Modular Design from Gene Regulation in a Cellular System

Song Zhan, Julian F. Miller, Andy M. Tyrrell

Abstract— **In nature, modules pervade organisms at all levels and construct complex and scalable organisms. Modularity is fundamental for achieving large and complex systems. In engineering design, as required systems become larger, the complex and modularity of the design becomes more important. Inspired by nature, this paper introduces a bottom-up construction method to construct digital circuits using gene regulation in a cellular system. It uses modular design to achieve scalable combinatorial systems.**

I. Introduction

With fast growing technology and increasing requirements of applications, complexity and scalability problems are growing in Engineering. However, it is apparent that nature presents us with enormously complex, yet well-organized intelligent systems. Biologists tell us that modularity pervades biological complexity [6]. All systems are characterized by some degree of coupling (loose or tight) between components and are to some degree, modular. Modularity is the general term describing the degree to which a system's components can be separated and recombined, and refers both to the tightness of coupling between components and the degree to which the "rules" of the system architecture enable (or prohibit) the mixing and matching of components [12]. It is already a central concept in many studies such as engineering, neurosciences, computer science and biology. In terms of design, it aims at identifying independent, standardized, or interchangeable units to satisfy a variety of functions [8] and makes complexity manageable, enables parallel work and is tolerant of uncertainty [2].

In biological research, modularity is considered in different aspects, for instance, evolutionary biologists perceive a module as a subunit of the whole organism while developmental biologists refer to it as a combination of lower level components [1]. Modularity is an important property of the genotype-phenotype mapping and is believed to benefit the evolutionary process. It enhances the ability of the genetic system to generate adaptive variants and enhances evolvability because with modularity, genetic changes tend to map to changes in a small number of phenotypic traits alone and hence the genome can respond to selection on these traits, independently of the rest of the phenotype.

The structure of modular organisms such as plants and many invertebrate phyla appear to be highly branched and iterative. Module reuse is one of the main construction process in building such organisms and is the major processes for dealing with system complexity. It is argued that primary modules and second-order modules are two aspects of modular growth that are relevant to complexity [5] where the primary modules are the fundamental building units and second-order modules are a higher level of organization constructed by primary modules. This statement has been supported in the evolutionary design field as the characteristics of modularity, regularity and hierarchy of objects appear to be determinants of object's complexity in [7]. In [3] the concepts of modularity, hierarchy and repetition are defined to exploit their efficiency in search algorithms. Preserving modules in evolutionary programming has become a subject of study such as in [9], automatically defined functions (ADFs) are defined to preserve the sub function calls.

Inspired by the mechanisms that occur in nature, we have modelled an evolutionary developmental gene regulation network process for electronic circuit design [13, 14]. This paper will investigate the modularity in this model for circuit designs.

II. The Model

In the model, a system is built up from its underlying gene regulation in a cellular system and mapped into circuit design by a process inspired by protein synthesis in biology.

A. Gene Regulation

A gene is structured by regulatory sites and product sites. This is illustrated in Fig. 1, where the first two integers are regulatory sites and the last two are expression products. Regulatory sites function as the regulation conditions and once a gene regulates, it will generate products. Regulatory proteins are either enhancers or inhibitors. They construct positive and negative feedback loops in the gene regulation network. These loops serve as important aspects in the system as explained in [13]. Fig. 1 illustrates a gene regulation network constructed with three genes. The regulation process includes regulatory protein binding and product protein generation. Each active gene will generate different amounts of products depending on the production rate. These products will regulate other genes and form a network structure. Regulatory proteins function transiently as after regulation they are consumed. Genes can be regulated multiple times (in Fig. 1 gene 0 is regulated by gene 1 and 2, gene 1 is regulated by gene 0 and gene 2 is regulated by gene 1 and itself).

Song Zhan was with the Department of Electronics, University of York, UK. songzhan2000@gmail.com

Julian F. Miller is with the Department of Electronics, University of York, UK. jfm7@ohm.york.ac.uk

Andy M. Tyrrell is with the Department of Electronics, University of York, UK. amt@ohm.york.ac.uk

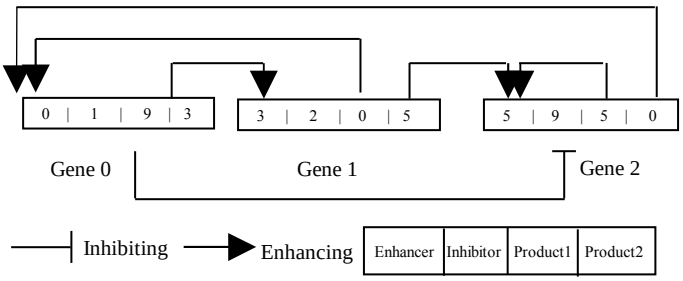


Fig. 1. A Three-Gene Regulation Network

B. Cellular Development

As in biology, in our model, the transcription process of protein synthesis produces mRNAs that are released from gene regulations in cells. Multicellularity is obtained from either cell division processes or multiple initial cells. Cell division is determined by predefined division proteins and occurs during gene regulation. It involves copying a mother cell's proteins through a pathway. Differential gene regulations in cells result from cell signalling, cell division and potentially different gene regulation loops and this leads to cell differentiation. Cell signalling includes protein diffusion through a cell membrane threshold and protein alteration as a consequence of a signalling pathway. Fig. 2 gives an example of signalling processes. It shows the signalling associated with a particular protein (4 in this case). When the level of diffusing protein is larger than the membrane threshold 30.0 it is able to diffuse and change to protein 1 in its neighbouring cells. The level of protein that diffuses is uniform and calculated from equation 1 [11].

