Aspects of Digital Evolution: Geometry and Learning.

Julian F. Miller[1] and Peter Thomson[1]

[1]Department of Computer Studies, Napier University, 219 Colinton Road, Edinburgh, EH14 1DJ, UK. Email: j.miller@dcs.napier.ac.uk, p.thomson@dcs.napier.ac.uk
Telephone: +44 (0)131 455 4305

Abstract. In this paper we present a new chromosome representation for evolving digital circuits. The representation is based very closely on the chip architecture of the Xilinx 6216 FPGA. We examine the effectiveness of evolving circuit functionality by using randomly chosen examples taken from the truth table. We consider the merits of a cell architecture in which functional cells alternate with routing cells and compare this with an architecture in which any cell can implement a function or be merely used for routing signals. It is noteworthy that the presence of elitism significantly improves the Genetic Algorithm performance.

1. Introduction

There is now a growing interest in the possibilities of designing electronic circuits using evolutionary techniques. [2, 3, 4, 5, 6, 7, 8, 10, 11, 12].

Arithmetic circuits are interesting examples to choose to use to examine the issue of evolving digital circuits [1, 9] since there are well known conventional designs with which the evolved solutions can be compared. Additionally arithmetic circuits are modular in nature so that there is practical interest in trying to evolve new efficient designs for the small building blocks. Indeed we showed in our earlier work that more efficient designs could indeed be produced using evolutionary techniques (e.g. the two-bit multiplier). We demonstrated how it was possible to evolve modular designs from which the general principle for building the larger system could be extracted (e.g. the two-bit ripple-carry adder). We also found that a modest increase in circuit complexity could mean that evolving the circuit became enormously more difficult (e.g. the three-bit multiplier). Previously all our attempts to evolve digital circuits have involved evaluating the fitness of chromosomes using the *complete* truth table. Clearly such a procedure can lead to very slow evolution if the size of the truth table is large. In this paper we set out to examine whether it is feasible to use randomly chosen examples to evaluate the fitness of the chromosome.

Recently we have begun to consider the relationship between the architectural platform used for circuit evolution and the ease with which circuit function can be evolved, and we examine a number of different configurations of functional cells and routing cells. In one scheme the rectangular array is divided into columns of functional cells alternating with routing cells, in another, the functional and routing cells make up a chequer-board pattern. These schemes are compared with one in which the GA decides whether a cell is to be used for function or routing. The results are presented in section 4.

All the experiments reported in this paper have involved a new chromosome representation that we have developed which closely follows the structure of the Xilinx 6216 FPGA. This differs from the previous structure on which we evolved arithmetic circuits which was modelled on less specific netlists. Although we have not yet employed the chip directly in the execution of our genetic algorithm, we are confident that the new representation will readily translate into the bit strings required to configure the device without violating the routing constraints.

2. A Chromosome Representation which Accurately Models the Xilinx 6216 Chip

Since we are considering only combinational designs (non-sequential) it is vital to allow only *feed-forward* circuits. To achieve this we chose a cell connection scheme in which inputs are fed into a cell which are East-going or North going only. If the cell is a Multiplexer (mux) then we allow the control input of the mux to arrive from the North (indicated by 'S'). This scheme is shown in Figure 1 below:

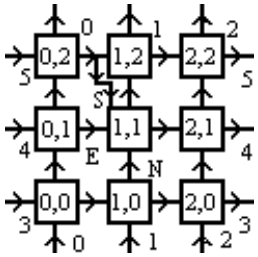


Fig. 1. The feed-forward connection of cells

The cells are numbered according to their column and row position with the origin at the bottom left hand corner of the array. All arrows pointing towards (outwards from) a cell represent inputs (outputs). The primary inputs connect to cells on the leftmost column and lowest row. The primary outputs exit the cells which are located on the topmost row and the rightmost column (in this case cells in column and row two). The cells are allowed to be one of the types shown in Table 1, where A and B represent 'E' and 'N' inputs and C represents 'S' input. If a cell at position (col, row) is a mux then the 'S' input is assumed to the 'E' output of the cell located at position (col-1, row+1). If the mux cell is located at column zero then we take the control input of the mux from the primary input located at position (-1, row+1). In such a scheme cells in the top row of the cellular array are not allowed to be multiplexers.

