MIGRATING FUNCTIONALITY FROM ROMS TO EMBEDDED MULTIPLIERS

Gareth W. Morris, George A. Constantinides, and Peter Y.K. Cheung

Imperial College, Circuits and Systems Group
Department of Electronic and Electrical Engineering
Exhibition Road, London, SW7 2BT

ABSTRACT

This poster proposes a technique, based on polynomial approximation, which can be applied to convert ROMs into a combination of arithmetic operations and smaller ROMs. We show that this technique highlights new areas of the multiplier/4LUT design space over existing methods.

1. BACKGROUND

Contemporary FPGA architectures include embedded RAM blocks and multipliers. By using the approach suggested it is possible to convert from ROMs and LUTs into embedded multipliers and a small number of LUTs. The conversion uses a variation on polynomial approximation. As a result, resources in FPGAs become more fluid and different possibilities for synthesising designs become available; for instance, allowing a high level synthesis system to target the number of embedded MULTs or RAM blocks used.

A major drawback of a simple polynomial approximation method is the worst case exponential approximation order. In order to mitigate this problem for common functions, uniform piecewise polynomial approximation [1], or bi-/multi-partite table method [2] have often been used in the past.

2. PROPOSED APPROACH

The proposed approach uses different coefficients depending on the address bus x, and these coefficients exist over independent ranges. This is achieved with the architecture in figure 1, with implications in the function domain shown in figure 2. Instead of single hardwired coefficients, a look up table containing many possible coefficient values is used. These coefficients are selected via an address bus input derived by a defined masking of the main address bus; requiring zero hardware overhead in an FPGA implementation. Simple and uniform piecewise polynomial approximations are subsets of this technique, for certain maskings.

The coefficient values are calculated via linear program solution, which allows each coefficient to be treated separately. For example, a set of linear equations representing

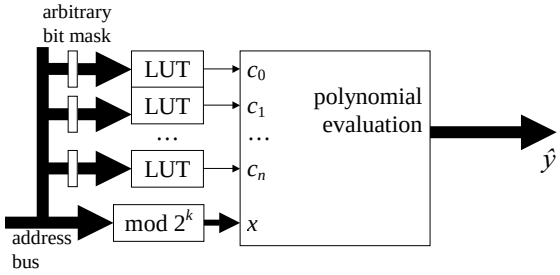

Fig. 1. Architecture of the proposed technique.

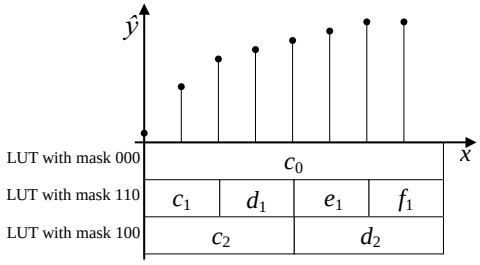

Fig. 2. Example showing implications of the proposed technique in the function domain.

the design in figure 2, having bit masks of 000, 110 and 100 for each coefficient LUT respectively, has 4 coefficients in the c_1 table and 2 in the c_2 table.

As well as allowing different values for each coefficient used in the polynomial evaluation, we make our architecture more general by applying a modulus of value 2^k to x before evaluating the polynomial. This modulus reduces the dynamic range, and therefore precision required, by the polynomial coefficients.

To incorporate bi- and multi-partite table methods into our design space we consider $k = 0$ as a special case. Normally in this case the x input to the evaluation would be constant 0; our modification is to take constant 1. The evaluation then reduces to a summation of the coefficient lookup table outputs. Bi- and multi- partite table methods are now a subset of the approach where the bit masks for each lookup table fit the relevant definition. Bi- and multi-partite table

methods described in the literature use Taylor series to calculate the table contents, whereas our system uses linear programming resulting in a minimax solution, thus reducing the precision requirement at the expense of increased computation time.

The coefficients stored in each LUT can be selected to reduce the number of 4LUTs required over standard techniques. Completely searching all combinations of masks is realistic only for smaller designs due to exponential computational complexity. Therefore we use a simulated annealing optimisation. The design space explored is that of the address bus masks and k. The objective function is the number of 4LUTs used in an FPGA implementation, calculated via a high-level model of logic synthesis.

