

B644 Final Project Report:

Cyclotron Beam Control

Gan Chen

Xiaoying Han

Albert Hartono

Jinkyung Hong

Chien-Wei Jeff Huang

Yu-Chun Wang

Computer Science Department

Indiana University

Bloomington, Indiana 47405

Abstract

Limitation of precision and difficulty of configuration slows the steps of development of Analog Computers. However, we also see the positive side of Analog Computers which could bring potentials to its' development and applications. A brief report on our project over the semester describes what we did, including experiments and analysis of XOR on EAC, plans to provide network traffic load balancing, the approach to control the alignment of the beam in the cyclotron facility using two current mirrors with four LLAs to produce possible control logic.

Keywords:

Genetic algorithms
Spatial traffic load balancing
Extended analog computer
Lukasiewicz logic array
Cyclotron Beam Control

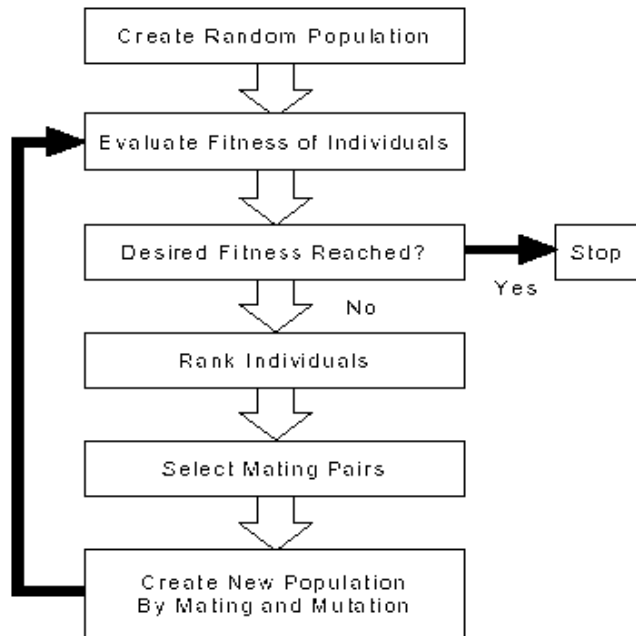
1 Team Work On XOR

Boolean exclusive-OR (XOR) is important because it is the simplest non-linearly-separable function. It also has an elegant correspondence to a network of leaky neurons [1].

1.1 Genetic Algorithm

The Genetic Algorithm is a model of machine learning that derives its behavior from a metaphor of the processes of evolution in nature. This is done by the creation within a machine of a population of individuals represented by chromosomes, in essence a set of character strings that are analogous to the base-4 chromosomes that we see in our own DNA. A GA Search proceeds as follows: Two chromosomes chosen randomly from the population are mated and they go through genetic operations like Crossover to yield better chromosomes for the next generation. This is repeated until about half of the populations are replaced with new chromosomes. Because the population size is fixed, chromosomes with lower fitness values tend to be eliminated from the population, therefore after several generations of GA search, relatively high fitness chromosomes remain in the population and some of them are chosen as solutions to the problem [1]. Therefore, when the Genetic Algorithm is implemented it is usually done in a manner that involves the following cycle: Evaluate the fitness of all of the individuals in the population. Create a new population by performing operations such as crossover, fitness-proportionate reproduction and mutation on the individuals whose fitness has just been measured. Discard the old population and iterate using the new population.

The process of developing a GA for a particular application is described in the following chief phases:



1.2 Design of XOR

Our version of genome is described below. Each chromosome has the following genes: Conductive Sheet Input, Mirror, LLA, and Output. The length of the chromosome is 54 bits.

Conductive Sheet I/O: 9-bit wide for a 3 x 3 EAC. Each bit specifies the I/O of a location on the conductive sheet.

Mirror: 9-bit wide for a 3 x 3 EAC. Each bit specifies the use of a voltage mirror at certain location.

LLA: 27-bit wide for a 3 x 3 EAC. 3 bits as one LLA function specifying one of the 27 piecewise linear functions. Each 3-bit unit specifies a LLA function for a location.

Output: 9-bit wide for a 3 x 3 EAC. Each bit specifies whether a location should be used as an output.

1.3 GA Experiments

We went through the XOR evaluation provided by Nathan and tested the code that simulate the XOR in GA algorithm with different parameters and made a few observations (see our first test and result in Appendices).

First we found that if we want to get perfect fits, the generation size (population) or number of generations should be big enough. For example, to find perfect fit from population of 1000, the generation number should be greater than 900. There is very few chances to find perfect fits with “evo-eac 100 1000 XOR -f output1.txt -m 0.05 &” due to limitations of the parameters.

Generally the number of generation that breeds the perfect fits is very random. But there is the trend that the larger the generation size is the sooner to find the perfect fits with the constant.

We classified the experiments to test how the generation size, the generation number or the mutation rate could affect the result. Since it is not guaranteed that every time that the program can breed best fits, we test the same parameters group for three times.