The chromosome has four parts; functional, routing, input, and output. The functional chromosome is a set of integers representing the possible cell types as indicated by gate type in Table 1. For N cells in the rectangular array there are N pairs (i,j), i ,j ε {0,1,2}, of routing genes which make up the routing chromosome, where the first (second) element of the pair represents the North (East) output. If i or j equals 0 then the North (East) output is the connected to East (North) input, and if i or j equals 1 then the North (East) output is connected to the North (East) input If a routing gene is 2 then the corresponding cell output is a function of the cell inputs. Thus, the routing chromosome ignores the corresponding gene in the functional chromosome if its value is set to 0 or 1. The input chromosome has #rows + #columns elements. Each element can take any integer from 0 to #primary_inputs - 1. The output chromosome has #primary_outputs elements which represent the places in the cellular array from which the primary outputs are to be taken. To illustrate the interpretation of the chromosome

consider the chromosome example (Table 2), which represents a chromosome for a 3x3 array of cells as seen in Figure 1. For the sake of argument imagine that the target function is a 1-bit adder with carry. This has three inputs A, B, Cin and two outputs S and Cout.

Table 1. Allowed functionalities of cells

Gate Type	Function	Gate Type	Function
-4	!A & !C + !B & C	6	A ^ B
-3	A & !C + !B & C	7	A | B
-2	!A & !C + B & C	8	!A & !B
-1	A & !C + B & C	9	!A ^ B
1	0	10	!B
2	1	11	A | !B
3	A & B	12	!A
4	A & !B	13	!A | B
5	!A & B	14	!A | !B

We read the chromosome in the following cell order (0,0) - (2,0), (0,1) - (2,1) and (0,2) - (2,2) as in Figure 1. The inputs and outputs are also read in as shown in Figure 1. Thus the cell at (1,1) is a multiplexer of type - 1 (see Table 1). Its outputs as indicated by (2,2) are routed out to North and East. The inputs on the bottom row are Cin, A, Cin, and along the left edge, A, B, B. The Outputs S and Cout are connected to the top row middle cell and top right hand corner cell (east output) respectively.

Table 2. Example chromosome for 3x3 array

Functional Part	Routing Part	Input part	Output part
2,9,11,12,-1,6,14,4,-3	0,1,2,1,1,2,2,2,2,2,0,0,1,1,1,0,2,2	2,0,0,2,1,1	5,1

In the genetic algorithm we used uniform crossover with tournament selection (size 2). Winners of tournaments are accepted with a probability of 0.7. The routing chromosomes are all initialised to 2; 0 and 1 are only introduced by mutation, this is found to be more effective than random initialisation. We use a fixed mutation rate which can be expressed as the percentage of all genes in the population to be randomly altered. The breeding rate represents the percentage of all chromosomes in the population which will be replaced by their offspring.

3. Aspects of the Evolution of Digital Circuits

3.1 How does the effectiveness of learning depend on the number of examples?

One of the fundamental problems with attempting to evolve digital circuits is that the size of the truth table grows exponentially with the number of inputs. This is not a problem if one is only trying to evolve quite simple circuits but would be a major difficulty if one were trying to evolve larger systems. The problem with digital circuits is that they are effectively useless even if a *single* output bit is incorrect. The degree of specification of the properties of an analogue circuit is not nearly so great as one is

usually trying to obtain output characteristics to within a certain error. The question of interest is whether it is necessary to present all the examples in the truth table to every individual chromosome, or whether one can merely distribute all the examples over an entire run of the GA. Secondly if one does present an incomplete set of examples to each chromosome for fitness evaluation, how should this be done? We decided to carry out a number of experiments to investigate this. We fixed the total number of examples presented to the GA while we varied the population size, the number of examples, and the number of generations. The examples were generated randomly and we looked at two scenarios: (a) a set of random examples was generated for each new population and each chromosome fitness was evaluated using this, (b) as set of random examples was generated independently for each individual chromosome. Using an incomplete set of examples for fitness evaluation means that the evaluated fitness for the chromosome is unlikely to be the same as the fitness calculated using the entire truth table. We refer to the former value as the apparent fitness, and the latter value as the real fitness. Throughout the GA run the apparent fitness is used except where the number of examples happens to equal that in the truth table. We always evaluate the real fitness of the best solution in each GA run after it has terminated. It is only by doing this that we can determine the effectiveness of this approach.

