Nonlinear Equalization of RF Receivers

Benjamin Miller[1], Gil Raz[2], Brandon Kam[1], Joel Goodman[1]
[1]Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, MA 02173-9108
{bamiller, bkam, jgoodman}@ll.mit.edu
[2]GMR Research & Technology, Inc., 1814 Main Street, Concord, MA 01742-3819
raz@gmrtech.com

Abstract

This abstract describes the need for High Performance Computing (HPC) to facilitate the development and implementation of a nonlinear equalizer that is capable of mitigating and/or eliminating nonlinear distortion to extend the dynamic range of Radar front-end receivers decades beyond the analog state-of-the-art. The search space for the optimal nonlinear equalization (NLEQ) solution is computationally intractable using only a single desktop computer. However, we have been able to leverage a combination of an efficient greedy search with the High Performance Computing technologies of LLGrid and MatlabMPI to construct a NLEQ architecture that is capable of extending the dynamic range of Radar front-end receivers by over 25dB.

Introduction

The linear dynamic range of RF receivers is a limiting factor in the performance of many high-end military Radar receivers. The nonlinear distortion generated in the analog-to-digital converter (ADC) and the analog receiver limits the capability of the backend Radar signal processor from unambiguously detecting targets with weak signatures. By reducing and/or eliminating nonlinear induced distortion after the ADC, NLEQ enables the Radar signal processor to detect weak target signals that would otherwise be masked by unwanted nonlinear artifacts.

The process of identifying an NLEQ architecture is computational onerous due to the vast size of the search space of candidate architectures and the computational complexity of evaluating each candidate. Historically, large polynomial representations were implemented using architectures such as the Volterra filter (see e.g., [1]). To overcome limitations in the over-parameterized Volterra representation, we have devised an alternate structure (Partitioned Horizontal Coordinate System - PHoCS) that enables us to efficiently search for and identify a robust and computationally efficient NLEQ architecture. In this abstract we describe PHoCS and the methods to identify computationally efficient equalization architectures by using a specialized greedy algorithm running on a large grid computer (LLgrid) using MatlabMPI [6] that is designed to run in parallel on host hardware.

This work is sponsored by the Defense Advanced Research Projects Administration under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Methodology

Using a Volterra filter model for our equalization problem is prohibitively expensive in terms of computational complexity; a p^{th} order Volterra kernel with a memory depth of L requires $\binom{L+p-1}{p}$ coefficients. For example a single 5th order Volterra kernel with a memory depth of 8 requires 792 coefficients. This makes the physical realization of a Volterra-based polynomial filter impractical for real-time implementation at high sample rates.

As an alternative to the Volterra filter method we have formulated the PHoCS architecture [2] which is a successor to the Diagonal Coordinate System (DCS) [3] architecture. The output of our nonlinear equalizer is the sum of the outputs of several PHoCS Processing Elements (PEs). Each PE as shown in Figure 1 is comprised of linear delays, linear filters and multipliers. Blocks $\alpha_l^j(n)$ are integer delay values and $g_{l,j}^{p_j}(q)$ are partitioning filters. The filters h_j are identified and optimized to best mitigate nonlinear distortion of the target receiver.

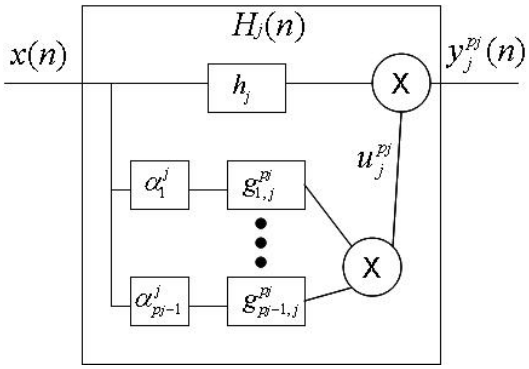

Figure 1: Representation of a PHoCS Processing Element

Software which we developed in Matlab is used to determine the appropriate PHoCS PEs that comprise an NLEQ processor. For a given NLEQ processor, each PE is chosen based on its composite equalization performance

from a pool of candidate PHoCS PEs. Because the processor needs to be implemented in hardware, we would like to get the most equalization performance from the least number of PHoCS PEs. So, the objective is to choose N PHoCS elements from a pool of M which have the very best performance. However, this search space can be very large; testing all permutations would require deriving coefficients for and testing $\binom{M}{N} \approx M^N$ PEs. For typical values of M and N this number can become prohibitively large.