1) Same mutation (0.08), the same size (1000), different #gen

500	1st time	no fits find	
500	2nt time	no fits find	
500	3rd time	no fits find	
700	1st time	fits find	450 gen
700	2nt time	no fits find	
700	3rd time	no fits find	
900	1st time	no fits find	
900	2nt time	fits find	40 gen
900	3rd time	no fits find	
1200	1st time	fits find	1111 gen
1200	2nt time	fits find	1087 gen
1200	3rd time	no fits find	
1500	1st time	fits find	814 gen
1500	2nt time	fits find	485 gen
1500	3rd time	no fits find	

Conclusion: The bigger that the number of generations is, the more chances to breed best fits.

2) Same mutation (0.08), the same #gen (1500), different size

1000	1st time	fits find	814 gen
1000	2nt time	fits find	485 gen
1000	3rd time	no fits find	
5000	1st time	fits find	592 gen
5000	2nt time	fits find	591
5000	3rd time	fits find	542 gen
8000	1st time	fits find	4 gen
8000	2nt time	fits find	16 gen
8000	3rd time	fits find	50 gen

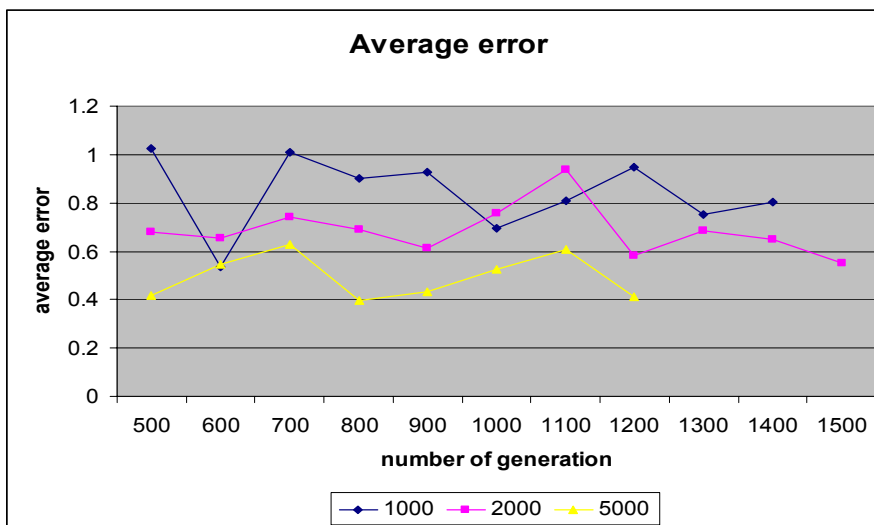
Conclusion: The larger the size of population the sooner to find the best fits.

3) Same size (1000), the same #gen (1500), different mutation

0.08	1st time	fits find	814
0.08	2nt time	fits find	485
0.08	3rd time	no fits find	
0.30	1st time	no fits find	
0.30	2nt time	no fits find	
0.30	3rd time	no fits find	946
0.40	1st time	no fits find	
0.40	2nt time	fits find	480
0.40	3rd time	no fits find	
0.50	1st time	fits find	155
0.50	2nt time	fits find	903
0.50	3rd time	no fits find	
0.60	1st time	no fits find	
0.60	2nt time	no fits find	
0.60	3rd time	no fits find	
0.80	1st time	fits find	224
0.80	2nt time	no fits find	
0.80	3rd time	fits find	175

Conclusion: The mutation rate did not seem to have that large of an effect on the chance to breed the best fits.

4) The following figure shows the error graph for simulations different generation size.



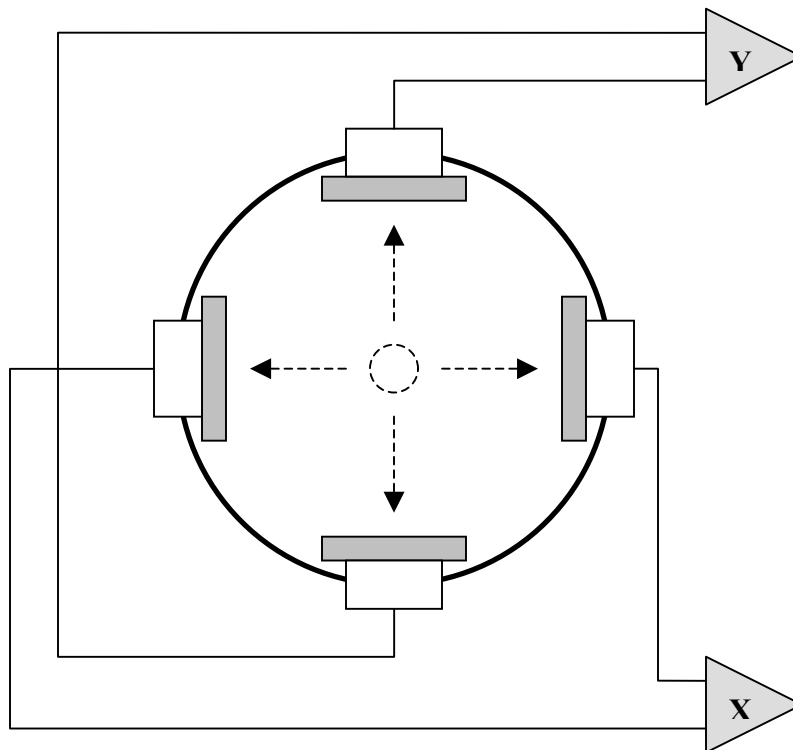
Conclusion: The figure shows that the average error decreases with the increasing of the number of generation.