3.2 Functionality and Routing

Our previous method of evolving pure netlists had been designed to limit the number of routing connections that each functional block could use to connect to its neighbours. We wondered whether it might make circuits easier to evolve using the new representation if this too was modified to limit routing and functionality. In other words, could the evolutionary process be forced, by limiting available resources, to converge more quickly to a desired 100% functionally correct solution. We conducted experiments in which the number of functional blocks and the number of routing blocks is pre-determined by the chosen geometry. In the linear chromosome, every odd numbered gene is chosen to represent a pure routing cell (we refer to this as a *differentiated* structure - an undifferentiated structure being one where there is no imposed difference between the cells). The functional cells were initialised to possess no routing, but could be mutated so that their outputs became routes. Figures 2 (a) and 2 (b) below show the arrangement which would result from the selection of 3 x 7 and 3 x 8 geometries respectively - where F represents a functional block and S is a switching block.

F S F S F S F	F S F S F S F S
S F S F S F S	F S F S F S F S
F S F S F S F	F S F S F S F S
(a)	**(b)**

Fig. 2. Two differentiated structures with different geometries

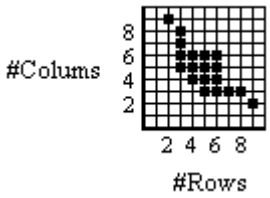


Fig. 3. Geometry map for the experiments on the 2-bit multiplier

Thus if the number of columns in the chosen geometry is odd the functional and routing cells are arranged in a chequer board pattern while if the number of columns is even the entire columns of functional cells alternate with columns of routing cells. We carried out a large number of experiments with the geometries shown in Figure 3.

4. Results

In section 4.1, experiments were carried out on a 3-bit binary multiplier, whereas all remaining experiments have been carried out on a 2-bit binary multiplier. The GA was always run with a 100% breeding rate, a 1% mutation rate. The fitness of the chromosomes was calculated as the percentage of correct output bits.

4.1 Is Learning by Example Effective?

In Figure 4, parameters and results are presented for experiments in which a fixed number of randomly chosen examples (input combinations) were evaluated by every member of the population. It should be noted that in this set of experimental results, the total number of examples, which is equal to the product of the following quantities: number of randomly chosen examples, number of generations and population size. This is the total number of examples evaluated by the GA in each run. In figure 4 this is shown in multiples of 10^6, whilst the number of generations is presented in multiples of 10^3. We varied the number of random examples presented to each chromosome in the range 8 to 32. Our motivation here was to determine whether we could evolve the function using a subset of the total number of examples. We did not look at numbers of examples between 32 and 64 because we were attempting to investigate whether the circuit could be evolved whilst making a significant saving in terms of numbers of truth-table comparisons. Each experiment involved 10 runs of the GA. The standard deviation was calculated using the best solutions from each run. From all of these experiments, the first obvious conclusion is that evolution is much more successful at finding circuit design solutions when all possible examples (i.e. the entire truth table) is presented for fitness evaluation (Figure 4, result 2, GA with population size 20). It would appear that whenever a subset of examples is used the information is too sparse to allow the GA to properly converge upon the optimal solution (a circuit which is 100% functionally correct). This apparent lack of information with which the GA has to work is borne out by the fact that when the average final generation at which there was a design improvement (the Δ generation) is analysed, it is found that fitness evaluations based upon the full example set produce a Δ which was approximately half the total number of generations. The Δ of all other runs was a factor of 10-20 smaller, thus, when a small example subset is used, the GA finds it very difficult to improve. An extreme case of this is when only 8 examples (from a total possible of 64) was used. In this case, the GA always achieved 100% apparent correct functionality for the examples given, and the Δ value was extremely small (<10 generations in all cases).

Additionally, further evidence to support this point, regarding the lack of information available to the GA when evaluating fitness on a reduced sample set, is that in each run the algorithm always identifies the best available fitness. This is demonstrated by the fact that the standard deviation from the sample of these results was always zero. Thus, the GA always converged to the same fitness value. However,

the standard deviation was not zero when the full truth table was used. It would seem that reducing the sample set to a subset of the entire truth table either makes the search space too discrete (by the removal of vital information), or that the chromosomes carry little memory about past generation performance because the basis for optimisation is apparently changing continually.

