26

Delay/Area Trade-Off Exploration Using an Architectural Jiggling Algorithm

C. J. Rouse and A. J. Carter
School of Electronic and Manufacturing Systems Engineering,
University of Westminster,
115 New Cavendish Street, LONDON, W1M 8JS, UK
Tel. +44 171 911 5083 Fax +44 171 580 4319
email : rouse@cmsa.westminster.ac.uk

Abstract

This paper describes a novel technique for module-selection exploration. The approach selects from a library containing a wide range of implementations to provide Delay/Area Trade-Off information for the architecture in question. The *Jiggle* Algorithm, which is a modified natural algorithm, is based on Darwinian and accelerated evolution ideas and provides effective *computation time - results* trade-off. Instead of starting the *Jiggle* Algorithm with a single seed, a re-generative multiple-seed approach is taken which drastically reduces the chance of getting trapped in a local minimum. A more appropriate 'cost function' is used in the natural algorithm and a suitable metric is used to evaluate the quality of the DATO graphs produced.

Keywords

Module-selection, binding, trade-offs, natural algorithms

1. INTRODUCTION

With both the increase in the use of high level synthesis and the increase in size and complexity of VLSI and digital systems, designers need better ways to increase the effectiveness of the synthesis process. Most high level synthesis systems cannot perform extensive design space exploration: they normally restrict one or more parameters while trying to optimise the remaining. A technique is presented here that allows Delay/Area Trade-Off (DATO) assessment of a system by provide a plot of timing and area estimates for a range of architectural configurations.

G. Saucier et al. (eds.), *Logic and Architecture Synthesis*
© IFIP International Federation for Information Processing 1995

2. DESIGN SPACE EXPLORATION

Design Space Exploration (DSE) includes assessing all aspects and all possible solutions to a particular problem, from the top level system to the device level. Architectural Exploration (Ae) is a sub-space concerned with looking at different architectures for a structural description, including the use of partitioning, scheduling, and allocation (Landman, 1993). A further sub-space can be identified as Module-Selection exploration (MSe), which in this paper is taken to mean experimenting with a fixed pre-determined architecture by changing the hardware module assignment.

Partitioning, resource allocation, and scheduling, can produce many different architectures, even before considering the design space at the functional block implementation level. Attempting to perform Ae and MSe at the same time complicates the process by introducing too many parameters to control. Currently the techniques described here concentrate on MSe so are used after a traditional high level synthesis algorithm has scheduled and allocated the functional units.

3. AN LDI IIR FILTER AS AN EXAMPLE SYSTEM

A lossless discrete integrator (LDI) ladder filter has been chosen to demonstrate the approaches proposed here. With the coefficients quantized to 12 bits (not including sign bit), the filter's transfer function closely matches the ideal characteristic (Turner, 1986) (Bruton, 1975). The LDI filter structure used here is shown in Figure 1. To ease the demonstration of the algorithms used here a one to one mapping of the SFG to functional blocks has been used.

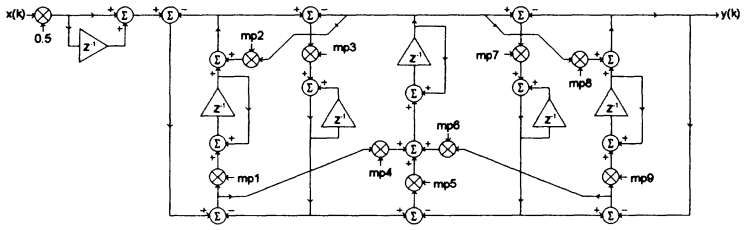

Figure 1 Fifth Order Digital Elliptic LDI Low Pass Filter from (Turner, 1986).

4. THE LIBRARY

For each function, an approach should be allowed to select from modules with a large variety of delays and areas, as well as other trade-off factors (Landman, 1993). This is demonstrated in (Timmer, 1993). All the multi-architecture functional blocks should contain all the 'tricks of the trade' (Rechtin, 1992) in different combinations, that the designer would use in a manual design. The novel approach described here relies on module generators to rapidly provide delay and area information (Bidet, 1993), as well as provide silicon layouts of the functional units when necessary.

The gate level implementation for constant multipliers can be optimised, the complexity being directly related to the number of nonzero digits in the coefficient. To show the type of delay/area information provided by the module generators, the coefficients for multiplier *mp2* and *mp9* in Figure 1 were fed into the multiplier generator. The DATO graphs are shown in Figures 2 and 3 respectively.

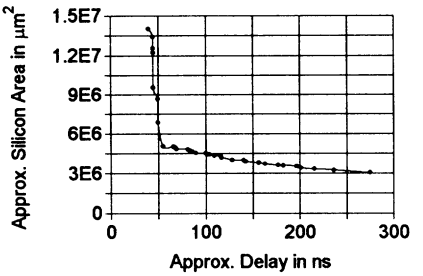

Figure 2 Thirty Five Possible Architectures for Multiplier *mp2*.