2 Team Project: Cyclotron Beam Line Controller

High-energy high-speed beam of charged particles is produced by accelerator for many applications such as research in nuclear physics, isotope production, medical therapies, and many other industrial applications. A high-energy ion radiation is delivered as a fixed horizontal beam to the object target. The ion beam is very attractive to many applications because of its unique characteristic that it can be positioned accurately in extreme position in three dimensions. The positioning methods are depth control that is accomplished by energy variation and lateral control that is affected by magnetic or electrostatic redirection.

Cyclotron Beam Line Controller adjusts the magnetic field of each dipole and quadrupole magnet attached to the accelerator structure to maximize the beam intensity and accuracy. Beam position monitor (BPM) is a device that detects and determines the exact location of the beam radiance going through an accelerator structure. BPM device consists of four metal strips located inside the accelerator structure. Wires are connected to the accelerator to extend to outside the structure and finally the wires must be grounded. These whole devices are joined together isolated from the accelerator structure itself.

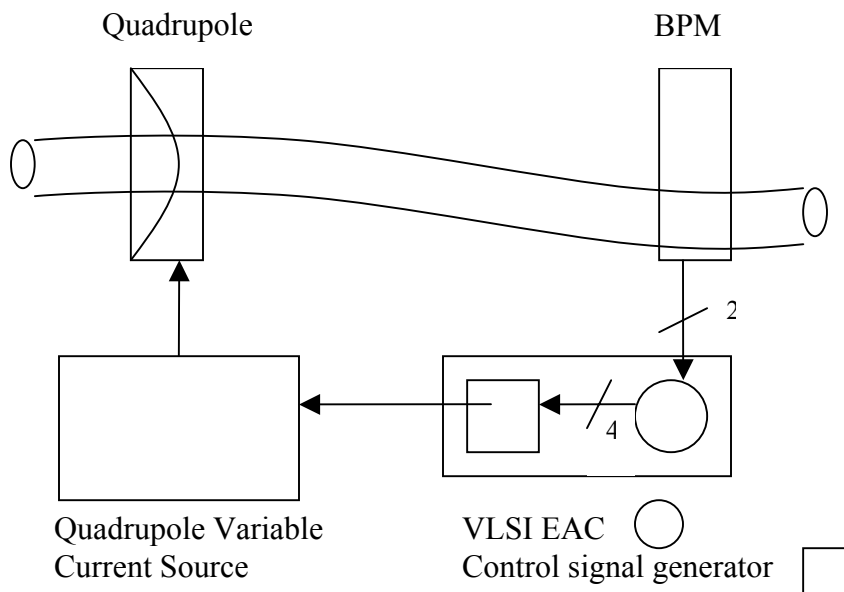
Below is the outline picture of a beam position monitor.



