Advances in the Application of the Evolvable Hardware Method to Autonomous Robot Design

By
Jason Plew, Dr. Antonio A. Arroyo, Dr. Eric 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 most 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), which uses Genetic Algorithms to apply evolutionary forces to the development process. The application of this method to the design of control systems for mobile robots is the focus of my master's thesis, and was begun in the spring of 2002. The topic was first introduced at FCRAR 2002 [1]. Since that conference, advances have been made, including the primary accomplishment of the design and production of an FPGA-based controller for a mobile robot. This system will be used as the platform for the EHW research, which is being implemented through ongoing code development. This paper will discuss both the FPGA Controller designed in the recent year, as well as the refined EHW design process that has resulted.

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 can be automated, then this approach would have potential. Several different methods are currently being investigated in an attempt to develop such a process, with neural networks being the most popular. Another method is Evolvable Hardware, which is the focus of this research. The EHW method allows evolutionary forces to produce a design that

fulfills a goal as efficiently as possible. It 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 [2]. We can consider the process to be similar to those undertaken in nature, and therefore use similar terminology. Thus, a set of designs can be considered a generation. Those chosen to be used to create a new generation can be called parents, and undergo reproduction and mutation to result in their design offspring. After several generations, the resulting designs should be able to successfully accomplish their objective.

Our goal in this research is to use EHW to develop a control system for a mobile robot. The initial step will be to evolve a mobile robot that is able to demonstrate obstacle avoidance. Work over the past year has been focused on accomplishing that objective. More complex scenarios will then follow.

2. EHW Research Platform

A major advance over the past year has been the development of a system to be used for phase one of the research. The robot used for this research is designated the TJ-PLU, or Talrik Junior: Programmable Logic Unit, and is shown in Figure 1. The development of the robot was used as a Senior Design Project at the University of Florida's Electrical Engineering Department.

2.1 TJ-PLU Robot Body and Sensors

The TJ-PLU is based on a TJ-PRO Robot, which is a stock mobile robot designed and used at MIL for a variety of research purposes. The TJ-PRO uses hacked servo motors for propulsion, and has both infrared sensors and contact switches for object detection. The TJ-PLU uses a TJ-PRO body with a widened undercarriage to fit a new controller system. The platform itself is made out of balsa wood, and is cut out using a CNC machine at MIL. The sensors and motors used for the TJ-PLU are identical to that of the TJ-PRO. While the contact switches produce a digital output, the infrared sensors require a 40 kHz signal to create the IR, and produce an analog output.

2.2 TJ-PLU Controller Board

Given the fact that multiple designs will have to be evaluated, existing robots at the Machine Intelligence Lab, which are based on microprocessors, were insufficient. Therefore, a new robot controller was designed and built for this research that would support the EHW design process.

The TJ-PRO uses a MTJPRO11 controller, which is built around an HC11 microprocessor. This device would of course have to be replaced, and the new design would have to be both based on a Programmable Logic Device (PLD) and be able to control the TJ-PLU sensors and motors. The result was a controller card built around an Altera FLEX10K Field Programmable Gate Array (FPGA). A system diagram of the TJ-PLU controller card is shown in Figure 2. Figure 3 shows a picture of the controller card.

