Applying an Evolvable Hardware Approach to Autonomous Robot Design

By
Jason Plew, A. A. Arroyo, E. M. Schwartz
Jason@mil.ufl.edu

Machine Intelligence Laboratory
Department of Computer and Electrical Engineering
Gainesville, FL 32611

Abstract

Microprocessors have long been the central controller for many autonomous robots. However, as a processor is a general-purpose device, it is possible that if a control system for a robot were designed specifically for a certain task, improved performance would result. However, such an approach would be impractical for anything but the most simplistic goals, assuming that a human were the designer. One possible solution is to approach the problem using Evolvable Hardware (EHW). This method uses Genetic Algorithms to apply evolutionary forces to the development process. In this research, the EHW approach will be applied to design a control system for a robot that has increasingly complex tasks to perform. This design process will then be compared with control systems developed by a human designer, to determine whether the EHW approach can benefit robot designs.

1. Introduction

In the past 50 years we have seen a transition in the control of autonomous robots. Initially, microprocessors were too expensive to consider using them for anything but the most essential processing requirements. Instead, designers did as much as possible outside of the device, including reading sensor data and determining actuator positions. As processors became cheaper, they also became more powerful, and as a result most necessary functions for controlling robots were moved inside the microprocessor. However, since the processor is a multi-purpose device, it is possible that improved performance can be obtained by designing a control system for a specific task.

Unfortunately, the amount of time involved in developing such a system can be rather time-consuming, making it impractical for most applications. However, if the design process could be automated, then this approach would have potential. Several different methods are currently being researched in an attempt to develop such a process, with neural networks being the most popular. This technology is currently being researched at MIL, as are techniques to have a robot determine for itself how to operate through a design architecture called Environmental Reinforcement Learning (ERL) [1, 2].

Another method would be to use nature as a guide, and allow evolutionary forces to produce a design that fulfills a goal as efficiently as possible. Such an approach is

being researched in a field that has been termed Evolvable Hardware (EHW). This involves applying evolutionary processes, in the form of genetic algorithms, to develop circuit designs, and then evaluating the resulting systems. The best approaches are then used as a basis for a new set of designs [3].

There are two methods of determining the efficiency of the design. The first, intrinsic evaluation, would involve simulating the design, from which the output characteristics can be obtained. While this is a valid approach for developing some applications through EHW, the complexity of simulating the environment of an autonomous mobile robot makes this approach extremely difficult.

The other method of determining the effectiveness of a design is extrinsic evaluation, or actually building the device [4]. In this case, it means transferring the resulting designs to a robot, and then observing its behavior. Extrinsic evaluation has been chosen to evaluate the designs that will be produced during this research.

The researcher will attempt to determine the effectiveness of the EHW approach to robot design by developing a means where a description of a robot control system can be modified by a reproduction process, which will use a genetic algorithm to combine elements of the best designs as well as make slight modifications to the results.

The new designs will then be applied to a robot, which is controlled by a Programmable Logic Device (PLD) instead of a microprocessor. Once this is accomplished, an analysis of the design will begin. The first phase will consist of an attempt to develop a robot through evolution that can perform a simple task, but do it very well. This will allow us to evaluate the reproduction process. These designs will also be compared to a similar robot whose control system has been developed in the manner of a microcontroller.

Once the above experimentation is complete, we will move on to applying the EHW process to actual robot designs. The first phase of this process will be developing robots for a predator / prey model. Future applications will follow.

2. Evolutionary Design Process

Those working in the field of evolvable hardware have taken to using terminology from biology to describe comparable methods in designing digital and analog circuits. This "reflects the conceptual similarity between genetic algorithms, natural evolution, and genetics" [4]. This practice will therefore also be used in describing the evolutionary process developed for this research.

We will start with a set of 10 empty designs. This will comprise the 0^{th} Generation. Subsequent Generations will be developed using the reproduction process described below. Two of the designs will be chosen, and then submitted to a program for reproduction. This program will combine the two designs, or "parents", in a manner such that two new designs, the "children", are created with characteristics of both parents. Furthermore, mutations will be introduced by having each child be replaced by 5 copies that are then slightly modified in random ways. In this manner the new generation will also have 10 members, and the reproduction process can begin again after the designs are evaluated.