These results suggest that there is going to be very little hope that circuit designs can be directly evolved in this way by using sample subsets of the entire truth table, and that one must be prepared to accept that evolution needs perfect information regarding the circuit to be implemented. Indeed, it would seem that the greater the loss of information regarding the circuit to be evolved, the greater difficulty the GA has in performing well.

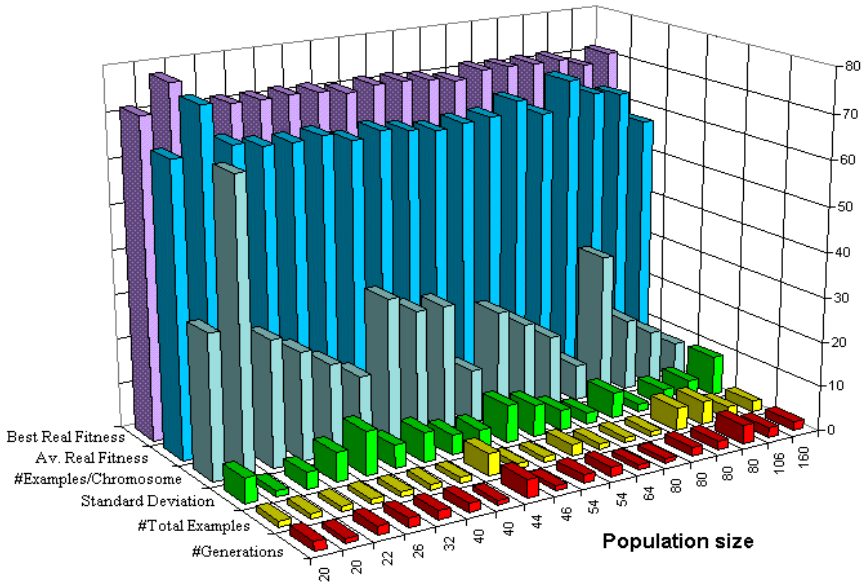


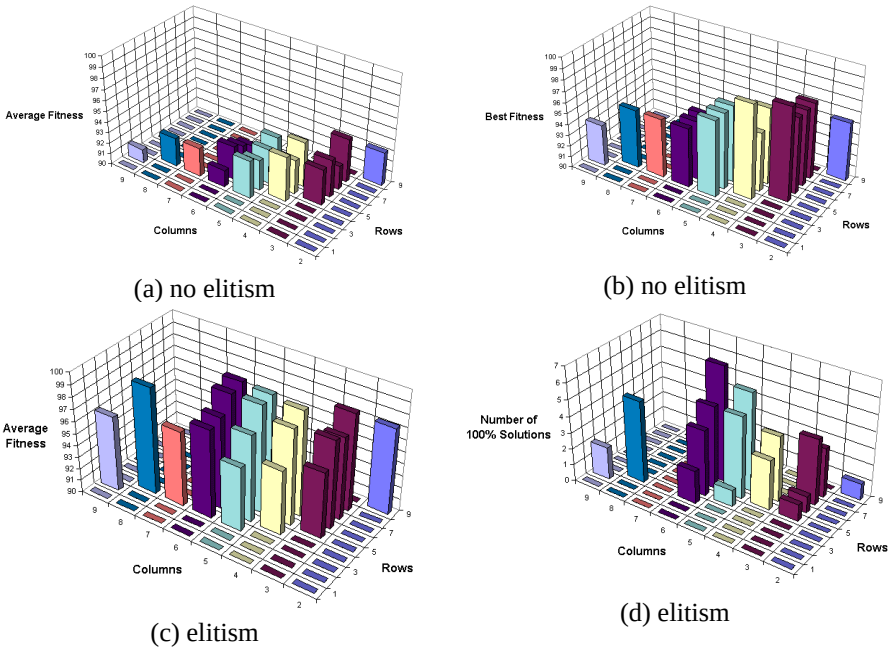
Fig. 4. Results for population based random examples.

In fact, it makes very little difference to the GA performance whether scenario (a) is used, where the same set of random examples is used for each population member, or scenario (b) is used, where a new random set of examples is generated for each chromosome within the population. Even increasing the total number of examples examined by the entire GA run does not produce a significant increase in performance (Figure 4, results 8, 15 and 16, population sizes 44 and 80). What seems to be much more important is the fact that the set examples being used has been reduced in the first place. The truly outstanding result in Figure 4 is result 2, with population size 22, where all examples from the entire truth table were used.