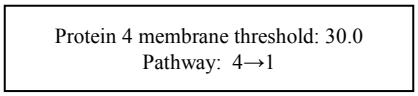


Fig. 2. Cell Signalling

$$P^{t+1}_{conc}=1/2P^{t}_{conc}+ 1/(2*\text{NumberofCellBorders})\sum P^{t}_{conc} \quad (1)$$

C. Circuit Construction

Fig. 3 presents the overall design process alongside its biological counterpart. A circuit is constructed during gene regulation and development by a process that is inspired by the protein synthesis process in biology. In biology, proteins are constructed by amino acids which are generated from gene regulation and mRNA translation. In electronic circuit design, elements required for design such as gates are generated from gene regulation and translation to gates. For example in Fig. 13, when gene 0, 1 and 2 are all active, they will generate mRNA: 6, 4, 1, 0, 6, 2. The mRNA is translated to gates: 2, 0, 1, 0, 2, 2. These gates are connected together inspired by the process of "linking" amino acids to construct polypeptide chains through a sequence of gene regulations. A Cartesian Genetic Programming (CGP) [10] construction method is used in this "linking" process. A netlist as in Fig. 3 represents the gate connections (first two integers in each group) and gate type (underlined integer in each group). The connection embedded in each gene in the square bracket e. g. [0, 1] at the left top of Fig. 3 is defined through evolution. The constructed circuits build cell functions as illustrated in Fig. 4. A mathematical modulo function is used to translate the gene products to gate types. Table 1 is a modulo translation look up table with 8 types of regulatory product proteins used in the example shown in Fig. 3. The circuit cells are then connected together to obtain the overall cellular function using another CGP netlist. As illustrated in Fig. 5, the netlist is 012, 012, 34, where each group represents three inputs of each cell, the last two integers represent which cells the outputs come from. Each cell is referenced by one integer indicating one connection point, e.g. 3 and 4 in Fig. 5.

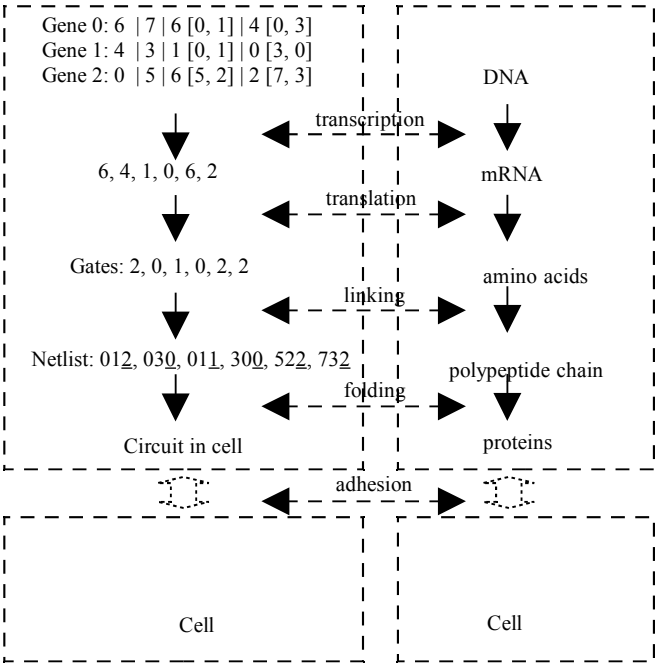
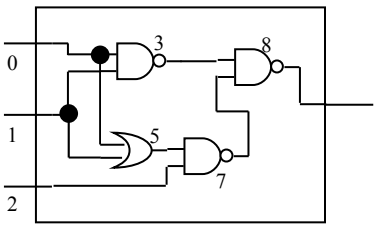


Fig. 3. Circuit construction processes

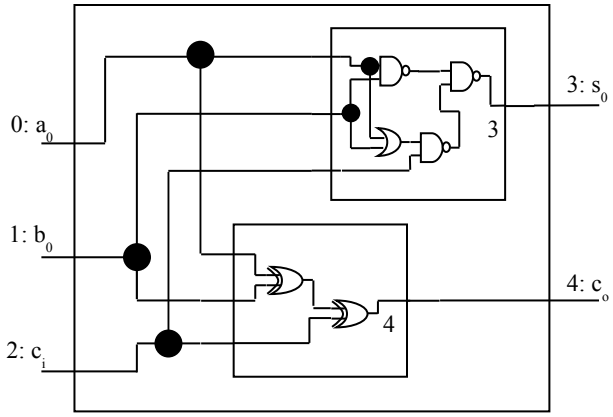
TABLE 1
PRODUCT-GATE MODULE TRANSLATION (MOD 4)

Gate Id	Product protein Id
0 (xor)	0, 4
1 (or)	1, 5
2(nand)	2, 6



netlist: 012, 030, 011, 300, 522, 732

Fig. 4. Constructed circuit in one cell



netlist: 012, 012, 34

Fig. 5. Cell Connection

D. Evolution Operation

Genotpye includes elements at the underlying biological level: genes in DNA (enhancers, inhibitors and gene products), cell pathway, cell membrane, production rate and elements at the application level (circuits): gate connection and cell connection. Fig. 13 and 18 show examples of evolved individuals. Mutation causes changes to elements within the predefined range: the number of regulatory proteins, protein concentration, gate and cell connections.