2.1 Design Layout

The development of a robot design will be based around an array of circuit blocks, called "cells". At first the array will be a 10 x 10

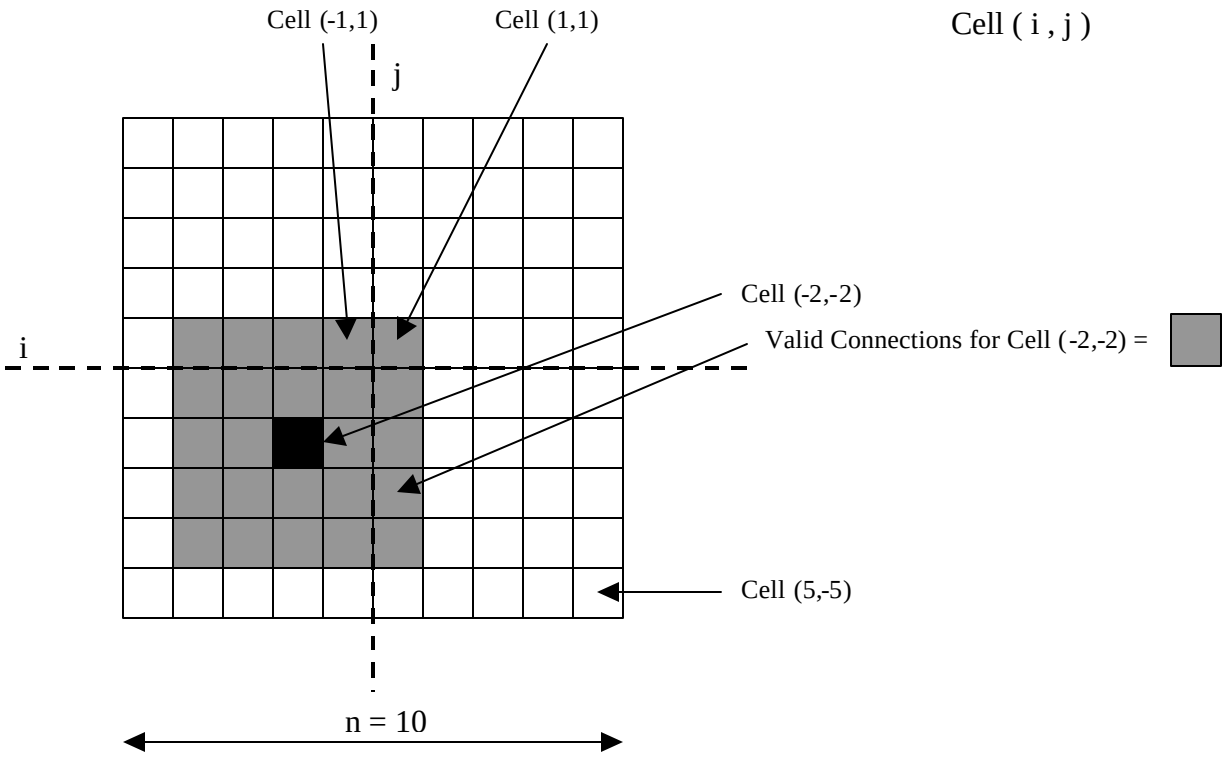

Figure 1: Cell Array

matrix (See Figure 1), though this size will be increased as the design grows more complex. Each cell will initially be set to provide an output of "0". Furthermore, each cell will be described by a sequence of binary data, referred to as a "gene", that describes its functionality, location in the cell array, and its connections to other cells. To simplify the reproduction processes, all genes will be the same size. An initial format of a gene is shown in Figure 2.

Specific Functions are listed in Table 1. The Function Identification Block in the gene will include both this information, as well as other necessary data particular to the specific function. Input Connections that are unused for a specific function will be set to null. Otherwise, they will contain the column and row location in the array of the cell whose output is connected to the specific input in question. Any device with an output is a valid option, but the cell in question must be within a certain distance from the cell this gene describes. This practice is used to maintain some control over the complexity of the design, and to simplify the reproduction process. The formula used will be dependent on the size of the array, n. For an array of size n = 10, which is the size of the initial cell array, only outputs that are at most two cells away will be allowed (See Figure 1).

