A Reduced-Memory FIR Filter Using Approximate Coefficients for Ultra-Low Power SoCs

Alicia Klinefelter and Benton H. Calhoun
University of Virginia, Charlottesville, VA 22903 USA, Email: {amk5vx, bcalhoun}@virginia.edu

***Abstract*-This paper presents an ultra-low power (ULP) finite-impulse response (FIR) filter using a method that approximates filter coefficients on-chip without reliance on dedicated memory such as SRAM. In a system-on-chip (SoC) context, this method allows for full power gating of the coefficient unit without coefficient state loss, and runtime modifications of filtering specifications, such as filter order and cutoff frequency. Using trigonometric approximation methods for the sinc and resource sharing of computational units, a single coefficient is generated in five clock cycles. The approximation unit is compared against standard-cell-based memories, such as register and latch files, for energy and area, and the design is synthesized in 130nm CMOS consuming 6.9nW at 300mV and 6.5kHz.**

I. Introduction

Supply voltage scaling into the subthreshold region of operation is becoming an increasingly attractive solution to save energy and power in cases where performance is not the driving factor. Applications with low-speed requirements, such as environmental and physiological monitoring, can operate at processing frequencies in the 10-100kHz range, while the acquired signals, such as electrocardiographs (ECGs), have sampling rates in the hundreds of Hz. This leaves hundreds to thousands of clock cycles between samples for processing, while enabling operation in the subthreshold region. In this region, where leakage energy begins to dominate, serializing logic at the cost of more clock cycles has little impact on energy consumption [1]. Flexible and programmable biomedical SoCs often require large sets of filtering coefficients during deployment for a variety of scenarios (e.g. removing 60Hz power line noise or band selection). This can lead to large amounts of coefficient storage, and for batteryless systems such as in [2], an intermittent supply can lead to complete coefficient state loss and the need for reprogramming.

Generally, an FIR filter's coefficients are stored in an on-chip SRAM or register file that must also operate in subthreshold. The SoC's data memory is rarely power-gated as it also stores chip data that may be unrelated to the filtering process. This leads to excess leakage in memory dedicated to stored coefficients when the filter is not being used. For ULP SoCs, on-chip SRAMs can consume ~60% of the total digital power, and if not power gated can dominate the digital power budget [3]. They are often the primary barrier to low-voltage operation and pose the first failure point in low-voltage designs [1]. Additionally, in the deep subthreshold region, decreased I_{on}/I_{off} ratios reduce the reliability of SRAM. Standard-cell-based coefficient memories that are local to the accelerator to hold data and coefficients are an alternative, but this puts an upper bound on the order of the filter and becomes energy and area inefficient for sizes >1kb [4]. By generating the coefficients in real-time using digital, synthesized logic, design robustness is improved and the minimum supply voltage and leakage is decreased without the need for SRAM-based storage.

II. Methods of Approximation

Assuming a symmetric FIR filter, the filter can be implemeted using a folded delay line shown in (1).

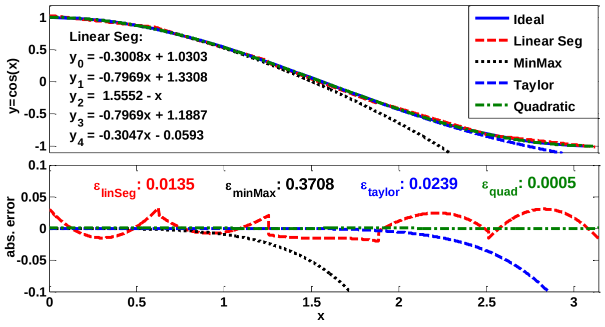

Fig. 1. Approximations (top) and absolute error (bottom) for cosine [0, π]. Average error, ε, is show for each method.