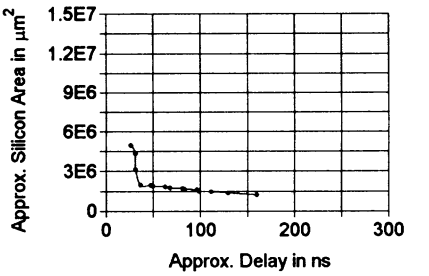

Figure 3 Fifteen Possible Architectures for Multiplier *mp9*.

The gate level implementation for each configuration is constructed from the most primitive standard gate logic elements for the 1.2µ ES2 CMOS process. The availability of specific hand crafted cells will provide much better results than those given here.(eg. adder cells with generate and propagate instead of carry outputs)

5. RANDOM SEARCH OF THE MODULE-SELECTION SPACE

For later comparison, this section discusses the use of a simple random search technique for finding an acceptable implementation. The technique uses a uniformly distributed pseudo random variable to select gate level implementations for each function in the SFG. The 'cost', which in this case is obviously speed and area, of the new architecture is evaluated, and to decide whether it should enter the DATO graph it is checked against every entry in turn. If both the speed and area are better, the one checked against is discarded and the new architecture substituted in its place. If either the speed or area is better, the new architecture is added to the DATO graph. If neither is better, the new architecture is discarded. The results of this search for the example are given later.

6. NATURAL ALGORITHMS FOR DSE

The *Jiggle* Algorithm used here is based on a Darwinian technique (Cain, 1990) which belongs to the class of natural algorithms, all of which model natural processes. Natural algorithms include among others, genetic algorithms (Wehn, 1991) and simulated annealing (Wehn, 1991) (Kale, 1993). Darwinian techniques are based on Darwin's theories of survival ONLY of the fittest, which implies (for MSe) never accepting an architecture that is slower and larger than one already discovered, even if it may lead to an even better architecture if it was kept and used for subsequent generations.

Natural algorithms are in general easier to implement and visualise than exact and heuristic algorithms, especially when other constraints have to be considered. The main characteristic of natural algorithms is that they are experimental in nature, ie. they experiment in the same way nature does in trying to overcome difficulties.

A combined cost function and single figure of merit are normally used to guide a random natural process. This type of cost function is abstract and detached from the reality of designing a system to a specification. For this reason the more appropriate 'cost' of separate delay and area has been

chosen. The 'cost' function is the same as the criteria for entering the DATO graph for the random selection exploration.

7. RANDOM JIGGLE AND ACCELERATED EVOLUTION

The simplest of Random *Jiggle* Algorithms takes a seed architecture, randomly chooses a function from the SFG and exchanges its current gate level implementation with one that is either slightly faster or slightly slower (with corresponding change in area). Figure 4 shows the growth of a seed to form a DATO graph. The results of each jiggle are numbered and it should be noted that the result of jiggle 1 would eliminate the seed from the DATO graph. The result of jiggle 4 eliminates 1 and 2, while the sixth jiggle eliminates 3.

The algorithm used here is a modified version of the basic Random *Jiggle* version in that it incorporates the idea of accelerated evolution, it is referred to as the *Jiggle* Algorithm. If the change of a particular functional block implementation results in an improvement, then a similar change is made in the hope that an even better architecture is found. This further slight change in the same direction is attempted every time the architecture is better than the previous one. The *Jiggle* Algorithm locks onto an improvement and takes it to its limit. Accelerated evolution is demonstrated in Figure 5 where a new architecture is accelerated three times.

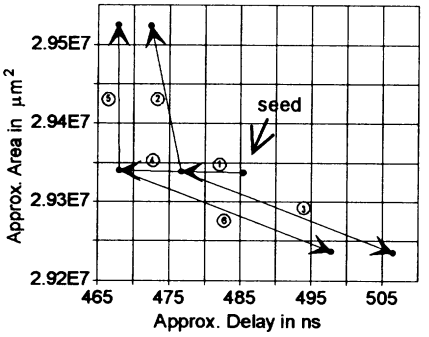

Figure 4 Growth of a Seed to form a DATO Graph.

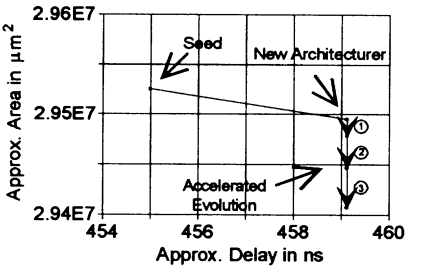

Figure 5 Accelerated Evolution of a New Architecture.

