support telephony services, but provide voice, high bit-rate data, video, image and multimedia capability. These systems must also interact with sophisticated systems like the internet. Human nature dictates that there will be a range of communication standards that evolve. There will also be a range of user terminals that need to be connected to this rich communications tapestry, including cell phones, video phones, satellite phones, PDAs, portable computers and other nomadic computing devices. To flourish and succeed in this dynamic environment equipment suppliers must build highly flexible systems that operate across multiple wireless and wired network standards. They must be able to rapidly adopt new business models as they evolve, and they must be able to incorporate new signal processing techniques that allow increased network capacity, increased coverage, increased quality of service, or a combination of the above. The answer to the diverse range of requirements is the *software defined radio*.

Software defined radios (SDR) are highly configurable hardware platforms that provide the technology for realizing the rapidly expanding third (and future) generation digital wireless communication infrastructure. Many sophisticated signal processing tasks are performed in a SDR, including advanced compression algorithms, power control, channel estimation, equalization, forward error control, adaptive antennas, rake processing in a WCDMA (wideband code division multiple access) system and protocol management. While there are a plethora of silicon alternatives available for implementing the various functions in a SDR, field programmable gate arrays (FPGAs) are an attractive option for many of these tasks for reasons of performance, power consumption and configurability. This tutorial will describe how many of the functions required in a software radio system can be realized in an FPGA. Topics discussed will include adaptive channel equalizers and channelization functions. Amongst the more arithmetically demanding tasks performed in a high data rate wireless system is channel equalization. We describe and examine the FPGA mechanization of equalizers for QAM (quadrature amplitude modulation) systems. The implementation of the equalizer using state-of-the-art FPGA semiconductor technology, that provides a high-performance multiplier fabric, is discussed. The design and simulation of equalizers that employ fixed-point arithmetic is described.

Figure 1: Future generation communication environments will need to support a multitude of modes of operation and air interfaces.

2 FPGA Architecture

FPGAs have experienced extensive architectural innovations in the past several years. Advanced process technology has enabled the development of high density[1] devices that are extremely well suited to the needs to high-performance real-time signal processing. The architecture of the Xilinx Virtex™-II is shown in Figure 2. The device is organized as an array of logic elements and programmable routing resources used to provide the connectivity between the logic elements, FPGA I/O pins and other resources such as on-chip memory, delay lock loops and embedded hardware multipliers

Figure 2: Virtex-II FPGA architecture. This FPGA family provides an array of 18x18-bit precision multipliers for addressing advanced sign al processing applications.

[1] State-of-the-art FPGAs like the Xilinx Virtex family provide devices with approximately quarter of a billion transistors and in excess of 3 million system gates.

The FPGA resources of particular interest to the signal processing engineer are configurable dual-port block memories, distributed memory and the multiplier array [10]. The multiplier array is composed of 18x18-bit precision mutlipliers that can operate in combinatorial mode (140 MHz) or they can be pipelined (1-stage) to support clock frequencies up to 250 MHz. The smallest Virtex-II device provides a modest 4 multipliers while the largest supplies an impressive 192 multipliers.

3 Software Radios

The ever-increasing demand for mobile and portable communication requires high-performance systems employing advanced signal processing techniques to allow operation as close as possible to the Shannon information theoretic bound [13]. However, not only must these systems provide exceptional performance, but due to market and fiscal pressures, they must be flexible enough to allow the rapid tracking of evolving and fluid standards. Software defined radios are emerging as a viable solution for meeting the conflicting demands in this arena. SDRs support multimode and multiband modes of operation to allow service providers an economic means of future-proofing these increasingly complex and costly systems.

The use of the term *software* may give the impression that the radio is realized entirely on a processor-based platform. This is not the case. The essence of the SDR is flexibility. The flexibility to support multiple air-interfaces and to have the provision to easily and rapidly change the nature of the signal processing chain that is the kernel technology in these systems. DSP microprocessors, even with advanced architectural extensions (very long instruction word (VLIW), super-scalar, etc.) do not satisfy the arithmetic or I/O requirements of a modern communication signal processing engine. Advanced field programmable gate array technology offers a solution. FPGA-based signal processors provide high-performance, while at the same time maintaining flexibility through static RAM configurability [10]. A receive subsystem of a concept FPGA-based base transceiver station (BTS) is shown in Figure 3. The figure also shows various feedback loops for providing digital gain control in the digital down converters (DDCs) in addition to a digitally controlled AGC (automatic gain control) loop. Typically this will be a low-bandwidth loop which allows the application of novel sigma-delta modulation techniques [1] for efficiently generating analog signals using FPGAs without the requirement of a digital-to-analog converter (DAC).