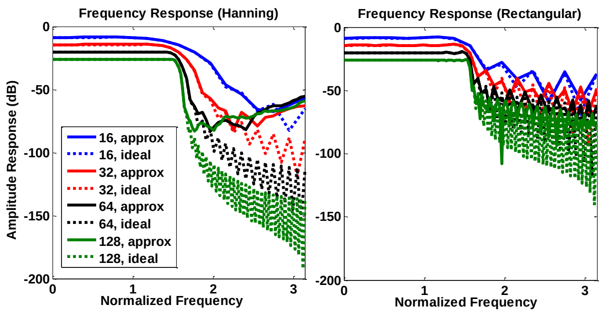

Fig. 2. Ideal and approximated frequency response functions for f_c=0.5π using a rectangular and Hanning window.

$$y(i) = \sum_{k=0}^{M/2} h(k)[x(i-k) + x(k)] \quad (1)$$

Here, *x* is the sampled and delayed input data, *h* is the array of coefficients, *M* is the filter order, and *y* is the filtered output. This form reduces the number of multiplications by a factor of two relative to the classical form. FIR filtering coefficients are sampled points along the sinc function, *sin(x)/x*, where windowing functions are applied to smooth the effects of truncating this infinite function. Although there are other methods of creating filter coefficients such as frequency sampling and least-squared, many of these are computationally complex requiring Fast-Fourier Transforms or optimization methods.

To generate the sinc, the sine function must be approximated in a way that isn't "too complex" to implement in hardware, but is also "accurate enough". Four methods were evaluated and the results are shown in Fig. 1. A common approximation used for trigonometric functions is the Taylor series. This series is expensive to compute for accurate results (i.e. more terms) and the resulting error is only acceptably low for small values of the input argument [5]. A 7th order approximation was used here, but it required seven multiplications, three divisions, and three additions to compute the result. An alternative method uses minmax polynomials. This method is often preferred over the Taylor polynomial method due to its distributed error over the range, which also decreases with additional terms, but suffers from the same computational complexity issue as Taylor's approximation [5]. The third method, a quadratic approximation, uses a parabola to approximate the sine/cosine function across the

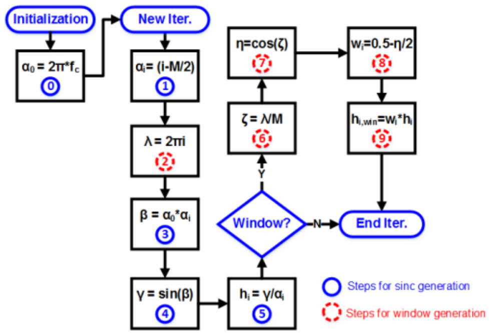

Fig. 3. Data-flow graph showing the coefficient generation algorithm using a Hanning window. Circled numbers indicate the current clock cycle (state).

[0, π] range, but requires three total multiplications per approximation. The final method uses linear segments to approximate the cosine and thus requires only one multiplication per computation. An unconstrained nonlinear optimization method was used to choose slope and intercept values that reduced the residual sum of squares of the fitted lines, and the segments are shown in Fig. 1. Five linear segments were used to achieve comparable error with the other, higher order methods, while requiring fewer hardware resources, and this was used for the final design.

I. Coefficient Generation

Using a low-pass filter as an example, the methodology for coefficient generation is described here. First, the filter specification is defined by providing the desired cutoff frequency, f_c, and the filter order. From this, an ideal impulse response function (sinc) can be generated (2).

$$h(i) = \frac{\sin(2\pi f_c(i-M/2))}{i-M/2} \tag{2}$$

This impulse response is delayed to keep the system causal, and the sinc function is multiplied by a specified window function and sampled to obtain coefficients. Many of the most common windows such as Hanning (3), Hamming, or Blackman contain cosine functions that require an additional trigonometric computation. For this purpose, the approximation unit is dual-purposed for both sine and cosine approximations, using a shift of π/2.

$$w(i) = 0.5 - 0.5\cos(2\pi i/M) \tag{3}$$

The approximated frequency response functions were compared with the ideal response in Fig. 2 for both rectangular ($w(i) = 1, \forall\, i$) and Hanning windows. Although the passband regions appear unaffected by the approximation, the stopband attenuation is slightly degraded for the approximated cases.