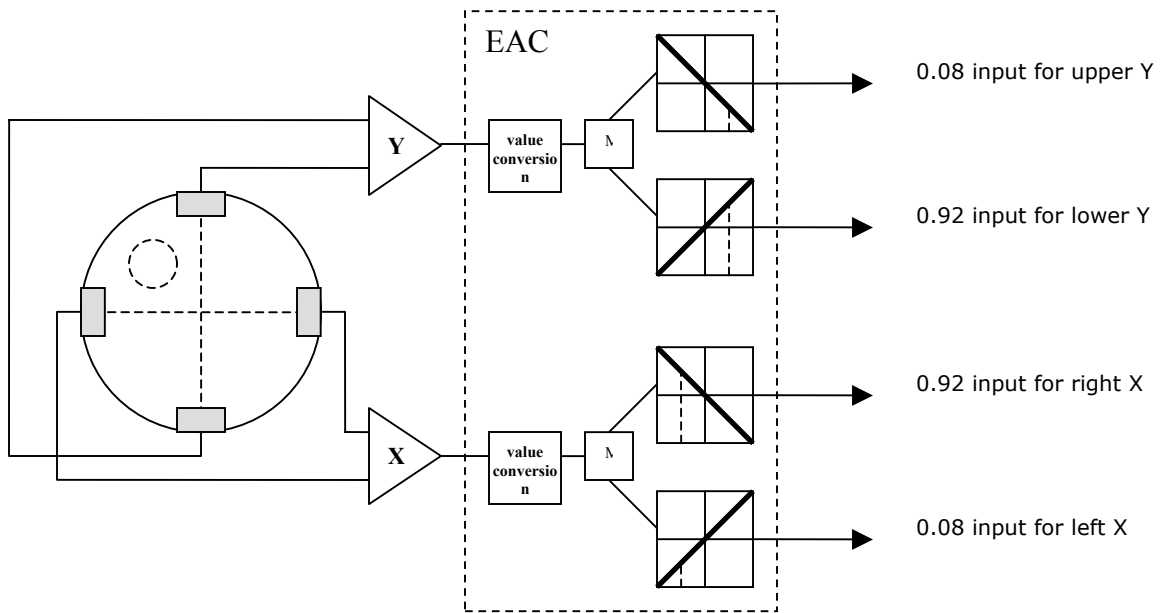
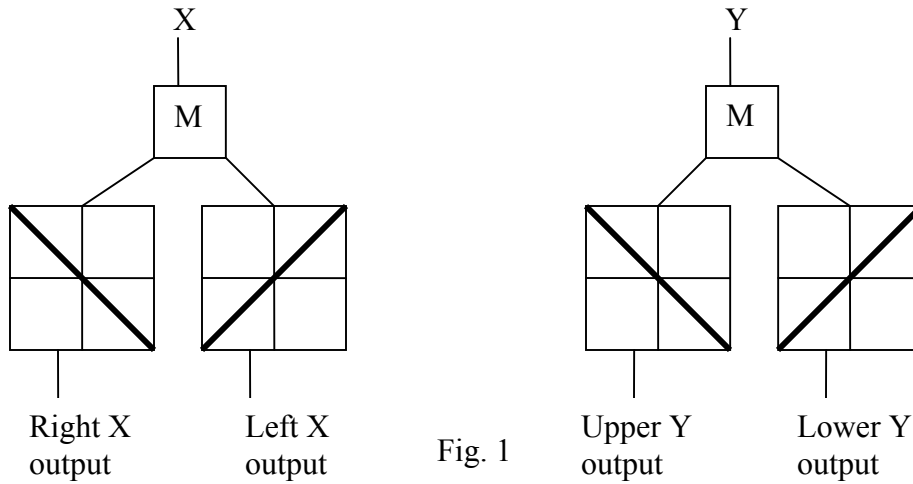
The use of EAC in real-time systems eliminates the need of complicated digital CPU or logic arrays to perform arithmetic and logic functions [2][3], increases the response speed, and gives better reliability over traditional digital controllers.

To control the alignment of the beam in the cyclotron facility, first we need to detect the whereabouts of the beam. The BPM used in IUFC has four sensors. Unsure of the output format, we are assuming the BPM produces two readings, one for x-axis value and the other one for y-axis value [4] [5].

With the pair of (x, y) values as inputs, we use VLSI EAC to produce four corresponding outputs to the quadrupole that adjusts the moments of the beam.



Intuitively, we can simply use two current mirrors with four LLAs to produce possible control logic as shown in Fig. 1 even without the use of the conductive sheet since these two inputs can be considered as two isolated controlling signals. However, the output values from BPMs may not be within the input range for LLAs; therefore we need to find an EAC configuration that could accept the signal from the BPMs and produce desired output at the same time.