3. RESULTS

The least-cost designs from the exploration of our design space have been compared against the *optimum* designs using bit masks of standard techniques.

The comparison was performed with ROMs of between 32 and 1024 entries, containing data of the functions $\log_2(1+2^{-x})$, $\log_2(1-2^{-x})$ (used for addition and subtraction in logarithmic number systems) and $\sin(x)$. Only designs faithful to the original ROM after rounding were considered.

Figure 3 shows the Pareto-optimal design curves for a 256 entry ROM with data from the LNS addition function. It shows four additional points on the Pareto-optimal design curve due to the proposed technique.

The improvement over uniform piecewise polynomial approximation increases with approximation order: figure 3 shows that our technique allows higher order approximations to be used with reduction in 4LUT usage up to at least 4th order, rather than an increase (found with uniform piecewise polynomials above 2nd order).

At least a single new point on the Pareto-optimal design curve was obtained with our technique in 94% of the designs examined, with 44% of designs having two or more new points. With the comparatively small ROMs under comparison, bi- and multi-partite tables are not able to reduce 4LUT usage. However savings can still be made with the proposed technique even without using multipliers, as shown in figure 3.

Five designs for implementing a 1024 entry $\log_2(1+2^{-x})$, look-up were constructed using coefficient values returned by our tools. The latency statistics, without pipelining, for each of these designs are shown in table 1. The proposed technique exhibits a 5.4 times increase in latency compared to a Block RAM solution, however the transformation has resulted in a complete transfer of Block RAMs to other resources. Similarly the increase in latency of 3 times compared to a 4LUT ROM has allowed a reduction in 4LUT use by 84%.

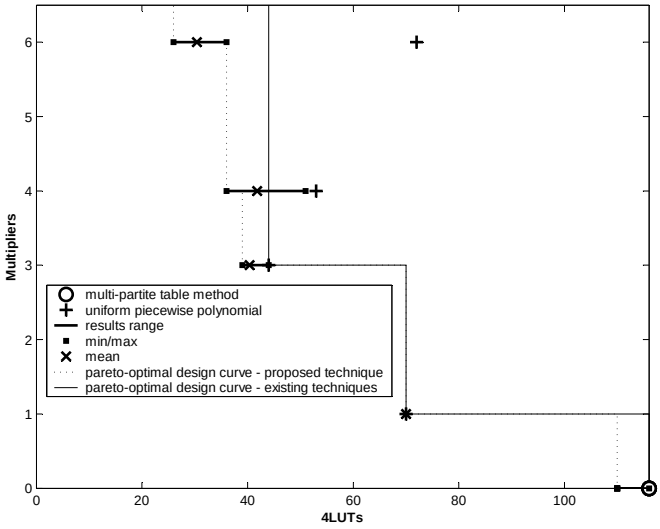

Fig. 3. Design space plot for a 256 entry ROM for the function $\log_2(1+2^{-x})$

Technique	Latency
Single Block RAM	4.57ns
Single LUT RAM	8.21ns
Multi-partite	8.21ns
Uniform piecewise polynomial	24.9ns
Proposed technique	24.9ns

Table 1. Comparison of design latency with our technique against standard approaches.

4. CONCLUSIONS

A novel method has been presented for converting functionality implemented in ROMs into multipliers and smaller ROMs. A polynomial approximation, modified by replacing each fixed coefficient with a lookup table addressed by arbitrary bits of the main address bus, is used. The design space is optimised using simulated annealing.

Hardware utilisation after applying the technique has been compared to standard methods. Results show an average of 18.6% improvement in 4LUT utilisation, with new areas of the multiplier/4LUT design space found in all designs examined.

5. REFERENCES

[1] T. Pavlidis and A. Maika, "Uniform piecewise polynomial approximation with variable joints," *Journal of Approximation Theory*, vol. 12, pp. 61–69, 1974.

[2] F. de Dinechin and A. Tisserand, "Some improvements on multipartite table methods," in *Proc. 15th IEEE Symposium on Computer Arithmetic*, 2001.