4.2 Choosing Cell Layout and Geometry

One of the potential problems with attempting to evolve circuits is the difficulty in routing the designs within the constraints of the implementation platform. This is

because devices do not possess an infinite routing resource with which to inter-connect its functional blocks. In our earlier representation we attached no cost to routing whatsoever and this made it easier to evolve functionally correct designs. When we carried out 10 runs of the GA using a 4x4 geometry of cells we obtained an average fitness of 98.12 (including numerous 100% solutions). However using our new chromosome representation we obtained an average of 94.06 with no 100% solutions. We decided to investigate this by initialising the circuit cell structure with increased routing resources, and further allowing initialised functional blocks to mutate into routing cells (the differentiated model from above). This could be compared with a scheme in which the Genetic Algorithm decides for itself how to employ cells, either as functional cells, routing cells, or indeed, a mixture of both (undifferentiated model). Thus we carried out ten runs of the GA (with 10,000 generations and population size 50) with geometries ranging from 9 rows by 2 columns to 2 rows by 9 columns (see Figure 3). It should be emphasised that when the number of rows and columns is fixed, this represents the *maximum* number of cells which may be used. In reality, the circuits may use any number of cells up to and including this maximum, and so the evolutionary process is in effect choosing the size. We also looked at the effect of using elitism. The results for the undifferentiated model are shown in Figures 5(a) - 5(d) while those for the differentiated model are seen in Figures 6(a) - 6(d). The first immediate conclusion which can be drawn is that the use of elitism has a very considerable favourable impact on the quality of the solutions. Without elitism no 100% functional solutions were obtained with the experimental parameters. Indeed for some geometries 10 runs of 20,000 generations were carried out again no 100% functional solutions were obtained without elitism. The reason why in Figures 5(d) and 6(d) the graphs show “Number of 100% solutions” was because when elitism was used a great many cases had a best solution (of the ten runs) which was 100% or extremely close. When Figures 5 and 6 are compared with one another it appears to show that an undifferentiated cell structure is more effective than the differentiated pattern. However, one should be cautious about this conclusion because the number of functional cells which are available in a differentiated structure is approximately half that of the undifferentiated pattern. We know from our previous results that seven functional cells are the likely to be the theoretical minimum required to build the 2-bit multiplier, and many of the differentiated patterns do not possess a realistic number of actual functional cells to be able to find a 100% correct solution (i.e. the functional resources allocated are close to the theoretical minimum). Formerly, in our original chromosome representation, we were able to evolve many more 100% correct solutions because there was no effective cost incurred for routing. In other words, only functional cells were considered as bearing any actual cost. However, in this model, which is much closer to the actual Xilinx chip, cells have to be allocated to routing tasks and so represent a real resource cost. The more cells one gives over to pure routing functions - thereby reducing functionality, as in the differentiated cell approach - the more closely one operates to the resource margin. Therefore, a direct comparison of the undifferentiated and differentiated cell pattern is perhaps not entirely appropriate.