Figure 3: SDR BTS employing FPGA reconfigurable DSP – receiver.

There are many advanced signal processing tasks performed in a modern digital receiver. Figure 3 illustrates a system consisting of two sub-systems – a front-end high-data rate (100-200 MHz) processor and a back-end symbol rate (or chip-rate in the context of WCDMA) processing fabric. The front-end high-data rate FPGA DSP implements channelization functions for a multi-carrier system. Each channelizer accesses the digital IF (intermediate frequency), translates a channel (e.g. a 5 MHz wide spectral segment for WCDMA) to baseband and using a multi-stage multi-rate filter adjusts the sample rate to satisfy Nyquist for the selected band. The back-end processor will typically operate on multiple slower rate sample streams performing functions like rake processing, adaptive rake processing, demodulation, turbo decoding, Viterbi decoding. In a QAM system, carrier recovery, timing recovery and adaptive channel equalization will be required.

The SDR BTS transmitter in Figure 4 implements multiple channels of digital up-conversion, modulation, forward error control, adaptive pre-distortion for high-power amplifier linearization and beam forming for smart antenna arrays.

Figure 4: SDR BTS employing FPGA reconfigurable DSP – transmitter.

4 DSP Functions in a Digital Receiver

In this section we consider the FPGA implementation of two functions commonly found in many receiver architectures: adaptive channel equalizers and digital down conversion.

4.1 Channel Equalizers

Most modern bandwidh efficient communication systems use quadrature amplitude modulation. The input data stream is partitioned into sets of *N* bits. These bits are used to select one of 2^N possible waveforms that are both amplitude and phase modulated. The waveforms are each directed to the channel, each with the same shape and bandwidth.

Figure 5: Equalized data demodulator.

Figure 5 shows an equalized receiver. Adaptive equalizers operate in a receiver to minimize intersymbol interference (ISI), due to channel-induced distortion, of the received signal. The equalizer operates in cascade with a matched filter (MF), synchronous sampler, and decision device (slicer) operating at the symbol rate. A gradient descent process such as the least-mean square (LMS) algorithm adjusts the equalizer weights to minimize the difference between the input and output of the decision device. In modern receivers the sampling process precedes the matched filter, and in order to satisfy the Nyquist criterion for the matched filter, the sample rate is greater than the symbol rate by a ratio of small integers p-to-q such as 3-to-2 or 4-to-3 and often is 2-to-1 to simplify the subsequent task of down sampling prior to the slicer. If the down sampling occurs prior to the equalizer, the equalizer operates at 1-sample per symbol and it is termed a symbol equalizer, and if the down sampling occurs after the equalizer, the equalizer operates on p/q-samples per symbol and it is termed a fractionally-spaced equalizer (FSE).

There are many excellent references in the open literature that deal with the design and development of various types of equalizers. An excellent introductory review of fractionally-spaced equalizers can be found in 0 while constant modulus algorithm based blind equalizers are extensively covered in [3] and [4]. These references also contain excellent bibliographies.
There are many options for realizing adaptive filters, including FIR, IIR lattice and transform domain architectures. There are also a large number of algorithmic choices for implementing the coefficient update process – least-mean-square, fast Kalman, conventional Kalman, square-root Kalman and recursive-least-square (RLS) lattice to name a few. One of the most successful, because of its simplicity and excellent behavior under finite arithmetic conditions, is the least-mean-square algorithm [5] [6]. A brief review of the procedure, based on the derivation in [6], follows.

The basic adaptive FIR structure is shown in Figure 6. The error signal e_k can be expressed as

$$e_k = d_k - \mathbf{X}_k^{\mathrm{T}} \mathbf{W}_k \tag{1}$$

where $\mathbf{X}_k$ is the regressor vector and the $\mathbf{W}_k$ are the filter coefficients. In the LMS algorithm $e^2{}_k$ is used as an estimate of the gradient on the performance surface [6].

