ROM to DSP block transfer for resource constrained synthesis

G.W. Morris, G.A. Constantinides and P.Y.K. Cheung

Abstract: Modern field programmable gate array (FPGA) architectures are moving towards heterogeneity with the increasing inclusion of coarse grained elements such as embedded multipliers and RAMs. This has given rise to a multi-dimensioned resource-based measure of design area, very different from the traditional application-specific integrated circuit figure of silicon area. In order for a designer to use these heterogenous elements in their design, they must usually specifically instantiate them. Heterogeneous elements not used in a design remain unused on the device, consuming leakage power, silicon area and manufacturing costs. Method of transferring functionality normally implemented in embedded ROMs and 4-input look-up tables (4-LUTs) onto unused digital signal processor (DSP) blocks is proposed. The paper proceeds to include this method in a synthesis system incorporating the idea of resource constrained synthesis, where a design is mapped to an FPGA considering user and device constraints on heterogenous element usage, based on an extension to the Altera Quartus II synthesis software. Results have been obtained, showing an improvement over existing methods in 76% of the 21 ROMs examined. Further results have been obtained from applying this approach with the synthesis system to benchmark algorithms. In the designs examined, the number of possible implementations has increased by two to four times over Altera Quartus.

1 Introduction

Field programmable gate arrays (FPGAs) are user-programmable devices for implementing generic digital circuits. FPGA technology is now widespread in many applications for reasons of reconfigurability, reduced development time and low cost for small to medium production runs compared to application-specific integrated circuits (ASICs).

A classic FPGA architecture consists of an array of 'configurable logic blocks' or 'logic elements', containing one or more multi-input look-up tables (LUTs) followed by D-type latches. These allow the implementation of simple logic functions, and more complex functions can be made by using combinations of CLBs. Switch boxes allow full routing between any two CLBs using a dense network of hierarchical routing channels. For interfacing with external circuitry, input/output blocks (IOBs) are used to connect the routing channels with the device pins. IOBs typically provide configurable buffering and D-type latching.

These three features give FPGAs their versatility and reconfigurability. However, the traditional disadvantages of FPGAs compared to ASICs are the high power consumption needed to drive the routing network, and performance disadvantage caused by routing delays.

In recent years, logic density and performance of FPGAs has increased to a point where they are able to compete with other products, such as digital signal processors (DSPs), in areas not traditionally associated with FPGAs. This application domain has led FPGA manufacturers to make their architectures more heterogeneous, with the inclusion of embedded coarse-grained 'hard cores' in addition to the traditional 'sea of gates' network of LUTs. Some of the most popular of these elements include RAM blocks, multipliers, MAC elements, PLLs, DLLs, processors and high-speed serial transceivers.

Conventional synthesis techniques usually require that these heterogeneous elements are instantiated in a user's design by use of specific syntax or symbols, which can only be mapped to that specific architecture element. Otherwise these elements remain unused on the device. Although existing synthesis tools can infer the use of embedded components, this is limited to an 'always-infer' or a 'never-infer' dichotomy.

Current and previous generation FPGAs from the major manufacturers, Altera and Xilinx, have included the Stratix I and II, and the Virtex II and IV. The heterogeneous elements that these device families all share in common are embedded multiplication, embedded RAM and a network of fine grained LUTs. In all architectures, except the Stratix II [1], the LUTs used are fixed 4-input LUTs (4-LUTs). As these elements are an accepted part of a modern FPGA device, possibilities for using them have been investigated in this paper.

The area metric of FPGA designs differs somewhat from traditional ASIC measures. In the case of ASICs, design area is measured in terms of μm^2 of silicon used for implementing a design using a particular fabrication process.

In the case of designs implemented onto FPGAs silicon area is not such a meaningful measure of design area, even across devices using the same process width. Manufacturers supply device families in certain discrete sizes, with the number of heterogenous elements present in a particular device fixed. For example, an Altera EP1S10 device from

doi:10.1049/iet-cdt:20060016

Paper first received 20th January and in revised form 17th June 2006

The authors are with the Department of Electronic and Electrical Engineering, Circuits and Systems Group, Imperial College, Exhibition Road, London SW7 2BT, UK

E-mail: gwem@preromanbritain.com

the Stratix family contains 10570 'logic elements' (each containing a single 4-LUT), 920448 bits of RAM and 48 embedded multipliers. Mapping a user's design onto a device will use some or all of these resource types, and different implementations of the same design might transfer functionality between different combinations of them, therefore resulting in different silicon areas. For example, a 9×9 multiplication might use a single Stratix embedded multiplier, but it could also be implemented from several logic elements. This vector area measurement is not easily comparable to a single-dimensioned silicon area measurement.

Existing techniques already provide an incomplete set of transfers between device resources as shown below:

- embedded multiplication → embedded ROMs (by tabulating the results of all possible multiplications);
- a network of LUTs → embedded ROMs (by tabulating selected combinatorial parts of the design) [2];
- embedded ROMs→a network of LUTs (with traditional logic synthesis [3]);
- embedded multiplication→a network of LUTs (with core generation [4, 5]).

These transfers are shown diagrammatically in Fig. 1. The main contribution of this paper is to propose a method of re-mapping resources between ROMs and embedded multiplication. This transfer allows unused embedded multiplication elements to be used for implementing functions normally associated with ROMs, possibly allowing a design to be targeted to a smaller, cheaper and less power consumptive device than that allowed by existing methods.

Fig. 2 shows a high-level representation of the technique proposed in this paper. Parts of the circuit that operate as ROMs (which may be implemented using embedded RAM blocks, or as a combinatorial circuit implemented with a network of LUTs) are swapped directly for alternative circuitry with identical behaviour and input/output ports. The alternative circuitry uses a modified form of faithful polynomial approximation, which is explained in Section 3. Polynomial approximation was chosen as a basis of the conversion as the calculation of polynomials uses a multiplier rich evaluation, in contrast to table-based techniques such as bi-partite lookup table [6, 7]. A multiplication-rich approximation technique is clearly suited for ROM to multiplier transfer.

Although the actual ROM to embedded multiplication method can be applied to any heterogenous FPGA architecture containing ROMs and embedded multipliers, the Altera Stratix family of devices has been selected for the actual implementation of our ideas.