III. Modularity and Reusability in the System Construction

A. Hierarchical Modules in the system

This paper defines the modularity in the model using the definition that developmental biologists use. There are two types, hierarchically primary modules and second-order modules (or composite module as in [3]) in the circuit designs. As the fundamental building units, primary modules are logic gates for gate level designs as the example in Fig. 6 and the functions built of these in each cell serve as second order modules. These second-order modules can be both small circuits such as the cells in Fig. 4 or may function as primary modules: logic gates as the example in Fig. 7.

B. Module Repetition

Since the second order modules are generated from gene regulations, the same regulations in different cells will allow the reusability of modules. Regulations are determined by regulatory proteins in cells, so the regulatory proteins in cells will also control the reusability of modules. In [14] we show one way of repeating modules by introducing cell movement behaviour. Here we introduce a more general solution by starting gene regulations from multiple initial cells, in parallel. In this way, gene regulation modules and the module reuse are obtained. Fig. 9 shows an example of the repetition of the gene regulations of (A) when the individual in Fig. 8 develops with different numbers of initial cells and cellular environment sizes. In each group, the figure at the left side represents cell initialisation. This is defined during evolution. Each cell is initialized with one protein, e.g. in Fig. 8 the "Cell Initialization" section shows the evolved initial protein 0 with concentration 100. The figure at the right side represents gene regulation in each cell. In the example during regulation each gene generates two product proteins represented by different colour levels. Two initial cells in four cell environment are shown in (B), three initial cells in six cell environment are shown in (C) and four initial cells in eight cell environment are shown in (D). This gives an indication that it is possible to scale up to larger circuits. This application is explained in more detail in section IV.A.

C. Module Combination

Different cell initializations can achieve different gene regulation modules within their local area. This provides a way to obtain combinations of modules in larger environments by the development of an individual using a number of initial cells distributed in a cellular environment. For example, when the individual in Fig. 10 is placed in a 2 cell environment with different cell initialization as in the top row of Fig. 11, it will develop into three different regulation patterns as in the bottom row of Fig. 12. When this individual is placed in a 2*3 environment with 3 initial cells that combines three cell initialization as in Fig. 11 with constrained vertical signalling and division, it will develop to the pattern in Fig. 13 which combines the three patterns in Fig. 12. This mechanism inspires the construction of combinatorial circuits and is explained in section IV.B.

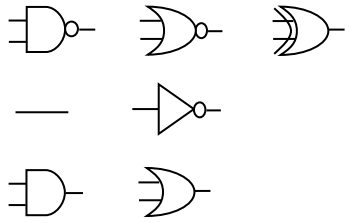


Fig. 6. Example of primary modules

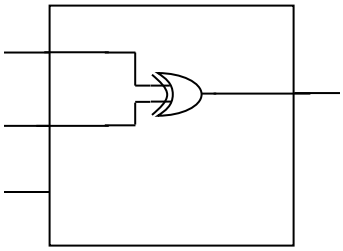


Fig. 7. Example of second order module: logic function

/----------DNA----------/
Gene 0 : 1 | 0 | 1 | 7
Gene 1 : 14 | 12 | 11 | 5
Gene 2 : 3 | 15 | 2 | 2
Gene 3 : 0 | 11 | 0 | 15
Gene 4 : 15 | 14 | 10 | 8

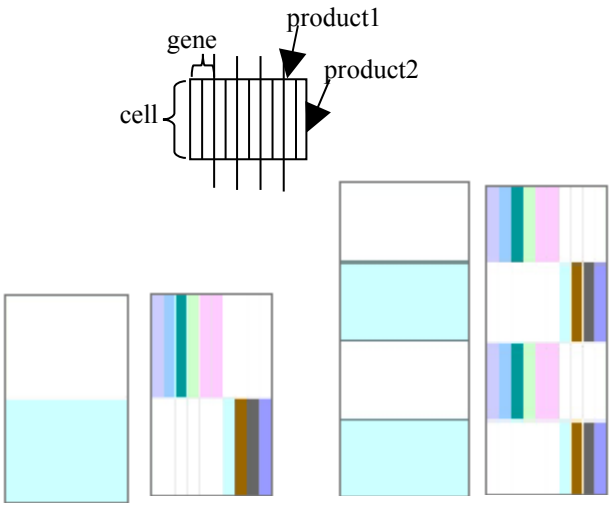
/----------Cell Pathway ----------/
0 | 11 , 1 | 1 , 2 | 0 , 3 | 13 , 4 | 0 , 5 | 6 ,
6 | 1 , 7 | 16 , 8 | 1 , 9 | 12 , 10 | 3 , 11 | 4 ,
12 | 3 , 13 | 11 , 14 | 7 , 15 | 6 , 16 | 14 ,
17 | 12 , 18 | 5

/------- Cell Membrane ---------/
0 | 112.94 , 1 | 26.69 , 2 | 99.57 , 3 | 27.64 ,
4 | 38.10 , 5 | 103.94 , 6 | 18.93 , 7 | 85.16,
8 | 64.53 , 9 | 7.79 , 10 | 46.67 , 11 | 88.49 ,
12 | 22.32 , 13 | 53.22 , 14 | 18.97, 15 | 64.24 ,
16 | 106.43 , 17 | 1.81 , 18 | 40.76 , 18 | 40.76

