PIPELINED IMPLEMENTATIONS OF THE A PRIORI ERROR-FEEDBACK LSL ALGORITHM USING LOGARITHMIC ARITHMETIC

Felix Albu[1], Jiri Kadlec[2], Nick Coleman[3], Anthony Fagan[1]

[1]DSP Group, UCD, Belfield 4, Dublin, Ireland, felix@ee.ucd.ie
[2]UTIA, Pod vodarenskou vezi 4, 182 08 Prague 8, Czech Republic, kadlec@utia.cas.cz
[3]Dept EE, The University, Newcastle, NE1 7RU, UK, j.n.coleman@ncl.ac.uk

ABSTRACT

In this paper we present several implementations of the Modified A Priori Error-Feedback LSL (EF-LSL) algorithm [1] on the VIRTEX FPGA. Its computational parallelism and pipelinabilty are important advantages. Internally, the computations are based on the logarithmic number system. We compare 32-bit (SINGLE-ALU or DUAL-ALU version) and 20-bit (QUADRI-ALU versions). We show that the LNS implementation can outperform the standard DSP solutions based on 32-bit floating-point processors.

1. INTRODUCTION

Adaptive signal processing algorithms are used widely in communications systems. The lattice algorithms represent a good alternative to LMS algorithms in many applications due to their fast convergence, modular implementation and less complexity than RLS (order N) [2]. The modified EF-LSL algorithm proposed in [1] is one of the most numerically accurate LSL algorithms and presents numerically stable behavior when properly implemented. There are many possible applications for this algorithm such us acoustic echo cancellation, noise cancellation or blind equalization [3].

In reference [1] an efficient implementation of this EF-LSL algorithm using a floating point Analog Devices 21061 DSP (SHARC) is presented. This 40 MIPS DSP has sixteen 40-bit internal registers, which are extensively used for parallel additions and multiplications. For an acoustic echo canceling application the authors showed that it is possible to use up to M=290 coefficients for an 8 kHz sampling rate [1]. However, this solution has a limitation imposed by the use of divisions that are implemented using denominators with reduced mantissa wordlength. We will present a solution that overcomes this problem.

As an alternative to floating-point, the logarithmic number system offers the potential to perform real multiplication, division and square root at fixed-point speed and, in the case of multiply and divide, with no rounding error at all. These advantages are, however, offset by the problem of performing logarithmic addition and subtraction. Hitherto this has been slower or less accurate than floating-point, or has required very cumbersome hardware. We employ an innovative solution developed by the HSLA project team under Coleman [4], which yields a drastic reduction in the size of the look-up tables required compared to those needed for conventional linear interpolation of both functions. Coleman's approach leads to a suitable solution for the FPGA implementation. The logarithmic number system (LNS) is well suited to the FPGA environment. The 32-bit core using external memory takes about 7% of the XILINX Virtex XCV2000E-6 device [5]. It implements all the basic operations of logarithmic arithmetic (ADD, SUB, MUL, DIV and SQRT), with precision equal to or better than the standard IEEE 32-bit floating point used in new DSPs or the TI 32-bit floating point standard used in the TMS320C30/C40 devices. The internal conversion to the log domain and the internal LNS operations can be hidden from the user. The conversion to/from log domain is based on evaluation of the rational polynomial approximation of log and antilog in the range (-1,1). The log ALU is used for the conversion [5]. However, their contribution to the processing time is quite insignificant in comparison with the overall requirements of the algorithm, especially for high orders and it wasn't considered in this paper.

The 32-bit LNS implementation uses 321,536 bits (95 blocks) of lookup tables. The hardware requirements are about 20,000 equivalent gates for the add/subtract unit and about 2,000 for all other units. A 20-bit LNS format is also considered. It maintains the same range as the 32-bit, but has precision reduced to 11 fractional bits. This is comparable to the 16-bit formats used on commercial DSP devices. The 20-bit version requires just 10,920 bits of lookup tables (4 blocks). The hardware requirements are about 12,000 equivalent gates for the add/subtract unit and about 1,600 for all other units.

We have implemented the 32-bit LNS (with external RAM), 32-bit LNS DUAL-ALU (with internal RAM) and 20-bit QUADRI-ALU LNS (with internal RAM), and we compare each implementation with floating-point alternatives.

In the adaptive lattice, coefficients are adapted within each stage, without the need for global feedback, making extensive pipelining of computations possible [6]. That leads to either increased clock speed or sample speed or to reduced power consumption at the same speed. This method is successfully used when the application permits the algorithmic delay to be increased. The task of mapping this lattice algorithm on to the pipeline is straightforward.

