Application of Self-Configurability for Autonomous, Highly-Localized Self-Regulation

N. J. Macias P. M. Athanas
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Email: {nmacias, athanas}@vt.edu

Abstract

In this paper, key features of biological vs. artificial systems are identified, along with a synopsis of important consequences of those features. An artificial substrate that contains these key features is introduced, and examples of the use of those features are given. The concept of an electronic stem cell is described, and its characteristics and subsystems are presented. Electronic stem cells offer a means of configuring a large number of autonomous computational "cells" for a given task. The electronic stem cell has properties similar to the biological counterpart, namely the abilities to self-replicate and to differentiate. This paper presents a design for a reconfigurable building block that can function similar to a biological stem cell, but in the domain of electronic circuitry. The goal is for a single electronic stem cell to, in effect, "grow" into an organism, or in this case, into a desired electronic circuit.

1. Introduction

As demonstrated commonly in nature, two keys to efficient operation in complex systems are autonomy and locality of control. Autonomy refers to the the ability of the system to operate without the need for external intervention beyond an initial startup or *incubation* period. Locality of control means that subsystems within the overall system are able to manage their routine operations without excessive dependence upon a centralized control system: in a sense, they exhibit their own autonomy relative to the rest of the system.

Autonomous systems are initially told what to do (via their initial design), and are then left on their own to do so. Once left on their own, they must handle whatever circumstances arise, reacting as they have been told, or as they have learned to do. Such systems can play a vital role in systems designed to operate in remote or hazardous environments, such as deep space or search and rescue operations [1]. Note that autonomy is generally the rule for biological systems, which operate naturally in a self-contained way, attempting to adapt to unexpected circumstances without external intervention or guidance [2]. Artificial systems, in contrast, are the extreme on non-autonomous operation: if an unexpected situation is encountered, they still continue to behave exactly as they were originally designed to. Autonomous systems tend to be more resilient and more versatile than externally-controlled systems.

Locally-controlled systems have within them a multitude of controllers for various subsystems. These local controllers are able to operate without excessive (or any) direction from less-local/centralized controllers. Here again, living systems lie at one end of the locally-controlled spectrum: within a complex biological system, centralized control (i.e., a central nervous system) may govern large-scale behavior, but individual cells operate largely on their own, performing their cell functions under local control [3].

In contrast, locality of control is the antithesis of the methodology exhibited by systems based on a Central Processing Unit (CPU). In a traditional CPU-based system, all behavior is governed by a single, centralized unit, responsible for managing every aspect of the system's operation. While some CPU-based systems may employ local control for subsystems such as video processing, network connections and mass storage components, this represents a degree of localized control only to a first order: each of these subsystems still rely upon their own centralized controller. Locally-controlled systems are generally able to operate faster and more robustly, and can be more adaptive to change, than centrally-controlled systems. While CPU systems are known to scale fairly well to thousands of nodes, and in some highly constrained situation to tens of thousands of nodes, issues of centralized control may limit scalability beyond this point.

It is thus interesting to consider whether the autonomy and locality-of-control found in biological systems can be applied to artificial, electronic systems, to improve the resiliency and flexibility of these artificial systems. Doing so requires a fundamentally different approach to the design and implementation of electronic systems: systems need to be highly-parallel in their operation, with an underlying structure that can be modified while the system is operating. This suggests the use of reconfigurable logic, such as FPGAs [4], as the building blocks for such systems. However, traditional FPGAs are not well suited to this: most FPGAs are configured by initially loading a configuration bitstream from outside the device, *FPGAs are inherently non-autonomous, non-locally-controlled devices.* The implementation of autonomous, locally-controlled systems requires an underlying hardware architecture that is itself autonomous and locally-controlled.

Section 2 presents a concise overview of the Cell Matrix architecture, a substrate that has the salient features of autonomy and local control. Section 3 will discuss the design of a general autonomous, locally-controlled system, built upon the idea of E*lectronic Stem Cells* (ESCs). Section 4 will conclude with remarks on future work.