Figure 6: Adaptive transversal filter.

At each iteration in the adaptive process we have a gradient estimate of the form

$$\hat{\nabla}_k = \begin{Bmatrix} \dfrac{\partial e_k^2}{\partial w_0} \\ \vdots \\ \dfrac{\partial e_k^2}{\partial w_L} \end{Bmatrix} = 2e_k \begin{Bmatrix} \dfrac{\partial e_k}{\partial w_0} \\ \vdots \\ \dfrac{\partial e_k}{\partial w_L} \end{Bmatrix} = -2e_k \mathbf{X}_k \tag{2}$$

With this simple estimate of the gradient we can specify a steepest descent adaptive algorithm that is described by the following equations

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k - m\hat{\nabla}_k \\ &= \mathbf{W}_k - 2me_k \mathbf{X}_k \end{aligned} \tag{3}$$

Figures 7, 8 and 9 show a decision directed FSE, decision feedback and constant modulus algorithm (CMA) blind equalizer respectively. Each equalizer shows the hardware realization of the LMS update algorithm described in Eq. (3).

Figure 7: Decision directed equalizer.

Figure 8: Decision feedback equalizer.

Figure 9: Blind CMA equalizer.

A fractionally-spaced equalizer was designed for a Virtex-II FPGA by starting with Matlab [14] floating-point m-file simulations. This design was migrated to a quantized system using the fixed-point blockset in Simulink. The Matlab simulations provided an environment to quickly compare the performance of symbol-spaced, fractionally spaced and blind equalizers. Various channel models can easily be simulated and monte-carlo simulations performed to evaluate equalizer performance. Figure 11 and Figure 12 illustrate the Matlab floating-point simulations. A 16-QAM uniformly distributed data stream was applied to the decaying exponential channel model shown in Figure 10.

Figure 10: Test channel impulse response.

Figure 11(a), (b), (c), (d), (e) and (f) show the modulator constellation diagram, the real component of the input data overlaid with the 1-to-4 upsampled and shaped modulation waveform, the modulation eye diagram, channel output time series. The ISI introduced by the channel is clearly observable in Figure 11(d).

The ISI data was applied to a 40-tap FSE. Figure 12(a), (b), (c) and (d) show the equalizer instantaneous and average error, equalized constellation, equalized constellation with the startup transient removed and the equalized eye diagram respectively.

Once a floating-point prototype of the FSE was developed and verified in Matlab, the Simlulink [15] fixed-point blockset was employed to quantize the design. Figure 13 shows a 24-tap filter specified in Simulink while Figure 14 is a single filter tap and LMS update processor.

Figure 11: (a) Modulation constellation. (b) Input time-series overlaid with shaping filter (root-raised-cosine) output (c) Modulator eye diagram. (d) Channel output time-series with intersymbol interference (ISI). (e) ISI constellation. (f) ISI eye diagram.

Figure 12: (a) LMS instantaneous error and average error. (b) Equalized constellation. (e) Equalized constellation – startup transient removed. (f) Equalized eye diagram.

Figure 13: Fixed-point FSE showing three 8-tap complex FIR and LMS building blocks, the output slicer and error signal computation block.

Figure 14: Simulink fixed-point blockset implementation of one FSE complex filter tap and complex LMS coefficient update engine.

The fixed-point system was designed with the Virtex-II arithmetic resources in mind – the 18x18b multipliers. For a number of channel models tested, 18b precision coefficients were determined to be sufficient to ensure convergence and provide an acceptable error-vector magnitude.

A $T/2$ ($T \equiv$ symbol rate) FSE was implemented. In this case data is provided to the equalizer at a rate of $T/2$ samples per second, but symbol decisions are made at the symbol rate T. The embedded 2:1 downsampling dictates that a polyphase architecture be used to most efficiently and economically implement the design. A block diagram of the implementation is shown in Figure 15. A transposed FIR architecture is employed. The accumulate-delay path in each polyphase segment permits a very regular highly pipelined datapath, maximized for speed, to be employed. One potential downside of this filter architecture, that needs to be considered by designers, is the high fan-out nets from the equalizer input port to the multiplier operand inputs. These nets frequently form the critical path in transposed FIR structures and can limit system throughput. With the transposed FIR structure the regressor vector, needed for the adaption processing, is not available as part of the filter state. A separate register file is used for this purpose. A compensating delay to correctly align the output decisions with the input samples is also required. The implementation employed 40 complex transversal taps, all operating in parallel. 20 LMS coefficient update engines were used. This allowed each the entire filter coefficient vector to be updated every two symbol decisions. An FPGA floorplan of the design is shown in Figure 16.