Results obtained by applying the proposed technique to discrete ROMs presented in Section 5 illustrate the viability of this method for ROM replacement. In 86% of cases at least one new point of the Pareto-optimal design curve was added over existing techniques, with an average case 13.9% improvement over these alternative methods [a point on the Pareto-optimal design curve represents a design which cannot have improved characteristics in one optimisation criterion, without sacrificing those in another].

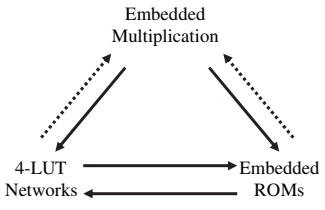

Fig. 1 *Transformations between FPGA resources*

Solid lines represent transformations already possible using existing methods

Dotted lines represent the new transformation made possible with the technique proposed in Section 3

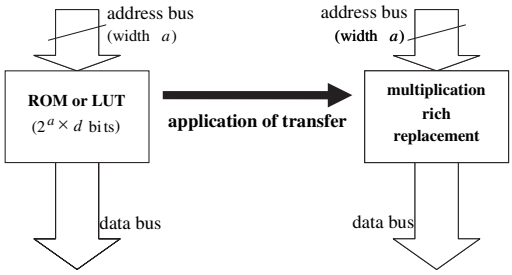

Fig. 2 *Transformation made possible with the ROM replacement technique*

These results have been augmented by including the transfer within an alternative synthesis system built on Altera Quartus II tools within the Quartus University Interface Program (QUIP) framework. This system provides automated re-mapping of functions between heterogeneous elements, specifically ROMs, multipliers or DSP blocks and LUTs as presented.

Five benchmark designs have been applied to this synthesis system, which resulted in a 2–4-fold increase in Pareto-optimal implementations over Altera Quartus II. In order to investigate the types of ROM data to which the proposed technique can usefully be applied, synthetically generated data sets have been used. Results show improvements in 4-LUT utilisation over standard techniques of up to 51%.

2 Background

A recent analysis of the differences in programming models needed for heterogeneous CPU/FPGA devices is presented in the work of Andrews *et al.* [8]. The article discusses the current high-level tools available (SystemC, Handel-C, System Verilog and so on) and suggests that more work must be done in order to fully support the heterogenous elements of modern FPGAs. The approach described in this paper addresses the issue of using heterogenous elements in designs where they are not specifically instantiated or inferred.

Pavlidis and Maika [9] examine the problem of uniform piecewise polynomial (UPP) approximation with variable joints, recognising the need for this in approximating discontinuous functions. A procedure for computing the position of these joints, when given the approximation order, is described. The procedure has guaranteed convergence and minimal approximation error. The alternative transformation scheme, proposed in this paper, achieves equivalent or better results, and in the context of the proposed synthesis system it does not require approximation order as an external input.

Comparisons to bi- and multi-partite table methods have been made in this paper. These schemes are first-order table-based methods, implemented without use of embedded multipliers, with the result being the summation of several table lookups. Several papers have been published developing the idea of these techniques. The first paper to suggest the idea of bi-partite tables was the work of DasSarma and Matula [6], where it was applied to tabulated values of the reciprocal function. In the work of Schulte and Stine [7],

this idea was extended. Muller [10] was the first paper to consider using additional tables and suggested a tri-partite approach using three lookup tables derived from a Taylor series analysis. Finally, work of de Dinechin and Tisserand [11] combines that of Schulte and Stine [7] and Muller [10] into a general definition of multi-partite tables.

The work of Defour *et al.* [12] is direct extension to the existing work on multi-partite table methods with a second-order approximation scheme using embedded multiplication. The method remains biased towards using tables, however, the use of multiplication has led to reduction in the 4-LUT usage needed to store them. Defour *et al.* [12] note that its reliance on Taylor series approximation leads to less accurate approximation and a larger table sizes to implement a faithful approximation. This paper proposes a method using a minimax polynomial approximation to reduce the 4-LUT for the same approximation error.

Application of polynomial approximation to the problem of emulating ROM functionality with DSP blocks was the subject of a previous paper by Morris *et al.* [13]. Overviewed are techniques for ROM to DSP block transfer using modified polynomial approximation, and the idea of resource constrained synthesis was suggested. The work described here expands on the work of Morris *et al.* [13], filling in details of the ROM to DSP block transfer, providing more explanation of the resource constrained guided synthesis system, and presenting additional results from applying different ROM data sets to the ROM to DSP block transfer technique.

Use of embedded RAM blocks in modern FPGAs for implementing logic functions has been the subject of considerable study [2, 14]. The work discusses the mapping of random combinatorial logic into embedded RAM blocks and the effectiveness of this mapping for different FPGA architectures. Logic packing techniques are described, which aim to find the most efficient cut of the design that can be taken to form the ROM contents. An average 69% reduction in LUT utilisation was achieved over 14 benchmark circuits, with a 6% decrease in critical path delay. The work described here complements this approach by adding ROM to multiplier transformations and combining the concepts of ROM to multiplier and LUT to ROM transformations in a unified synthesis approach.

3 Replacing ROMs with DSP blocks

This section describes the proposed ROM to multiplication transfer method. It uses a lossless approximation technique based on a modified polynomial evaluation. The general formula for polynomial evaluation is reproduced in the following equation

$$y = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots + c_n x^n \tag{1}$$

A simple polynomial approximation, using (1) directly, can be applied to any set of ROM data using a Lagrangian interpolating polynomial. Equation (2) shows such a polynomial passing through a general set of p points (x_1, y_1) to (x_p, y_p)

$$\begin{aligned} y = {} & \frac{(x-x_2)(x-x_3)\cdots(x-x_p)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_p)} y_1 \\ & + \frac{(x-x_1)(x-x_3)\cdots(x-x_p)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_p)} y_2 + \cdots \\ & + \frac{(x-x_1)(x-x_2)\cdots(x-x_{p-1})}{(x_p-x_1)(x_p-x_2)\cdots(x_p-x_{p-1})} y_p \end{aligned} \tag{2}$$

Reducing (2) to give an equation of the form (1) gives a highest order term n given by $n = p - 1$. The number of values p stored in a ROM with an a bit address bus is given by $p = 2^a$. As the Lagrangian polynomial represents the polynomial of worst case order which passes through p points, the approximation order for a general set of data stored in a ROM with an a bit address bus is given by $n \leq 2^a - 1$.