2. The Cell Matrix architecture: an autonomous, locally-controlled substrate

The Cell Matrix [5] is a fine-grained reconfigurable logic device which is designed around the concept of *self-configuration*. Self-configuration means that the device is able to configure and re-configure itself, with a minimum amount of external intervention. This is in contrast to other reconfigurable devices such as FPGAs, which typically require an external system for the complex design flow associated with generation of a configuration string, which is then loaded into the FPGA from outside the FPGA itself [4].

The Cell Matrix achieves self-configuration via the following characteristics:

- the matrix is composed of a number of low-level logic blocks called “cells”;
- interactions between cells are limited to the continuous exchange of two pairs of bits (called “C” and “D”) between any cells considered “neighbors,” according to a pre-defined, system-wide cell topology;
- each cell has local configuration information (an internally-stored “truth table”), which indicates how a cell's outputs are set in response to the D inputs it receives from its neighbors;
- each cell has the ability to read and write the configuration information of any of its neighboring cells; and
- the only non-local signals are a single system-wide clock (and the power supply).

This means that configuration is necessarily a local-operation, since there are only a small number of cells that can configure any given cell (initial configuration from outside the system can only be performed on edge cells, which are missing some neighbors in the cell topology). Figure 1 shows a two-dimensional Cell Matrix composed of four-sided cells.

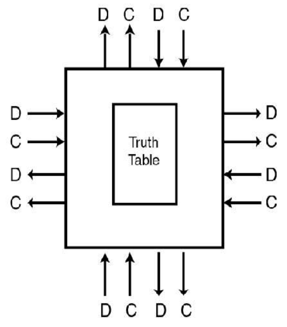

Fig. 1 Four-Sided Cell Matrix Cell
C inputs determine the cell's mode (C mode if any C input=1, otherwise D mode). In D mode, all outputs are derived from Truth Table contents and D inputs. In C mode, D outputs display old Truth Table contents, and D inputs supply new Truth Table contents.

At any given time, each cell is operating in one of two *modes*: *D-Mode*, in which the cell is mapping inputs to outputs based on its internal truth table configuration; and *C-Mode*, in which the cell's internal truth table is being read and written by one of more of the cell's neighbors. A cell's mode is determined by certain of its neighbors' outputs (the “C” outputs), and those outputs are determined, of course, by those neighbors' own truth tables and “D” inputs. If any of a cell's C inputs are set to 1, then the cell is in C-Mode, and its truth table can be read and written via its D outputs and inputs; otherwise, the cell is in D-Mode, and is mapping D inputs to C and D outputs based on its truth table contents.

The situation inside the matrix is thus one where any cell can configure, or be configured, by any of its neighbors at any time. This makes it possible, in effect, to build circuitry that processes *circuit configurations* as well as standard Boolean data. Figure 2 shows an example arrangement of cells: in this example, Cell X is asserting its northern output, which causes Cell Y* to read Cell Y's configuration information. Cell Y* passes that information to Cell X, which then sends it to Cells Z1*-Z4*, which are acting in parallel to create four copies of Cell Y in Cells Z1-Z4 (also in response to Cell X's northern output). Note that Cell Y* is also re-configuring Cell Y with Cell Y's original truth table, which preserves Cell Y. Reference [5] describes further details of cell behavior.

The elegance of this cell architecture is its simplicity, making it well suited for tiling in very large two-dimensional, or even three-dimensional arrays.

3. Electronic stem cells

The present work seeks to define a reconfigurable building block that can function similarly to a biological stem cell [6], but in the domain of electronic circuitry [7,20]. The goal is for a single Electronic Stem Cell

