FPGA Implementation of an Adaptive Noise Canceller

Tian Lan[1], Jinlin Zhang[2]
[1]Department of Electronics and Information Engineering, Huazhong University of Science and Technology
[2] Wuhan Radar Institute
taran.lan1986@gmail.com

Abstract

This paper proposes an FPGA implementation of an Adaptive Noise Canceller using the Least Mean Square (LMS) algorithm. The hardware architecture is synthesized using the Xilinx Spartan-3e Starter Kit as the target board. The experimental result of the hardware implementation shows the performance of LMS algorithm under different conditions and the feasibility of our architecture. A comparison between hardware and pure software implementation is then made with different filter taps.

1. Introduction

Digital signal processing, which spans a wide variety of application areas including speech and image processing, communications, networks, and so on, is becoming increasingly important in our daily life. Digital signal processing applications impose considerable constraints on area, power dissipation, speed and cost, thus the design tools should be carefully chosen. The most commonly used tools for the design of signal processing systems are: Application Specific Integrated Circuit (ASIC), Digital Signal Processors (DSP) and FPGA. DSP is well suited to extremely complex math-intensive tasks, but can not process high sampling rate applications due to its serial architecture. ASIC can meet all the constraints of digital signal processing, however, it lacks flexibility and requires long design cycle. FPGA can make up the disadvantages of ASIC and DSP. With flexibility, time-to-market, risk-mitigation and lower system costs advantages, FPGA has become the first choice for many digital circuits' designers.

Nowadays, the FPGA technology has achieved conspicuous development. System on a Chip (SoC) and Intellectual Property (IP) designs can be integrated and downloaded into FPGA to work with an embedded processor. The latest 65 nm technology makes the function of FPGA even more powerful. The 65nm Virtex-5 FPGAs offer unprecedented performance at speeds on average 30 percent faster and 65 percent increased capacity over previous 90nm FPGAs, while consuming 45 percent less area and reducing dynamic power consumption by 35 percent than previous generation devices [10].

In this paper, in order to show the performance of FPGA in digital signal processing applications, we implement an Adaptive Noise Canceller on an FPGA and use the LMS algorithm as the adaptive filtering algorithm of the Adaptive Noise Canceller. Adaptive Noise Canceller is a real-time system that requires high sampling rate, thus FPGA is a good choice. LMS algorithm, originally proposed by Widrow and others, is widely used for adaptive filter [1], [11]. After that, delayed LMS (DLMS) algorithm [2] has been derived to achieve low latency. The previous works of Very Large Scale Integrated Circuit (VLSI) implementations of LMS and DLMS are shown in [3]-[6]. However, they are mainly concerned with the convergence behavior of LMS or DLMS, the detailed implemental process of VLSI and the advantage of VLSI implementation are not mentioned clearly. This paper is based on the performance of FPGA implementation in an Adaptive Noise Canceller.

This paper is organized as follows: section 2 gives the necessary background of Adaptive Noise Canceling (ANC) and LMS algorithm. Section 3 gives the detailed process of FPGA implementation. Section 4 presents the experimental results, in which, the performance of the LMS algorithm in our hardware architecture is shown and a pure software implementation is introduced and compared with its hardware counterpart for different filter taps.

2. Application of LMS algorithm in ANC system

A. ANC

As the name implies, ANC is a technique used to remove an unwanted noise from a received signal, the operation is controlled in an adaptive way in order to obtain an improved signal-to-noise ratio (SNR) [7]. As shown in Fig.1, an ANC is typically a dual-input, closed-loop adaptive feedback system. The two inputs are: the primary input signal d(n) (the desired signal corrupted by the noise) and the reference signal x(n) (an interfering noise supposed to be uncorrelated with the desired signal but correlated with the noise affecting the desired signal in an unknown way). The adaptive filtering operation achieved the best results when the system output is noise free, which means that the output SNR is

978-0-7695-3151-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ISIP.2008.107

infinitely large. Therefore, in the application, on obtaining the best result, we should put the reference sensor into the most appropriate place where the signal component of the primary sensor output is undetectable in the reference sensor output and the noise component of the primary sensor output is highly correlated with the reference sensor output.