To mitigate this problem of exponentially increasing approximation order a set of different coefficients values are used for the polynomial evaluation, depending on the value of the input address bus of the ROM being replaced. The coefficients used to evaluate a particular address are selected from a bus derived from the most significant bits of the main address bus. The masking function used to derive this bus is expressed in (3), where 'div' represents an integer division operation. Each coefficient is selected from the results of separate masking functions, this is shown in the example architecture of Fig. 3.

$$m_i = M_i(x) = x \operatorname{div} 2^{a-j_i} \tag{3}$$

Note here that the well-known UPP [9] and simple polynomial approximations are subsets of this technique corresponding to certain mask types. UPP approximations are a subset of MSB masking types, where the masks generating each coefficient selection bus are identical, that is $M_0 = M_1 = M_2 = \cdots = M_n$. Simple polynomial approximation corresponds to architectures where each coefficient can take only a single value, therefore each coefficient selection bus has zero width, that is, $j_0 = j_1 = j_2 = \cdots = j_n = 0$.

A modulus can optionally be applied to the address bus input of the polynomial evaluation, which may reduce 4-LUT overhead in the case of periodic data sets. This modulus is restricted to type 2^k, as modulus operations of this type can be easily calculated by simply ignoring most significant bits of the input address bus. The output of the modulus function r has a reduced range compared to x, and is expressed in (4)

$$r = x \bmod 2^k \tag{4}$$

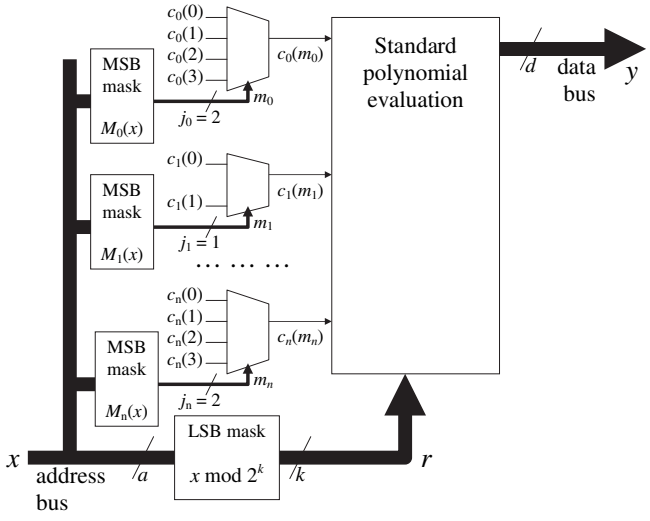

Fig. 3 *Example of the architecture proposed for ROM replacement*

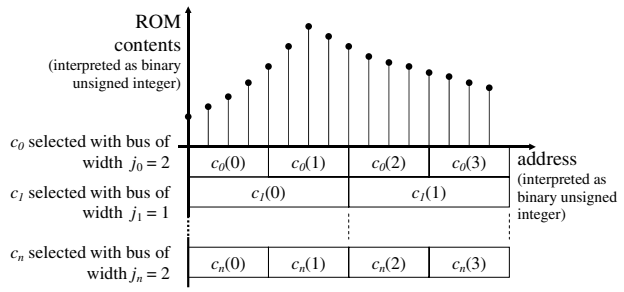

Fig. 4 *Implications of Fig. 3 architecture on the coefficients used in the polynomial evaluation*

Combining (1), (3) and (4) the proposed technique can be described by

$$y = c_0(m_0) + c_1(m_1)r + c_2(m_2)r^2 + c_3(m_3)r^3 + \cdots + c_n(m_n)r^n \quad (5)$$

3.1 Proposed architecture

An example of the architecture proposed for implementing (5) is presented in Fig. 3, illustrated with arbitrarily chosen coefficient selection buses for clarity. The MSB and LSB maskings of the input address bus, used to select coefficient values and calculate r, respectively, are achieved with zero overhead in an FPGA implementation. The coefficient multiplexers are generated in an FPGA with LUT based ROMs. The proposed implementation uses an Estrin's method [15] polynomial evaluation, although any other evaluation technique can also be used. The polynomial evaluation only requires DSP blocks for implementation in an Altera Stratix FPGA.

The distribution of the different coefficient values of the example architecture of Fig. 3 over the full range of x is shown in Fig. 4. The diagram illustrates how different regions of the function domain use different coefficients for the polynomial evaluation. The values used for each coefficient are distributed over uniform regions of the function domain.

The use of r confers benefits when applying the ROM reproduction technique to ROMs containing periodically repeating data sets. In this case, a lower approximation order may be required, reducing 4-LUT overhead needed to store the higher level coefficients. For example, Fig. 5 illustrates a ROM containing a periodic function. Instead of approximating the whole range of this function, the modulus 2^k is adjusted to generate identical ranges of r for each period of the input function. This means the calculated coefficients need only consider a single period of the stored ROM data, reducing the required approximation order, and therefore number of coefficient values, and 4-LUT overhead.

By allowing each coefficient to have an independent number of values, many functions found in practical circuits can be reproduced needing fewer stored coefficient values

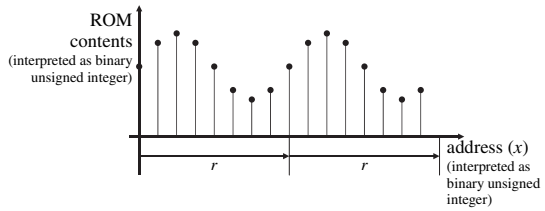

Fig. 5 *Illustration of the advantage of applying a modulus to the evaluation of a periodic data set*

Fig. 6 *Representation of the example ROM to be reproduced with the ROM replacement architecture*

than a UPP polynomial approximation with equivalent worst-case error compared to the original function. As UPP approximation is a special case of the proposed scheme, the results achieved are at least as good as this well-known technique.

3.1.1 Motivational example: An optimisation method for this architecture is described in the following section, however, first we present an example to illustrate the usefulness of the proposed architecture for ROM reproduction. A ROM with 2 bit address bus and 8 bit data bus containing the values 0, 97, 181 and 236 is used as an example as shown in Fig. 6. These values are the first 90° of an 8 bit sin(x) function. In binary format, the stored values are as shown below

0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1
1	0	1	1	0	1	0	1
1	1	1	0	1	1	0	0

When this ROM is synthesised to an FPGA, it would consume an estimated seven 4-LUTs when implementing the storage in 4-LUTs alone. This estimate is derived from the seven non-constant bits in the binary representation shown above, only bit 1 is common for all the values. Fig. 7 shows a 4-LUT-based circuit implementing this ROM.