2. SIMULATIONS AND PERFORMANCE

The Modified EF-LSL algorithm implementations (FLOAT or LNS) based on *A Priori* Errors was used for a system identification example. The input signal is a composite source signal according to the G.168 ITU recommendation for digital echo cancellers. The length of the adaptive filter is $M = 128$. No noise was added and the echo path is taken from that standard. An accurate standard for comparison of the outputs was obtained by presenting this input data to a double precision version. The squared errors are presented in Fig.1 and the absolute sum of errors is presented in Fig. 2. The forgetting factor was $\lambda = 0.999$ and the parameter $\varsigma = 2^{-20}$ [1]. It can be seen from Fig.1 and Fig.2 that the finite implementations (32-bit full precision FLOAT or 32-bit LNS) have about the same performance. The effect of using a 20-bit precision is clearly visible in Figs. 1-2. The magnitude of the sum of errors for 20-bit implementation is much higher than the 32-bit implementations (FLOAT or LNS). However, it is evident from Fig.3 that by using reduced precision divisions (with 8 bits mantissa wordlength) the performances of the floating-point implementation are affected and the accumulation of errors is higher for this case. Note that in Fig.3 we changed the value of ς to 2^{-7} for both FLOAT implementations in order to assure the numerical stability for the reduced precision FLOAT implementation. Fig.2 and Fig.3 show the superior performance of the LNS implementation over the reduced precision FLOAT implementation.

2.1. Performance of LNS FPGA vs. TMS320C3x and SHARC

Clock cycles for each type of operation are presented in Table 1. The results for C3x processors (for the full precision floating-point implementation) were estimated using the routines presented in [7].

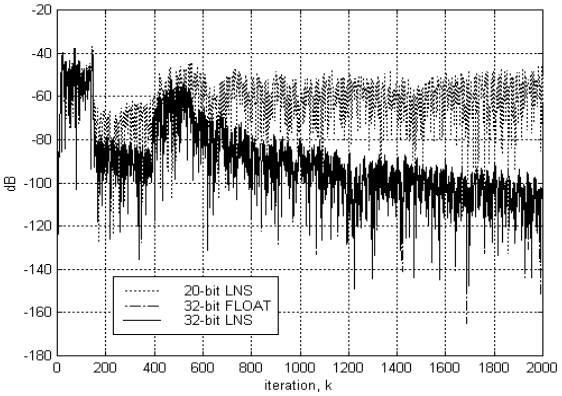

Fig. 1. The square errors for 32-bit FLOAT, 32-bit LNS and 20-bit LNS implementations of the modified EF-LSL algorithm (32-bit traces co-incident)

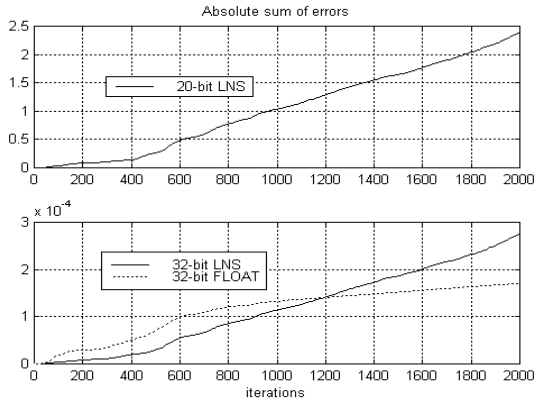

Fig. 2. The absolute sum of errors for FLOAT and LNS implementations of the modified *A priori* EF-LSL Algorithm

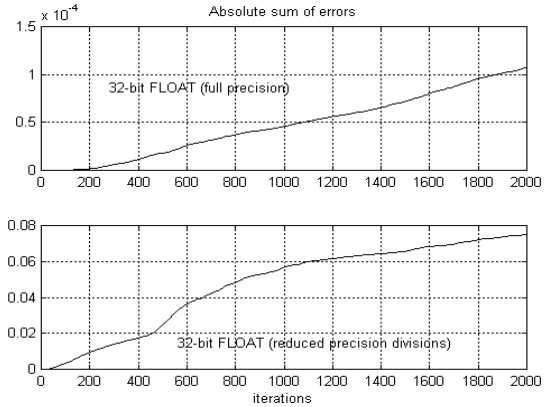

Fig. 3. The absolute sum of errors for 32-bit FLOAT (reduced precision divisions) and 32-bit FLOAT (full precision) implementations of the modified *A priori* EF-LSL Algorithm

Our FPGA implementations can use on-chip or off-chip RAM. The on-chip RAM yields a substantial decrease in the latency of addition and subtraction operations.