ANC technique has been successfully applied to many applications, such as acoustic noise reduction, adaptive speech enhancement and channel equalization. In these cases and other related high speed required applications, pure software implementation would bring about long processing time, as shown in the last part of section IV, thus can not meet the requirement. An effective way can be represented by a hardware implementation on FPGA.

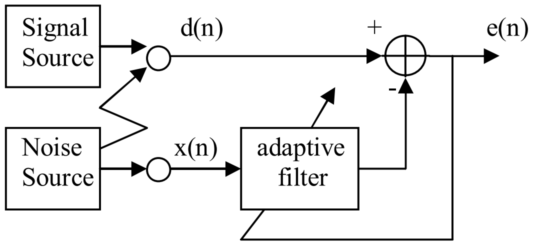

Fig.1. ANC system

B. LMS Algorithm

The LMS algorithm is a widely used algorithm for adaptive filtering. The algorithm is described by the following equations:

$$y(n) = \sum_{i=0}^{M-1} \mathbf{w}i(n) * x(n-i); \quad (1)$$

$$e(n) = d(n) - y(n) \quad (2)$$

$$\mathbf{w}i(n+1) = \mathbf{w}i(n) + 2ue(n)x(n-i); \quad (3)$$

In these equations, the tap inputs x(n), x(n-1),……,x(n-M+1) form the elements of the reference signal x(n), where M-1 is the number of delay elements. d(n) denotes the primary input signal, e(n) denotes the error signal and constitutes the overall system output. **w**i(n) denotes the tap weight at the nth iteration. In equation (3), the tap weights update in accordance with the estimation error. And the scaling factor u is the step-size parameter. u controls the stability and convergence speed of the LMS algorithm. The LMS algorithm is convergent in the mean square if and only if u satisfies the condition:

$0 < u < 2$ / tap-input power

where tap-input power = $\sum_{k=0}^{M-1} E\,[|u(n-k)|^2]$

The flowchart of the signals of the LMS algorithm is shown in Fig.2.

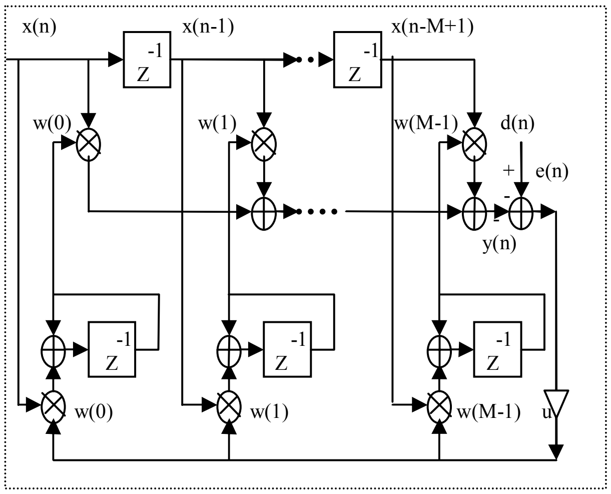

Fig.2.Flowchart of the LMS Algorithm

There are usually two ways to implement the LMS algorithm, hardware implementation and software implementation [8], [9]. The hardware implementation of the algorithm in an FPGA has good real-time ability, but requires large resources. From Fig.2, we can see that an N-tap adaptive filter requires at least 2N multipliers and 2N adders. The software implementation consumes trivial amount of resources, however, the low speed of which makes it uncommonly used.

3. Hardware implementation

In this work, the ANC described in the previous section is implemented in the Xilinx Spartan-3E Starter Kit board that provides a convenient development board for embedded processing applications. The board contains a Xilinx XC3S500E Spartan-3E FPGA with up to 232 user-I/O pins，320-pin FPGA package and over 10,000 logic cells. The Spartan-3E Starter Kit also provides the MicroBlaze 32-bit embedded RISC processor and the Xilinx Embedded Development Kit (EDK) [10]. With those features, the Xilinx Spartan-3E Starter Kit is well suited for hardware implementation of our ANC system.

The hardware implementation process is described using the flowchart shown in Fig.3.

A. Hardware Architecture

The whole embedded system consists of a MicroBlaze core [10], two Fast Simplex Link (FSL) bus systems, an On-Chip Peripheral Bus (OPB), a Local Memory Bus (LMB), an OPB peripheral (RS232 Controller), the on-chip block RAM and the user core. A block diagram of the architecture is shown in Fig.4.