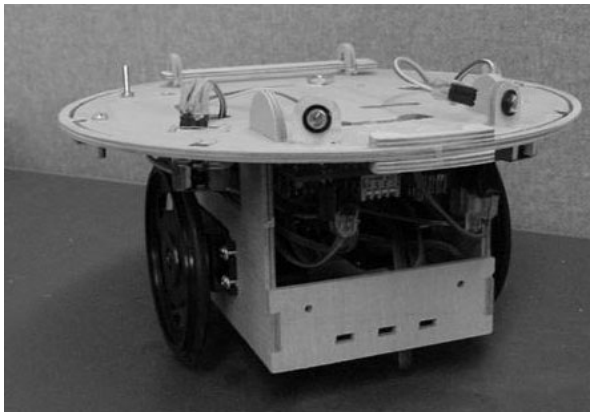

Figure 1: TJ-PLU

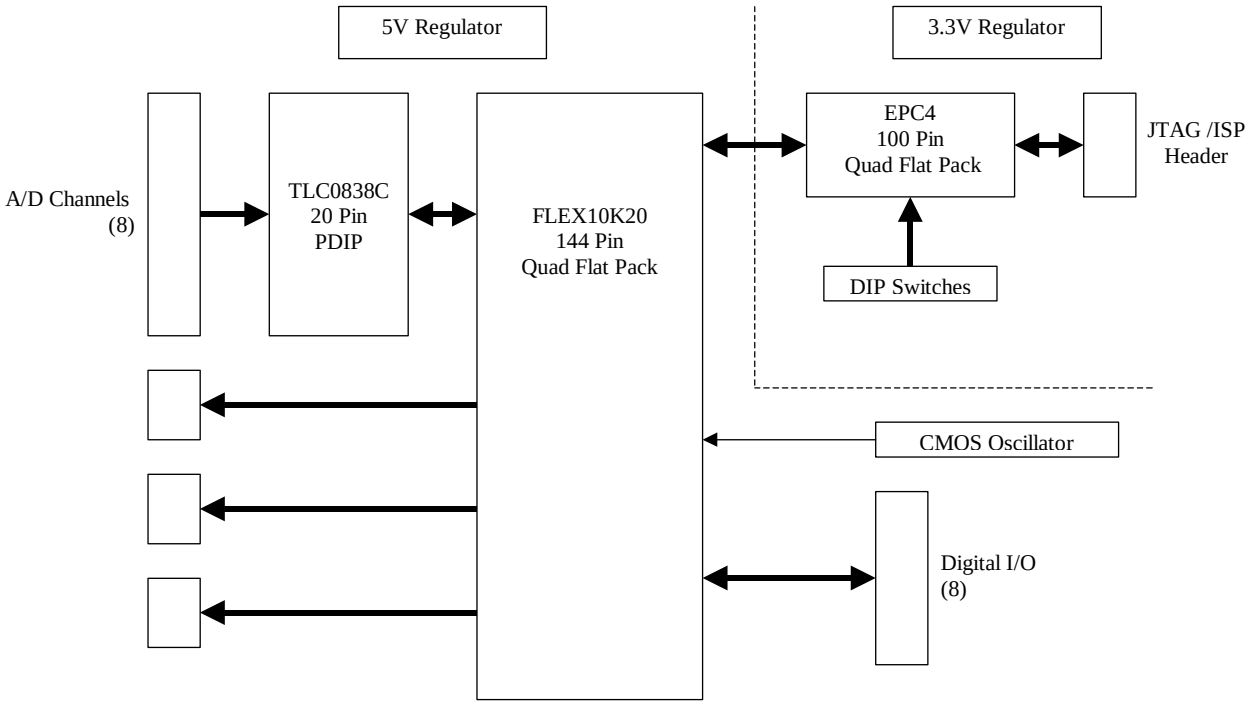

Figure 2: TJ-PLU Controller Board System

The specific FPGA is a FLEX10K20 in a 144-pin thin quad flat pack, which can contain a maximum of 63,000 logic gates. FPGAs are an SRAM-based technology, and therefore lose their configuration when power is shut down and must be reprogrammed when restarted. This would make the operation of the robot rather difficult, and so a FLASH-based configuration device from Altera, an EPC4, was used. An added benefit of using the EPC4 is that the device can contain multiple design files, allowing an entire generation to be downloaded from a PC to the controller card.

The controller has a series of DIP Switches that can be set to have the EPC4 use a specific design stored in its memory. The FPGA will then be programmed with this design whenever the robot is turned on.

FPGA designs for the FLEX10K can be created through Altera's MAX+PLUS II software, and multiple designs can then be combined into a programming file for the EPC4 through Altera's new software package, Quartus II. Quartus can also download the resulting file to the EPC4 through a JTAG interface.