/------- Cell Initialization ------/
Protein 0 : Concentration : 100.0

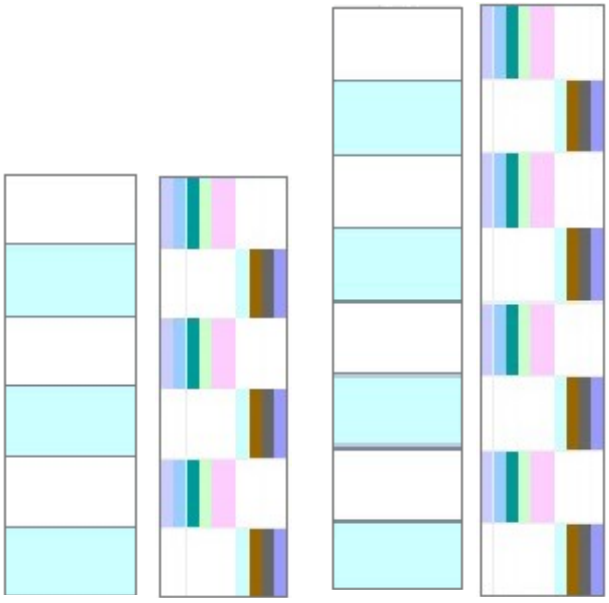
Key:
Gene Id: Enhancer | Inhibitor | Product1 | Product2
Pathway: protein id | transferred protein id
Membrane: protein id | threshold

Fig. 8. Genotype of the individual for gene regulation repetition



A: The development of a 1 zygote individual in a 2 cell environment

B: The development of a 2 zygote individual in a 4 cell environment



C: The development of a 3 zygote individual in a 6 cell environment

D: The development of a 4 zygote individual in an 8 cell environment

Fig. 9. Gene regulation repetition in different sized cell environments and different numbers of initial cells. The left figure in each group represents cell initialisation, the right figure in each group represents gene regulation in each cell.

/----------DNA----------/
Gene 0 : 1 | 0 | 1 | 7
Gene 1 : 14 | 12 | 11 | 5
Gene 2 : 3 | 15 | 3 | 2
Gene 3 : 0 | 11 | 0 | 15
Gene 4 : 15 | 14 | 10 | 8
/----------Cell Pathway ----------/
0 | 11 , 1 | 1 , 2 | 0 , 3 | 13 , 4 | 0 , 5 | 6 ,
6 | 1 , 7 | 16 , 8 | 1 , 9 | 12 , 10 | 3 , 11 | 4 ,
12 | 3 , 13 | 11 , 14 | 7 , 15 | 6 , 16 | 14 ,
17 | 12 , 18 | 5
/------- Cell Membrane ---------/
0 | 112.94 , 1 | 26.69 , 2 | 99.57 , 3 | 27.64 ,
4 | 38.10 , 5 | 103.94 , 6 | 18.93 , 7 | 85.16,
8 | 64.53 , 9 | 7.79 , 10 | 46.67 , 11 | 88.49 ,
12 | 22.32 , 13 | 53.22 , 14 | 18.97, 15 | 64.24 ,
16 | 106.43 , 17 | 1.81 , 18 | 40.76 , 18 | 40.76

/------- Cell Initialization ------/
Protein 0 : Concentration : 100.0

Fig. 10. Genotype of the individual for gene regulation combination

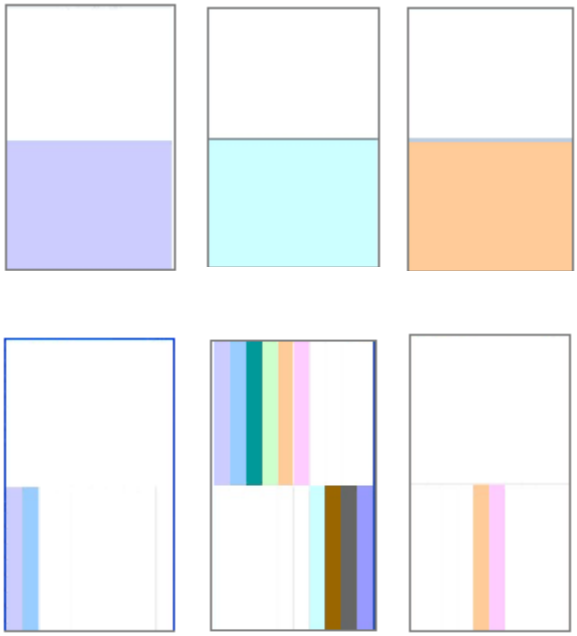


Fig. 11. The development of the individual in Fig. 10 in a 2 cell environment with different initialisation

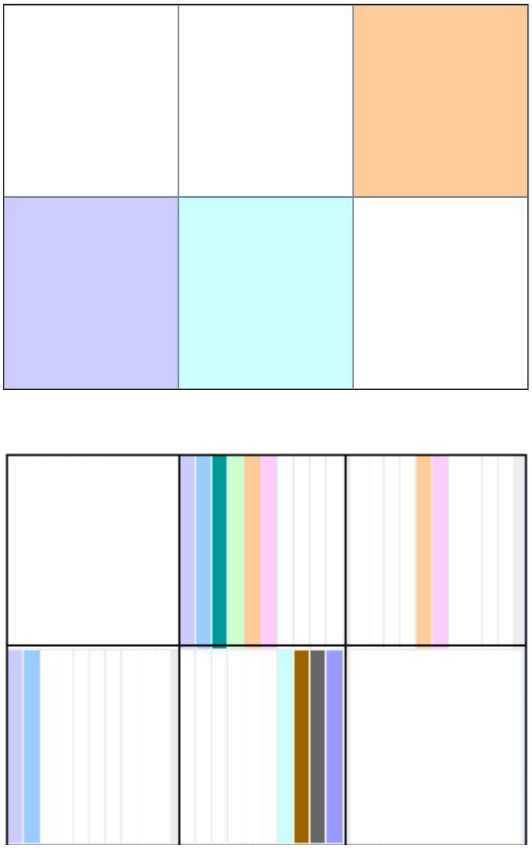


Fig. 12. The development of the individual in Fig. 10 in a 2*3 cell environment with a combinatorial initializations of Fig. 11 (top). This results in a combinatorial gene regulation patterns of Fig. 11 (bottom)

