LOW POWER SPARSE POLYNOMIAL EQUALIZER (SPEQ) FOR NONLINEAR DIGITAL COMPENSATION OF AN ACTIVE ANTI-ALIAS FILTER

Karen Gettings, Andrew Bolstad, Show-Yah Stuart Chen, Michael Ericson, Benjamin A. Miller, Michael Vai

{karen.gettings, andrew.bolstad, stuart.chen, michael.ericson, bamiller, mvai}@ll.mit.edu

MIT Lincoln Laboratory

ABSTRACT

We present an efficient architecture to perform on-chip nonlinear equalization of an anti-alias RF filter. The sparse polynomial equalizer (SPEq) achieves substantial power savings through co-design of the equalizer and the filter, which allows including the right number of processing elements, filter taps, and bits to maximize performance and minimize power consumption. The architecture was implemented in VHDL and fabricated in CMOS 65 nm technology. Testing results show that undesired spurs are suppressed to near the noise floor, improving the system’s spur-free dynamic range by 25 dB in the median case, and consuming less than 12 mW of core power when operating at 200 MHz.

Index Terms— Nonlinear equalization, filter compensation, low power architectures, algorithm/hardware co-design

1. INTRODUCTION

Nonlinear equalization (NLEQ) can reduce or eliminate unwanted spurs in signals. This enables detection of weak target signals that would otherwise be masked by interference [1], as shown in Figure 1.

Previous work [2] has implemented NLEQ algorithms in an application-specific integrated circuit (ASIC) to drastically improve the intermod-free dynamic range of commercial ADCs by up to 21 dB over a 500 MHz receiver passband, and up to 30 dB for narrower bands. This is a significant improvement, because according to [3] the dynamic range of commercial ADC has historically improved at a rate of approximately 1 dB per year.

The stand-alone ASIC in [2] was built using systolic arrays to achieve the processing power required to implement the full NLEQ algorithm, requiring over $10^{12}$ operations per second. The design was implemented with custom cells using 0.25 μm (1.2V) CMOS technology to achieve core power consumption of 122 mW at 500 MSPS. This architecture is able to compensate a large array of front ends for a large variety of applications.

In contrast, system-on-chip (SOC) implementations tend to cater to specific needs, but are an attractive solution for size, weight, and power (SWaP) restricted applications, such as mobile devices and unmanned aerial vehicles. We took advantage of the capabilities of NLEQ by designing a processor that linearizes the operation of an active anti-aliasing filter [4], as part of a low-power homodyne SOC receiver design. Targeting a specific system implementation as opposed to using a general purpose solution allows significant savings in both the power consumed and the die area used for compensation.

The filter design in [4] targets a highly linear anti-alias filter with 30 MHz bandwidth and 90 dB stopband rejection, which is higher than industry standard. It is also SWaP constrained, therefore large LC filters are unsuitable. As a trade-off, the filter was designed with limited linearity, and was co-designed with the sparse polynomial equalizer (SPEq) processor presented in this paper to achieve the desired spur-free dynamic range. The analog filter and digital compensator have been fabricated and tested independently, and will be integrated together in a SOC in the near future.
2. NONLINEAR SIGNAL PROCESSING OVERVIEW

To compensate for the nonlinear behavior of the RF circuitry, we derive a nonlinear inverse function of the system response. Memory effects (i.e., state-dependent behavior) of RF devices often complicate the modeling of nonlinear behavior. A general nonlinear finite impulse response (FIR) model for systems with significant memory is the Volterra series [5], whose output is given by:

\[
y_{NL}(n) = \sum_{p=0}^{P} \sum_{m_1=0}^{M} \cdots \sum_{m_p=0}^{M} h_p(m_1, \ldots, m_p) \prod_{i=1}^{P} x(n - m_i).
\]

This model generalizes the linear FIR filter to polynomial combinations of the input.

While this representation captures general nonlinear behavior, its complexity is combinatorial in memory depth (\(M\)), and systems requiring real-time performance will typically use a “pruned” version of the kernel. The approach we take is similar to that of [1], in which the full coefficient space is divided into subspaces, only a few of which are selected for use in the compensator.