One issue was that the EPC4 is only available as a 3.3V device, since the industry is moving forward to low voltage devices to conserve power. However, most of the sensors used by our mobile robots have not yet made that

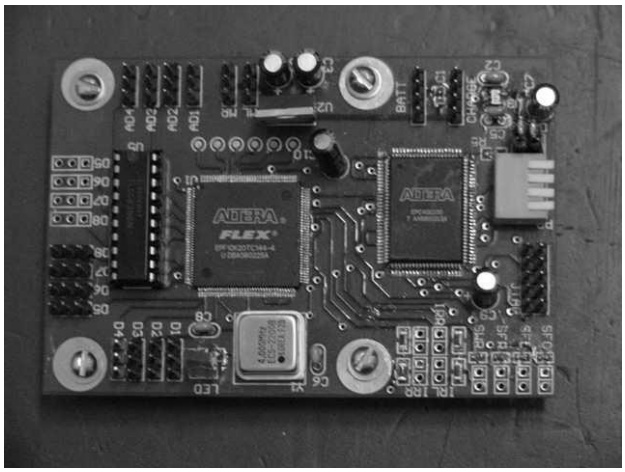

Figure 3: TJ-PLU Controller Card

transition, and still operate at 5V. So that buffers would not be required as an interface between the sensors and the FPGA, the 5V version of the FLEX10K was used. Fortunately, the EPC4 was designed to operate in mixed-logic designs, and so the only added complexity was having two separate power systems.

However, even though the FLEX10K is a 5V device, its maximum output was discovered to be only about 3.8V [4]. Though all the peripheral systems currently used in the TJ-PLU view this as an acceptable high input voltage, it will bear watching as the project continues.

One final design issue also had to be resolved. Using the TJ-PRO IR Sensors meant that some method of transferring their analog values to the FPGA was required. Comparators could have been used, but the resulting system would not have been a true replacement for a TJ-PRO. Furthermore, while translating each analog signal to a single binary value might be enough for simple obstacle avoidance, future research will most likely require more robustness. Therefore, an Analog to Digital Converter (ADC) was included in the system – specifically, TI's TLC0838, an eight channel serial ADC. Controlled by the FPGA, it sends the system an eight-bit approximation of the analog inputs from the IR detectors, and therefore allows the FPGA design to determine what threshold should result in special actions.

2.3 TJ-PLU Controller Software Design

The actual control of the TJ-PLU is located in a state machine that resides in the FPGA. The state machine reads the digital signals from the bump sensors, and also access the data from the IR Sensors through the ADC interface. The result of the state machine is speed and direction information for each motor, which is used by yet another module to create the actual Pulse Width Modulation (PWM) signals to control the servo motors used by the TJ-PLU. A system diagram of the controller software can be found in Figure 4.