IV. Circuit design

A. Scalable Circuits

Adders and parity checking circuits are scalable and can be built up to arbitrary sizes by repeating small circuit functions. The fact that gene regulation can be repeated (in section III.B) should therefore be useful in tackling these problems. Once the evolutionary algorithm finds the basic circuits, e.g. 1 bit adder, it can be used as a module and reapplied in a larger environment and through the evolution of the cell to cell circuit connections, large circuits can be obtained. Thus the evolutionary search space can be reduced to only search cell connection. Fig. 13 shows the genotype of an evolved 1 bit adder in a 2 cell environment. Its gene regulation patterns and initial cell position are shown in Fig. 14. By applying multiple initial cells in larger environments, gene regulation pattern repeat the regulation pattern of the 1 bit adder and provide circuit cells for larger functions as the example in Fig. 15 illustrates. When the evolved one bit adder in Fig. 13 is placed in 4, 6 and 8 cell environments, the developed patterns repeat the gene regulation pattern in the 2 cell environment in Fig. 14. This provides the circuit cells for 2, 3 and 4 bit adders.

```
/-----------------------------DNA---------------------------------/
      Gene 0 : 0 |  2 | 1 | 0 | 2 0  | 0 1  | 0.99
      Gene 1 : 0 |  3 | 5 | 0 | 2 0  | 2 4  | 0.83
      Gene 2 : 4 |  2 | 0 | 3 | 1 2  | 3 3  | 0.91
      Gene 3 : 3 |  3 | 4 | 1 | 4 7  | 6 8  | 0.47
      Gene 4 : 4 |  4 | 5 | 4 | 3 6  | 11 5 | 0.55
/---------------------Cell Pathway -------------------------------/
          0 | 4 , 1 | 1 , 2 | 0 , 3 | 0 , 4 | 3 , 5 | 4
/---------------------Cell Membrane -----------------------------/
 0 | 20.0 , 1 | 100.0 , 2 | 20.0 , 3 | 20.0 , 4 | 80.0 , 5 | 70.0
/---------------------Cell Initialization -----------------------/
                Protein 0 : Concentration : 100.0
/--------------------Cell Connection -----------------------------/
                    1 2 0 1 3 0
Key:
Gene Id: Enhancer | Inhibitor | Product 1 | Product 2 | [Connection 1] |
[Connection 2]| rate
Pathway: protein id | transferred protein id
Membrane: protein id | threshold
```

Fig. 13. Genotype of an evolved one bit adder

Fig. 14. The gene regulation pattern and zygote position of the one bit adder in Fig. 13 developing in a 2 cell environment. The left figure represents cell initialisation, the right figure represents gene regulation in each cell.

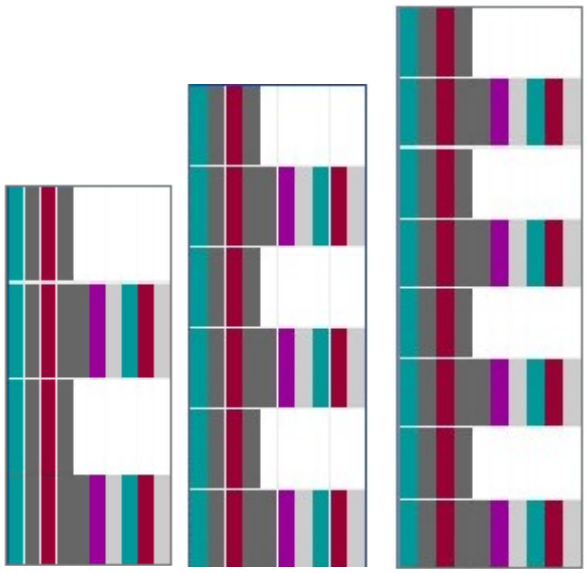


Fig. 15. The gene regulation pattern of the one bit adder in Fig. 13 developing in a 4, 6 and 8 cell environments. It repeats the pattern in Fig. 14 and prepares the circuit cells for 2, 3 and 4 bit adders as only the connections between them need to be evolved.

B. Combinatorial Circuits

Combinatorial circuits are often constructed by many sub functions, e.g. an ALU combining arithmetic functions and logic gates. When the size of the circuit increases, it is necessary to use other basic circuits in circuit construction as discussed in [4]. In the model presented in this paper circuits are mapped from gene regulations so that the method described in section III.C could be used to design circuits that combine many sub functions.

This section attempts to achieve the ALU function illustrated in Fig. 16, which combines a 1 bit full adder, 1 bit subtractor, 2 bit multiplier and logic functions: AND, OR, NAND, NOR and PASS using an evolutionary algorithm. It investigates how the modules could be used in the evolved functions. In Fig. 16, an ALU architecture with 4 inputs and 4 outputs is constructed by 4*4 (or 3*4) cells. Each cell has 4 inputs and 2 outputs. As in CGP the integer in each cell is used for assigning order of connections.

Table 2 and 3 list the parameters used for development and circuit construction: 15 genes and 15 regulatory proteins are used to evolve an individual in a 4*4 environment. Development starts from 4*4 (or 3*4) cell initialization. Each cell is initialised with one regulatory protein. For instance, at the cell initialization section of Fig. 18, each integer represents which regulatory protein is initialized in each cell. Each circuit cell is set to have 4 inputs and 2 outputs but only the first 3 are used for adder and subtractor functions. To provide a PASS function, during mapping from gene regulation to circuit, when there is no active gene, the first input of the cell will pass to the outputs. Outputs of each function are obtained in sequence from the cell outputs. This is indicated in the evolved example shown in Fig. 17 by the different shapes on integer labels. Since each cell has two outputs, each cell will provide two new labels for later cells to connect to, function output labels are allocated in sequence starting from the last cell output.