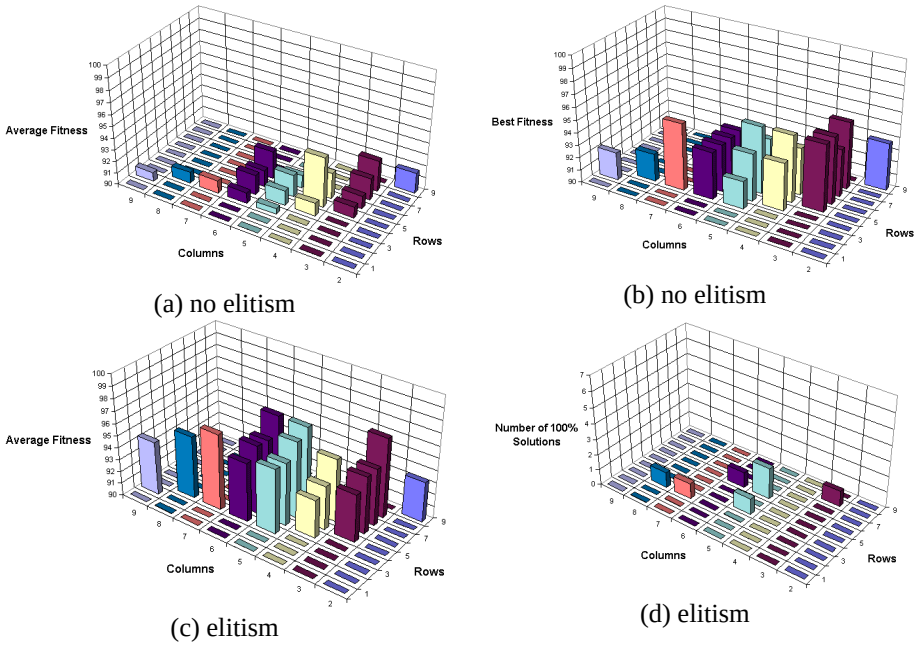
(a) no elitism (b) no elitism

(c) elitism (d) elitism

Fig. 5. Experimental results of runs with undifferentiated cell structure

Another feature which shows up the strong dependence of evolutionary success on functional and routing resources is depicted in Figure 6(d) where the number of 100% correct designs increases dramatically as resources are increased (by increasing cell geometry - with 6 x 6 as the most effective). What needs to be established is whether the differentiated cell pattern can become more effective than the undifferentiated approach when equal numbers of functional resource are allocated.

These results suggest that there needs to be a lot more work done to establish the relative importance of routing and functionality in attempting to evolve circuits. It is very tempting to concentrate purely on functionality when designing the GA for circuit evolution, and ignoring the inter-connectivity of cells and the available routing resources on the target device. There also needs to be a more open view of available architectures for evolution. It may be that the cell structure and connectivity arrangements currently employed by Xilinx are not best suited to the direct evolution of circuits, and that the entire process would benefit greatly from a more sympathetic internal architecture.



(a) no elitism (b) no elitism

(c) elitism (d) elitism

Fig. 6. Experimental results of runs with differentiated cell structure

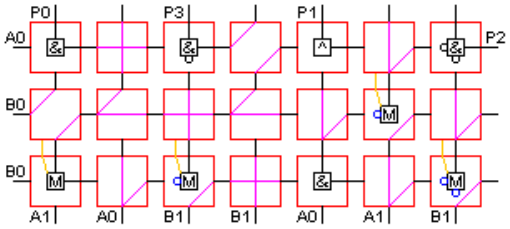


Fig. 7. Cell layout for fully evolved 2-bit multiplier

Figure 7 shows one of the 100% functionally correct evolved circuits. Each cell closely mimics the actual cells used on the Xilinx FPGA parts. As in our chromosome representation, the North and East (North and East going) connections to each cell are inputs, whereas the South and West going connections are the outputs. This means, therefore, that the cell at location 2,0 on the grid is a multiplexer (designated by the M), with one input B1 and the other A0 (both primary inputs on the truth table) routed from neighbouring cell (1,0). The control line for the multiplexer is passed down from the cell layer above and is actually the output of the multiplexer located at (0,0). The smaller square inside the cells indicates the cell's use as a functional logic gate, and the small circles on inputs and outputs of these denote logical inversion. The actual chromosome from which this circuit is generated is shown below:

-1 0 -3 0 3 0 -4 0 8 0 4 0 -3 0 3 0 4 0 6 0 8 (Functional)
2 2 1 0 2 0 1 1 2 2 1 0 2 0 0 0 0 1 1 1 0 1 1 0 2 0 1 0 2 2 1 1 2 2 0 0 2 2 1 0 2 2 (Routing)
0 1 2 2 1 0 2 3 3 1 (Inputs)
2 9 4 0 (Outputs)

Fig. 8. The chromosome which produced the circuit of Figure 9

Note that the zero entries in the functional part of the chromosome denote the absence of cell functionality. In other words, the routing part of the chromosome will never refer to these genes because these cells are only ever used as routing paths.

5. Conclusions

This paper has looked at a number of the issues that face researchers who are using evolutionary methods to create digital circuits directly. We are attempting to do this on a structure which is very closely modelled on that used by Xilinx on their 6216 FPGA part.

Experiments were conducted with regard to the evolution of a 2-bit multiplier circuit design on this new representation, and results gathered which examined: (a) the possibility of using truth table sample sets (as opposed to the entire truth table), and (b) the issue of providing the circuit representations with greater routing facilities (by differentiating cells as either functional or routers).