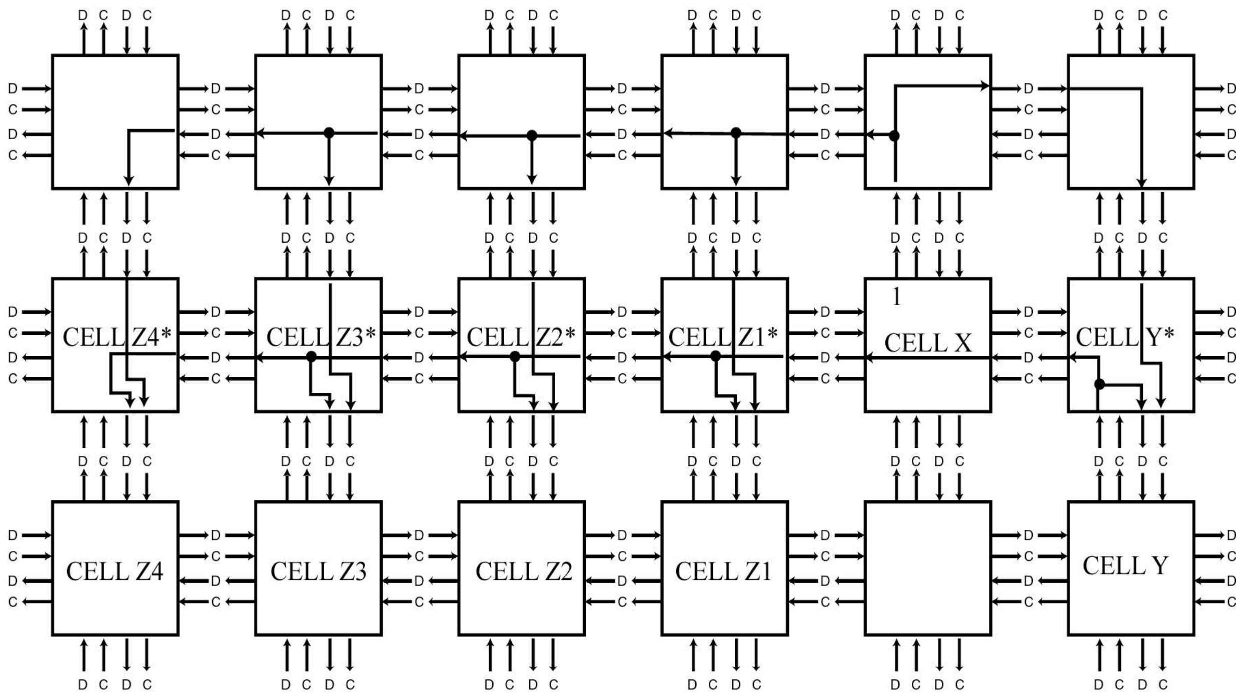

Fig. 2 A collection of cells operating in Parallel.
Cell X asserts a 1 on its northern output, which cell Y* copies to cell Y's C input, thus placing cell Y into C mode. Cell Y* reads cell Y's truth table, and immediately re-writes it, thus performing a non-destructive read of cell Y. Cells Z1*-Z4* transfer their northern D input to their southern C output, thus placing cells Z1-Z4 into C mode. Cells Z1*-Z4* also receive a copy of cell Y's truth table, and use it to configure cells Z1-Z4 to be copies of cell Y. This circuit thus makes four copies of cell Y. The copies are created in parallel, in the same time that would be required to make a single copy.

(ESC) to, in effect, “grow” into an organism - in this case, some desired electronic circuit. This block needs to perform the following three functions:

1. Self-Replication: beginning with a single artificial stem cell, it should make multiple copies of itself, populating a region of an initially-empty Cell Matrix with its copies;
2. Differentiation: the artificial stem cells should then differentiate in the components of a desired high-level circuit; and
3. Interconnection: these components should build connections among themselves, thereby wiring together the components and implementing the target circuit

Moreover, these steps should be performed in as much of an autonomous, locally-controlled way as possible. For example, cell replication should be governed by the self-replicating cells themselves, rather than some external system that monitors and directs their activity. Similarly, detection- and avoidance-of faults during the building of the final circuit should be handled locally, as much as possible.

The operation of the system of electronic stem cells can thus be broken into the following three steps: self-replication; differentiation; and interconnection.

Self-replication begins with the creation of a single ESC. This ESC will check for signals from neighboring ESCs, but, being the only ESC present, will determine that it is alone in the matrix, and will then begin the processing of self-replicating into multiple copies. Of course, each of these copies will appear identical to the original ESC, but upon inspection of its surroundings, will discover that it is not the initial ESC. This observation will modify the behavior of subsequently-generated ESCs. Alternatively, rather than deducing their position within the collection of ESCs, each ESC can be synthesized with a unique internal ID.