MicroBlaze is a standard 32-bit RISC Harvard-style soft processor, which is used to run the whole system. MicroBlaze communicates with user core through FSL channels; it interfaces with local memory by LMB, and interfaces with different peripherals by OPB. The FSL channels are dedicated unidirectional point-to-point data streaming interfaces, they are 32-bit wide, and support either control or data communication. The detailed process on how the embedded system works is as follows:

Firstly, the software application writes two data (primary input signal and reference signal) from memory to FSL0, the LMS core reads the data and calculates the result. In this case, MicroBlaze is the Master on the FSL bus and the LMS core is the Slave, thus MicroBlaze controls the data sent to the LMS core. Then, when the result is available, the LMS core writes it to FSL1, and MicroBlaze reads the data back from it. Here the LMS core is the Master on FSL1 and MicroBlaze is the Slave. After that, MicroBlaze writes the result to the PC through serial interface. At last, we use Matlab to plot the wave of the output signal.

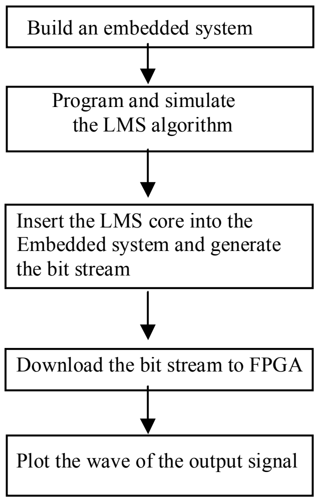
Build an embedded system
Program and simulate the LMS algorithm
Insert the LMS core into the Embedded system and generate the bit stream
Download the bit stream to FPGA
Plot the wave of the output signal

Fig.3. Flowchart of the FPGA Implementation Process

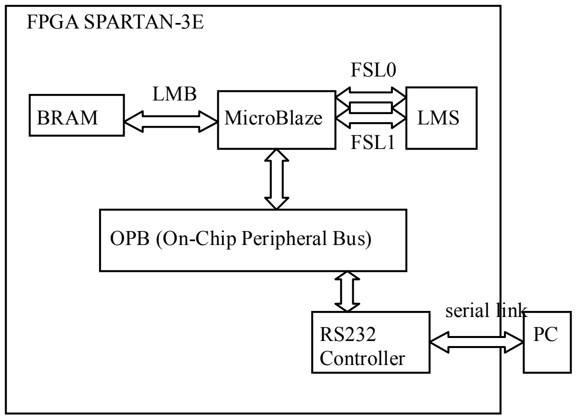

Fig.4. Hardware Architecture

B. LMS Core Implementation

The LMS core is the central part of our hardware architecture. We programmed the LMS core with VHDL under the platform of Xilinx ISE 9.1i, and simulate it with ModelSim 6.1b. Then we test the validity of the LMS core by using System Generator 9.1.

1) Block Diagram

The block diagram of the LMS core is shown in Fig.5. The LMS core is divided into five blocks：

1. The Control block arranges the timing of the whole system. It produces four enable signals: en_x, en_d, en_coee, en_err, which enable the Delay Block, the Weight Update Block and the Error Counting Block separately.

2. The Delay Block receives the reference signal x_in and the primary input signal d_in under control of the enable signal en_x and en_d. And it produces the M tap delay signal x_out.

3. The Multiply Accumulator (MAC) Block multiply the M_tap reference signal x_out with the M_tap weight w separately, and add them together, then we get yn.

4. The Error Counting Block subtract yn from dn and get the error signal e_out, which is also the output of the whole system. And it produces signal xemu as a feedback by multiplying e_out, x_out and the scaling factor u.

