Proceedings of the 2007 IEEE Workshop on
Evolvable and Adaptive Hardware (WEAH 2007)

Simultaneous Multi-Module Evolution: The Need for Simultaneous Evolution of Multiple Neural Net Modules for Brain Building

Hugo de Garis
Jian Yu Tang
International School of Software
Wuhan University
Wuhan, Hubei Province, China
Email: profhugodegaris@yahoo.com

Di Huang
Computer Science School
China University of Geosciences
Wuhan, Hubei Province, China
Email: aaron192032@gmail.com

Abstract- **This paper discusses a problem that our brain building research group is confronted with increasingly each year, namely, "How can we evolve, one neural net module at a time, while building artificial brains, when the technology, i.e. Moore's Law, keeps increasing the number of neural net modules that a PC can process in real time, by a factor of two nearly every year or two?" In 2006, it was possible to have a PC process the neural signaling in real time, of an artificial brain of some 50,000 modules, each to be evolved individually, by a human BA (Brain Architect) or EE (evolutionary engineer). This number is already too large to be practical, so the obvious question arises, "How to automate the simultaneous evolution of multiple numbers of neural net modules?" This paper presents the problem and gives some tentative answers.**

I. INTRODUCTION

This paper describes a problem that our research group is currently grappling with, namely, how to evolve neural network modules quickly enough, to build artificial brains, when today's processor chips and memory chips allow us to put 50,000 modules in a PC, then have the PC run the neural signaling of the artificial brain in real time. The problem is that 50,000 are just too many. In a year it will be 100,000, due to Moore's Law. Evolving one module at a time is now simply too slow, if we want to build artificial brains with 50,000 modules, as today's PCs make possible. This exponential explosion of electronic capacity ("more electronics than ideas") is forcing us to consider ways to evolve more than one neural network module at a time, but how to do this?

This paper presents some tentative ideas on how multi-modules can be evolved simultaneously.

We begin by saying a little about what brain building is and what it can be used for. The basic idea of our approach to building artificial brains is to evolve neural network modules, one at a time, each with its own little function, with its own fitness definition.

These modules are evolved in an electronic accelerator board fifty times faster than on a PC. Once each module is evolved on the board, it is downloaded into the PC. This process is repeated many thousands of times, to generate thousands of neural net modules in the PC. Special software is then used to specify the inter-connectivity of the modules to build artificial brains, according to the designs of human "BAs" (Brain Architects). The artificial brain in the PC can then be used, via radio antennas, to control the behaviors of such things as autonomous robots, vision systems, speech understanding systems, etc.

With Moore's Law supplying trillion trillion component machines by 2020, there will be plenty of scope for building artificial brains with vast capacities. Our research group is investigating how artificial brains can be built using the "evolved neural net module approach". If we are successful, other research groups could use it, because this approach is not expensive. The accelerator board costs less than $2000, a PC costs less than $1000, and a robot is also about $1000, so any computer science or engineering research group could get a small research grant and begin building artificial brains for many different purposes.

For over a decade the first author has been investigating the possibility of building artificial brains [2], [3]. Our approach recently has been to evolve neural network (NN) modules in programmable electronic accelerator boards to shorten the evolution times. These neural network modules are evolved one at a time in the board, and then downloaded into an ordinary PC.

Our group has written many papers in the past [3] on the evolution of individual NNs, which are typically fully connected, and consist of 12-16 artificial neurons. Each connection is weighted by a signed binary fraction of typically 8 or 9 bits. The dot product of the (signed binary fraction) incoming neural signal vector and the weight vector of each neuron is calculated. To this product is added the external signal values to a neuron. The sum is then sent through a sigmoid or squashing function that limits the absolute size of the signal to be less than 1.0 Copies of this output signal value are then sent to all neurons in the network, to become input signals in the next (tick) cycle.

Each NN module performs some little function that is evolved. Several thousand NN modules can be evolved individually in this way and downloaded into the PC's memory, where they are then interconnected according to the designs of human "BAs" (Brain Architects) to construct an artificial brain. The PC then performs the neural signaling of the whole brain, i.e. thousands of NN modules, sequentially, in real time (defined to be 25 Hz per neural signal) for all neural signals of all NN modules in the brain.