After a certain number of ESCs have been generated (or, perhaps, a certain-sized area has been populated with ESCs), one or more ESCs will broadcast a signal indicating initiation of the differentiation phase. Since differentiation occurs essentially in parallel, no arbitration is necessary among the ESCs during this step.

Since each ESC contains a genome for the final circuits, each ESC can deduce how long the differentiation step requires to complete. Following an appropriate delay, the ESCs begin their final step: interconnection. Interconnection may be performed sequentially, in which case the ESCs arbitrate an ordering among themselves; or, depending on the interconnection strategy, some degree of parallelism may be available.

In general, under this strategy, there should be no contention during the self-replication, differentiation and interconnection processes. However, it is interesting to consider the possibilities if *two* initial ESCs were placed on the matrix, each thinking they were alone (due to the lack of immediately-neighboring ESCs). These ESCs would each begin their own process of synthesizing a complete final circuit, with these circuits perhaps eventually colliding with each other.

3.1 Self-Replication

Synthesis of the target circuit (or ”organism”) begins with the configuration of a single ESC inside the matrix. This can be performed using a standard bootstrap algorithm, which is necessarily initiated from outside the matrix (assuming the matrix is initially “empty,” i.e., all of the cells are configured to outputs 0s regardless of their inputs). However, once this single ESC has been configured, additional ESCs are created from within the matrix itself, via self-replication processes. This proceeds in two phases: in the first phase, a single row of ESCs is created, one ESC at a time; in the second phase, multiple columns of ESCs are synthesized in parallel.

3.1.1 Single-ESC replication. Each ESC contains within itself machinery for producing an identical copy of itself. Figure 3 shows the basic self-replication algorithm, which consists of two main parts. The first part, called the “Main Grid,” is a state machine for building three wires (one into a region of itself, and two into regions of a future ESC), reading cell configurations from the end of one wire, writing those cell configurations at the end of the other two wires, and sweeping the location of the end of all three wires in order to read and write two-dimensional regions of the matrix. The second part, called the “Exploded Grid,” is an exact copy of the Main Grid, except that there is extra space between each row of the Main Grid. The Exploded Grid is essentially a “data” version of the state machine that is the Main Grid. It is a non-operating copy, suitable for reading by the state machine (Main Grid). The Main Grid makes two copies of what it reads: one will be a new Main Grid, the other a new Exploded Grid, and these two copies will comprise the new ESC. This is analogous to the process of Translation and Transcription associated with replication of biological cells.

3.1.2 Multi-ESC replication. Once a single ESC has been configured, populating the matrix with a number of ESCs proceeds in two steps. The first step builds an entire row of ESCs across a region of the matrix. To do this, the initial ESC makes a copy of itself to the east. That copy then makes a copy if itself, further to the east, after which that copy makes a copy of *itself*, and so on. Within each ESC (remember, each ESC is identical to every other ESC) is the pre-coded specification of the target circuit's size, in terms of the number of ESCs required (X and Y dimensions, i.e., width and height). Each ESC receives a *column number* from its western neighbor. The initial ESC, located on the western edge of the matrix, has no western neighbor, and thus receives a “0” on its unconnected western inputs. Each ESC increments its incoming western column number, and thus computes its own position within the matrix (which is the same as the number of ESCs to its west). Since each ESC knows how wide the collection of ESCs should be, the easternmost ESC can identify itself as such, and decline to self-replicate. This limits the number of ESCs in the top row.

In the second step of Multi-Cell Replication, the easternmost ESC sends a signal back to all ESCs to the west (each ESC receives the signal from the east, and relays it to the ESC immediately on its west). This signal causes each ESC to make a copy of itself to the south, immediately below itself. Again, as in the first step, each ESC makes a copy, and each copy makes a subsequent copy, and so on. And again, each ESC receives a *row number* from the ESC above itself, and thereby determines the length of the row it occupies. This is used by the southernmost ESC to determine when to not replicate. This causes the collection of ESCs to be limited to a pre-defined width and height.