We further restricted the architecture to conserve power. Rather than operate over the entire Volterra kernel coefficient space as in [1], SPEq uses coefficients of the generalized memory polynomial (GMP) architecture of [6]. In this model the nonlinear output (neglecting the constant \(h_0\) and linear \(h_1\) terms) is given by

\[
y(n) = \sum_{p=2}^{P} \sum_{m_1=0}^{M} \sum_{m_2=0}^{M} h_p(m_1, m_2) x(n - m_1) x^{p-1}(n - m_1 - m_2),
\]

i.e., it is restricted to the coefficients lying on a 2-dimensional plane within the larger coefficient space. This decision costs us flexibility—we can no longer choose coefficients from arbitrary portions of the space—but allows a simple, power-efficient implementation. From the set of possible GMP coefficients, we select a small number (up to five in the current implementation) of non-zero coefficients using a sparse signal estimation procedure similar to the orthogonal matching pursuit (OMP) algorithm [7] popular in compressed sensing. Using architectures that include several contiguous coefficients as in [1] can be more power-efficient per coefficient in a larger system, but we have empirically observed that allowing the procedure to choose individual coefficients allows a greater initial dynamic range improvement with few coefficients. While constrained, this implementation is sufficient to compensate the targeted RF front-end of [4], as we demonstrate in subsequent sections.

3. IMPLEMENTATION

The SPEq processor was designed in VHDL and was subsequently synthesized into a standard cell design, using IBM 65 nm CMOS technology (1.2V) and focusing on a low power implementation. The VHDL description was written with portability in mind, which allows easy conversion to other technologies and/or needs.

A block diagram of the equalizer is presented in Figure 2. It consists of the following building blocks: Two’s Complement Conversion, Global Exponentiation, Processing Elements (PEs), Shifters and a Final Accumulator (\(\Sigma\)).

![Figure 2: Block diagram of equalizer.](image)

3.1. Two’s Complement Conversion

Our algorithm works on two’s complement format. The Two’s Complement Conversion block changes the front-end receiver’s data format to two’s complement if needed.

3.2. Global Exponentiation

In order to save power, the input signal from the front-end receiver is truncated to the most significant 8 bits. These 8 bits are subsequently exponentiated to powers ranging from 2 to 4, hence polynomial combinations of the input, \(x(n)\) to \(x^4(n)\), are available for processing, as shown in Figure 3.

![Figure 3: Global exponentiation block.](image)
of creating custom exponentiated signals per processing element (PE), these signals are created only once, and the output can feed all of the processing elements with no more exponentiation needed. Each PE can independently select which order it will use for processing via a multiplexer, as shown in Figure 4. This order is determined during training, and it is fed to each PE as a control signal. We compute all 16 bits per multiplication, and then truncate to 8 bits for the rest of the processing. This is to avoid losing accuracy due to the rounding at early stages during the chain multiplication.

3.3. Processing Element

The processing element is the main building block of the equalization circuitry. Its architecture includes a multiplexer, two delay blocks, and two multipliers, as depicted in Figure 4.

The top delay block (labeled Sub-PE in Figure 4) takes in the top (MSB) 8 bits of the input signal from the front-end receiver. The bottom delay block takes in either the same top 8-bit input signal or one of its exponentiated forms, from \( x(n) \) to \( x^4(n) \). The signals are delayed in each Sub-PE independently and then multiplied. The delay required is determined during training, and is fed to each Sub-PE as a control signal. For power savings only the top 8 bits of the product of the two Sub-PE outputs are kept, and it is subsequently multiplied with the coefficient calculated for that specific PE during training.

Each PE implements a single term in the GMP series rather than a contiguous block, as described in Section 2. It was determined via simulation that 8 delays per delay block and 3 PEs were sufficient to compensate our front-end circuitry while maintaining our low power requirement; however, two additional PEs were included to mitigate nonlinear effects potentially not captured in simulation. The separation of exponentiation from individual PEs allows additional PEs to be added with little additional power consumption (< 1mW per additional PE).