Figure 15: Virtex-II FSE polyphase equalizer. The design employs a 2:1 decimated coefficient update.

Figure 16: Complex QPSK FSE. The design has 40 filter taps and 20 LMS coefficient update engines operating concurrently for a total of 180 multipliers. The floorplan shows a Virtex-II XC2V10000BF957 FPGA.

The transversal filter taps are implemented using the embedded hardware multipliers. Each complex tap is realized using three 18x18b precision multiplications, one addition and one

subtraction. There are 120 multipliers in the filter. The LMS engines operate on complex valued data. The twenty complex multipliers (requiring a total of 60 real multipliers), involved in the coefficient adaption are implemented in the FPGA logic fabric. There are a total of 180 multipliers in the equalizer. The clock speed achieved was 74 MHz. This is equivalent to 13.31 giga-MAC/s.

4.2 Channelization

Virtually all digital receivers perform channel access using a digital down-converter (DDC). Modern basestation transceivers will often require a large number of DDCs to support multi-carrier environments or for coherently down-converting and combining a number of narrow-band channels into one wide-band digital signal. The DDC is typically located at the front-end of the signal processing conditioning chain, close to the A/D, and is usually required to support high-sample rate processing in the region of 100 to 200 mega-samples-per-second. The high data rate, coupled with the large arithmetic workload, are not well suited for DSP microprocessor implementation. Application specific standard products (ASSP) are a common solution. A more flexible, and typically higher-performance alternative, is to implement the DDC using programmable logic. Since DDC functions only require a modest amount of FPGA silicon resources, many other receiver functions can be implemented in the same device. Consider the implementation of a single channel of the GC4016 quad digital receiver [7]. A simplified block diagram is shown in Figure 17.

Figure 17: Digital down converter architecture.

The desired channel is translated to baseband using the digital mixer comprising the multipliers *M1, M2* and a direct digital synthesizer (DDS). The sample rate of the signal is then adjusted to match the channel bandwidth. This is performed using a multi-stage multi-rate filter consisting of the filters $C(z)$, $G(z)$ and $H(z)$. The functions performed in the system are waveform synthesis (DDS), complex multiplication and multirate filtering. We will benchmark an FPGA version of the DDC that is suitable for GSM wireless applications. The spectral mask requirement for GSM is shown in Figure 18. The input sample rate is chosen to be 52 MHz. The GSM channel can be supported with an output sample rate of 270.8333 kHz. This corresponds to a sample rate change of 192.

Figure 18: GSM spectral mask.

The baseband channel is highly oversampled so a simple cascade of boxcar filters, implemented as a cascaded integrator comb (CIC) [12] will be employed to initially reduce the sample rate by a factor of 48. The CIC filter $C(z)$ is multiplierless consisting only of integrator and differentiator sections. For this application a cascade of 4 integrators followed by 4 differentiators, with an embedded 48:1 rate change, will be employed. A desirable characteristic of the CIC filter in the contest VLSI design is of course that it is multiplier free. However, as described in [12], the precision required in the integrator and differentiator sections can be non-trivial, growing to 40-bits or more. This can potentially lead to operating speed issues in some technologies because of the serial dependency in the adder/subtractor carry chains. Modern FPGA devices like Virtex-II provide extremely high performance carry chains, and even the long carry-chains that can be required in some CIC filters can be supported at very high speed. There is a rich range of intellectual property (IP) from FPGA vendors and third parties to assist designers with their ever compressed development schedules. In this case we can use the CIC filter Core supplied with the Xilinx Core Generator System [16]. A screen shot of the user interface is shown in Figure 19. The complex filter $C(z)$ is automatically generated using this approach and consumes 532 logic slices [10] in total for the I and Q data streams.