3.1.3 Advantages. The above approach to tiling the matrix with ESCs has a number of advantages over a more-traditional *externally-directed* configuration approach:

- external intervention is required for only the configuration of an initial ESC, a relatively small object;
- the configuration is extremely fast: an *nxn* collection of ESCs would requires only *2n-1* time steps (a single time step being the time required to configure a single ESC). For a three-dimensional matrix, configuring an *nxnxn* collection requires roughly 3n time steps. Put another way, configuring 10^{24} ESCs in a three-dimensional matrix would require roughly 300 million time steps, vs. a trillion trillion time steps for sequential configuration of each ESC. Assuming a hypothetical time step of one millisecond, this amounts to an *incubation time* of 83 hours, or approximately 3-4 days, compared with ***31 trillion***

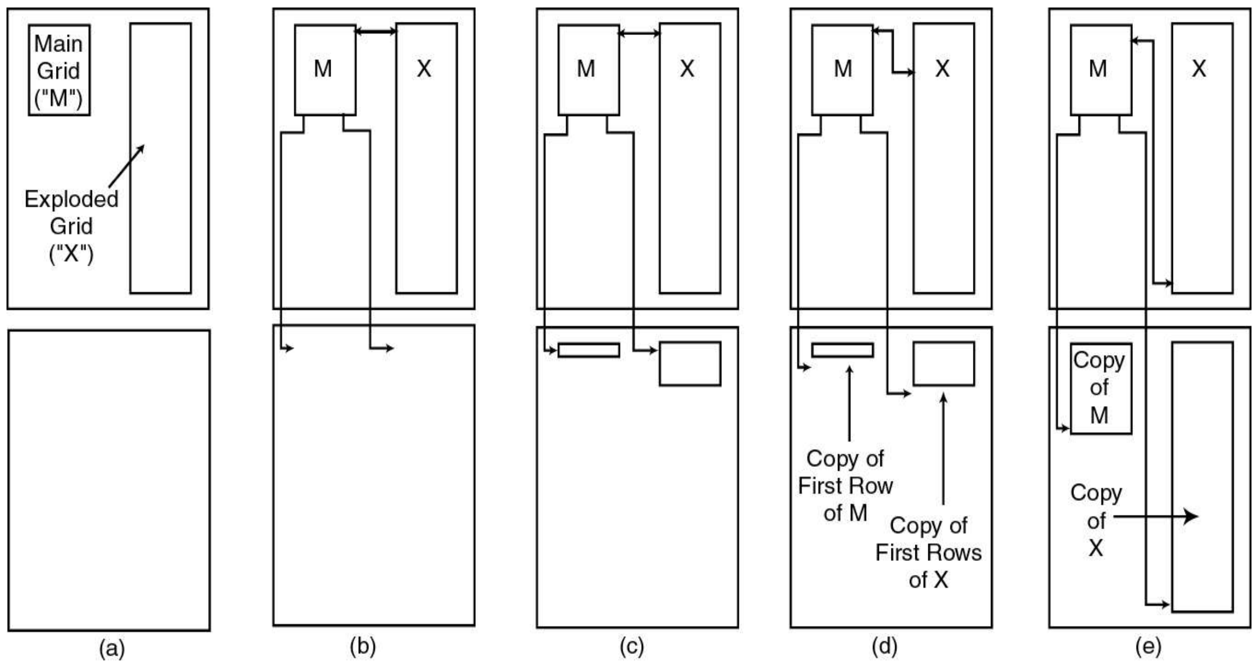

Figure 3 - Self-Replicating Circuit. In 3(a), the upper rectangle contains a state machine (called the Main Grid "M") as well as a copy of the state macnine (called the Exploded Grid X). The Exploded Grid includes rows of unconfigured cells between each row of M. In 3(b), the Main Grid has built three wires: one for reading cells from X, and two for making copies of the cells being read. In 3(c), the first row of M has been read from X, and used to make copies of the top row(s) of M and X. In 3(d), all three wires have been extended to the South, in preparation for reading the next row of M from X. In 3(e), all of M has been read from X, resulting in the creation of new copies of M and X in the region to the South. The lower rectangle now contains an exact copy of the original upper rectangle, and now itself begins a new self-replication cycle.