The same values can be reproduced by using a second-order variant of the proposed architecture, with

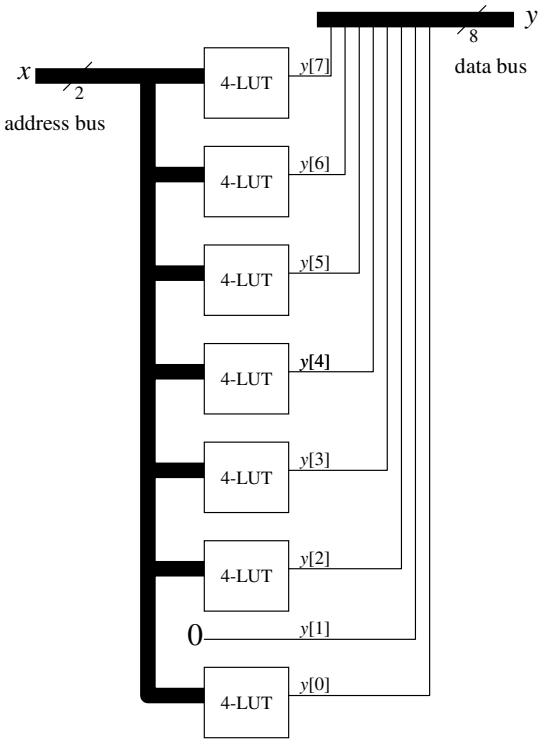

Fig. 7 *4-LUT implementation of the example ROM*

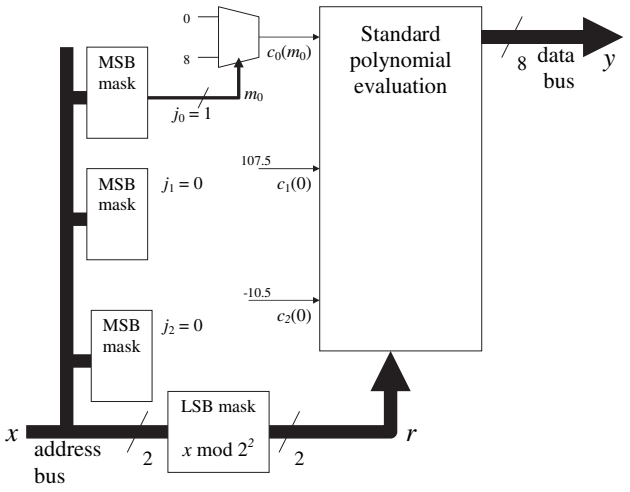

Fig. 8 *Proposed replacement architecture for the example ROM*

modulus 2^k set to 4, and coefficient selection buses j_0 to j_2 set as:

$$\begin{aligned} j_0 &= 1 \\ j_1 &= 0 \\ j_2 &= 0 \end{aligned} \tag{6}$$

There are therefore two possible values for the zeroth-order coefficient, with the remaining orders having a single constant value. This specific architecture is shown in Fig. 8

Coefficient values that reproduce the example ROM with this architecture are shown in the following equation:

$$\begin{aligned} c_0(0) = 0, \quad c_0(1) &= 8 \\ c_1(0) &= 107.5 \\ c_2(0) &= -10.5 \end{aligned} \tag{7}$$

The original ROM values can be recovered as shown in the following equations:

$$\begin{aligned} y &= c_0 + c_1 x + c_2 x^2 \\ y_0 &= 0 + 107.5 \times 0 - 10.5 \times 0^2 = 0 \\ y_1 &= 0 + 107.5 \times 1 - 10.5 \times 1^2 = 97 \\ y_2 &= 8 + 107.5 \times 2 - 10.5 \times 2^2 = 181 \\ y_3 &= 8 + 107.5 \times 3 - 10.5 \times 3^2 = 236 \end{aligned} \tag{8}$$

As coefficients for the first and second orders are constant values, they require no 4-LUTs for storage. The two possible values for the zeroth-order coefficient, 0 and 8, only differ in a single bit. Therefore this replacement architecture requires only one 4-LUT compared to the seven 4-LUTs required in the straightforward 4-LUT implementation, an 86% reduction in this example. However, the replacement architecture requires embedded multiplication elements for the polynomial evaluation, making this method useful for ROM to multiplier transfer.

3.2 Optimisation method

A design based on the architecture described in Fig. 3 is defined by the approximation order that the address bus masks, selecting each coefficient, the modulus applied to the input address bus and the ROM contents. These parameters can be set to significantly reduce LUT usage, maximising the benefits of using a multiplier rich approximation.

Completely searching all mask and modulus combinations is realistic only for smaller designs due to exponential computational complexity. For larger designs, a simulated annealing optimisation [16] is used to find the address bus masks and modulus that minimise the number of 4-LUTs used in implementing the design.

As the main stage in the optimisation procedure uses simulated annealing, firstly an initial constructive solution must be found. Any random starting position could be used, however, to ensure that the worst-case performance of the whole optimisation procedure is at least as good as UPP approximation, and masking functions corresponding to the optimum UPP approximation are used as the starting point for the simulated annealing stage. To find the optimal UPP mask set, a complete search of all possible UPP masks sets is carried out. Due to the restrictions of UPP approximation, this process is carried out by testing only $a + 1$ architectures.

While the simulated annealing algorithm is running, it produces mask sets and a modulus with each iteration. Since polynomials are linear in the coefficients, linear program solution may be used to find the coefficient values. The mask set and modulus are used to formulate a set of linear equations, which are solved by a linear program solver to find coefficient values that best reproduce the original ROM.

The linear program solver returns the coefficient values in floating-point format. Before LUT utilisation can be calculated, these values are subject to a word-length optimisation scheme. Each coefficient is scaled to a fixed point representation with zero redundancy in the MSBs. The precision of the representation is then found by successively increasing the precision of the coefficients separately, retaining full floating-point precision in the other coefficients until faithful rounding is achieved. Next, all coefficient values are rounded, and the error of the resulting approximation is calculated again. If necessary, the precision of all coefficients is then increased by the same quantity until faithful rounding is achieved to a maximum of 64 bits precision.