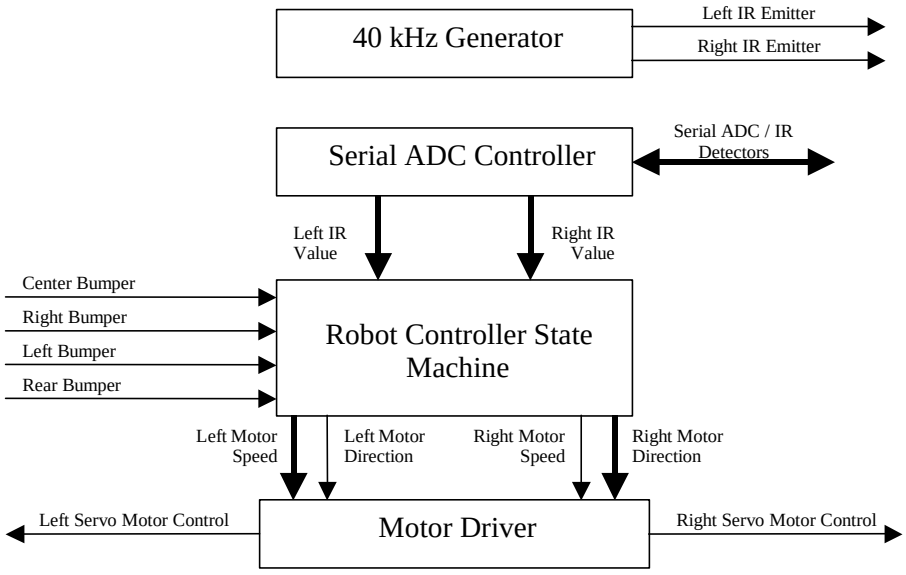

Figure 4: TJ-PLU Controller Software Design

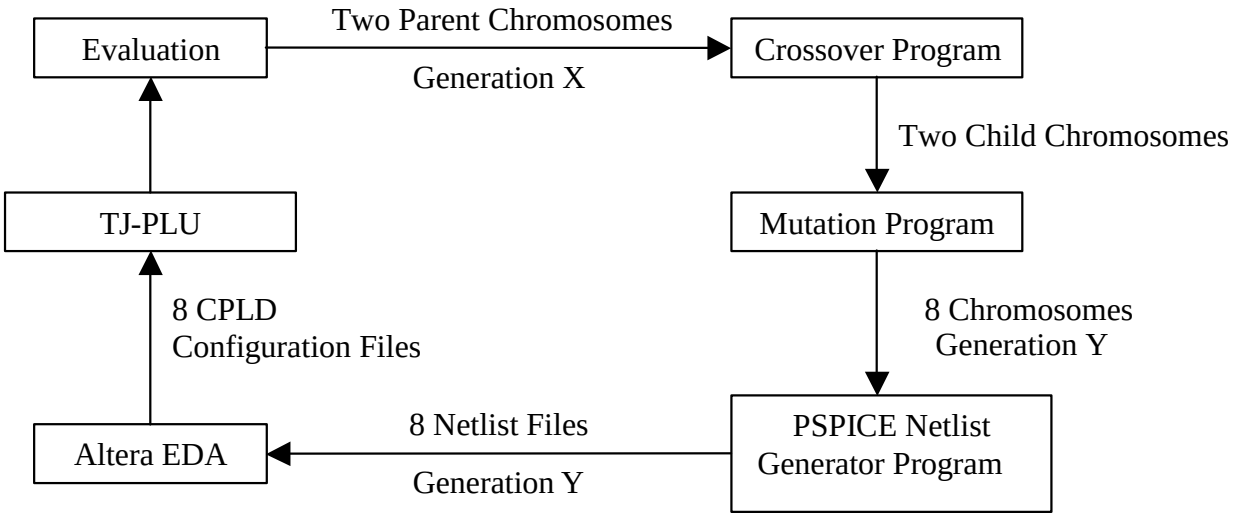

Figure 5: Design Process

3. Evolvable Hardware Design Method

For phase one of the research, the state machine shown in Figure 4 that will be replaced by a system produced through the Evolvable Hardware method. This design process was first explained in a paper at the 2002 FCRAR conference [1]. Since then, code has been developed that builds the underlying infrastructure of the process, namely creating the actual designs. During this process, the actual method of evolving a system has been modified from that discussed previously. The revised design process can be seen in Figure 5, and is described below.

3.1 Design Layout

We can start with an initial generation of eight designs. Eight is chosen because this is the maximum number of distinct designs that can be stored in the EPC4. In the event that more variety is required to result in optimal solutions, the number of designs in a generation will be increased. Each design is described by a file which, again using nature as a reference, we call a chromosome.

These chromosomes consist solely of a string of integers. At the beginning of this chromosome is a header section containing information about the design, including the generation it belongs to, its identification within that generation, the identities of its parents, and also the size of the design in the form of a variable, *n*. This relates to the actual layout of the design, which can be seen in Figure 6.

The design exists as a $(2n+1) \times (2n+1)$ matrix, or Cell Array. Each individual cell is described by yet another string of numbers, called a gene, which follow the header in the chromosome. Each gene describes the cell's location in the array, the function that exists in that cell, and where that function's inputs come from. To simplify the reproduction process, each function consists of at most only two inputs, and only one output. Furthermore, a cell can only receive signals from outputs produced at most two cells away. This will hopefully maintain some control of the complexity of the design, and reduce the risk of a bad mutation resulting in a defective design.