5. The Weight Update Block updates the weight vector w(n) to w(n+1) that will be used in the next iteration.

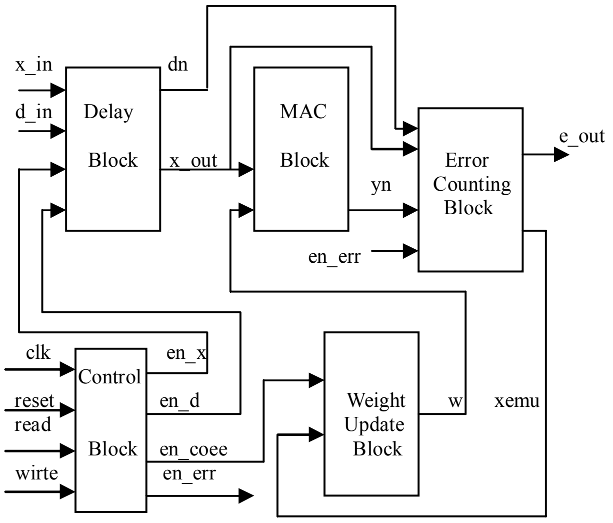

Fig.5. Block Diagram of the LMS Core

2) Fixed-point Calculation

Since logic capacities on the FPGA chips are limited for floating-point calculations, we design a fixed-point system to implement the LMS algorithm. In a fixed-point system, the word length of the I/O signals and intermediate signals should be considered carefully in terms of truncation effect and cost. Fig.6 shows the block diagram of the circuit and the fixed-point bit format is shown in Fig.7.

Firstly, the bit format is chosen. The radix point is located just after the sign bit in achieving the highest precision. Then the word length of the I/O signal is considered. An 8-bit I/O signal is determined to be a good compromise between the resource cost and filter stability [6]. The word length of intermediate signals is also critical in reducing the truncation effect and hardware complexity. We make bit-truncation twice in the circuit, each of which is located before the multiplier in saving the resource cost. Apart from that, the location of the first bit-truncation determines the convergence performance of the filter. To meet fast convergence rate and low SNR required by ANC system, the filter coefficients should be as accurate as possible. Increased word length achieves more accuracy but results in large cost of hardware resource. To make a compromise, we truncate the coefficients to 8-bit right after the adder, before the multiplier. That is to say, in the update process, the coefficients are in 16-bit level, while in the filtering process, they are truncated to 8-bit level. Truncation effect would be serious if the first bit truncation occurs before Adder1, namely the filter coefficients are 8-bit throughout the circuit, and the experimental result of this case is shown in the next section.

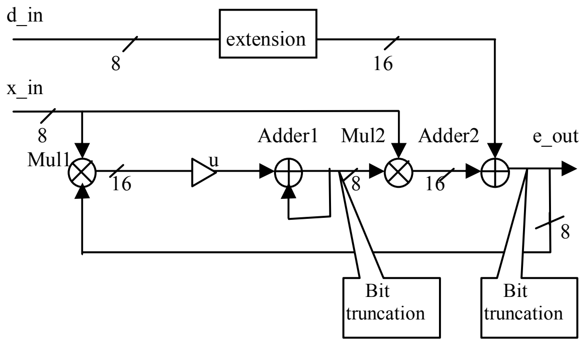

Fig.6. Block Diagram of the Circuit

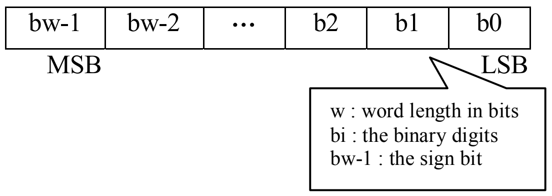

Fig.7. Bit format

4. Experimental results

A. Convergence Performance

The convergence performance of LMS algorithm applied in hardware architecture is analyzed with respect to an 8-tap LMS adaptive filter with desired signal of constant 0.5, and input signal of 0.25. Since the convergence speed of LMS algorithm has closed relationship with step-size u, three cases of performance are compared with different step-sizes. The following 3 curves plotted in Fig. 8 show the traces of error between the desired signal and the output signal. The step-sizes of the curves from left to right are: 1/2, 1/4 and 1/8. From the cases, we can see that with the increase of step-size, convergence speed rises, which is in accord with the theory of LMS algorithm. Convergence performance can also be affected by bit-truncation effect discussed in the Section III. The convergence trace of the error signal in the case that the first bit truncation occurs before Adder1 in Fig.6 is plotted in Fig.9 (8-tap filter, u = 1/2). We can see that the error signal does not converge to zero. In contrast, under the same condition, the error signal of our implementation plotted in red line in Fig.8 has much better convergence performance.