years for sequential configuration [8];

- the configuration process can be made more *fault tolerant*, by performing pre-configuration tests of each region prior to configuring a new ESC. Of course, these tests would be performed locally, by the same ESC that it preparing to self-replicate, and thus testing is performed in parallel as well. The above observations regarding parallel- vs. sequential-operation apply here also. This can also be used to facilitate self-replication on a matrix that is not initially empty, by testing regions for pre-existing configurations prior to self-replicating. The simplest strategy for handling detected defects (or occupied regions) is simply to terminate the replication process in the given row, though this approach causes entire sections of a column to be eliminated by a single defect (and, if a defect were found in the top row, this would eliminate all remaining columns to the east). Numerous embellishments are available though to improve this fault-handling strategy [9].

3.2 Differentiation

Once the matrix has been tiled by a number of ESCs, each ESC enters a *differentiation* stage, during which it changes from being identical to every other ESC, and self-modifies into a particular piece of the final circuit. This self-modification is directed by two pieces of information: (1) the location of this ESC within the collection (which can be determined locally by using the same row/column technique described above for self-replication); and (2) consultation of an internally-stored representation (called the “genome”) of the final circuit.

For maximum flexibility, the number of possible differentiated circuit components should be large, while for efficiency, the granularity of each component should also be large. These goals, however, would both contribute to an extremely large ESC, **if** each ESC were required to contain inside itself *every possible* differentiated circuit. Here again, the self-configurability of the Cell Matrix is key, because it allows the ESC to *dynamically generate* its target circuit component, using a library of sub-components and a map of how to assemble them into a complete component.

There are several possible approaches to the dynamic generation of a routing solution, each making different tradeoffs between size and flexibility. One approach is to store, within each ESC, a library of basic building blocks (logic gates, multiplexers, etc.) and then embed within the genome the instructions for assembling those blocks into each ESC's desired component. A second approach would store within each ESC both a library of basic building blocks *and* instructions for assembling those blocks into components. The genome would then only need to specify which component is desired. In cases where a small number of components are used repeatedly, this would be more space-efficient than the first approach.

A third approach, combining the above two, could place a library of blocks inside each ESC, embed instructions for building **each** component in the beginning of the genome, and then have the genome refer to specific components for each ESC. This requires more space than the second approach, since the component building instructions are repeated in each copy of the genome (one per ESC), but it also more flexible, since the choice of possible components can be changed through simple genome modification vs. re-architecting the ESC.

Reference [10] uses the simplest form of the first approach, wherein each ESC contains a small library of single Cell Matrix-cell building blocks, and the genome specifies which of those blocks should be assembled into the ESC's differentiated component.

3.3 Interconnection of differentiated components

Once the ESCs have differentiated into their target components (i.e., have synthesized within themselves the component they are responsible for implementing in the final circuit), these components must be connected to each other. Note that, because the underlying hardware may contain defects, it is difficult to predict the exact location of each ESC prior to completion of the self-replication and differentiation steps. Therefore, it is generally not feasible to pre-compute a routing solution and simply store that inside the genome. Instead, the routing must be determined "on the fly," once the ESCs have differentiated. Reference [10] describes a straightforward, sequential, greedy algorithm for doing this, by storing an abstract wiring netlist inside the genome, and having each component dynamically locate any other components to which it must make connections. The process of component location is performed collectively by all components. Alternatively, a parallel, distributed place-and-route algorithm [11] could be implemented by the ESCs, in order to develop a more efficient routing solution.

Once the desired inter-component routes have been determined, they must be synthesized. One approach is to use a standard multiplexer-based, hierarchical routing scheme, such as that employed by traditional FPGAs [12]. This is a fairly space-efficient solution, but is limited in the scope of routings options, and thus may increase route-computation time, as well as final circuit size. A second approach is to use a series of cross-bars [13] at the junction of all ESC paths, allowing signals to pass from any channel to any other channel. This provides maximum routing flexibility, in exchange for consuming a large amount of space (since most routing options will remain unused in the final circuit). Reference [10] describes a routing approach that incorporates, within each ESC, a series of routing channels, and then synthesizes connections from channel-to-channel. This is a compromise between the two above approaches.