	+	-	*	/
C3X	4	4	4	62
32-bit Log (external RAM)	10	10	1	1
32-bit Log (internal RAM)	5	5	1	1
20-bit Log (internal RAM)	5	5	1	1

Table 1. Comparison of LNS and FLOAT execution time (clock cycles)

Operation counts were determined from the modified EF-LSL algorithm equations. Table 2 compares the number of operations for the processing of one input/output sample for the chosen application. The last line of the tables, denoted Log/par, indicates additional savings gained by parallel execution of mul and div operations in the FPGA. It can be seen from Table 2 that these are significant.

	+	-	*	/
C3X	1024	384	1664	256
32-bit Log	1280	128	1664	256
32-bit Log/par	1280	128	256	128

Table 2. Elementary operation counts (EF-LSL algorithm) TMS320C3x in comparison with FPGA

	Cycles	Freq-uency (MHz)	% of slices+ Block RAMs	Latency in samples	Speedup
C3X	28160	50	-	-	1
32-bit Log (external RAM)	14464	45	27+0	-	1.7
32-bit DUAL-ALU Log (internal RAM)	3712	42	54+95	1	6.3
20-bit QUADRI-ALU Log (internal RAM)	1856	50	40+8	3	15.1

Table 3. Comparison TMS320C3x versus FPGA

Table 3 shows the number of clock cycles, the operating frequency, the percentage of VIRTEX slices, and the number of block RAMs for each implementation. Based on these data, an estimate is made of their relative speed. The last column of Table 3 shows that the LNS implementations of the EF-LSL algorithm are likely to be faster than the C3X implementation. The speedup is higher for higher lattice filter orders. The 20-bit QUADRI-ALU implementation of the modified EF-LSL algorithm needs fewer slices than the 32-bit DUAL-ALU ones. Therefore, for some applications that do not require much precision, the 20-bit LNS version could be an attractive alternative.

The following small section is taken from the Handel C code. It outlines the style of parallel programming of the LNS ALU. The subtract operation `lsub()` executes in parallel with 4 instances of the HW macros for logarithmic multiplication `lm()`.

```
par
{
        lsub(lpsi_old[lj-1], ltemp2, lpsi[lj]);
        {
                ltemp6= lm(lf, leta[lj-1]);
                ltemp8= lm(lbn_old[lj-1],leta[lj]);
                ltemp4= lm(llambda, lF_old[lj-1]);
                ltemp5= lm(llambda, lB_old[lj-1]);
        }
}
```

The algorithm has been coded in Handel-C 2.1 and finally Celoxica DK1 [5, 8] has been used. The reported performance has been achieved by the same path presented in [9]:

1. Celoxica DK1 (using the Handel C2.1 compatible code) with export to VHDL.
2. Synplify 5.3 from Synplicity to create EDIF.
3. XILINX Alliance 3.3i tools to place and route from the EDIF netlist for the FPGAs.
4. The Virtex XCV2000E-6 on the RC1000 board [10] has been used for the implementation.
5. MSVC code has been used for interfacing of RC1000 board to Matlab.

We produced an echo signal using a recorded voice signal and a measured car impulse response of 2048 samples used in [11]. The echo return loss enhancement (ERLE) was computed using the mean of 64 consecutive measurements. For an 8 kHz sampling rate the 32-bit LNS implementation using external RAM can use up to 48 coefficients with no delay, while the implementation using internal RAM can use 175 coefficients with 1 sample delay. The QUADRI-ALU 20-bit LNS implementation using the internal RAM can use up to 420 coefficients, with 3 samples delay. In Fig.4 and Fig.5 we compare the ERLE performances for the 32-bit FLOAT versions (290

coefficients), 32-bit DUAL-LNS (175 coefficients) and 20-bit QUADRI-LNS (420 coefficients) implementations. The same parameter values were used in the following simulations. It is evident from Fig.4 that the 32-bit LNS, with only 175 coefficients, is almost equivalent to the reduced-precision 32-bit floating-point with 290 coefficients. Yet the LNS version is implemented as an FPGA whereas the floating-point version runs at the limits of the DSP's capability. Fig.5 shows that a 20-bit LNS FPGA implementation, in which 420 coefficients are possible, offers around the same performance as full-precision 32-bit floating point.

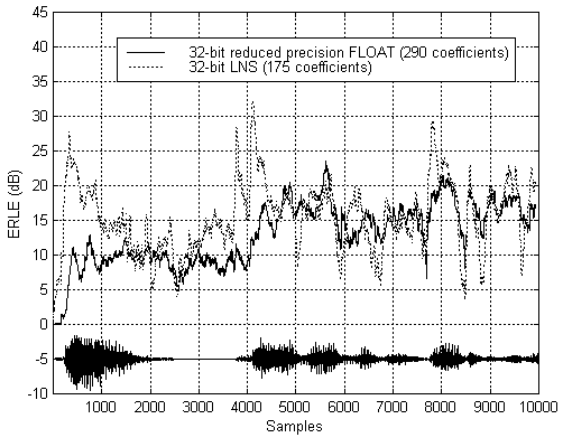

Fig. 4. The ERLE performances for 32-bit reduced precision FLOAT (290 coefficients) and 32-bit LNS (175 coefficients) implementations of the modified *A priori* EF-LSL Algorithm. Bottom trace: echo signal prior to cancellation