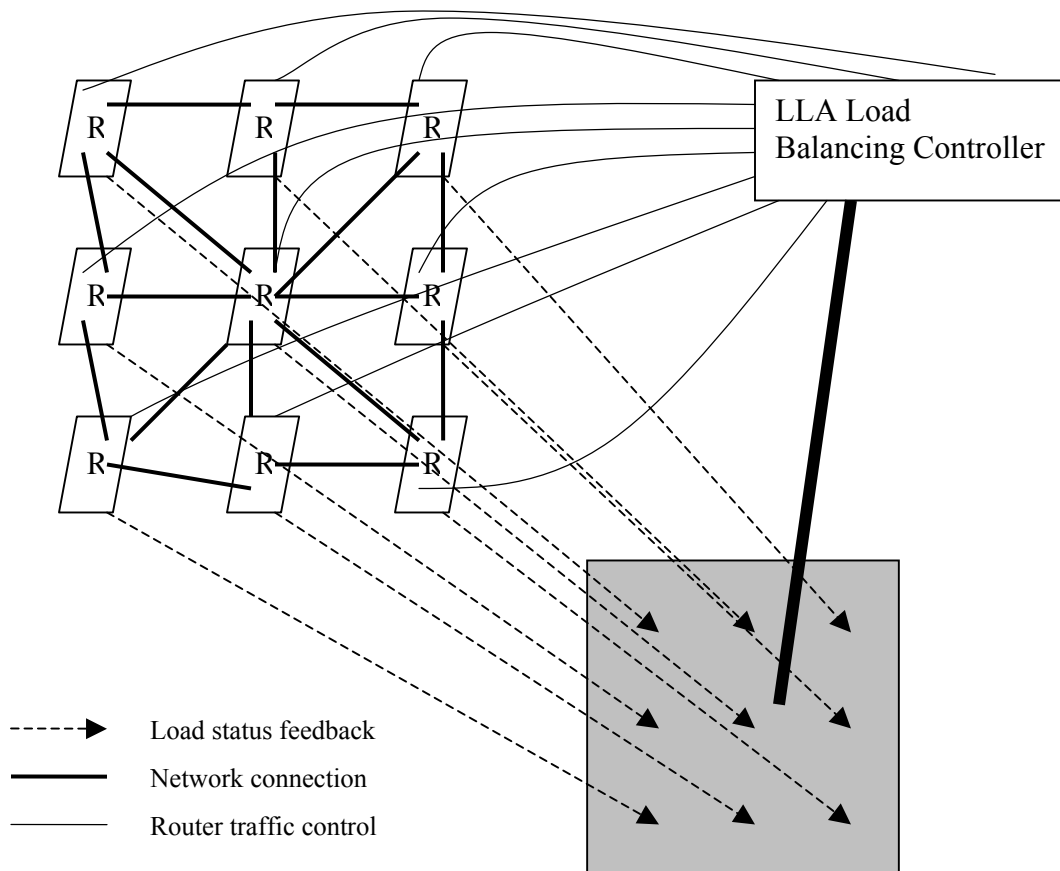


On the other hand, this implementation would only be able to produce linear control values. If non-linear control values are needed, such as exponential intensity with respect to the distance, using GA will be a better way to find a suitable configuration. If GA is to be used, simulation of x and y values are to be used as inputs in the evaluation of fitness and desired non-linear results will be used to be compared with the outputs.

3 Spatial network traffic load balancing

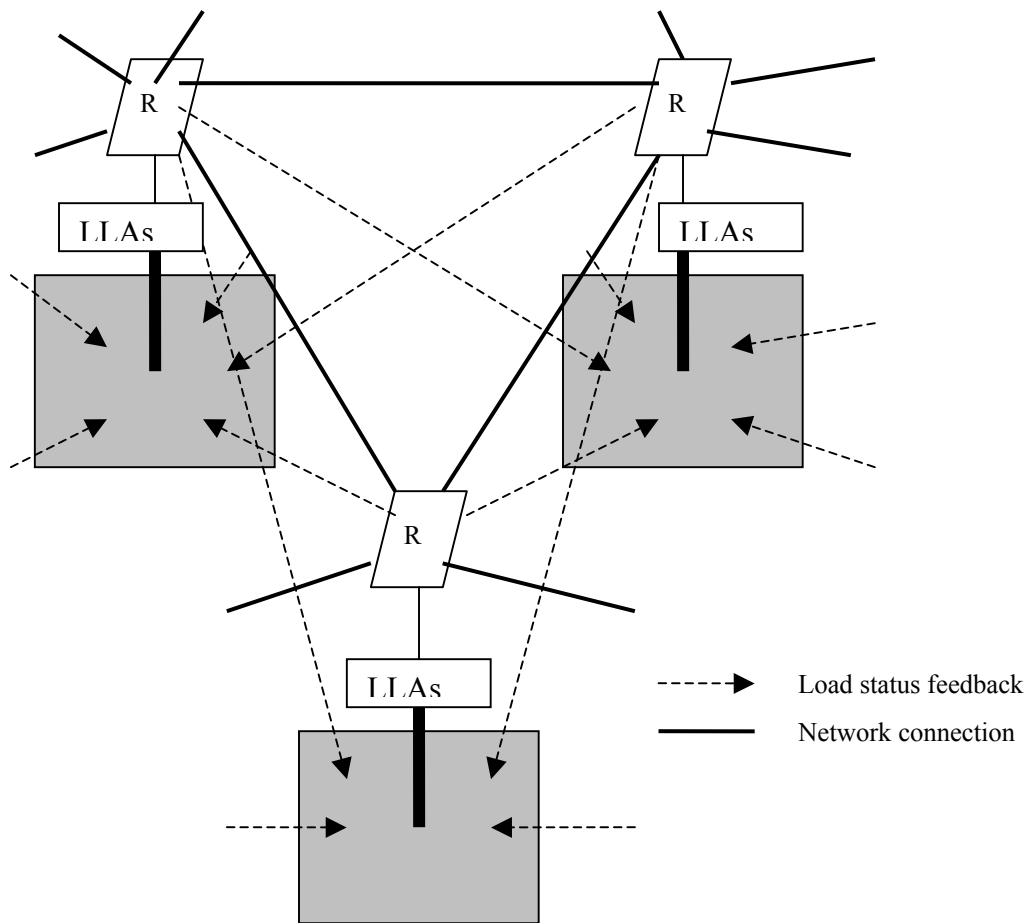
To prevent overload some part of the network and redirect network traffic away from already congested area, network traffic load balancing provide such function without modification to the network client or server. Because of the nature of network traffic, load balancing needs to check large amount of data and provides immediate redirecting control. The characteristics of EAC, like imprecise but very fast, simple, and reliable, seem to be more suitable in such task than traditional digital solutions.