These modules can be evolved to perform a large variety of tasks, e.g. for pattern recognition, motion control, decision-making, memory, etc. The art of brain building is knowing which modules (i.e. their functions) are needed for a given application, and how they should be connected together. These decisions are made by BAs (Brain Architects).

A few years ago, our group did an experiment to see how many NN modules a normal PC of that year, could process sequentially in real time. We were shocked at the answer, which was, about 20,000 modules. That was several years ago. With another Moore's Law doubling, this number is now around 50,000. In another year or so it will be over 100,000. This is in effect an "embarrassment of riches". We now have more modules than we can handle. Even with quite a large research team of say 20 people (mostly graduate students) with each person evolving single NN modules separately, 20 people is not enough, when the total number of modules is about 50,000.

The question then arises "Is it possible to evolve several modules simultaneously?" These individual modules, once evolved one at a time, as we do currently, are subsequently interconnected in the PC to build artificial brains. "Could several NN modules be evolved together and then downloaded as a "unit", consisting of several interconnected NN modules, that functions as a subcomponent in the artificial brain design?"

Our group has barely begun to think about such issues, let alone trying to find solutions to the problem. However, due to the explosive, exponential growth of electronic capacity, due to Moore's Law, we are now being forced to. In earlier years, it was not an issue, because the size of the annual increments in the number of extra NN modules was too small to be concerned about. But, this is no longer the case.

Not only are the PC processing speeds, and the memory capacities doubling, so too are the capacities of the programmable chips (FPGAs) on the electronic accelerator boards we use to perform the evolution of the NN modules. We use these boards to shorten the evolution times of individual NN modules. To evolve a single module in a PC can take many hours to over a day. We find that an accelerator board can perform the same task about *50 times faster* than the PC.

At present, with our evolution of one NN module at a time, we find that the percentage of the logic gates used in the programmable chip is over 50%. So if we were to use the same chip to evolve several NN modules at the same time, we would probably run out of logic gates on the board.

Perhaps a little background knowledge on the use of the board would be helpful here. A user of the board writes high-level "C-language-like" code that consists of a genetic algorithm to evolve a NN module with a specific fitness function, to perform some task. This code is then *hardware-compiled* into instructions to configure the wiring of the programmable chip (FPGA). The more code, the greater number of programmable logic gates in the FPGA are used. But the FPGAs on the boards are themselves also subject to Moore's Law. They too are doubling the number of logic gates they contain every year or two, so they are now large enough for the idea of simultaneous multiple NN module evolution to be tested. (In fact it is even possible that the programmable board we are using currently may be big enough to allow two NN modules to be evolved simultaneously.)

The big question is "how"? The remainder of this paper attempts to give some tentative ideas to this tantalizing question. In fact, section 2 gives an outline of some ideas on how to proceed. Section 3 shows how two modules of a 2-module subsystem could be evolved individually. Section 4 shows how they could be evolved simultaneously. Section 5 discusses some points and questions arising from simultaneous multi-module evolution.

II. INITIAL IDEAS

Before starting the discussion we need to explain how we evolve currently an individual module in the accelerator board (from the British company "Celoxica", by the way [1]). The C-like language whose compilation the Celoxica board accepts is called "Handel-C" [4], based on the name of the composer. This language is about 80% ordinary C, with the remaining statements concerned mostly with the parallelism that the board can handle.

A user of the board writes Handel-C code to perform a genetic algorithm (GA) that is applied to the evolution of the weights of a single fully connected neural net module. The fitness definition of the module is part of the code. The module is then evolved at electronic speed, and the elite chromosome (coding the weights of the NN module) is then downloaded into the RAM memory of the PC, to be later connected to other such downloaded NN modules, to create an artificial brain, consisting of thousands of such NN modules. The artificial brain can then be used to control such devices as autonomous robots and such systems as speech recognition, and artificial vision, etc. There are many possible applications.

To aid thinking about how to evolve multiple-modules simultaneously, we restrict ourselves to the simplest case, i.e. in attempting to evolve only *two* such modules simultaneously.

However, before talking about how to do this, we talk about how the two modules could be evolved independently, and in the then talk about how they could be evolved simultaneously. Even simultaneous evolution can be done in several ways. One is called "independent simultaneous evolution" and the other is called "dependent simultaneous evolution".