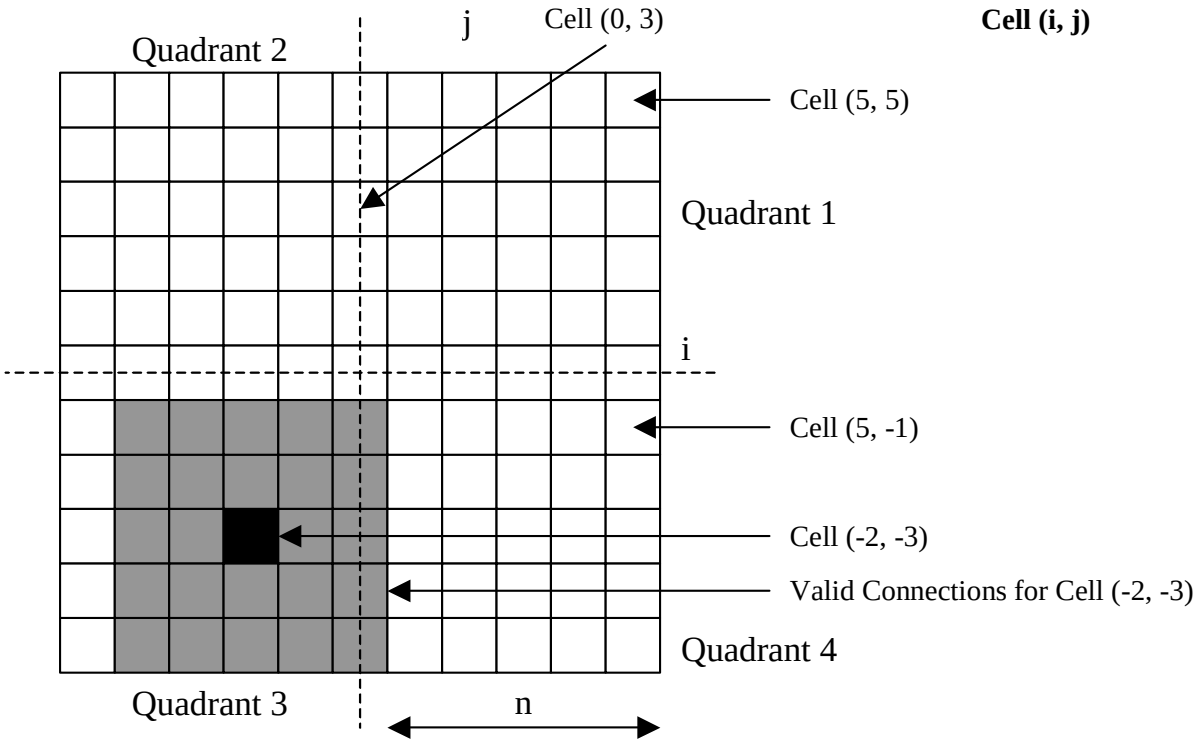

Figure 6: Cell Array

These functions include the basic building blocks of digital logic - an inverter, AND, OR, and XOR gates. Constant outputs of high and low signals are also available. Though the possibility of asynchronous design was considered, it was decided that the use of a global clock would be more suited to the first phase of the research, especially given the fact that the robot motors and IR sensors, as well as the FPGA modules that interface to them, are synchronous designs. Therefore, another function available is D-flip flops, all of which are tied to a single global clock

The final two functions available are global inputs and output buffers. Each serves to connect the evolved system to the outside world. Multiple buffers can be placed in the system, allowing different sections to be connected to the same signal. In the case of global outputs, all of the buffer signals for a specific output will be fed into an OR gate. The signals to the state machine that will be evolved during phase one can be seen in Figure 4. Each will be divided up into individual binary signals.