With the following designs, the router may be traditional digital router enhanced with the ability to accept traffic control signals or entirely new router design with built-in control signal acceptance.



A centralized load-balancing layout is used in this design. All the routers feedback load status to a centralized EAC that computes and produces control signals to each router to redirect network traffics. The EAC needs to be configured to produce control signal to instruct the routers to send traffic toward less congested paths. Such EAC oversees the entire network load status and may be configured to provide better balancing strategies over the other more modular design, but we have yet to develop a design for the control signal with this setup. On the other hand, without the modularity found in our next

design, configuration may grow significantly in complexity when the network layout/connection is expanding physically.



In this layout, there is a load balancing EAC within each router. The task of load balancing resides in every router on the network. Each router communicates its current load status with the other routers that are connected to it directly. No centralized load balancer is used. The EAC inside of each router has the number of input equals to the number of other routers that are connected to it. The EAC may generate outputs that will be able to differentiate the number of routers that are connected to it and the specified router is having lower traffic load than other routers. When the specified value (connected router with lower/lowest traffic load) is presented, the router that this EAC resides should direct more packets to the specified router. However, it has one potential problem is that the router reporting lowest load may not be a path to the destination point. A possible solution is to determine potential routers without load balancing first, and then feed such information into EAC to produce a selection among the potential routers based on their load status feedback.

Reference

- [1] Kenneth V. Noren, John E. Ross. Analog Circuit Design Using Genetic Algorithms.
- [2] <http://www-elsa.physik.uni-bonn.de/~keil/pdf/dipac99.pdf>
- [3] <http://www.slac.stanford.edu/econf/C011127/TUAP016.pdf>
- [4] <http://www2.slac.stanford.edu/vvc/accelerators/bpm.html>
- [5] <http://www.bnl.gov/atf/experiments/BPM/Bpm2.htm>
- [6] Mills, J., "Lukasiewicz' Insect: The Role of Continuous-Valued Logic in a Mobile Robot's Sensors, Control, and Locomotion", Proc. 23rd Int. Symp. On Multiple-Valued Logic, IEEE Computer Society, 1993.
- [7] Mills, J. W., M. G. Beavers, and C. A. Daffinger. 1990. Lukasiewicz Logic Arrays. Proceedings of 20th International Symposium on Multiple-Valued Logic.
- [8] Mills, J. W. 1992. Area-Efficient Implication Circuits for Very Dense Lukasiewicz Logic Arrays. Proceedings of 22nd International Symposium on Multiple-Valued Logic. Sendai, Japan: IEEE Press.
- [9] Mills, J. W. 1995. Programmable VLSI Extended Analog Computer for Cyclotron Beam Control.
- [10] Mills, J. W. 1995. The Continuous Retina: Image Processing with a Single-sensor Artificial Neural Field Network.

Appendices:

Codes:

The source code is written by Nathan Ainslie and consists of the following classes:

- CConductiveSheet: Simulates the conductive sheet
- CLLA: Simulates a single LLA
- CGenome: Contains the genome data and breeding functions
- CEval: The base class for all fitness evaluators
- CXOREval: Finds the error for XOR running on a Genome
- CIMPEval: Finds the error for implication running on a Genome
- CANDEval: Finds the error for AND running on a Genome
- CBreedingPop: A list of genomes sorted by error used for breeding the next generation
- CGod: The overseer function and associated data

Also included with the code are:

- main.cpp: the main function
- makefile: a *NIX makefile

Here we just post the main.cpp and you can find more codes from Nathan's website.

<http://www.cs.indiana.edu/~nainslie/evo-eac.html>

```
/*
EVO_EAC
Nathan Ainslie
nainslie@indiana.edu
Oct 2002

main.cpp

*****/
#include "ConductiveSheet.h"
#include "Genome.h"
#include "LLA.h"
#include "Eval.h"
#include "XOREval.h"
#include "ANDEval.h"
#include "IMPEval.h"
#include "BreedingPop.h"
#include "God.h"

#include <stdio.h>
#include <string.h>

#define RES 2.0
```

```

int NUM_GENS = 100;
int GEN_SIZE = 10000;

int main(int argc, char* argv[]) {
    int i = 4;
    FILE* pOutputFile = NULL;
    CEval* pEval;
    if (argc >= 4) {
        NUM_GENS = strtol(argv[1], NULL, 0);
        GEN_SIZE = strtol(argv[2], NULL, 0);
        if (!(strcmp("AND", argv[3]))) {
            pEval = new CANDEval(RES);
        } else if (!(strcmp("XOR", argv[3]))) {
            pEval = new CXOREval(RES);
        } else if (!(strcmp("IMP", argv[3]))) {
            pEval = new CIMPEval(RES);
        }
    } else {
        fprintf(stderr, "Usage: %s num_generations generation_size function
[-f output file] [-m mutation_rate] [-v]\n", argv[0]);
        exit(0);
    }

    CGod oGod(NUM_GENS, GEN_SIZE);

    while(i < argc) {
        if(!(strcmp("-f", argv[i]))) {
            pOutputFile = fopen(argv[i+1], "w");
            oGod.setOutputFile(pOutputFile);
            i += 2;
        } else if (!(strcmp("-m", argv[i]))) {
            oGod.setMutationRate(strtod(argv[i+1], NULL));
            i += 2;
        } else if (!(strcmp("-v", argv[i]))) {
            oGod.setVerbosity(1);
            i++;
        } else if (!(strcmp("-e", argv[i]))) {
            oGod.setMinErr(strtod(argv[i+1], NULL));
            i += 2;
        } else {
            i++;
        }
    }

    oGod.setEvaluator(pEval);
    oGod.go();

    delete pEval;
    if (pOutputFile)
        fclose(pOutputFile);

    return 0;
}