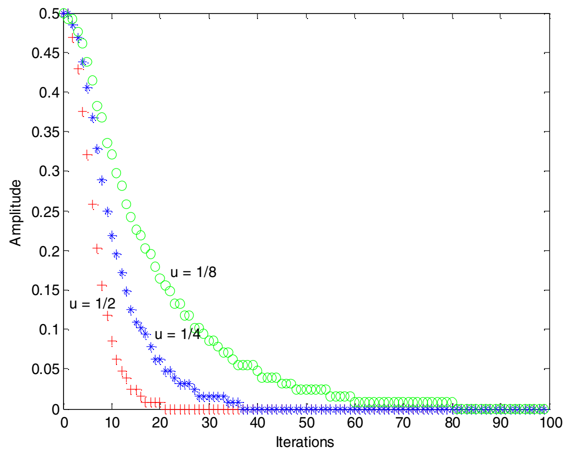

Fig.8. Convergence behavior of the LMS algorithm for three step sizes

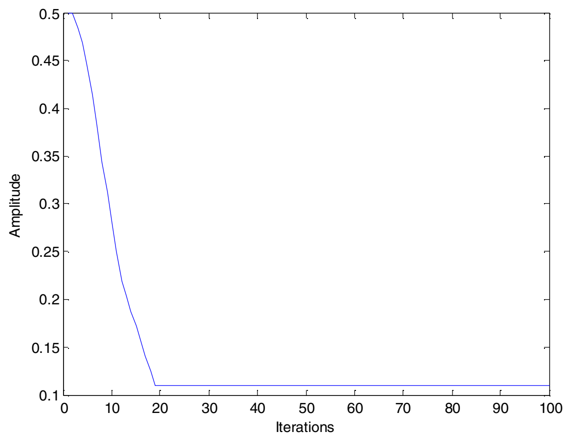

Fig.9. Convergence behavior of the case

that takes bit truncation inappropriately

B. Tracking Ability

Tracking ability is also an important property of LMS algorithm. We use a LMS filter (8-tap, u = 1/4) with an input signal shown in Fig.10, which is a sinusoidal signal at 1/32 of the sampling frequency corrupted by sinusoidal noise at 1/3.2 of the sampling frequency. The SNR of the input signal is 6 dB. Fig.11 shows the tracking ability of the LMS filter, the desired signal is plotted in blue dashed line, and the red real line represents the result signal. We can see that, after about 200 iterations, the result signal is converged to the desired one.

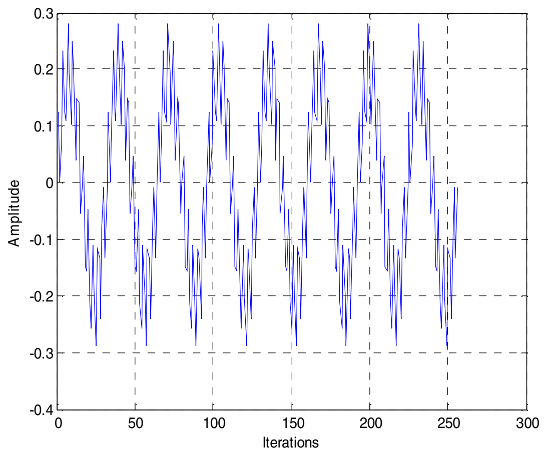

Fig.10. Input signal

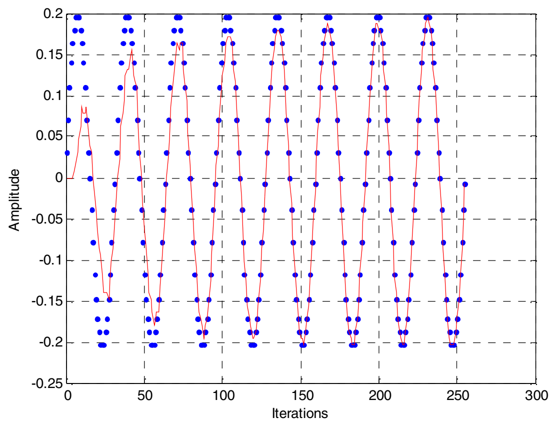

Fig.11. Tracking ability of the LMS filter

C. Software Implementation

In order to show the advantage in speed of hardware implementation, we use a software implementation to compare with it. In this architecture, we use the MicroBlaze embedded processor to run the whole system and the LMS algorithm is written in C code. The pseudo C code of the software implementation is shown in Fig.12.