The constant parameters used in the simulated annealing algorithm for initial temperature, cooling factor and final temperature have been selected empirically using the fifth-order test ROMs shown in Table 2 to obtain an acceptable trade-off between repeatability, solution quality and computation time. Once determined, identical simulated annealing parameters have been used for all the results we provided here. The actual values of the parameters used are shown in Table 1.

The simulated annealing algorithm uses 4-LUT utilisation as the figure of merit to optimise. Using an actual 4-LUT utilisation from synthesising a design in a typical logic synthesis tool, such as Altera Quartus, is impractical because of the large computational overhead of logic synthesis. Therefore, the simulated annealing algorithm uses an estimate of 4-LUT utilisation which is derived using a simplified model of a typical logic synthesis stage [17].

Table 1: Constant values used in the simulated annealing optimisation

Parameter	Value
Initial temperature	1
Cooling factor	0.9
Final temperature	$0.075/n + 1$

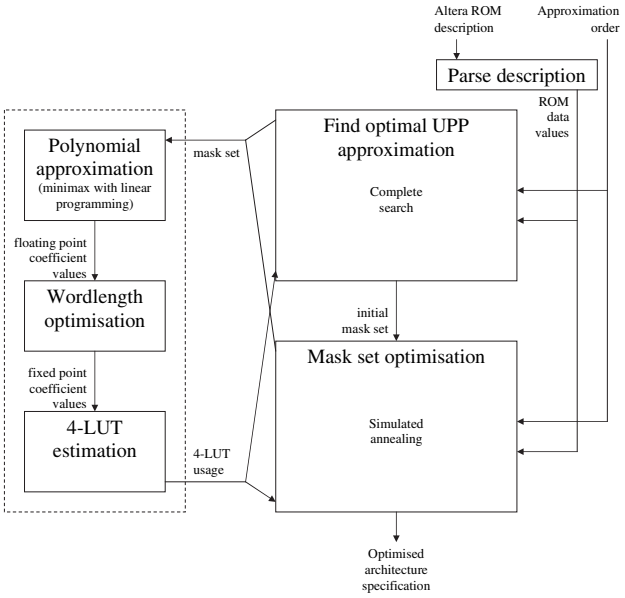

Fig. 9 *Represention of the ROM to arithmetic conversion technique*

An overview of the whole technique and the interaction between the ROM data parser, simulated annealing algorithm, linear program solver, word-length optimisation and 4-LUT estimation is provided in Fig. 9.

4 Proposed synthesis flow

The transformation technique described in the previous section has been combined with other resource transformations into a resource-constrained guided synthesis system. This system can be used to implement designs on FPGA devices, that may not have been possible using existing synthesis techniques. A subset of the resource transformations shown in Fig. 1 are incorporated within the synthesis system, this subset is shown in Fig. 10.

First, the user's design is specified with any usual mechanism (e.g. VHDL, Verilog, schematic capture, Altera DSP builder). This is taken as the input to the synthesis system. The design is technology mapped to a particular architecture, and this mapped design is preserved for later manipulation. As with a usual synthesis flow, the design is then fitted to a particular part and a timing simulation is made. The tool then parses the Altera timing and fitting reports to compare the synthesised design to user constraints on system clock speed in addition to ROM bit, DSP block and LUT utilisation. If the constraints are met, then the programming bit stream is assembled as usual. If the constraints are not met, then resources in the design must be transferred depending on the constraint violation. The resources are transferred by substituting replacement components into the previous implementation. The process is repeated until the user constraints are met, or no further optimisation is possible.

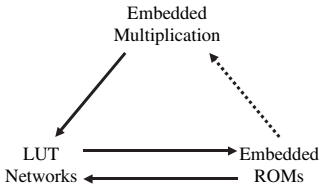

Fig. 10 *Transformations between FPGA resources integrated within the proposed synthesis system of Section 4*

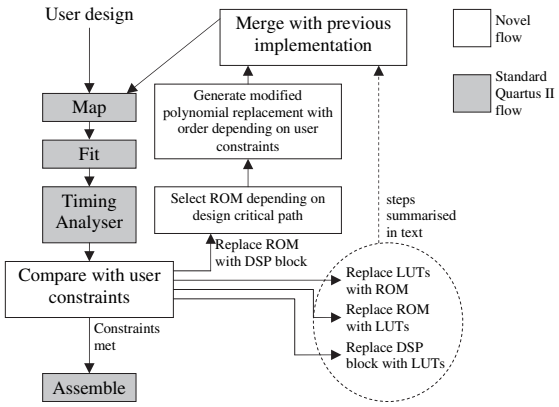

Fig. 11 *Block diagram of proposed synthesis system*

A block diagram representing the proposed design-flow and its relationship with the existing Altera Quartus II tools is represented in Fig. 11. The existing tools are interfaced within the QUIP framework [18].

In the case of a ROM to DSP block transfer, the design is analysed in order to select the ROM furthest away from the critical path for replacement. The parameters of the ROM replacement circuitry described in Section 3 depend on the timing slack from the critical path and the estimated delay of the replacement, which is calculated via a heuristic based on the target architecture's timing specifications.

As the focus of this paper is on the automation of the ROM to DSP block transfer, the remaining transformations shown in the ellipse in Fig. 11 are carried out by hand. LUT to ROM and ROM to LUT replacements are performed by tabulation of stored data and forcing the Quartus II synthesis system to implement the tables with ROMs or LUTs as appropriate. DSP block to LUT replacements are performed by substitution of references to DSP block internal multipliers and adders by alternative circuits produced with Synplicity's Synplify design suite. DSP blocks furthest away from the critical path are selected for replacement first.

5 Results

5.1 Benchmark ROMs

The results presented in this section are intended to compare the effectiveness of the ROM to DSP block replacement technique to the standard approaches of simple polynomial and UPP approximation, as well as bi- and multi-partite table methods [6, 7, 10].

DSP blocks in the actual Altera Stratix architecture have a fixed input bus width of 9 bits and can be flexibly reconfigured. In these results, we have considered the DSP blocks to be configured to sum the outputs of two multiplications, which is just one possible configuration of the real DSP blocks. We have also assumed the DSP blocks have sufficient width to accommodate the required precision at each stage of the evaluation, without the need to combine blocks.