Before launching into the details of the evolution of the 2 modules (whether evolved individually or simultaneously) we define the differences between the two types of *simultaneous evolution*.

a) 2-module, Simultaneous Independent Evolution (SIE)

One is the very simple idea of evolving two modules independently on the same board. This is a "no-brainer", since it consists of simply having two independent NN module evolutions going on at the same time in the board. This is not an interesting case conceptually speaking. Of course, it speeds up the evolution when one can evolve two modules at once, compared to doing one at a time, but this is not what this paper is aiming at. This brings us to the second approach.

b) 2-module Simultaneous Dependent Evolution (SDE)

In this second case, two modules are evolved at the same time, but they are not independent, i.e. they are connected, with the outputs of one connecting to the inputs of another, and both NN modules evolving at the same time. The evolution of one module influences the behavior of the other module. One could also have a mutual influence, where the output signals of the second module feed back to the first, in a non-feed-forward manner. Hence we can make a further distinction between

a) feed-forward multi-module simultaneous evolution
b) feed-back multi-module simultaneous evolution

In this paper, to keep things simple, we limit ourselves to a simple feed-forward 2-module simultaneous evolution.Take two NN modules that are connected to form a 2-module neural subsystem, i.e. both are needed to allow the subsystem to perform its function.

To keep the discussion concrete, imagine the two modules are components of a neural subsystem that detects a visual pattern and responds by pressing the left lever or the right lever of a device, depending on which of two different patterns it saw, and on the values of two switching input control signals.

III. MULTI-MODULE EVOLUTION

More specifically, let module A be a detector of either a triangle or a square in its visual field. It outputs two signals Stri (i.e. triangle) and Ssq (i.e. square). If the pattern presented to the subsystem is a triangle/square, the output signal Stri should be strong/weak, and the signal Ssq should be weak/strong correspondingly. These two output signals are fed into a module B which uses these two (now input) signals to help decide which lever to press, the left one or the right one, depending also upon two further switching control signals Sthi (i.e. triangle high) and Sshi (i.e. square high).

These two signals tell Module B that if a triangle is shone onto the grid, then either the left lever or the right lever should be pressed. Module B has two output signals Sleft, and Sright. If Sleft is strong, and a triangle is shone on the grid, then the left lever will be pressed. If Sright is strong and a triangle is shown on the grid, then the right lever will be pressed.

The two "switching" input signals, in effect, shunt or divert the Stri and Ssq signals to the Sleft and Sright or to the Sright and Sleft output signals. If these two switching signals did not exist, then the input signals to module B, i.e. Stri and Ssq could be connected directly to the output signals Sleft and Sright of module B, respectively. These shuntings can be summarized in symbolic form, as follows:

IF (triangle) & (Sthi = high) & (Sshi = low)
THEN (Sleft = Stri) & (Sright = Ssq)

IF (triangle) & (Sshi = low) & (Sshi = high)
THEN (Sleft = Ssq) & (Sright = Stri)

IF (square) & (Sthi = high) & (Sshi = low)
THEN (Sleft = Ssq) & (Sright = Ssq)

IF (square) & (Sshi = low) & (Sshi = high)
THEN (Sleft = Stri) & (Sright = Ssq)

Now that the function of the 2-module neural subsystem is fairly clear, again, to aid thinking, the evolution of these two modules is now discussed, and on the assumption that each module is evolved independently.

We need to know what the fitness function of each neural net module is, and its neural architecture. These details are discussed now, and will carry over to a large extent, when we discuss the simultaneous evolution of both modules.

Module A: The Pattern Detector Module

Assume this detector module receives input signals from the output signals of an 8*8 pixel grid of photocells, so that if an image of a triangle is shone on it, those photocells positioned inside the triangle of light will emit a strong output signal to the detector module A.

Let us assume that every photocell output signal S_{ij} (where *i*, *j* ranges over 0 to7) is sent to every neuron in the module A, which in turn is assumed to be a fully connected neural network of 16 neurons. This implies a rather large number of connections between the photocells and the neurons, namely 8*8*16 = 1024. We assume for reasons of generality, that each grid-neuron connection is given its own neural weighting factor W_{ijk}, where the *i, j* identify the photocell pixel, and the *k* is the neuron number in the net. We assume that the neural network is fully connected, i.e. each neuron has a connection