Still another solution is to dynamically construct entire paths from empty Cell Matrix cells, using wire building techniques such as described in [14]. This allows the maximum amount of freedom in terms of routing choices (since all paths are constructed on the fly), while also being extremely space-efficient, since only cells needed for the wires themselves are used. Moreover, this approach allows considerable flexibility in terms of fault handling, since, as in the self-replication step, Cell Matrix cells can be tested for defects before being used in wires, with defective cells being avoided by simply routing around them. Of course, this approach requires a more-sophisticated ESC design.

4. Realization

A number of the individual subsystems described above have previously been implemented on the Cell Matrix. A fully-autonomous self-replicating structure was implemented in 1996, with a total circuit size of *332x150=49,800* cells. This circuit was simply a state machine for the production of pre-defined sequences of bits so as to build wires, read cell configurations, and replicate those configurations in other cells, thus achieving the behavior shown in Figure 3. The ciruit was designed so that the last cell configured at the end of a self-replication cycle would output a 1 into the newly-configured state machine, thereby beginning the newly-built circuit's own self-replication cycle.

Each complete cell-replication cycle required approximately 21 million clock cycles. This was, however, only a prototype design only, and could be greatly improved upon in terms of both size and efficiency.

The work described in [10], which explored various self-assembly techniques on the Cell Matrix, demonstrated a number of concepts relevant to the design of an ESC, including:

- the parallel configuration of identical circuits, by transmission of a single configuration string to multiple locations within the Cell Matrix;
- local, parallel fault testing within the Cell Matrix itself, through the creation of temporary fault-detection circuits and the distribution of a single test sequence to those multiple circuits;
- locally-controlled fault isolation, through the activation of various *guard circuits* in response to localized fault testing; and
- design of a position-independent circuit description (genome) with subsequent *on-the-fly* translation into a complete circuit layout using a distributed routing algorithm implemented inside the Cell Matrix itself.

While [10] utilized a simple greedy placement and routing algorithm, the work performed under [15] implemented a more-sophisticated place-and-route algorithm, based on simulated annealing [18] and the Pathfinder placement algorithm [19]. While [15] employs global knowledge, which is not suitable for an ESC routing system, it does develop a number of routing concepts relevant to an ESC router, including:

- routing on top of an architecture with an underlying nearest-neighbor topology;
- routing in a system that utilizes extremely fine-grained blocks, i.e., blocks that function as individual gates or other simple logic blocks; and
- routing in a system where low-level blocks operate interchangeably as processing elements and as routing elements.

Hardware implementation of the Cell Matrix has been limited to small initial cMOS prototypes, as well as implementations on top of commercial FPGAs [16]. It is hoped that Cell Matrices with a large number of cells may be implementable using emerging fabrication techniques, including such techniques as assembly of carbon nanotubes on top of DNA scaffolding [17].

5. Future work and conclusions

Current plans for ESC development involve both enhancement of existing work as well as development of new algorithms and structures. Specific plans are:

- development of a more space-efficient self-replicating circuit, employing sequence compression and a general-purpose sequence generation system;
- addition of a payload to a self-replicating circuit, so that additional circuitry can be self-replicated;
- addition of support circuitry to control the number of replication cycles;
- extension of the parallel configuration algorithms in [10] to achieve parallel self-replication;
- development of an appropriate component library and library storage system;
- incorporation of distributed fault-detection techniques into the ESC-based system; and
- development of a *repairable routing solution*, where information about routing resources are stored locally, so that, in the event of a component failure, the routing solution can be **modified** vs. re-compute from scratch.

The Electronic Stem Cell approach seems to be both feasible and ideal for implementation of robust, efficient, general-purpose circuitry inside a Cell Matrix self-configurable substrate. Moreover, this approach represents an important step in the movement towards autonomous, locally-controlled systems, which are inevitable as system complexity continues to scale up.