The simulated annealing optimisation has been applied five times to each combination of test ROM and number of DSP blocks, up to a maximum of 7 DSP blocks, corresponding to a fifth-order Estrin's method polynomial evaluation. This limit was selected as adding additional DSP blocks to the design after fourth order resulted in an decrease in 4-LUT utilisation in only 9.5% the ROMs examined, as can be seen from the results later presented.

The estimated 4-LUT and DSP block utilisation from the best designs found by the simulated annealing optimisations are then compared against the optimum designs using sets

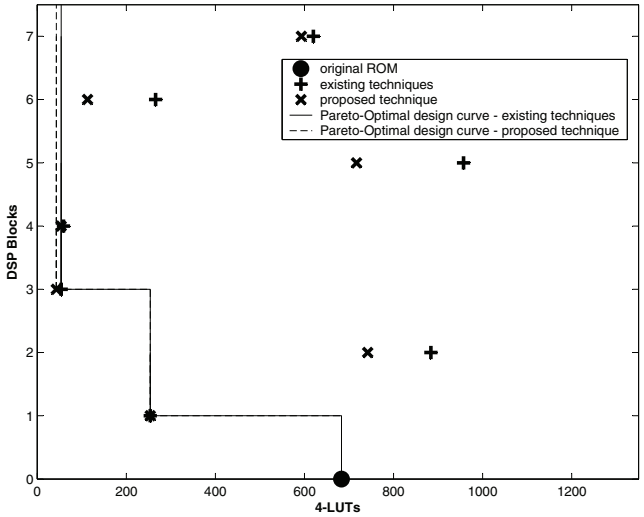

Fig. 12 *Design space plot for a 2048 entry ROM for the function* $log_2(1 + 2^{-x})$

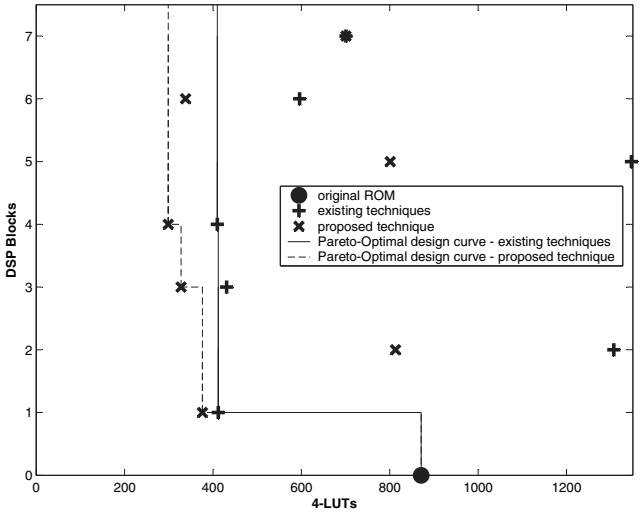

Fig. 14 *Design space plot for a 2048 entry ROM for the function sin(x)*

of masking functions corresponding to UPP approximation and multi-partite table method. In the multi-partite table method designs, the adders have been assumed to be implemented in DSP blocks rather than 4-LUTs for fair comparison to the other techniques under test.

The results of these design space comparisons are shown in Figs. 12–14 and Tables 2 and 3.

In total 21 different ROMs have been examined, with numbers of entries ranging from 32 to 2048. For the purposes of these tests, three different mathematical functions have been sampled for use as the contents of these ROMs. Two of these functions are $\log_2(1 + 2^{-x})$ and $\log_2(1 - 2^{-x})$, which are used for addition and subtraction in a logarithmic number system (LNS). The third function sampled is the first 90° of the sin(x) function.

Figs. 12–14 show graphically the design spaces for three of these ROMs, all with 2048 entries and one for each of the sampled functions. On these diagrams, the results obtained using the proposed technique are compared to existing methods with the change in Pareto-optimal design curve highlighted, points away from the design curve are highlighted for interest only.

The design space of the LNS addition function shown in Fig. 12 shows just one additional point on the Pareto-optimal design curve due to the proposed technique. However, apart from the case of using a single DSP block, the proposed technique consistently offers reduced 4-LUT utilisation over existing methods.

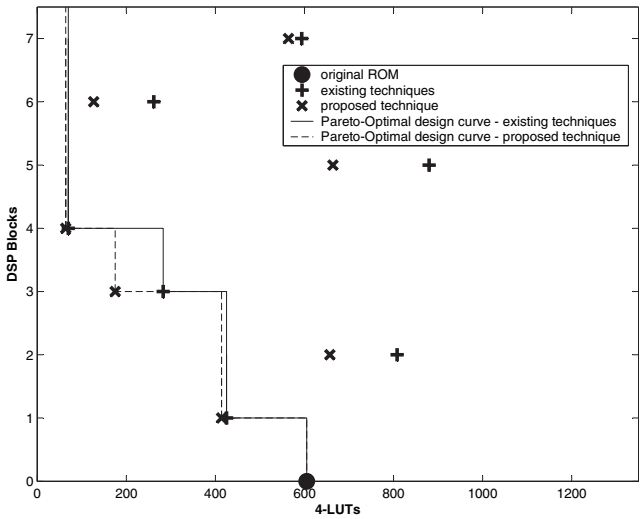

Fig. 13 *Design space plot for a 2048 entry ROM for the function* $log_2(1 - 2^{-x})$

Fig. 13 shows the design space of a ROM used for the LNS subtraction function. For this ROM, the proposed technique has found three additional designs on the Pareto-optimal design curve over UPP approximation and multi-partite table method. For every number of DSP blocks tested, the proposed technique obtains a saving in 4-LUTs over existing methods.

The design space for a 2048 entry sin(x) function is shown in Fig. 14. The proposed techniques add three extra points on the Pareto-optimal design curve over the original ROM. The proposed technique out performs the existing techniques for any number of DSP blocks below seven. Excepting one case, the existing techniques actually cause an increase in 4-LUT utilisation when using higher numbers of DSP blocks.

The complete set of raw results for the tests are reproduced in Table 2. The selection of sin(x) ROMs offers the biggest reduction in 4-LUT utilisation when using the proposed technique, on average a 24.7% reduction in 4-LUT usage over existing techniques is possible, up to a maximum of 80.6%. The remaining two functions sampled, LNS addition and subtraction, show reductions of 15.4% and 11.8%, respectively, which shows the sensitivity of the technique to the data stored in the ROM under conversion.