The functions available to the cells include

Cell Column Location	Cell Row Location	Function Identification	Input 0 Connection	Input 1 Connection

Figure 2: Genome Description

Device	# of Inputs	# of Outputs	Comments
Logic LOW	0	1	Provides Binary 0 to local devices
Logic HIGH	0	1	Provides Binary 1 to local devices
Global Input Buffer	0	1	Provides a Global Input connection to local devices. Gene contains identity of specific device.
Global Output Buffer	1	1	Provides a Connection for Local Devices to a Global Output. Gene contains identity of specific device. The Buffers are connected to the actual Global Output through OR Gates. The output of the buffer in the cell array is the current state of the Global Output.
Inverter	1	1	
D Flip Flop	1	1	No Set/Reset Functionality. Global System Clock automatically provided.
AND Gate	2	1	
OR Gate	2	1	
XOR Gate	2	1	

Table 1: Initial Functions Available to the Cells

the standard logical primitives - an inverter, AND, OR, and XOR Gates, as well as nodes providing the binary values '1' and '0'. D Flip Flops are also available and are automatically provided with a Global System Clock. The Flip Flops contain no Set or Reset circuitry – any such functionality will have to be developed in other parts of the design.

It remains to be seen whether a design with the complexity to control a robot will develop within a reasonable number of generations, when limited to just the fundamental building blocks of Digital Logic. This will be decided during Phase 1. For example, any design will have to be able to track time using counters. It is probable that at some point in the reproduction process a counter will develop, but it is uncertain how long this might take. If after several generations, there appears to be no such improvement, then the design process will be adjusted to make it more likely that such complexity appears sooner rather then later. One possibility is allowing a single cell to contain more complex circuits, such as Counters, Arithmetic Devices, more complex Flip Flops, or just random combinations of the logic gates. Another, more preferable, possibility would be to have the reproduction program select a random grouping of cells, and then develop a new gene that describes that circuit in a single cell. This new function would then be available in future reproduction.

Extraneous to the cell array are the connections to the outside world. Initially, all sensors will provide digital inputs, though there will be no distinction as to what the values mean. Any necessary analog to digital conversion will take place outside of the cell array, and therefore, outside the scope of the evolved design. These devices will connect to logic circuits in the array through the Global Input Buffers.

Similarly, Global Output Buffers will connect devices in the Cell Array to the outside world. All Global Output Buffers for a specific output will enter an OR Gate that is outside of the Cell Array. The outputs will initially be motors and IR emitters, where a high voltage drives the device. Timing control for the IR Emitters will be provided outside of the cell array. However, any motor speed or direction control will have to be developed through evolution. Inside the cell array, an output of a Global Output Buffer is equal to the current

Generation ID	Parent A ID	Parent B ID	Child ID	Size of Array (n)	# of Inputs	# of Outputs	Gene (-5, 5)	Gene (-4, 5)
…	Gene (1, 5)	…	Gene (5, 1)	Gene (-5, -1)	Gene (-4, -1)	…		Gene (-5, 5)

Figure 3: Chromosome Layout

state of the Global Output itself, not the output of the Buffer. This will allow the current state of the output to control future decisions.

A specific number will identify each input and output, and the number of inputs and outputs will be included at the beginning of a string of binary data that describes the overall design. Following the tendency to parallel Biological Processes, this will be called a "chromosome". At the beginning will be information on the identity of the specific design. After the input and output data will be the genes for the different cells in the design, organized by row and then column. An initial chromosome configuration is in Figure 3.

2.2 Reproduction Process – Crossover

The first step in producing a new generation is to select the two best designs in the preceding generation to use as parents. The specifics of selecting these designs will be discussed on the sections describing the particular experiments. Once the two parents have been selected, their chromosomes will be split, and then recombined, to produce the two children. This is a genetic operator referred to as crossover. [3]. Specifically, the designs' cell arrays are divided into 4 equal partitions, and these partitions are then combined to produce the offspring (Figure 4).

As the reproduction process splits the design apart, there will be cells along the boundary of the partitioning whose original inputs are from cells that have been replaced. Since for the initial design process, all cells have a single output node, there should be no broken connections as a result. In the event that devices are used with more then 1 output, or none at all, the program will look for any errors produced by the reproduction process, and randomly select a new cell that is valid for the specified connection.