Since this algorithm can be mapped to hardware in a variety of ways, the specific, serialialized method is shown in Fig. 3. Due to the resource-shared hardware, state order had to be carefully chosen to prevent combinational loops.

II. Results

To determine the suitability of this method in a real system, an approximation and multiply-accumulate (MAC) unit were taped out in a 130nm commercial process with chip micrograph and logical area distribution shown in Fig. 4. Fig. 5 shows the measured energy results of the approximation unit against the simulated results of 16-bit register/latch files of different

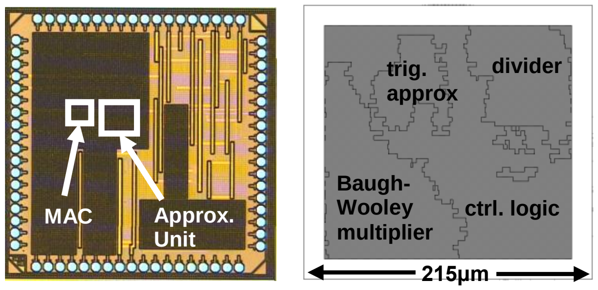

Fig. 4. Chip micrograph (left) and approximation unit division of area (right).

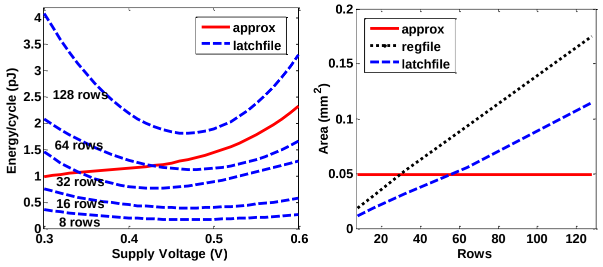

Fig. 5. Energy and area comparison of the approximation unit with 16-bit register and latch files of various depths.

depths/rows, corresponding to number of taps. From this, the point at which this method becomes more advantageous than standard-cell-based coefficient storage is seen with respect to supply voltage. Since the area of the approximation unit remains constant for any number of tap values, it becomes more area efficient than a register file at ~30 rows/taps and a latch file at ~55 rows/taps. The design was synthesized in 130nm CMOS and was tested to operate down to 300mV where it consumed an average energy of 1pJ/state.

Conclusion

This work presents a methodology for generating coefficients on-chip in lieu of storing them in high-leakage SRAMs operating below the threshold voltage. This design can enable reliable, ULP operation for applications requiring large banks of coefficient data and diverse filtering operations. By evaluating a variety of trignometric approximation methods for the complexity-accuracy tradeoff, a linear segment-based approximation was selected to generate coefficients. Although this coefficient generation adds latency to the filtering operation, there is often available slack for serial processing in low-throughput, subthreshold systems.

Acknowledgements

This work is funded by the NSF NERC ASSIST Center (EEC-1160483).

References

[1] Verma, N., et al., "Nanometer MOSFET Variation in Minimum Energy Subthreshold Circuits," T-ED, 2008.
[2] Zhang Y., et al., “A batteryless 19 µW MICS/ISM-band energy harvesting body area sesor node SoC for ExG applications,” JSSC, 2013.
[3] Khayatzadeh M., et al., "A 0.7-V 17.4-µW 3-Lead Wireless ECG SoC," BIOCAS, 2013.
[4] Meinerzhagen, P., et al., "Towards generic low-power area-efficient standard cell based memory architectures," MWSCAS, 2010.
[5] Muller, J-M., “Elementary Functions: Algorithms and Implementation,” Birkhäuser Publishing, 2005.