To resolve this problem we have implemented a greedy algorithm that uses an incremental approach and searches for a locally optimal solution. In short, we choose the processing element that works best in conjunction with the PEs we have already chosen until we have N elements. This algorithm drastically reduces our search space since the greedy algorithm only requires the examination of approximately $M \times N$ cases (see Table 1).

Typical Values	$\binom{M}{N}$	$M \times N$
M = 1,200, N = 20	10^{43}	24,000
M = 11,400, N = 80	10^{205}	912,000

Table 1: Greedy algorithm provides computational tractability for NLEQ

The Need for High Performance Computing Technologies

While the real-time operation of the equalizer is parsimonious, the computational complexity of researching new NLEQ architectures based on PHoCS is quite challenging. Each step in the construction and evaluation of an NLEQ architecture requires many floating point operations (FLOPs). To put the amount of computations into perspective, we estimate the computational cost of identifying an NLEQ architecture that determines the performance limit of the NLEQ software.

There are three major steps in the NLEQ software. In the first step, the major computational bottleneck is the translation of the time domain data to the frequency domain via the FFT, the computational cost of which is given by $5N\log_2(N)$ [4]. The QR-decomposition for determining a least squares solution dominates the computation in the second step. This uses modified Gram-Schmidt which requires $2mn^2$ FLOPs [5]. The third step is dominated by sending all signals through the NLEQ processor. In our example simulation, the numbers of FLOPs for of these three steps are approximately 456 trillion, 19 trillion, and 55 trillion, respectively, for a total of 530 trillion FLOPs. Each step requires significant computational power and the simulation would be intractable if not for such enabling HPC technologies as MatlabMPI [6] and the LLGrid [7] supercomputing system.

Results

MatlabMPI is a HPC technology that leverages the Message Passing Interface (MPI) standard for parallel computing within the Matlab environment. It was developed to be an easy to use technology that incorporates itself almost seamlessly with Matlab. The source code can be found on the MatlabMPI website [6]. LLGrid is the grid computing system used at MIT Lincoln Laboratory that enables individual users easy access to a powerful computing cluster. The LLGrid and MatlabMPI platforms in conjunction are particularly suitable to our computational needs, as they provide the power of parallel computing in a scientist-friendly way.

Parallelizing the NLEQ software over the LLGrid cluster significantly reduces the run times. Figure 2 depicts the achieved speedup compared to running on a single processor. For reference we show here the ideal linear speedup. Compared to the run time of 35.4 hours on a single processor for a small NLEQ simulation run, on 4, 16, and 64 processors we can achieve run times of 10.9, 3.9, and 1.3 hours, respectively.

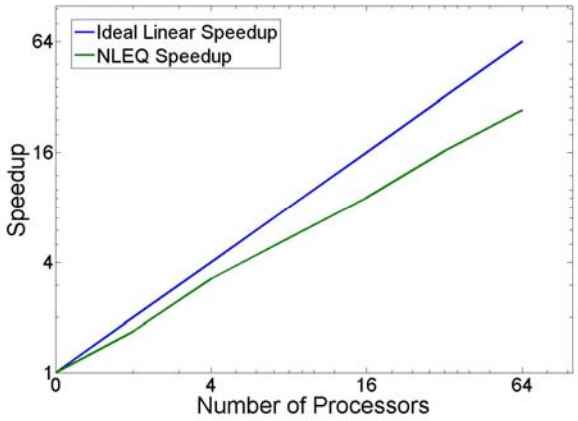

Figure 2: Speedup from parallelization vs. the "ideal" linear speedup

References

[1] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*, John Wiley & Sons, 2000

[2] C.P. Chan, G.M. Raz, *Nonlinear Equalization Preliminary Design Review: PHoCS and HCS Signal Processing – Improving Linearity in Receiver Systems*, Lincoln Laboratory Project Report NLEQ-2, September 2005

[3] G. M. Raz, *Highly Linear Analog-to-Digital Conversion System and Method Thereof*, U.S. Patent #6,639,537, 2003

[4] M. Arakawa, *Computational Workloads for Commonly Used Signal Processing Kernels*, Lincoln Laboratory Project Report SPR-9, May 2003

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations: Third Edition*, The Johns Hopkins University Press, 1996

[6] J. Kepner, http://www.ll.mit.edu/MatlabMPI

[7] A. Reuther, htttp://www.ll.mit.edu/HPEC/agendas/proc04/Talks/Wed/Focus2/reuther.pdf