with every other neuron, including itself. Hence a fully connected neural net of N neurons will have N*N connections and hence N*N weights. We choose to have N=16 (which divides 64, the number of pixels in the grid). Hence, there will be 16*16 = 256 such weights. We assume also that two "output neurons" in the network exist which provide the two output signals of module A, i.e. the signals Stri and Ssq. We assume each of these output neurons is connected to all 16 neurons of the neural net module, and that each connection has a weighting factor W_{rs}, where r = 0,1, and s = 0 to 15. So the total number of connections/weights in this module = 8*8*16 + 16*16 + 2*16 = 1024 + 256 +32 = 1312. If we assume that each weight is a signed binary fraction, with one bit for the sign (+/-), and 9 bits for the binary fraction, the total length of the bit string chromosome for neural net module A is 13120, a long chromosome. But it is a very general architecture for pattern recognition tasks. If the evolution of the module "decides" that certain pixel output signals should go only to a particular neuron in the module, then the weights of the connections from those pixels that go to other neurons will be driven down to zero, i.e. corresponding to "no connection". The above architecture is very general and very versatile. It is also has very long chromosomes, but manageable with today's electronic capacities.

To evolve module A independently, we assume the following target or desired output signals. Let the triangle be shone onto the grid for 50 "ticks" (i.e. time units, where 1 "tick" is the time taken for all the neurons in the network to calculate their outputs from their inputs). Then let the square be shone onto the grid for a further 50 "ticks". When the triangle is shone, the target output is to be a strong signal on Stri, i.e. 0.8, and a weak signal on Ssq, i.e. 0.2

When the square is shone, the target output is to be a weak signal on Stri, i.e. 0.2, and a strong signal on Ssq, i.e. 0.8 The fitness function is then the inverse of the sum of the squares of the differences between the actual outputs of the two signals over the 100 ticks, and the target signals. This will be quadruple sum where the target value will be 0.8 for Stri in the triangle case, for the first 50 ticks, and 0.2 for the second 50 ticks, and 0.2 for Ssq in the square case for the first 50 ticks, and 0.8 for the second 50 ticks. Based on our group's evolutionary experience, this module A should evolve successfully.

Module B: The Switching Module

We now specify the function, architecture and fitness definition of the switching module B, which has 4 inputs and 2 outputs, i.e. the 2 outputs from the detector module A, and two switching control signals, Sthi and Sshi (i.e. "triangle high", and "square high"). Its 2 outputs, Sleft and Sright, control the lever pressing. The function, i.e. the behavior, of this module is as follows. Its function is essentially to switch or shunt the input signals from module A to the outputs of module B in various ways.

For example, if a triangle is shone on the grid, and the signal on Sthi is strong and the signal on Sshi is weak, then the input signal Stri is shunted to Sleft with as little change in signal strength as possible, and Ssq is shunted to Sright. In other words, if a triangle is shone on the grid and Sthi is strong, then Stri goes straight through to Sleft. If a triangle is shone on the grid and Sshi is strong, Stri gets diverted into Sright, and correspondingly for Ssq. If a square is shone on the grid, the outputs of module B are reversed compared to the triangle case.

Thus we have two separate modules, one that detects a pattern, and the other that switches signals depending on the pattern detected and its control signals. We expect fully that we can evolve these two neural net modules independently and then combine them into a 2-module neural subsystem that will function as desired.

The interesting question is how would it evolve if both modules were to be *evolved simultaneously*? Before discussing this question further, we continue with the fitness definition and architecture of the switching module B.

Since its function is rather simple, i.e. merely shunting signals to different outputs, we think that the number of neurons needed to perform this task is less than needed to perform the more complex pattern recognition task, so let NB, the number of neurons for this module be 8.

How to connect the 2 input signals from module A to the neurons of module B? Let us provide a general solution, i.e. specify that there are two input neurons NB1 and NB2 in module B that receive the outputs from module A, i.e. Stri goes to NB1 and Ssq goes to NB2. NB1 and NB2 then connect to all 8 neurons of module B. These 8 neurons are fully connected.

There are two input neurons for the control signals as well, and they too connect to all of the 8 neurons. There will also be 2 output neurons of module B, creating the 2 output signals Sleft and Sright. Hence the total number of neurons in module B will be 2+2+2+8 = 14 (i.e. 2 input neurons for Stri and Ssq, 2 input neurons for Sthi and Sshi, and 2 output neurons for creating Sleft and Sright. The number of weights will be 8*8 + 2*8 + 2*8 + 2*8 = 112 (i.e. the 8*8 for the fully connected neural net of module B, 8 weights for each input neuron of module B to each of the 8 neurons of the fully connected module B, similarly for the two control input neurons, and the two output neurons. Hence the length of the bit string chromosome to evolve module B independently would be 1120 bits, much smaller than the 13,120 bits of the detector module A.