There are four major steps of the software implementation: 1). delay the input signal; 2). multiply accumulate to get the intermediate signal y; 3). error calculation; 4). weight update. We make a comparison between software and hardware implementation measured with clock cycles in one iteration, the profiling results of the four steps are shown in table I for three different values of N.

```
#define SIZE 256    //calculate 256 points of the signal

for j = 0 to SIZE-1
{
  for i = 1 to N-1
      x[i] = x[i-1];                    //input delay
      x[0] = ref;

  for i = 0 to N-1
      y[j] = y[j] + w[i]*x[i]; // multiply accumulate
      error = input - y[j];      //error calculation

  for i = 0 to N-1
      w[i] = w[i] + 2*u*error* x[i]; // weight update
}
```

Fig.12. Pseudo C Code of the Pure Software implementation

Table I Comparison of the profiling results using two architectures (clock cycles)

step	N = 8		N = 16		N = 32	
	SW	HW	SW	HW	SW	HW
1	8	1	16	1	32	1
2	8	8	16	16	32	32
3	1	1	1	1	1	1
4	8	1	16	1	32	1
total	25	11	49	19	97	35
HW Speed up	2.27		2.58		2.77	

The results in table I clearly indicate that the speed up of hardware implementation increases with filter tap N. The main difference of speed between software and hardware implementation lies in step 1 and 4. The software implementation compute each tap in one clock cycle, while in hardware implementation, N taps is calculated in parallel in one clock cycle.

5. Conclusion

In this paper, hardware architecture is presented to implement the ANC system. The performance of the LMS algorithm implemented by hardware is comprehensively analyzed in terms of convergence performance, truncation effect and tracking ability. The experimental results ensure the feasibility of the high-speed FPGA architecture of the LMS algorithm. Also, a software implementation is presented. A comparison between the two architectures shows that hardware implementation accelerates the LMS filtering process, and the speed up over pure software implementation increases with the filter tap. The ANC system is chosen to validate the performance of FPGA in digital signal processing applications; FPGA implementation will be more effective for more complex digital signal processing systems.

6. References

[1] B. Widrow, J. R. Glover, J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong and R. C. Goodlin, "Adaptive noise canceling: Principles and applications", *Proc. IEEE*, vol. 63, Dec. 1975, pp. 1692-1716.

[2] G. Long, F. Ling and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. on ASSP,* vol. 37, Sept. 1989, pp. 1397-1405.

[3] C.-L.Wang, "Bit-serial VLSI implementation of delayed LMS adaptive FIR filters," *IEEE Trans. Signal Process.*, vol. 42, Aug. 1994, pp. 2169–2175.

[4] L. K. Ting, R. F. Woods and C. F. N. Cowan, "Virtex FPGA Implementation of a Pipelined Adaptive LMS Predictor for Electronic Support Measures Receivers," *IEEE Trans. VLSI Syst.,* vol. 13, Jan. 2005, pp. 86-95.

[5] M. D. Meyer and D. P. Agrawal, "A high sampling rate delayed LMS filter architecture," *IEEE Trans., Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, Nov. 1993, pp. 727–729.

[6] L.-K. Ting, "Algorithms and FPGA implementations of adaptive LMS-based predictors for radar pulse identification," Ph.D. dissertation, Queen's Univ. Belfast, N. Ireland, Jul. 2001.

[7] B. Widrow, J. R. Glover, J. M. McCool and et al., "Adaptive Noise Canceling: Principles and Applications", *Proc. IEEE,* vol. 63, Dec. 1975, pp. 1692-1716.

[8] P. Waldeck and N. Bergmann, "Evaluating software and hardware implementations of signal-processing tasks in an FPGA," in *Proc. IEEE International Conference on Field-Programmable Technology*, Brisbane, Australia, Dec. 2004, pp. 299-302.

[9] A. Elhossini, S. Areibi and R. Dony, "An FPGA Implementation of the LMS Adaptive Filter for Audio Processing," in *Proc. IEEE International Conference on Reconfigurable Computing and FPGAs*, Sept. 2006, pp. 1-8.

[10] www.xilinx.com

[11] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, third edition, 2002.