2.3 Reproduction Process – Mutation

Once the two children have been developed, another program will take each new chromosome and produce five copies that are then slightly modified by mutation. The mutation process randomly selects some cells, and makes changes to them, including modifying their connections or their function. The randomization process is based on a principle of costs. Only so many changes may be made to a design. More dramatic changes to a cell are more expensive, and this limits the amount of change that occurs from one generation to the next.

Parent A	
A_1	A_2
A_3	A_4

+

Parent B	
B_1	B_2
B_3	B_4

=

Child A	
A_1	B_2
B_3	A_4

,

Child B	
B_1	A_2
A_3	B_4

Figure 4: Crossover Process

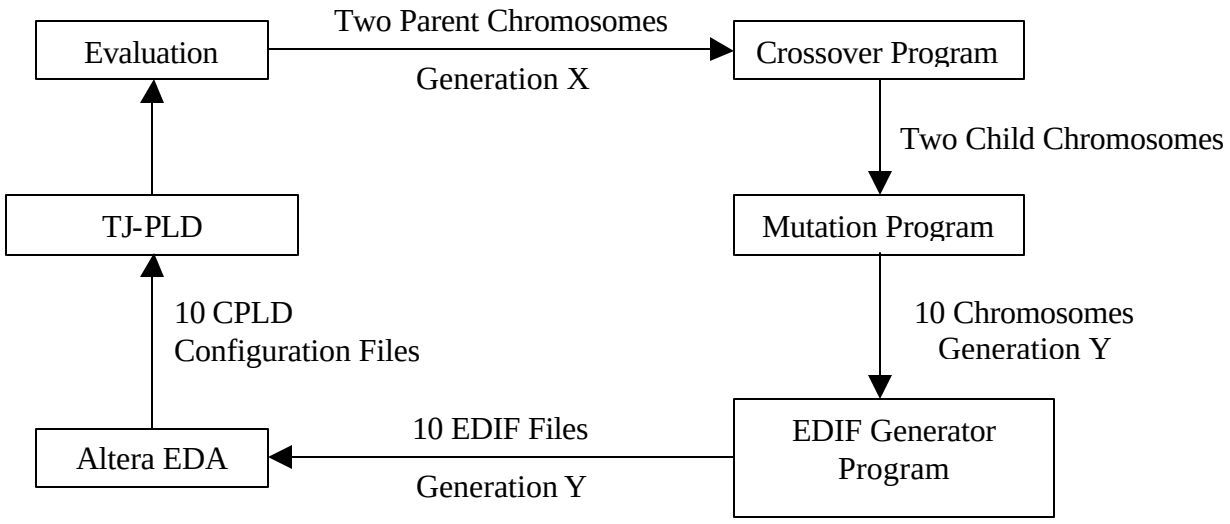

Figure 5: Design Process

Once the Reproduction Process is complete, we will have a new generation, containing 10 similar designs that are ready to be evaluated. The designs can then be downloaded into the robotic agent for experimentation. See Figure 5 for an overview of the entire reproduction process.

3. Robot Agent Used

The robot used for this research will be referred to as a TJ-PLU (Talrik Junior – Programmable Logic Unit). This robot will be based on a TJ-Pro, which is a standard robot platform that was designed at MIL by Scott Jantz. The TJ-Pro provides both Contact Switches and IR Sensors. In the original design, these devices are controlled by a MIL designed microcomputer, the MTJPRO11, which utilizes the MC68HC11 microcontroller [5].

The TJ-PLU will be controlled by a CPLD from the Altera FLEX10K family, rather then an HC11 microcontroller. The board will emulate the functionality of the MTJPRO11 by including an A/D subsystem that can be used to translate the analog sensor data from the IR detectors, and deliver it to the CPLD.
The design developed through the reproduction process is described, as explained above, by a string of binary data, called a chromosome. This information will be converted to an EDIF file, which will then be imported to the Altera EDA package. It will then be downloaded either directly to the CPLD through a JTAG interface, or into a serial configuration PROM. The latter device will be included on the board so that the CPLD may be initialized on power-up.