Now that we know the architecture of the switching module B, we can begin to define its fitness function. To evolve module B independently, we need a set of input "training vector" signals, that we present to the module, say for 50 ticks each, sequentially, clearing out the internal neural signals between presentations of the input pair of values. More concretely, imagine we input 100 pairs of input signals (Stri, Ssq), where each independently takes a value that ranges from 0.05 to 0.95 in steps of 0.10 So we have 100 input pairs. But we also have two control pairs of signals, so in total we have 200 cases. We give each case 50 ticks of input time, clearing

out the internal signals between cases. We can incorporate these 200 cases into the fitness definition of module B.

It will be the inverse of the sum of 200 summation terms corresponding to the combination of two control input pairs of (Sthi = high, Sshi = low) and vice versa, and the 100 pairs of input values (ranging over 0.05 to 0.95). Each of the 200 combinations of the 4 input values lasts 50 ticks, thus each bit string chromosome per generation takes 200*50 = 10,000 ticks to measure its fitness.

When Sthi/Sshi is high/low (i.e. a strong/weak signal) we want the output signal Sleft to be as close as possible to the input signal Stri from module A to module B, and correspondingly for Ssq. Let us present the 100 input signal pairs with Sthi high, and then again with Sthi low. There will be 400 sum terms in the fitness definition, 200 for each output signal Sleft and Sright.

For the 100 input combinations (Stri, Ssq), we want Sleft to be as close as possible to the target (desired) value of Stri, when Sthi/Sshi is high/low. So we will have a summation term for each of the 100 input pair combinations, over 50 ticks. This term will look like

$$\sum_{t=1}^{50}[\mathrm{Sleft}(t) - \mathrm{Stri}(t)]^2$$

where t is the tick number. Similarly for S_{right}, when S_{thi}/S_{shi} is high/low, i.e.

$$\sum_{t=1}^{50}[\mathrm{Sright}(t) - \mathrm{Ssq}(t)]^2$$

After 100*50 = 5000 ticks, the two control signals Sthi and Sshi change from high/low to low/high (e.g. from 0.8/0.2 to 0.2/0.8), so the sum terms become

$$\sum_{t=1}^{50}[\mathrm{Sleft}(t) - \mathrm{Ssq}(t)]^2$$

and

$$\sum_{t=1}^{50}[\mathrm{Sright}(t) - \mathrm{Stri}(t)]^2$$

Since there are more training combinations for module B, it is likely that its evolution will take longer. This completes the description of the independent evolution of the two modules.

IV. SIMULTANEOUS EVOLUTION

We now turn our attention to the evolution of both modules *simultaneously*, treating the 2-module subsystem as a black box, with the 64 inputs from the grid and the 2 control inputs (Sthi/Sshi) and the two output signals (Sleft/Sright). The two internal signals (Stri/Ssq) are hidden and are not taken into account this time.

In this case, there will be 4 input combinations, i.e. when the triangle or the square is presented on the grid, and when the control signals (Sthi/Sshi) are (high/low) or (low/high).

When the triangle is presented to the grid, we want the output on Sleft to be high, e.g. 0.8 (i.e. a fixed target value) and Sright to be low (0.2). Correspondingly, when the square is presented on the grid, i.e. we want the output on Sright to be high (0.8) and Sleft to be low (0.2).

We can shine the triangle on the grid for 50 ticks, and then the square for 50 ticks, measuring the outputs at Sleft and Sright at each tick.