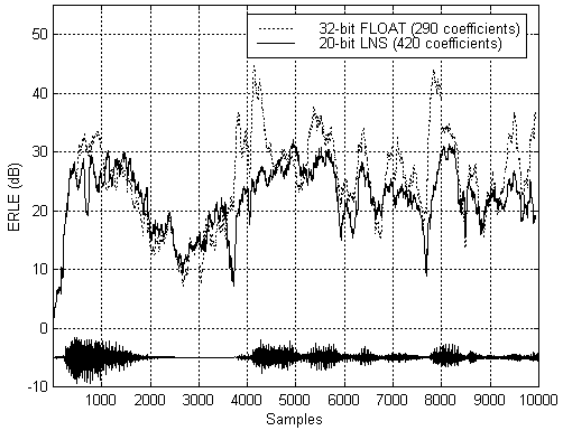

Fig. 5. The ERLE performances for 32-bit full precision FLOAT (290 coefficients) and 20-bit LNS (420 coefficients) implementations of the modified *A priori* EF-LSL Algorithm. Bottom trace: echo signal prior to cancellation

3. CONCLUSIONS

We have made a practical demonstration that an LNS implementation of the modified EF-LSL algorithm can run as accurately on an FPGA as a floating-point implementation on a contemporary microprocessor. A 20-bit LNS FPGA implementation would also be of interest, its much reduced complexity permitting functional replication and consequent improved performance arising from a higher filter order. In view of the superior fabrication technology of the microprocessor, these results suggest that a custom LNS based device, or an LNS microprocessor, might offer very substantial benefits indeed.

4. REFERENCES

[1] A. Carezia, P.M.S. Burt, M. Gerken, M. Miranda, M.T.M. da Silva, "A stable and efficient DSP implementation of a LSL algorithm for acoustic echo cancelling", pp. 921-924, ICASSP 2001, U.S.A

[2] S. Haykin, *Adaptive filter theory*, Prentice Hall Intl., 2nd edition, 1991

[3] Jin-Nan Li, Mou-San Ya, "Fast Maximum Kurtosis Decovolution (MKD) Blind Equalization Using Least Square Lattice (LSL) Algorithms", pp. 50-54, ICSPAT, 1997

[4] JN Coleman EI Chester CI Softley J Kadlec 'Arithmetic on the European Logarithmic Microprocessor' IEEE Trans. Computers, vol. 49 no 7 July 2000, pp 702-715 and erratum vol. 49 no 10 p 1152

[5] J. Kadlec, A. Hermanek, Ch.Softley, R. Matousek, M. Licko "32-bit Logarithmic ALU for Handel C 2.1 and Celoxica DK1 (53 MHz for XCV2000E-6 based RC1000 board)"http://www.celoxica.com/programs/university/academic_papers.htm

[6] M. D. Meyer, D. Agrawal, "Adaptive Lattice Filter Implementations on Pipelined Multiprocesor Architectures", IEEE Transactions on Communications, vol. 38, No1, pp. 122-124, 1990

[7] TMS320C3x General-Purpose Applications User's Guide http://www.ti.com/sc/docs/psheets/abstract/apps/spru194.htm

[8] http://www.celoxica.com/products/design_suite/index.htm

[9] F. Albu, J. Kadlec, C. Softley, R. Matousek, A. Hermanek, N. Coleman, A. Fagan, "Implementation of (Normalised) RLS Lattice on Virtex", FPL2001, pp. 91-100, Belfast, UK.

[10] RC1000-PP Hardware Reference Manual http://www.celoxica.com/products/boards/DATRHD001.2.pdf

[11] C. Breining, P. Dreitseitel, E. Hansler, A. Mader, B. Nitsch, H. Pudeer, T. Scheirtler, G. Schmidt, and J.Tilp, ' Acoustic echo control- An application of very high order adaptive filters,' *IEEE Signal Processing Magazine*, pp. 42-69, July 1999

The codes for the proposed algorithms can be obtained from
http://falbu.50webs.com/List_of_publications_lns.htm

The reference for the paper is:
F. Albu, J. Kadlec, N. Coleman, A. Fagan, "Pipelined Implementations of the *A Priori* Error-Feedback LSL Algorithm Using Logarithmic Number System", Proceedings of ICASSP 2002, pp. 2681-2684, Orlando, Florida, U.S.A, May 2002