4. Experimentation

Once the CPLD has been programmed, the evolutionary design may be evaluated. Determining which designs are suitable for reproduction will depend on the design goal. For the moment, two different applications have been developed.

4.1 Phase 1 –Design Process Evaluation

Phase 1 is essentially an evaluation of the Design Process developed above. A maze will be constructed, and two different types of

TJ-PLUs will attempt to escape the maze. The first design will be our control robot – a TJ-PLU whose design emulates a TJ-Pro programmed with avoidance software. The second TJ-PLU will contain designs that are a result of our EHW approach. Hopefully, later generations will show improvement over their predecessors to the point were the EHW designs are tailored to the specific layout of the maze. Once this point is reached, the TJ-PLU that utilizes EHW should show superior performance compared to the control robot.

The evaluation of an EHW design in this phase will consist of a formula that weights different factors of the design's performance. These include the time it takes for the robot to exit the maze (or whether it successfully exits at all), how many times it bumps into the sides of the maze, and also the size of the design.

As mentioned before, this phase will also involve the fine-tuning of the evolutionary process. If improvement is not apparent as subsequent generations are generated, the process will be modified as necessary.

4.2 Phase 2 – Predator / Prey

By this point, the development of Evolutionary Design Process should be complete. We will then be able to accurately determine whether the EHW approach is applicable to autonomous robot design. This phase will evaluate its affects on robots emulating the predator-prey relationship.

Two variations of the TJ-PLU will be used. One will consist of a regular TJ body that has been painted black, as this will drastically reduce the ability of IR sensors to detect the robot. For obstacle avoidance, it will instead use sonar, along with the contact sensors. It will not utilize IR emitters, but will have IR detectors available, which will enable it to detect the prey robots.

The prey robots will be nearly identical to the original TJ-PLUs used in Phase 1. The difference is that the bumper around the robot used with contact sensors will also have a wire wrapped around it. A positive voltage on this wire will "kill" the prey robot. Likewise, the predator robot will also have a wire pulled high around its bumper. The result is that if a predator robot bumps into a prey robot, the prey will be shut off.

Each generation will consist of 10 predator and 10 prey robots. For the actual experimentation, each predator will face off against five prey robots at a time. The two predator robots that "kill" four prey robots the quickest will be chosen for the reproduction process. Likewise, the two prey robots that survive the longest will also be chosen. Eventually, there will come a point when control robots will be used as well. These will consist of TJ-PLU predator and prey models whose designs are emulations of TJ-Pro software that would perform the same function. In the process, we will be able to determine whether the EHW approach can produce a predator or prey design superior to those produced in the more normal method.

5 Future Goals

Once Phase Two has been completed, other applications, involving more complex requirements, will be attempted. Furthermore, the reproduction process will not remain static. Possibilities include allowing devices to access more then one clock. Another possibility would be using Programmable Logic Devices that include more advanced features, such as PLLs, or using internal RAM to create state machines. The Reproduction process would then be modified to make use of such options. Finally, the

possibility of expanding the EHW beyond digital logic remains. While actually building such designs in discrete components would remain difficult, development of Field Programmable Analog Arrays may provide a solution to automating the entire design process of a robot control system.

Bibliography

[1] Nechyba, M. C. and Xu, Y., *"Human Control Strategy: Abstraction, Verification, and Replication"*, IEEE Control Systems Magazine, vol. 17, no. 5, pp. 48-61, 1997.

[2] Choi, TaeHoon A., Yim, Eubin A., and Doty, Keith L., *"Environmental Reinforcement Learning: A Real-time Learning Architecture for Primitive Behavior Refinement"*, Machine Intelligence Laboratory, Dept. of Electrical and Computer Engineering, University of Florida.

[3] Tomassini, M., *"Evolutionary Algorithms"*, Towards Evolutionary Hardware: The Evolutionary Engineering Approach, p. 19-47, Springer-Verlag, 1996.

[4] Hounsell, Ben I. and Arslan, T., *"A Novel Genetic Algorithm for the Automated Design of Performance Driven Digital Circuits"*, Proceedings of the 2000 Congress on Evolutionary Computation, Volume 1, p. 601-608, 2000.

[5] Doty, Keith L., TJ-Pro Assembly Manual, Mekatronix, 1999.