```

Postings:

Date: Thu, 19 Sep 2002 00:05:17 -0500

From: Chien-Wei Jeff Huang chihuang@indiana.edu

Newsgroups: ac.csci.b644

Subject: AMD Announces Technology to Enable Ten-fold Performance Leap in Future Transistors

<http://news.moneycentral.msn.com/ticker/article.asp?Feed=BW&Date=20020910&ID=1913519&S=ID=1913519&Symbol=US:AMD>

Date: Mon, 30 Sep 2002 16:03:29 -0500

From: Chien-Wei Jeff Huang chihuang@indiana.edu

Newsgroups: ac.csci.b644

Subject: Higher clock frequency achieved by increasing number of circuit paths

<http://www17.tomshardware.com/cpu/02q3/020821/athlonxp-03.html>

An additional interconnect layer within the latest Athlon processor code name Thoroughbred "B" gives the processor a boost in operating frequency. Test sample of Thoroughbred "B" is showing a possible 2.8Ghz while the original Thoroughbred "A" maxs out at ~1.8Ghz. "The increased number of circuit paths results in better conductivity for the connections within the processor, which, in turn, automatically decreases the amount of thermal loss."

Date: Mon, 30 Sep 2002 16:03:29 -0500

From: Chien-Wei Jeff Huang chihuang@indiana.edu

Newsgroups: ac.csci.b644

Subject: Higher clock frequency achieved by increasing number of circuit paths

Our version of genome is described below. Each chromosome has the following genes: Conductive Sheet Input, Mirror, LLA, and Output. The length of the chromosome is 54 bits.

Conductive Sheet I/O: 9-bit wide for a 3 x 3 EAC. Each bit specifies the I/O of a location on the conductive sheet.

Mirror: 9-bit wide for a 3 x 3 EAC. Each bit specifies the use of a voltage mirror at certain location.

LLA: 27-bit wide for a 3 x 3 EAC. 3 bits as one LLA function specifying one of the 27 piecewise linear functions. Each 3-bit unit specifies a LLA function for a location.

Output: 9-bit wide for a 3 x 3 EAC. Each bit specifies whether a location should be used as an output.

Emails:

Date: Wed, 16 Oct 2002 17:01:08 -0500 (EST)
From: Chien-Wei Jeff Huang chihuang@indiana.edu
To: VLSI Group Members -- Albert Hartono <ahartono@cs.indiana.edu>, Gan Chen <chengan@cs.indiana.edu>, Jinkyung Hong <jihong@cs.indiana.edu>, Xiaoying Han <xhan@cs.indiana.edu>, Yu-Chun Wang yucwang@cs.indiana.edu
Cc: Chien-Wei Huang chihuang@indiana.edu

For those of you haven't got to check out what Genetic Algorithm actually do, I found a site which explains GA pretty well. http://www.biz.uiowa.edu/class/6K299_menczer/PPT/Davis There are a lot of great sites out there explaining GA in pretty good details...so you could find more detailed information about GA.

Date: Mon, 28 Oct 2002 03:09:41 -0500 (EST)
From: Chien-Wei Jeff Huang chihuang@indiana.edu
To: VLSI Group Members -- Albert Hartono <ahartono@cs.indiana.edu>, Gan Chen <chengan@cs.indiana.edu>, Jinkyung Hong <jihong@cs.indiana.edu>, Xiaoying Han <xhan@cs.indiana.edu>, Yu-Chun Wang yucwang@cs.indiana.edu
Cc: Chien-Wei Huang chihuang@indiana.edu
Subject: XOR code

Just ran the genetic algorithm code posted on the news group and it was pretty straight forward. Nathan's group wrote this code in C++ which put the kind of conceptual ideas of GA into a practical matter. If you guys still haven't started to get yourself drown in the sea of the GA, you really should get it going. Anyway, the idea about Mill's EAC is to have a piece of conductive plastic sheet w/ 9 inputs/outputs, 9 current (voltage) mirrors w/ 1 input and 2 outputs each, 9 LLA w/ 1 input and 1 output, and last, 9 outputs to the outside world from any of the above outputs.