A 1+4 evolutionary algorithm is used to search for individuals. Table 4 lists the parameters used during evolution. Initially 10 individuals are randomly generated to provide the possibility of population diversity. The best individual is chosen as the parent in each generation. This parent generates 4 children through mutations of the current generation. Mutation operations randomly mutate elements in genotypes as in the examples in Fig. 10 and 13 in each generation. Selection and generating children processes occur at every generation and stop at a predefined evolution generation: 50000. In the case that no better individuals are generated in the population, the newest child with the same best fitness as the parent will be selected to allow some genetic drift [9]. Fig. 18 lists an evolved genotype.

Fitness is evaluated at the 10th development cycle. In order to obtain a stable circuit construction, it is continually evaluated for the next 10 cycles. Individual fitness is by the average of the sum of the each subfunction's fitness at each development cycle based on their truth table. In the experiment case, each function fitness is: 16 for 1 bit adder, 16 for 1 bit substractor, 64 for multiplier, 2 for pass, 4 for NAND, 4 FOR AND, 4 FOR OR, 4 FOR NOR. Total full fitness of individual is 16+16+64+2+4+4+4+4=144.

TABLE 2
PARAMETERS OF INDIVIDUALS FOR CIRCUIT CONSTRUCTION

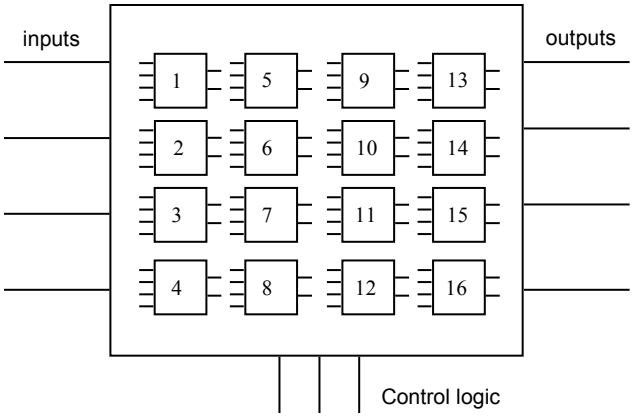
Number of outputs per cell	Number of inputs per cell	Number of inputs to ALU.	Logic gate set (id)
2	4	4	XOR (0), AND (1) OR (2), NOT (3) PASS (4)

TABLE 3
PARAMETERS OF INDIVIDUALS DURING DEVELOPMENT

Gene No.	Regulatory protein No.	Environment size	Initial protein No. (concentration)	Initial cell position
15	15	4*4 (3*4)	1 (100)	Every cell

TABLE 4
PARAMETERS IN EVOLUTION

Initial random individuals	10
Start fitness evaluation at development cycle	10
Continue fitness evaluation cycles	10
Mutation rate	0.025
Evolution Generations	50000
Evolution run number	30



NAND : 0, AND : 1, OR : 2, NOR : 3, PASS : 4
1 bit Adder : 5, 1 bit Sub : 6, 2 bit Multiplier : 7

Fig. 16. an ALU architecture with 4 inputs and 4 outputs constructed by 4*4 cells. Each cell has 4 inputs and 2 outputs. Control logic controls the output functions and also where the outputs are taken from are shown. As in CGP the integer in each cell is used for assigning order of connections

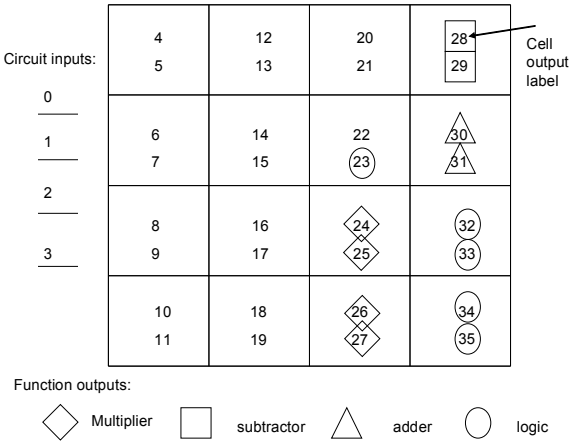


Fig. 17. The allocation of circuit outputs: different shapes represent the outputs of different sub functions. It shows which ALU outputs are taken from the circuit cells.

Fig. 18 shows one evolved individual's genotype with north and south direction cell signalling. Fig. 19 shows the cell initializations and developed gene regulations. It shows that although cells start from different initializations, there are only 3 different gene regulation which will contribute to 3 modules in the circuit. However, these modules are repeatedly reused many times to construct an ALU function. Fig. 20 shows the distribution of sub-function reuse in the cellular system. Different shapes represent different sub functions. It shows which circuit cells the sub functions are constructed with and shows the overlaps of cell reuse among sub functions.