References

[1] A. Birk and S. Carpin, "Rescue Robotics - a crucial milestone on the road to autonomous systems," *Advanced Robotics Journal*, 20 (5), VSP International Science Publishers, 2006.

[2] K. Ruiz-Mirazo, J. Peretó, and A. Moreno, "A universal definition of life: Autonomy and open-ended evolution," *Origins of Life and Evolution of the Biosphere*, 34, pages 323-346, 2004.

[3] J.D. Simnett and J.M. Fisher, "Cell division and tissue repair following localized damage to the mammalian lung," *Journal of Morphology* 148(2), pages 177-184, Feb 1976.

[4] W. Wolf, *FPGA-Based System Design*, Prentice Hall Modern Semiconductor Design Series, 2004.

[5] N. Macias, "The PIG paradigm: the design and use of a massively parallel fine grained self-reconfigurable infinitely scalable architecture," *Proc. The First NASA/DOD Workshop on Evolvable Hardware,* A. Stoica, D. Keymeulen and J. Lohn, Eds. pages 175-180, 1999.

[6] J.A. Thomson, J. Itskovitz-Eldor, S.S. Shapiro, M.A. Waknitz, J.J. Swiergiel, V.S. Marshall, and J.M. Jones "Embryonic stem cell lines derived from human blastocysts," *Science* 282, pages 1145-1147, 1998.

[7] L. Prodan, G. Tempesti, D. Mange and A. Stauffer, "Embryonics: Electronic Stem Cells," *Artificial Life VIII*, Standish, Abbass, Bedau, Eds. MIT Press, pages 101-105, 2002.

[8] L. Durbeck and N. Macias, "The Cell Matrix: an architecture for nanocomputing," *Nanotechnology* vol 12, pages 217-230, Bristol, Philadelphia: Institute of Physics Publishing, 2001.

[9] Cell Matrix Corporation, "Autonomous self-repairing circuits," NASA SBIR Phase II proposal number 00-II 01.01-8587, 2001.

[10] N. Macias and L. Durbeck, "Self-assembling circuits with autonomous fault handling," *Proc. The 2002 NASA/DOD Conference on Evolvable Hardware,* A. Stoica, J. Lohn, R. Katz, D. Keymeulen and R.S. Zebulum, Eds. pages 46-55, 2002.

[11] R.J. Brouwer, *Parallel algorithms for placement and routing in VLSI design*, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1992.

[12] S. Brown, M. Khellah and G. Lemieux, "Segmented Routing for Speed-Performance and Routability in Field-

Programmable Gate Arrays," *Journal of VLSI Design*, Vol. 4, No. 4, pages 275 – 291, 1996.

[13] M. Khalid and J. Rose, "A hybrid complete-graph partial crossbar routing architecture for multi-FPGA systems," *Proc. ACM Symposium on FPGAs*, pages 45-54, Feb 1998.

[14] N. Macias, *US Patent #6,297,667*, 2001.

[15] N. Macias and L. Durbeck, *Final Report to Los Alamos National Laboratory, Subcontract #90843-001-04*, 2006.

[16] Cell Matrix Corporation, "User manual and tutorials for the Cell Matrix MOD 88 (TM)," http://www.cellmatrix.com/entryway/products/mod88/mod88.pdf, 2005.

[17] C. Dwyer, S.H. Park, T. LaBean and A. Lebeck, "The design and fabrication of a fully addressable 8-tile DNA lattice," *Proc. of the 2nd Conf. on the Foundations of Nano.*, pages 187-191, April 2005.

[18] A. Sharma, "Development of a Place and Route Tool for the
RaPiD Architecture," *Master's Project*, *University of Washington*, December 2001.

[19] L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs," in *International Symposium on Field Programmable Gate Arrays*, Monterey, Ca., Feb. 1995.

[20] Logic Systems Laboratory, "The Embryonics Project," http://lslwww.epfl.ch/pages/embryonics/ Swiss Federal Institute of Technology, accessed April 2007.