3.4. Shifter

The 8-bit output of each PE is sign-extended and shifted (multiplied) before being added to the uncompensated 16-bit input signal \( x(n) \). Shifting allows a greater range of coefficients \( h_p(m_1, m_2) \) to be used for more accurate compensation. The system’s input is 14 bits, and the shifter extends up to 16 bits to allow for larger dynamic range compensation. The case statements used to build the shifter are shown in Figure 5.

### Shifter Case Statements:

- if sft\_control = 0 → sign[15:10] & pe\_value[7:0] & [0 0]
- if sft\_control = 1 → sign[15:9] & pe\_value[7:0] & [0 ]
- if sft\_control = 2 → sign[15:8] & pe\_value[7:0]
- if sft\_control = 3 → sign[15:7] & pe\_value[7:1]
- if sft\_control = 4 → sign[15:6] & pe\_value[7:2]
- if sft\_control = 5 → sign[15:5] & pe\_value[7:3]
- if sft\_control = 7 → sign[15:3] & pe\_value[7:5]
- if sft\_control = 9 → sign[15:1] & pe\_value[7:7]

Figure 5: Shifter implementation.

3.5. Final Accumulator

After the output of each PE has been shifted into the appropriate 8-bit range within the 16 bits of the output word, these values are added to the input signal in the Final Accumulator. This can be viewed as subtracting nonlinear effects from \( x(n) \), where subtraction is implemented by negating coefficients.

4. CHIP TESTING RESULTS

Our SPEq design was implemented in VHDL, synthesized and fabricated in CMOS 65 nm (1.2V) technology. The core of the chip measures 200 μm by 200 μm. It was fabricated
as a stand-alone chip for testing purposes, but will be integrated with the filter in a future system. The small core is roughly the size of 4 bonding pads, and a large amount of compensation capacitors were added to fill in the die area inside the pad ring, as shown in Figure 6.

![Figure 6: SPEq chip.](image)

We excited the active filter [4] with 118 two-tone signals at various frequencies and tone spacings. These signals were divided into two groups: 94 were used to train the SPEq compensator, choosing the delays and computing the coefficients; and the remaining data were used for validation. The results shown here were computed in Matlab using floating point math and measured front end filter data. The SPEq processor hardware was successfully verified with a subset of the data.

SPEq was able to suppress undesired spurs of polynomial nature close to the noise floor, as shown in Figure 7 and Figure 8. By lowering these nonlinear distortions SPEq increases the spur free dynamic range by approximately 16 dB in this example. The dataset shown was one of the hardest datasets to compensate, and greater suppression is possible in other datasets, usually driving the undesired spurs to the noise floor. SPEq can achieve 25 dB improvement for median dynamic range, as shown in the cumulative density functions of largest distortion levels before and after equalization in Figure 9. The results shown here were calculated using 5 processing elements (PEs), and additional work has shown that performance improves with a larger number of PEs, but power is adversely affected.

The core power of the chip was measured at 12 mW when operating at 200 MHz (targeted frequency), and 15 mW at 240 MHz.

![Figure 7: FFT of filter output before SPEq.](image)

![Figure 8: FFT of filter output after SPEq.](image)

![Figure 9: Maximum distortion CDF for filter before and after SPEq.](image)
5. CONCLUSIONS

Our sparse polynomial equalizer (SPEq) was codesigned for a specific front-end: an active low-pass anti-alias filter. The digital equalizer improves the system’s dynamic range by 25 dB in the median case. The processor was implemented in CMOS 65 nm technology and consumes only 12 mW of power at a sampling rate of 200 MHz, with a footprint of approximately 0.04 mm². The architecture, which is implemented in RTL, is both modular and technology independent, and can easily be used and modified for other front-end designs.

6. ACKNOWLEDGEMENT

This work is sponsored by the Department of the Air Force 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.

The authors wish to thank Helen Kim, Merlin Green, Daniel Santiago, Ford Ennis, and Joel Goodman for their significant contributions to this work.

7. REFERENCES