V. Summary

Modularity pervades all levels of biology process and is the building block in biological construction. In the engineering field, it is also an important aspect in designing large systems. This inspires us to consider the contribution of a bio-inspired method for circuit designs. This paper investigates the modularity in a developmental gene regulation network model for circuit constructions. It shows how gene regulations might be controlled in cellular systems to obtain regulation repetition for the purpose of achieving scalability in evolutionary circuit design. Using an ALU circuit as the example of combinatorial circuits, it shows that the second order modules are repeatedly used in combinatorial circuit constructions. Compared with traditional evolutionary technique, using gene regulation in a cellular system achieves modular design from gene regulation differentiation in cells and the combination with differentiation. This suggests that large systems (many cells) can be evolved with small genotypes. For example, compare the minimum number of elements required in evolution. If a circuit needs 1000 gates and gate input number is 2, a direct representation using CGP would need at least 1000*3=3000 elements. Using the method described in this paper the minimum number of elements required can be calculated. Assume that there are 25 cells, the number of gene products is 2, the least number of genes would be gateNoRequired/(cellNo*productNo)=1000/(25*2)=20. If the number of proteins is 10 and the number of cell inputs is 4, then the number of elements involved in evolution will be geneNo*(regulatorySiteNo+productNo+gateInputNo+productRateNo)
+cellPathwayNo+cellMembraneNo+cellIinitialisationNo+cellConnectionNo=20*(2+2+2*2+1)+10+10+25+25*4=325.
Since cell initialisation and connections are involved in evolution, the equation also shows that there is a trade off between the number of cells and the number of genes required. Table 5 lists a few comparison of this when other parameters are used. It shows that cell number around 30 requires the least number of elements.

```
/------------------------------DNA--------------------------------/
    Gene 0 : 1   | 5  | 1   | 5  | 2 1    | 2 0     | 0.77
    Gene 1 : 11  | 1  | 11  | 3  | 5 1    | 0 6     | 0.65
    Gene 2 : 12  | 10 | 1   | 1  | 6 4    | 4 7     | 0.50
    Gene 3 : 10  | 7  | 9   | 6  | 9 1    | 2 5     | 0.18
    Gene 4 : 5   | 5  | 12  | 3  | 4 6    | 6 9     | 0.72
    Gene 5 : 14  | 11 | 3   | 8  | 4 5    | 13 0    | 0.41
    Gene 6 : 7   | 4  | 12  | 3  | 0 13   | 10 1    | 0.17
    Gene 7 : 7   | 8  | 10  | 7  | 12 5   | 3 12    | 0.82
    Gene 8 : 9   | 1  | 9   | 2  | 13 17  | 19 20   | 0.004
    Gene 9 : 9   | 2  | 7   | 14 | 7 0    | 16 4    | 0.67
    Gene 10 : 9  | 8  | 0   | 8  | 0 13   | 17 20   | 0.37
    Gene 11 : 11 | 11 | 12  | 0  | 3 18   | 5 0     | 0.02
    Gene 12 : 4  | 8  | 11  | 0  | 9 10   | 22 13   | 0.62
    Gene 13 : 12 | 11 | 14  | 9  | 1 18   | 15 5    | 0.14
    Gene 14 : 0  | 4  | 0   | 6  | 28 6   | 10 16   | 0.74
/---------------------Cell Pathway ---------------------------------/
0 | 7 ,  1 | 13 , 2 | 3 ,  3 | 5 ,  4 | 7 ,  5 | 4 , 6 | 0 , 7 | 7 , 8 | 1 ,
9 | 2 , 10 | 6 , 11 | 1 , 12 | 1 , 13 | 4 , 14 | 10
/---------------------Cell Membrane ----------------------------/
 0  | 30.0 ,  1 | 80.0  , 2 | 60.0 , 3 | 70.0 , 4 | 0.0 , 5 | 80.0 ,
 6  | 0.0 ,  7 | 80.0  , 8 | 30.0,  9 | 40.0 , 10 | 50.0 , 11 | 20.0 ,
 12 | 40.0 , 13 | 100.0 , 14 | 70.0
/----------------------Cell Initialization --------------------------/
        8 12 2 9 9 3 0 13 9 6 3 13 2 7 6 6
/---------------------Cell Connection ----------------------------/
0 0 3 1 0 2 1 2 0 2 0 4 6 4 6 4 1 6 2 2 4 3 5 5 0 5 2 9 2 4 10 4 1 3 2
10 8 3 8 3 8 10 0 10 4 11 13 14 4 11 11 2 8 16 16 6 7 17 17 1 4 2 13 11
```

Fig. 18. Genotype of an evolved ALU

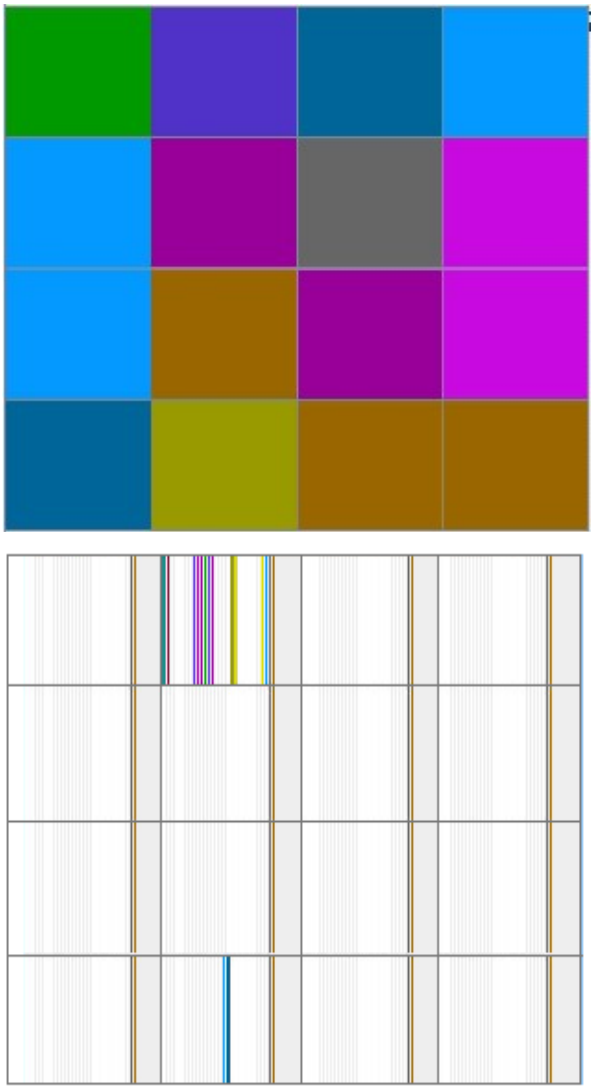


Fig. 19. The evolved cell initialisation (top) and gene regulations (bottom) of the ALU in Fig. 18.

