Word-Length Optimization of Folded Polynomial Evaluation*

George A. Constantinides, Abunaser Miah, Nalin Sidahao
Electrical and Electronic Engineering Department
Imperial College London
george.constantinides@ieee.org, abunaser.miah@ic.ac.uk, nalin.sidahao@ic.ac.uk

Abstract

This extended abstract presents further results from the word-length optimization system Right-Size described at FCCM 2003. The system is used to quantify the compile-time specialization savings made possible through unfolding a polynomial evaluation architecture.

1. Introduction

A general system for word-length optimization of non-linear systems was introduced at FCCM 2003 [1], and exemplified through the design of least-mean-square adaptive filters. This extended abstract presents some further results from the Right-Size system relating to word-length optimization of polynomial evaluation schemes, typically used to approximate to elementary and composite functions in hardware design [2].

In particular, the efficient design of *folded* polynomial architectures is explored. These architectures, described in [4], allow a controlled throughput/area tradeoff by reusing portions of the architecture to perform multiple operations over several different clock cycles.

Although there are many existing schemes for polynomial evaluation [3], to our knowledge, the circuit area implications of roundoff error toleration have not been significantly explored for any of these schemes. In addition, folded polynomial evaluation forms an interesting case study for word-length optimization, as the more resource sharing there is, the less specialized each resource can be, leading to an area overhead for folded architectures which has not previously been studied.

This abstract therefore makes the following novel contributions:

- to our knowledge, the first description of the application of word-length optimization techniques to polynomial evaluation schemes.
- the first quantification of the area savings possible due to tolerable round-off errors through increased *compile-time specialization* of polynomial evaluation circuitry.

[0]This work was supported by the Nuffield Foundation under grant number URB/01106/G

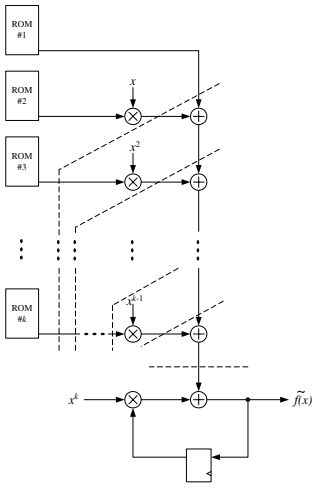


Figure 1. General structure of a folded polynomial evaluation architecture. The dashed lines represent pipeline stages.

2. Folded Polynomial Evaluation

A diagram showing the general form of a folded polynomial evaluation architecture is shown in Fig. 1. This architecture, adapted from [4], is capable of producing one nth order polynomial evaluation every $\lceil (n+1)/k \rceil$ cycles and each ROM contains $\lceil (n+1)/k \rceil$, entries where k is an unfolding factor.

Clearly as k increases, the degree of *specialization* increases, as the number entries in each ROM is reduced. While for any folded architecture, the multipliers are general, rather than constant-coefficient multipliers, this in-

Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)
0-7695-2230-0/04 $ 20.00 IEEE

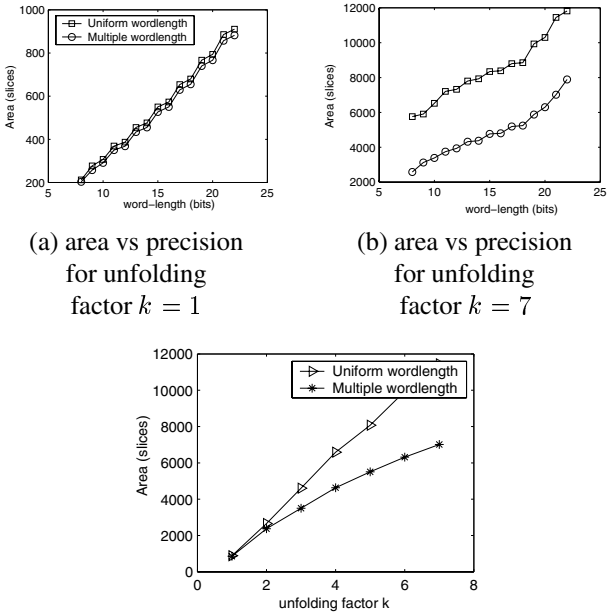
creasing degree of specialization can have significant implications on reducing the number of 'guard' bits required to achieve a faithfully rounded result.

3. Nonlinear Modelling

The Right-Size tool described in [1] requires each node to compute be a differentiable function of its inputs, and uses signal-to-noise ratio at the system output as a metric of signal quality.

While the addition and multiplication nodes in Fig. 1 are differentiable, the ROM nodes are not. Thus we merge the ROMs and the counter driving the ROMs into a single zero-input node for modelling in Right-Size, avoiding the problem. This is possible because we do not aim to optimize the precision required on the ROM address bus.

On the output side, the user is typically uninterested in intermediate values not corresponding to a complete polynomial evaluation. Thus we insert a decimation-by-$\lceil (n+1)/k \rceil$ block at the system output. For the purposes of system modelling, this is considered to be a *differentiable nonlinear time-varying* block, with unit first derivative every $\lceil (n+1)/k \rceil$ samples, and zero first derivative for the remaining samples.



(a) area vs precision for unfolding factor $k = 1$

(b) area vs precision for unfolding factor $k = 7$

(c) area vs unfolding factor k for 21-bit precision

Figure 2. Area required for a sin function approximated by a 7th order polynomial

4. Results

Fig. 2(a) illustrates the change in area requirements for implementing a sin function as the precision of the required function increases, for a fully-folded architecture, with a throughput of one evaluation every 8 cycles. Two plots are shown: one is the *optimum* uniform word-length implementation, and one is the optimized multiple word-length implementation resulting from the Right-Size tool. The latter consistently consumes less area than the former, but only by an average of 5%.

By contrast, Fig. 2(b) illustrates the same plots for a significantly unfolded architecture, *i.e.* one with a throughput of one evaluation every 2 cycles. This time the difference between the two design approaches averages 50%.

This trend is brought out in Fig. 2(c), where area is plotted against unfolding factor for a fixed precision. As the unfolding factor increases, the two curves separate. This is a novel observation, as usually the area is modelled as an approximately linear function of the unfolding factor, as the number of multipliers and adders grows linearly with unfolding factor. This is indeed a visible trend in the uniform word-length implementation. However the multiple word-length area grows *sub-linearly* in unfolding factor. This result can be explained, as Right-Size will automatically select the necessary scaling and word-length for each signal in order to maintain the required precision. The increasing specialization of the arithmetic components as the unfolding factor increases, results in significantly increased scope for optimization.

5. Conclusions

This extended abstract has illustrated the significant effect of word-length optimization on the problem of designing polynomial evaluation circuits. In future we intend to extend Right-Size to be usable on systems containing non-differentiable nonlinearities.

References

[1] G. A. Constantinides. Perturbation analysis for word-length optimization. In *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, Napa, CA, April 2003.

[2] J. Muller. *Elementary Functions*. Springer-Verlag, Berlin, 1997.

[3] N. Sidahao. Optimized FPGA-based function evaluation, 2003. M.Phil/Ph.D. Transfer Report, Imperial College London.

[4] N. Sidahao, G. A. Constantinides, and P. Y. K. Cheung. Architectures for function evaluation on FPGAs. In *Proc. IEEE International Symposium on Circuits and Systems*, 2003.

Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04)
0-7695-2230-0/04 $ 20.00 IEEE