Figure 19: CIC user interface. The module generator is parameterized for input sample precision, sample rate change and supports up-sampling and down-sampling operations.

The CIC filter is followed by a cascade of two 2:1 polyphase decimators to produce the required input-to-output sample rate change of 192:1. A 21-tap filter is used for the polyphase decimator $G(z)$ while a 63-tap filter is employed for $H(z)$. Figure 20(a), (b) and (c) illustrate the spectral responses for the filters $C(z)$, $G(z)$ and $H(z)$ respectively. Figure 20(d) is the aggregate response, obtained by convolving the impulse response of the three filters with each other. We observe that the combined response satisfies the GSM spectral mask requirements.

Figure 20: Frequency characterizations. (a) $C(z)$. (b) $G(z)$. (c) $H(z)$. (d) Aggregate frequency response and GSM spectral mask overlay.

The multipliers *M1* and *M2* are implemented using the Virtex-II embedded multipliers. Using the optional pipelined mode of operation the embedded Virtex-II multipliers can support samples rates in excess of 200 MHz. In this design, with an input sample rate of 52 MHz, a single multiplier could be time-shared to implement the input heterodyne. The DDS employed is a phase-dithered look-up table-based synthesizer. The FPGA block memory is used to store one quarter of a cycle of a sinusoid. The dual-port memory enables both the in-phase and quadrature components of the local oscillator to be generated simultaneously using a single block RAM. The single block RAM implementation can generate a 4096-sample full-wave 16-bit precision complex sinusoid. With phase dithering, the synthesizer will generate a mixing signal with a spurious free dynamic range (SFDR) of approximately 84 dB. Space constraints inhibit a comprehensive description of the DDS. Details can be found [8].

One very successful technique for implementing filters, including multirate structures, in FPGAs is distributed arithmetic (DA) [9]. This approach is an option for realizing $G(z)$ and $H(z)$. The functional requirements of a DA filter, large shift registers, lookup tables and accumulation, closely match the silicon resources provided in an FPGA technology like Virtex-II. The two polyphase structures can be realized using parameterizable IP [16]. The complex filters $G(z)$ and $H(z)$ 372 and 652 logic slices respectively. To put these figures in context, the Virtex-II family of FPGAs consists of devices with slice counts ranging from 40 to 61,400.

Because the sample rate at the output of $C(z)$ is only $52e^6 / 48 = 1.0833$ MHz, another option for implementing the multirate filters is to simply employ a time-shared multiply-accumulator (MAC) approach. The arithmetic work load $F_{G(z)}$ (using millions of MAC/s as the metric) for $G(z)$ is

$$F_{G(z)} = 21 \times \frac{52e^6}{48 \times 2} = 11 \text{ MMAC/s} \tag{4}$$

The workload $F_{H(z)}$ for $H(z)$ is

$$F_{H(z)} = 63 \times \frac{52e^6}{48 \times 4} = 17 \text{ MMAC/s} \tag{5}$$

The total workload for the two filters is $2 \times (11+17) = 56$ MMAC/s. This processing requirement can be met with a single Virtex-II multiplier. The coefficient sets could be stored in either block or distributed memory [10].

The complete DDC requires two multipliers, one block RAM and a small number of logic slices for coefficient storage and control. An XC2V1000 device, with 40 multipliers and 5,120 logic slices could easily support 8 DDCs while an XC2V6000 with 144 multipliers and 33,792 slices would enable 60 DDCs to be realized on a single chip.

5 Conclusion

FPGA based signal processors are being employed in a diverse range of signal processing applications for reasons of performance, economics, flexibility and power consumption. The telecommunication industry has been quick to embrace FPGA technology. Nearly 50% of all FPGA production finds its way into telecommunications and network equipment of one sort or another - wireless base stations, switches, routers and modems to name a few. FPGAs offer great flexibility, which can, for instance, enable designers to service multiple standards. An example is a universal cellular handset that would automatically recognize different signaling standards, such as GSM, CDMA, TDMA, or AMPs, and would reconfigure itself to accommodate the identified protocol. The flexibility and performance provided by FPGAs also allows designers to easily track evolving standards like MPEG, and provide a methodology for dealing with fluid standards such as ADSL.