In 76% of ROMs examined, at least one new point has been added to the Pareto-optimal design space as a result of the technique, with 38% of designs having more than one new point added.

Solution times from the range of ROM sizes examined are shown in Table 3. When the proposed synthesis framework shown in Fig. 11 conducts a ROM to DSP block transfer, the overall synthesis time is increased by the amounts shown in this table. As solution time depends somewhat on the data of the ROM being replaced and the order of the approximation, this figure is the mean of the three functions sampled for these tests.

For smaller designs, the solution time is dominated by operating system overheads, but as the ROM size increases the most significant factor becomes the solution of the coefficient finding linear program for each optimisation step. As ROM size increases further, the time needed for 4-LUT

Table 2: Raw results of the investigations

Function	Entries in ROM (all 8bit)	4-LUTs required Orginal ROM	1 DSP blocks		2 DSP blocks		3 DSP blocks		4 DSP blocks		5 DSP blocks		6 DSP blocks		7 DSP blocks	
			prop.	exist.	prop.	exist.	prop.	exist.	prop.	exist.	prop.	exist.	prop.	exist.	prop.	exist.
$\log_2(1+2^{-x})$	32	20	15	15	29	34	15	17	15	17	29	34	15	15	15	17
	64	41	37	37	55	65	24	25	22	28	57	65	30	34	20	26
	128	78	52	52	101	115	26	29	30	34	101	123	23	28	43	69
	256	146	62	62	172	194	31	36	32	44	172	225	33	58	60	74
	512	262	202	202	281	324	40	41	39	46	281	364	46	68	96	96
	1024	427	145	145	446	509	39	46	40	55	460	561	47	90	203	288
	2048	683	254	254	742	782	43	54	54	59	717	957	113	266	593	620
$\log_2(1-2^{-x})$	32	20	15	15	22	32	15	17	15	17	29	32	15	15	15	17
	64	37	37	37	43	62	35	42	30	30	62	62	37	37	41	42
	128	79	76	76	12	112	32	32	36	44	118	118	43	57	46	73
	256	140	106	106	184	187	56	86	43	45	181	212	54	61	63	85
	512	241	201	210	283	300	103	103	54	54	283	336	52	67	91	95
	1024	384	202	202	433	463	117	117	67	67	411	511	67	89	199	281
	2048	605	414	425	657	808	175	283	64	70	664	880	127	262	564	594
$\sin(x)$	32	23	12	12	22	37	13	15	14	15	22	37	0	0	0	0
	64	49	17	17	33	64	17	21	18	23	47	64	11	29	7	36
	128	97	34	48	97	117	26	26	26	37	97	163	23	49	67	71
	256	181	58	58	170	207	27	32	30	41	170	243	41	58	43	83
	512	321	218	218	272	346	156	185	103	159	272	386	118	214	96	96
	1024	542	296	324	452	650	263	374	196	323	531	700	174	259	273	349
	2048	871	376	412	813	1307	328	431	299	410	801	1347	338	596	701	701

'prop.' column contains the number of 4-LUTs used for the least cost implementation using the proposed technique after five simulated annealing runs. 'exist.' contains results for the optimum design using the existing techniques of UPP approximation and multi-partite tables

Table 3: Mean solution time when running a design-space optimisation on a 3.4 GHz Pentium 4 PC under Fedora Linux with 2 Gb RAM

Entries in original ROM	Mean solution time, s
32	22
64	32
128	69
256	99
512	228
1024	450
2048	817

estimation computation increases in significance, however, in the designs examined it does not dominate.

5.2 Benchmark circuits

To investigate the larger set of possible implementations produced using the proposed techniques for a given design, and to examine impact on latency, the synthesis system incorporating the proposed ROM to DSP block transfer has been applied to five benchmark designs taken from various sources.

The first benchmark is an audio synthesis system including oscillators and additive mixing. The second benchmark is an implementation of the CORDIC algorithm producing simultaneous 21 bit sine and cosine outputs. The third benchmark is a hybrid DSP system consisting of signal generation, combination and filtering. The fourth benchmark is a fully folded 32 tap FIR filter with 17 bit data bus output and coefficient width. The final is an FM receiver, which includes a PLL based oscillator, phase detector and filtering. The complete set of Pareto-optimal results for the tests are shown in Table 4 and are discussed below, the non-Pareto-optimal results that were obtained during the design space exploration have not been reproduced.

The audio synthesis engine has two additional design space points highlighted as a result of the design space exploration. The first of these transfers all the ROM resources into 2 DSP blocks, with a small increase of 4% of LUTs utilisation and a decrease in clock frequency of 5%. The second design maps all the ROM resources to LUTs. The LUT utilisation is also increased by 4%, however, the clock frequency is decreased by a higher 8% in this case.

In the case of the CORDIC algorithm, the proposed system achieves a 2-fold increase in Pareto-optimal implementations over the point solution provided by the standard Quartus II flow. The additional design shows the potential for completely freeing-up embedded ROM blocks, as well as improvements in system clock frequency of 11% at the expense of a 3% increase in LUT utilisation, respectively. This is the only design examined where no ROM to DSP block transfers resulted in new Pareto-optimal designs. The improvement in system clock frequency is attributable to lower routing delay.

For the mixed DSP system, a single extra implementation has been discovered as a result of our design space exploration. A reduction in ROM bit usage by 94% with a resource transferral to DSP blocks has increased LUT utilisation by 17% and reduced maximum clock frequency by 77%. This is the only case where the maximum clock rate has been reduced significantly. The ROMs are replaced with a structure of the type shown in Fig. 3, which has inherently higher delay than an equivalent ROM. In the other designs, this was mitigated by replacing ROMs away from the critical path of the design. Unusually in the mixed DSP system, the ROM was on the critical timing path so the logic delay of the polynomial evaluation had a direct impact on system performance. This performance drop could be mediated with additional pipelining, at the expense of increased latency and re-timing the design.