The fitness definition will be the inverse of the sum of 8 sum terms, each over 50 ticks. When the triangle is presented at the grid and Sthi/Sshi is high/low, we want the corresponding sum term to be

$$\sum_{t=1}^{50}[\mathrm{Sleft}(t) - 0.8]^2$$

and similarly for Sright

$$\sum_{t=1}^{50}[\mathrm{Sright}(t) - 0.2]^2$$

When the triangle is presented at the grid and Sthi/Sshi is low/high, we want the corresponding sum term to be

$$\sum_{t=1}^{50}[\mathrm{Sleft}(t) - 0.2]^2$$

and similarly for Sright

$$\sum_{t=1}^{50}[\mathrm{Sright}(t) - 0.8]^2$$

When the square is presented at the grid and Sthi/Sshi is high/low, we want the corresponding sum term to be

$$\sum_{t=1}^{50}[\mathrm{Sleft}(t) - 0.2]^2$$

and similarly for Sright

$$\sum_{t=1}^{50}[\mathrm{Sright}(t) - 0.8]^2$$

When the square is presented at the grid and Sthi/Sshi is low/high, we want the corresponding sum term to be

$$\sum_{t=1}^{50}[\text{Sleft}(t) - 0.8]^2$$

and similarly for Sright

$$\sum_{t=1}^{50}[\text{Sright}(t) - 0.2]^2$$

To evolve both modules simultaneously, we simply concatenate the two bitstring chromosomes for the two modules to make one longer bitstring chromosome to evolve the 2-module subsystem considered as a unit.

V. DISCUSSION

This paper is an initial attempt to answer the question as to whether it is possible to evolve several neural net modules that comprise a useful integrated neural subsystem, simultaneously. Being honest, our research group has not really bothered to consider the question seriously before, for several reasons, the main one being that we were not forced to. Until recently, the maximum number of interconnected, evolved neural net modules in an artificial brain that could be neurally signaled in real time was about 10,000 to 20,000. At the time (only a year or two back) we thought that that number could be reached reasonably by a sizable human team of BAs (brain architects) of say 20 people working for a few years. Of course, we extrapolated up the Moore's Law curve and anticipated that we would soon run into a situation of 50,000 modules, then 100,000, but it had not yet happened so we forgot about it. Now it is real, so if we want to take advantage of it, then we need to confront the fact that we seem to have "more electronics than ideas". The bottleneck now is not the lack of electronic capacities, but the lack of an approach on how to evolve many modules at once. So the motivation level to solve this problem increased to the point of writing this paper. We are now hoping that the experience we gain in working with the ideas of this paper in which we evolve only 2 modules at once, will help us evolve 3 modules at once, then 4, etc.

There are many remarks we can make and questions we can pose regarding multi-module simultaneous evolution. For example, how will the evolvability of the larger system compare with that of the two smaller systems? For example, will :-

$$ET(A) + ET(B) > ET(A+B)$$

where ET(X) is the evolution time of module X?

We are fairly sure the evolution of the two separate modules A and B will be successful. (Actually, since this paper was written just before the submission deadline, the experimental coding and testing is yet to be done, but that can now proceed quickly, given that the specifications are now clear). Will the 2-module (A+B, or alternatively, D+S (i.e. detector and switcher module)) evolution be quicker than the time taken for D and S to evolve separately? Is it possible that the 2-module system will not evolve at all, whereas the two modules separately will?

Another possibility, is that since the size of the search space of the simultaneous evolution is hugely greater than that of the two separate search spaces of the individual evolutions, then perhaps the 2-module evolution will find more interesting and surprising solutions. One thing that experience has taught us is that when one tries new things, one should expect to be surprised.

Another consideration is that the natural biological world has been doing simultaneous multi-module evolution for millions of years. The mammalian brain for example has been evolving as a whole for a long time, with each component of the brain evolving along with other components, and interacting with them. So, if nature can do it, can we, as brain builders, do the same?

Another question relates to how we could compare the evolution times of the two cases, i.e. evolving S+D simultaneously, compared with evolving S and D separately. How to define the stopping condition of each evolution, so that a comparison is meaningful?

REFERENCES

[1] Celoxica 2006, www.celoxica.com

[2] Hugo de Garis, Michael Korkin, "THE CAM-BRAIN MACHINE (CBM) An FPGA Based Hardware Tool which Evolves a 1000 Neuron Net Circuit Module in Seconds and Updates a 75 Million Neuron Artificial Brain for Real Time Robot Control", *Neurocomputing* journal, Elsevier, Vol. 42, Issue 1-4, February, 2002. Special issue on Evolutionary Neural Systems, guest editor: Prof. Hugo de Garis. Downloadable at http://www.iss.whu.edu.cn/degaris/papers

[3] Hugo de Garis http://www.iss.whu.edu.cn/degaris/coursestaught.htm Click on the CS7910 course.

[4] Handel-C 2006, www.celoxica.com