For a single seed, when changing one randomly chosen block by a slight change, you will eventually jiggle every block. There is a chance that every new architecture tried will fail to enter the DATO graph. When this happens the algorithm is trapped in a local minimum of the set of DATO graphs. As many architectures will be generated for the DATO graph it is more sensible to prevent local minima by totally changing the seed at some stage.

Instead of starting the *Jiggle* Algorithm with a single seed, a seed DATO graph is used. This contains the fastest possible and the smallest possible architecture, as well as a range of points in between. The points are produced by heuristic based module selection from the available implementations in the library. The *Jiggle* Algorithm goes through each architecture in the seed DATO graph in turn and jiggles it, accelerates it if possible, and then continues to the next DATO point when the current one fails to enter the graph. When all the points in the current graph have been jiggled another pass is performed if the required number of jiggles has not been reached. Each full DATO graph pass will generate extra points in the DATO graph which are then used as the seeds for the next pass. Using a constantly changing seed drastically reduces the chance of getting trapped in a local minimum of the set of DATO graphs.

8. RESULTS

Extensive trails were carried out on the demonstrator system. To show the *Jiggle* Algorithm in action and demonstrate its advantages, results for various numbers of iterations are shown and compared to the same number of iterations for the Random Search method. DATO graphs for 10 to 100 million iterations (in increasing powers of 10) for the Random Search method are shown in the cascade plot in Figure 6 and for the *Jiggle* Algorithm in Figure 7. (The graph at the back is the seed graph.)

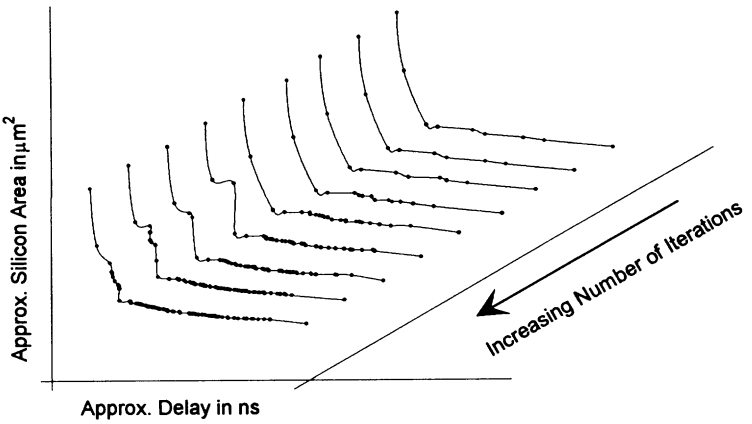

Figure 6 Cascaded DATO Graphs for LDI IIR Filter Using Random Search.

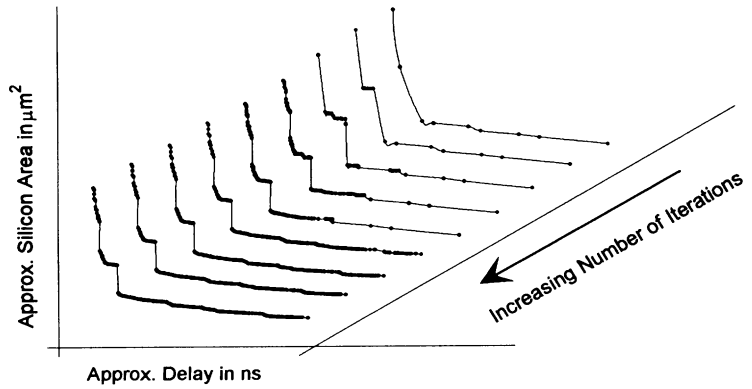

Figure 7 Cascaded DATO Graphs for LDI IIR Filter Using Jiggle Algorithm.

As can be seen the *Jiggle* Algorithm produces much superior DATO graphs in terms of accuracy at every number of iterations, but at the expense of extra computation time, as shown in Table 1.

Table 1 Computation Times for Jiggle Algorithm and Random Search (for a 50MHz 486)

Number of Iterations	*Jiggle Algorithm*	*Random Search*
10	0.05 sec	0.11 sec
100	0.16 sec	0.22 sec
1000	0.55 sec	0.44 sec
10 000	2.52 sec	2.30 sec
100 000	27.79 sec	20.48 sec
1 000 000	5 min 44 sec	3 min 31 sec
10 000 000	58 min 22 sec	37 min 49 sec
100 000 000	9 hr 45 min	6 hr 52 min

The metric of the number of DATO points in the graph is not appropriate if you are judging the quality of the DATO graph. The exact number of points has little relation to the quality of the graph as was demonstrated in Figure 4 where new architectures nullify previous entries. For this reason it has been chosen to use the area under the DATO graph as the metric for comparison. Figure 8 shows the areas under the DATO graphs when using the Random Search and *Jiggle* Algorithm. The entry at '1 iteration' is for the simple heuristic used as the seed for the *Jiggle* Algorithm and is also used to kick start the Random Search. This is done for comparison purposes.