The FIR filter has the largest number of new Pareto-optimal implementations. These new implementations give alternatives for increasing all four figures of merit. An increase in LUT usage by 11% has led to a 27% improvement in clock frequency. A transferral of ROMs into DSP blocks resulted in a total reduction in ROM utilisation, at the expense of using eight times more DSP blocks and 32% more LUTs, this transfer reduced maximum system clock frequency by only 17%. The last implementation comes from a resource transfer from ROM bits to DSP blocks, and then from the DSP blocks to 4-LUTs. A total reduction in used ROM bits has increased the

Table 4: Post place and route Pareto-optimal results

Algorithm	ROM bits	DSP blocks	LUTs	System clock (MHz)
Audio synthesis	**256 (0%)**	**0 (0%)**	**507 (5%)**	**210**
	0 (0%)	2 (4%)	529 (5%)	200
	0 (0%)	0 (0%)	567 (5%)	194
Cordic	**544 (0%)**	**0 (0%)**	**526 (5%)**	**163**
	0 (0%)	0 (0%)	543 (5%)	182
DSP system	**1768 (0%)**	**0 (0%)**	**2024 (19%)**	150
	104 (0%)	34 (71%)	2370 (22%)	34
FIR filter	**544 (0%)**	**2 (4%)**	**168 (2%)**	133
	0 (0%)	18 (38%)	221 (2%)	110
	544 (0%)	2 (4%)	187 (2%)	169
	0 (0%)	2 (4%)	255 (2%)	166
FM receiver	**8192 (1%)**	**0 (0%)**	**564 (5%)**	**63**
	0 (0%)	2 (4%)	1000 (9%)	68

Bold-faced results represent ones produced by the Altera Quartus II system with default synthesis options. Values in brackets give a percentage utilisation for that particular resource in an Altera Stratix EP1S10 FPGA

number of LUTs by 52% and also resulted in a 25% improvement in system clock rate.

As a result of the design space exploration of the FM receiver, there is an additional implementation possibility. Complete transfer of ROM resources to other elements is achieved, using 2 DSP blocks and increasing LUT utilisation by 77%. This transfer resulted in an 8% improvement in clock frequency over the standard Quartus implementation. The improvement in clock speed has been achieved due to lower routing delays.

These results illustrate the applicability of the ROM to DSP block transfer in real designs, with a 2–4-fold increase in possible Pareto-optimal implementations over the single point solution provided by the Quartus II synthesis system and, in most cases, minimal impact on system clock speed.

6 Conclusions and further work

This paper has presented a ROM to DSP block resource transferal technique using a hybrid polynomial approximation based approach. This method has further been used in a synthesis system that manages resource utilisation within a design, allowing heterogeneous resource-based synthesis constraints. This synthesis method was implemented with a modified Quartus II design-flow using the QUIP framework.

Some example ROMs have been applied to the technique and the results show new points on the Pareto-optimal design space found in 76% of these, and an average of 17.3% reduction in 4-LUT utilisation over standard techniques.

Five benchmark designs were applied to the synthesis system and the multi-dimensional LUT, ROM bit, clock period, DSP block design-space was expanded in each case, with significant reductions in ROM usage. An average 3-fold increase over the solutions provided by the standard Quartus II flow was found by using benchmark algorithms.

Further work in this area will be to examine the design space of more benchmark algorithms using synthesis system, and also to fully automate all possible transformations. In addition, the technique could be extended to constrain other heterogeneous resource types. The implications of the adaptive logic elements found in Stratix II devices [1] on the architecture optimisation procedure will also be examined.

7 Acknowledgment

The authors thank the Engineering and Physical Sciences Research Council (EP/C512596/1 and EP/C549481/1) for financial support that made this research possible.

8 References

1 Hutton, M., Scleicher, J., Lewis, D., Pedersen, B., Ynan, R., and Kaptanoglu, S.: 'Improving FPGA performance and area using an adaptable logic module' in Becker, J., Platzner, M., and Vernalde, S. (Eds.): 'Proceedings of the Field programmable logic' (Springer LNCS, 2004)
2 Wilton, S.: 'Implementing logic in FPGA memory arrays: heterogeneous memory architectures'. Proc. IEEE Int. Conf. on Field-Programmable Technology, 2002
3 Murgai, R.: 'Logic synthesis for field-programmable gate arrays'. PhD Thesis, University of California at Berkeley, 1993
4 Xilinx CORE Generator Guide Xilinx 1994–2002
5 Gupta, R.K., and Zorian, Y.: 'Introducing core-based system design', *IEEE Des. Test Comput.*, 1997, **14**, (4), pp. 15–25
6 DasSarma, D., and Matula, D.: 'Faithful bipartite ROM reciprocal tables'. Proc. 12th IEEE Symp. on Comp. Arith., 1995
7 Schulte, M., and Stine, J.: 'Symmetric bipartite tables for accurate function approximation'. Proc. 13th IEEE Symp. on Comp. Arith., 1997
8 Andrews, D., Niehaus, D., and Ashenden, P.: 'Programming models for hybrid CPU/FPGA chips', *IEEE Comput.*, 2004, **37**, (1), pp. 118–120
9 Pavlidis, T., and Maika, A.: 'Uniform piecewise polynomial approximation with variable joints', *J. Approx. Theory*, 1974, **12**, pp. 61–69
10 Muller, J.: 'A few results on table-based methods', *Reliable Computing*, 1999, **5**, (3), pp. 279–288
11 de Dinechin, F., and Tisserand, A.: 'Some improvements on multipartite table methods'. Proc. 15th IEEE Symp. on Comp. Arith., 2001
12 Defour, D., de Dinechin, F., and Muller, J.: 'A new scheme for table-based evaluation of functions'. Proc. 36th Conf. on Signals Syst. Comput., 2002
13 Morris, G.W., Constantinides, G.A., and Chenug, P.Y.K.: 'Using DSP blocks for ROM replacement: a novel synthesis flow'. Proc. Field Programmable Logic, 2005
14 Wilton, S.: 'Heterogeneous technology mapping for area reduction in FPGAs with embedded memory arrays', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2000, **19**, (1), pp. 56–68
15 Aravena, J., and Soh, S.: 'Architectures for polynomial evaluation'. Proc. 21st Southeastern Symp. on System Theory, 1989
16 Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller E.: 'Equation of state calculations by fast computing machines', *J. Chem. Phys.*, 1953, **21**, pp. 1087–1092
17 Morris, G.W.: 'Migrating functionality from ROMs to embedded multipliers'. MPhil/PhD Transfer Report, Imperial College of Science, Technology and Medicine, 2004
18 Altera Corporation, Quartus II University Program (QUIP) Version 2.1, Altera 2004