The first conclusion we may draw from our results is that it is extremely difficult to evolve functionally correct designs - even for a circuit as simple the 2-bit multiplier - using samples which are subsets of the truth table. In other words, it seems that the GA cannot use incomplete training data to derive the operational circuit, but requires the full truth table as a basis for fitness evaluation. There appears to be too great a loss of information concerning the circuit to be evolved for this method to succeed. This is clearly unfortunate as it means that the GA must be able to examine the entire truth table for a particular function - and this may be very large if the number of input variables is large. It is worth noting that the performance of the GA was not markedly different for the various runs involving different numbers of examples (where this number was less that the total number available in the full truth-table). This is surprising, but it may be due to the fact that without a very large percentage of the truth-table to work with, the GA is not much different from a random search. Clearly, this is a matter for further investigation.

We saw earlier that the number of cells used by the chromosome is variable but is bounded by the maximum number of cells available within a fixed geometry. When one considers allowing the maximum number of cells to be evolved, one would need to take into account the fact that initially smaller numbers of cells tend to produce higher fitnesses, although they eventually can be insufficient to realise the desired function. We feel that one needs to get the GA to reliably produce 100% functional solutions before one attempts to evolve the geometry.

Another conclusion is that using a GA which employs elitism is a considerable advantage in finding designs - using this chromosome representation - which are 100% functionally correct. Without elitism the GA struggled to find any fully correct solutions for what is essentially a very simple circuit, but with elitism the results were markedly improved. Further, there appears to be a strong dependency upon the cell resources given to the GA and its ability to deliver fully working solutions. We are intend to carry out further experiments in which we allocate equal functional resources to both differentiated and undifferentiated cell structures to ascertain if one technique is indeed more effective. Our original, more successful, chromosome representation employed effectively resource free routing, and we feel that an cell pattern or

architecture which devotes more to routing must provide greater opportunity to evolve circuits which are functionally correct. It is possible that we could improve the performance of the GA by using the long-line routing that is incorporated into the Xilinx 6216 architecture.

References

[A] *Lecture Notes in Computer Science - Towards Evolvable Hardware*, Vol. 1062, Springer-Verlag, 1996.

[B] Higuchi T., Iwata M., and Liu W., (Editors), *Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), Lecture Notes in Computer Science*, Vol. 1259, Springer-Verlag, Heidelberg, 1997.

1. Fogarty T. C., Miller J. F., and Thomson P., "Evolving Digital Logic Circuits on Xilinx 6000 Family FPGAs" in Soft Computing in Engineering Design and Manufacturing, P.K. Chawdhry,R. Roy and R.K.Pant (eds),Springer-Verlag, London, pages 299-305,1998.
2. Goeke M., Sipper M., Mange D., Stauffer A., Sanchez E., and Tomassini M., “Online Autonomous Evolware”, in [B], pp. 96 -106
3. Higuchi T., Iwata M., Kajitani I., Iba H., Hirao Y., Furuya T., and Manderick B., “Evolvable Hardware and Its Applications to Pattern Recognition and Fault-Tolerant Systems”, in [A], pp. 118-135.
4. Hemmi H., Mizoguchi J., and Shimonara K., “ Development and Evolution of Hardware Behaviours”, in [A], pp. 250 - 265.
5. Iba H., Iwata M., and Higuchi T., Machine Learning Approach to Gate-Level Evolvable Hardware, in [B], pp. 327 - 343
6. Kitano H., “Morphogenesis of Evolvable Systems”, in [A], pp. 99-107.
7. Koza J. R., *Genetic Programming*, The MIT Press, Cambridge, Mass., 1992.
8. Koza J. R., Andre D., Bennett III F. H., and Keane M. A., “Design of a High-Gain Operational Amplifier and Other Circuits by Means of Genetic Programming”, in Evolutionary Programming VI, Lecture Notes in Computer Science, Vol. 1213, pp. 125 - 135, Springer-Verlag 1997.
9. Miller J. F., Thomson P., and Fogarty T. C., "Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study", in Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: D. Quagliarella, J. Periaux, C. Poloni and G. Winter (eds), Wiley, 1997.
10. Sipper M., Sanchez E., Mange D., Tomassini M., Perez-Uribe A., and Stauffer A., “A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems”, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No 1., pp. 83-97.
11. Thompson A., “An evolved circuit, intrinsic in silicon, entwined with physics”, in [B], pp. 390 - 405.
12. Zebulum R. S., Pacheco M. A., and Vellasco M., “Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications”, in [B], pp. 344 - 358.