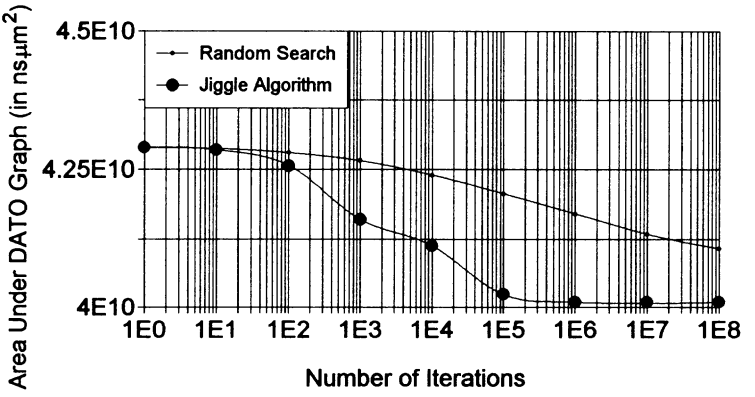

Figure 8 Area Under DATO Graphs.

9. CONCLUSIONS

An exhaustive search of all possibilities will take an excessively long time but a well chosen subset of these will be a useful guide for becoming familiar with the task in question. The basic shape and vital points of the curve are established quickly. Even 28 seconds of jiggling will give a valuable insight, showing that long run times of random based natural algorithms need not be tolerated, but

for good results run times can be as long as the user desires or has to spare. This is useful to the designer who dose not want to wait long for a rough guide and designers who want a final solution.

The *Jiggle* Algorithm which includes a heuristic for accelerated evolution, is an easily implementable method that it is worth considering as an alternative to exact approaches. For finding an optimal solution immediately, exact and heuristic approaches (Sakouti, 1993) (Karkowski, 1993) are normally better, but at a cost, especially for very large or complex systems. These results demonstrate that natural algorithm based MSe is an acceptable method of design space exploration if sufficient DATO data is available for the individual blocks in the system. Module generators are of prime importance.

10. REFERENCES

Bidet E., C.Joanblanq and P.Senn, (1993) GENRIF: An integrated VLSI FIR Filter Compiler, European Design Automation Conference, 466-471.

Bruton, L.T., (1975) Low-Sensitivity Digital Ladder Filters, IEEE Transactions on Circuits and Systems, Vol. CAS-22, No. 3.

Cain G.D., A.Yardim, (1990) Darwinian Design of Digital Filters, Proc. IEEE Workshop on Genetic Algorithms, Simulated Annealing and Neural Nets Applied to Problems in Signal/Image Processing and Communications, 1-14.

Kale I., P.Belloni, G.D.Cain, E.Del Re, (1993) Guided Simulated Annealing with Tempering for FIR Filter Design Via Z-Plane Zero Manipulation, Proc. Int. Conf. on Digital Signal Processing, 390-395.

Karkowski I., (1993) Circuit Delay Optimization as a Multiple Choice Linear Knapsac Program, European Design Automation Conference, 419-423.

Landman P.E., J.M.Rabaey, (1993) Power Estimation for High Level Synthesis, European Design Automation Conference, 361-366.

Rechtin E., (1992) The Art of Systems Architecting, IEEE Spectrum, 66-69, October.

Sakouti, K., P.Abouzeid, M.Belrhiti, M.Crastes, and G.Saucier, (1993) Coherent Optimization Strategies for Multilevel Synthesis, European Design Automation Conference, 378-385.

Timmer A.H., M.J.M.Heijligers, L.Stok, and J.A.G.Jess, (1993) Module Selection and Scheduling using Unrestricted Libraries, European Design Automation Conference, 547-551.

Turner, L.E and B.K. Ramesh, (1986) Low Sensitivity Digital LDI Ladder Filters with Elliptic Magnitude Response, IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 7.

Wehn N., M.Held, and M.Glesner, (1991) A Novel Scheduling / Allocation Approach for Datapath Synthesis based on Genetic Paradigms, Proc. Int. IFIP Workshop on Logic and Architecture Synthesis, 47-56.

11. BIOGRAPHY

Chris J. Rouse received the B.Eng(Hon) degree in electronic engineering and the MSc degree in digital signal processing systems from the University of Westminster, London, in 1990 and 1992, respectively. He is currently working towards the PhD at the University of Westminster, London. His research interests include DA, DSP, and VLSI. He is a member of the IEEE, the ACM, and the IEE.

Alison J. Carter received the B.Sc degree in Mathematics from Imperial College, London, in 1977. She is currently a senior lecturer at the University of Westminster, London. Her research interests include all aspects of CAD.