Now, like the DNA works in life, we use a block of information to describe the properties of an EAC (cell). Like a very long string of bits describing the connections between the 4 main components above. For example, for the connections on the conductive plastic sheet, we can use 101001000 to describe the cs(conductive sheet) part:

```
101  in out in
001 => out out in
000  out out out
```

As how we are going to connect the wires on the cs. Similiar to the other main components. Now, the program starts w/ random strings of bits (with the length set by our genome, of course) to create a pool of chromosome samples. Then we run these samples w/ fitness functions to determine if the output from an EAC described by a sample suit our requirement (in XOR, we feed 2 inputs to it and see if the output is correct w/ the spec of a XOR function). Then we keep the best portion of these

chromosomes (in Nathan's code, best 10% of the population is kept) and have them breed with each others and replace the rest (90%) of the population with the new offsprings of these 10%. Then we run fitness function on these again....we keep it running until number of generations we set is reached or a specified level of fitness. Usually w/ the longer generation and bigger pool we'll be able to find a few chromosome that works well. Like the sample output from Nathan's site, his pool had a amazing chromosome producing a near perfect output (0.000001 error) for XOR (see his site) at generation 6084. My run had a pretty good one (0.000674) at a mere 43th generation! Try to play with the code posted on the news group a bit and you may be able to understand about this more. We'll need our knowledge well for our paper on cyclotron controller.
-----here's my best output-----

Running 10000 generations of size 10000
Mutation rate: 0.050000

Successful organism found in generation 43

Spacing: 1.128474
Sheet Inputs: |0|0|0|1|0|0|0|1|0|
Sheet Outputs: |0|0|0|0|0|0|1|0|0|
LLA's: |0|0|16|16|8|16|0|0|16|0|16|0|16|8|8|8|16|16|0|16|0|0|16|16|8|8|8|
Error: 0.000674

Test1 (0 XOR 0): f(0.000000) = 0.000000, Error: 0.000000
Test2 (1 XOR 0): f(0.499916) = 0.999832, Error: 0.000168
Test3 (0 XOR 1): f(0.499916) = 0.999832, Error: 0.000168
Test4 (1 XOR 1): f(0.999832) = 0.000337, Error: 0.000337

Date: Sat, 14 Dec 2002 19:51:30 -0500 (EST)
From: Albert Hartono <ahartono@cs.indiana.edu>
To: Xiaoying Han <xhan@cs.indiana.edu>
Cc: Chien-Wei Jeff Huang <chihuang@indiana.edu>, Gan Chen
<chengan@cs.indiana.edu>, Jinkyung Hong <jihong@cs.indiana.edu>,
Yu-Chun Wang <yucwang@cs.indiana.edu>
Subject: b644 documentation

Jeff, I think we still need the EAC in cyclotron. I think we cannot just take the output of BPM and then plug it as an input of LLAs. The EAC acts as the sensor and the output of EAC will be compatible with LLA. BPM output may not be compatible with LLAs. That's my opinion.

I also switched the LLA of Upper Y with LLA Lower X, LLA of Right X with LLA of Left X. I changed Jeff's words a little bit.

Date: Sat, 14 Dec 2002 22:12:36 -0500

From: Chien-Wei Jeff Huang <chihuang@indiana.edu>

To: Albert Hartono <ahartono@cs.indiana.edu>, Xiaoying Han <xhan@cs.indiana.edu>

Cc: Gan Chen <chengan@cs.indiana.edu>, Jinkyung Hong <jihong@cs.indiana.edu>, Yu-Chun Wang <yucwang@cs.indiana.edu>

Subject: Re: b644 documentation

Here's the copy of the paper I updated this afternoon. It does NOT contain the updates made by Albert so they need to be compared and combined.

Definition of EAC: conductive sheet with current mirrors and LLAs. So in a sense, what we have right now is still considered as an EAC, it's just that we didn't make use of the conductive sheet. However, after reading and some thinking, I think I know what Albert was trying to say and I think my design does have a major flaw which Albert pointed out.

No, EAC (remember that a EAC is cs, mirror, and LLA) does not act as a sensor in anyway. The EAC is merely a computing device sitting between the BPM (sensor) and quadrupole (alignment). It takes 2 output values as inputs from BPM, produce output that is recognizable by quadrupole, which we are assuming to be 4 output values. " We cannot just take the output of BPM and then plug it as an input of LLAs." "BPM output may not be compatible with LLAs." Albert is totally right! I wasn't thinking about it clearly so I simply thought about feeding the LLAs with the values from BPM and producing desired output, but I totally forgot that LLA accepts 0.0 ~ 1.0 as input while we have no idea about the range of the output from the BPM. So we do need to use the GA to produce a configuration that'll accept the output values from BPM as inputs, and produces output values acceptable by quadrupoles (assuming 4 separate inputs, again).

About the LLA that Albert switched, I think it doesn't really matter as long as we state clearly our assumption whether a higher output value means increased attracting force or increased repulsing force. After all we don't know the specification of the control values for the quadrupole.