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 functions described above. This will be decided during phase one. 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 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.

3.2 Transferring a Design to the FPGA

The design process will begin with two initial designs that have been randomly generated to contain some preexisting functions. They will then undergo reproduction, where different sections of each design are combined to produce children with characteristics of each parent. For example, child A may get quadrants 1 and 3 from the first parent, and quadrants 2 and 4 from the second, while child B may get the exact opposite.

Four copies of each child will be made. Mutation will then occur, by randomly modifying existing cells, including both their functions and their connections. This randomization process will be based on a principle of costs. Only so many changes will be made to a design during mutation. More dramatic changes to a cell will be more expensive, and thus the amount of change that occurs from one generation to the next will be limited.

The new chromosomes will then be converted into a netlist. These files will be imported into MAX+PLUS II, which will map the functions into preexisting HDL code, and then use the netlist to produce FPGA configuration files. The files will then be combined into a single program file for the EPC4, and downloaded to the TJ-PLU for evaluation.

The method of evaluating the designs will be different for every phase of the research. In the case of evolving the state machine, we are looking for the demonstration of obstacle avoidance. In this case, several factors may be considered, including whether the robot moves at all, how often it hits an object, the maximum number of times it hits the same object, or the number of times it successfully avoids an object. These values will then be used in a fitness function that will produce an overall measure of the success of the design. Those designs with a better fitness value will not be assured of being chosen to reproduce, but they will have a greater chance. Once the two parents have been chosen, the design process from Figure 5 can begin again.

4 Future Work

During the course of evolving the state machine controller for the TJ-PLU, the EHW design process will be refined as necessary. Once this is complete, the research will move into phase two. This will explore the possibilities in evolving two distinct robot systems, a predator and a prey. A major portion of evaluating the designs will involve which predator has the most "kills", and which prey are able to survive the longest. It is believed that the relationship between the two distinct types of robots will result in designs that reach optimal solutions more quickly then in the case of the single system worked on in phase one. Improvements in the predator design will result in a "culling" of the prey designs. The rapid changes in the prey designs will result in the predator being forced to change, and the cycle will continue. Furthermore, the EHW design method itself can be tested by testing the evolved predator and prey robots against traditionally designed counterparts.

Another improvement being considered once the EHW design process is refined is to modify the TJ-PLU controller board to have an onboard computer system that can both generate the designs, download them into the FPGA, and then monitor the sensors itself to help determine the fitness of the design. This would allow greater autonomy and speed up the design process.

Finally, yet another step will be to transition from evolving just the internal state machine controllers to developing the overall robot control system itself. This could be

accomplished by moving as much of the interface modules into the evolved system. The motors could be fully hacked, and then controlled directly by the internal controller. Any speed control and direction would then need to be evolved. Newer IR detector technologies would also remove the need for a separate 40 kHz generator. Other sensor systems would also have their interfaces evolved, instead of designed through the traditional methods.

Many of these sensors would still result in an analog signal, however. And the process of attempting to evolve a controller for the serial ADC would be both complex and time intensive. One potential method would be to use EHW to evolve not only digital systems in an FPGA, but also analog designs in a Field Programmable Analog Array (FPAA), which is an emerging market. This would allow the entire scope of a robot controller design to fall within the purview of EHW.

The purpose of attempting to evolve more of the controller system is to allow us to fully explore alternatives to the traditional methods of controlling robots. The system described above, though designed through EHW, is still tied to the microprocessor roots of robot control. The sensors are intended to be controlled by a processor, the modules are synchronous designs, and even the state machine that will be evolved in the first phase of the research is itself essentially a simplified microprocessor. The true power of EHW will be demonstrated once we allow it to move beyond our methods of design and explore the other possibilities that we cannot presently foresee.

Bibliography

[1] Plew, J., "*Applying an Evolvable Hardware Approach to Autonomous Robot Design*", Florida Conference on Recent Advances in R 2002

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

[3] 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.

[4] FLEX10K Embedded Programmable Logic Family Data Sheet v4.02, Altera, May 2002

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