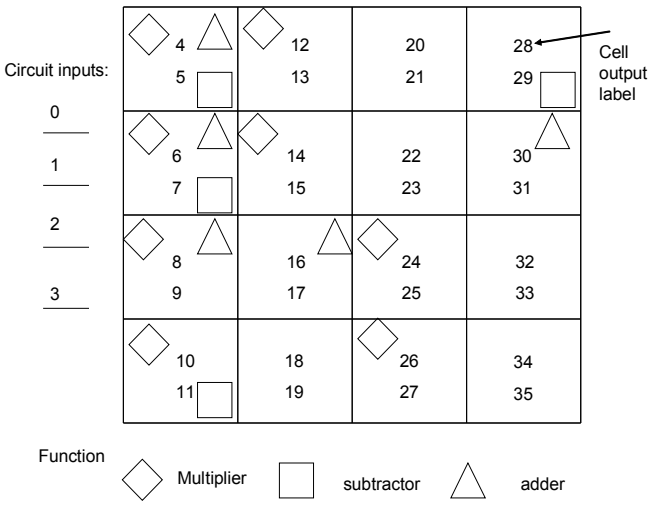


Fig. 20: Reuse of modules in the evolved ALU shown in Fig. 17. Different shapes represent different sub functions. It shows which circuit cells the sub functions are constructed with and shows the overlaps of circuit cell reuse among sub functions.

TABLE 5
CASE COMPARISON OF DIFFERENT CELL NUMBERS AND REQUIRED ELEMENTS IN EVOLUTION

CellNumber	1	20	30	40	60	80
Direct CGP			3000			
The system	4525	345	323	337	401	483

In the approach we describe, sub functions outputs are allocated to be taken from the last circuit cell outputs. This is to reduce the circuit cell connection length and reduce the evolution space so that it can focus on gene regulations. However, this will restrict the allocation of module usability of sub functions. Future work will improve cell connection mechanism to be more biology inspired and the evolution algorithm to take special consideration in different layers of genotype on how and when operations take place. Future experiments will concentrate on larger systems.

References

[1] Bolker, J. A. Modularity in development and why it matters to Evo-Devo. American Zoologist, 40: p770-776, 2000.

[2] Clark, Kim B. and Baldwin, Carliss Y. Modularity after the Crash, Harvard NOM Research Paper No. 01-05. Available at SSRN: http://ssrn.com/abstract=270292 or DOI: 10.2139/ssrn.270292, 2000.

[3] De Jong, E.D., Thierens, D., and Watson, R.A. Defining Modularity, Hierarchy, and Repetition. Proceedings of the GECCO Workshop on Modularity, regularity and hierarchy in open-ended evolutionary computation, pp.2-6. ACM. 2004.

[4] Cancho, R. F., Janssen, C. and Solé, R. The topology of technology graphs: Small world patterns in electronic circuits, Phys. Rev. E 64, pp. 32767, 2001.

[5] Hageman, J. S. Complexity Generated by Iteration of Hierarchical Modules in Bryyoza, Integer. Comp. Biol. 43: 87-98. 2003.

[6] Hartwell, L.H., Hopfield, J.J., Leibler, S. and Murray, A.W, From molecular to modular cell biology. Nature, vol. 42 supp, pp. 47-52. 1999.

[7] Hornby, G. Measuring, Enabling and Comparing Modularity, Regularity and Hierarchy in Evolutionary Design, GECCO, pp. 1729-1736. ACM. 2005.

[8] Huang, C.C. and Kusiak, A. Modularity in design of products and systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 28(1): 66-77, 1998.

[9] Koza, J. Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, Cambridge Massachusetts, 1994

[10] Miller, J. F and Thomson, P. Cartesian Genetic Programming, Third European Conference on Genetic Programming, Proceedings published as LNCS, Vol. 1802, pp. 121-132. Springer. 2000.

[11] Miller, J. F. Evolving a self-repairing, self-regulating, French flag organism, Proceedings of GECCO, Part I. LNCS, 3102 Springer. pp. 129-139. 2004.

[12] Schilling, M.A. Towards a general modular systems theory and its application to inter-firm product modularity. Academy of Management Review, Vol 25: 312-334. 2000.

[13] Zhan, S., Miller, J. F. and Tyrrell, A. M. An Evolutionary System using Development and Artificial Genetic Regulatory Networks, Proceedings of 9th IEEE World Congress on Computational Intelligence. Congress on Evolutionary Computation, Special Session on Evolving Gene Regulatory Networks, IEEE Press 2008. 812-822.

[14] Zhan, S., Miller, J. F and Tyrrell. A. M. A Developmental Gene Regulation Network for Constructing Circuits, International Conference on Evolvable Systems: From Biology to Hardware (ICES), LNCS Vol. 5216, pp. 177-188. Springer. 2008