Even though FPGA DSP systems represent a significant faction of the signal processing arena, we are witnessing an exponential growth in the insertion of FPGAs in DSP hardware. This explosive growth is enhanced by access to FPGA intellectual property (IP) cores from all the major FPGA suppliers as well as 3rd-party IP designers. With these resources, the system implementor is able to focus on the design rather than the details of lower-level modules like filters and transforms.

The continuing evolution of communication standards and competitive pressure in the market place dictate that communication system architects must start the engineering design and development cycle while standards are still in a fluid state. Third and future generation communication infrastructure must support multiple modulation formats and air interface standards. FPGAs provide the flexibility to achieve this goal, while simultaneously providing high levels of performance. The SDR implementation of traditionally analog and digital hardware functions opens-up new levels of service quality, channel access flexibility and cost efficiency.

The software in a SDR defines the system personality, but currently, the implementation is often a mix of analog hardware, ASICs, FPGAs and DSP software. The rapid uptake of state-of-the-art semi-conductor process technology by FPGA manufacturers is opening-up new opportunities for the effective insertion of FPGAs in the SDR signal conditioning chain. Functions frequently performed by ASICs and DSP processors can now be done by configurable logic. This paper has provided an overview of how several signal processing functions can be implemented in an FPGA.

6 References

[1] Chris Dick and fred harris, ``FPGA Signal Processing Using Sigma-Delta Modulation'', *IEEE Signal Processing Magazine*, Vol. 17, No. 1, pp. 20-35, Jan. 2000.

[2] J. R. Treichler, I. Fijalkow and C. R. Johnson Jr., " Fractionally Spaced Equalizers", *IEEE Signal Processing Magazine*, pp. 65-81, May 1996.

[3] J. R. Treichler, M. G. Larimore and J. C. Harp, " Practical Blind Demodulators for High-Order QAM Signals", *Proc. IEEE*, Vol 86, No. 10, pp. 1907-1926, Oct., 1998.

[4] C. R. Johnson Jr., P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown and R. A. Casas, " Blind Equalization Using the Constant Modulus Criterion: A Review", *Proc. IEEE*, Vol. 86, No. 10, pp. 1927-1950, Oct. 1998.

[5] B. Widrow, J. M. McCool, M. G. Larimore and C. R. Johnson, "Adaptive Noise Cancelling: Principles and Applications", *Proc. IEEE*, Vol. 63, No. 12, pp.1692-1716, Dec. 1975.

[6] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice Hall, New Jersey, 1985.

[7] GC4016 Multi-Standard QUAD DDC Chip, Datasheet, Graychip, Pal Alto CA, USA, Aug., 2000.

[8] C. H. Dick and f. j. harris, "Direct Digital Synthesis – *Some options for FPGA Implementation*, *SPIE International Symposium on Voice Video and Data Communication,* Boston, Mass., 19-22 Sept. 1999.

[9] Peled and B. Liu, "A New Hardware Realization of Digital Filters", *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. 22, pp. 456-462, Dec. 1974.

[10] Xilinx Inc., *Virtex-II Platform FPGA Handbook*, 2000.

[11] S. A. White, "Applications of Distributed Arithmetic to Digital Signal Processing*", IEEE ASSP Magazine*, Vol. 6(3), pp. 4-19, July 1989.

[12] E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE. Trans. Acoust., Speech Signal Processing*, Vol. 29, No. 2, pp. 155-162, April 1981.

[13] B. Sklar, *Digital Communications Fundamentals and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[14] The Mathworks Inc, *Matlab, Getting Started with Matlab*, Natick, Massachusetts, U.S.A, 1999.

[15] The Mathworks Inc, *Simulink, Dynamic System Simulation for Matlab, Using Simulink*, Natick, Massachusetts, U.S.A, 1999.

[16] Xilinx Core Generator System, http://www.xilinx.com/products/logicore/coregen/index.htm

[17] C. H. Dick and f. j. harris, "Direct Digital Synthesis - Some Options for FPGA Implementation", *SPIE International Symposium On Voice Video and Data Communication: Reconfigurable Technology: FPGAs for Computing and Applications* Stream, Boston, MA, USA, pp. 2-10, September 20-21 1999.