FPGA Based Implementation of Hysteresis Neural Network

Ryota Aoki, Toshiya Nakaguchi, Tsuyoshi Otake and Mamoru Tanaka

Sophia University
7-1 kioi-cho, chiyoda-ku, Tokyo, Japan
Phone:+81-3-3238-3878, Fax:+81-3-3238-3321
Email: ryo@mamoru.ee.sophia.ac.jp

Abstract— In this paper, we focus on an analog and digital mixed implementation, called 'Hybrid Architecture', for Hysteresis Neural Modules. Our proposed architecture makes connection weights of the modules software programmable by using FPGA devices. In order to investigate our proposed architecture, we apply them to quantizers, four color problems and N-Queens problems.

1. Introduction

Hysteresis Neural Networks(HNNs) proposed in 1991[1] are one of the Artificial Neural Networks. The mathematical model of the HNNs have been applied to various combinatorial optimization problems[2, 3].

Investigating the efficiency of parallel processing of the HNN, the first hysteresis neural modules(HNMs) have been implemented with discrete components[4]. But the first HNMs can solve only N-Queens problems because of their non-flexibility. In the HNN, constraints of the target problem are expressed by connection weights. Therefore, if connection weights can be reconfigure flexibly, we can apply them to various combinatorial optimization problems. In order to solve various problems, an Universal HNM(UHNM) which has five kinds of flexibility, output voltages, input voltages and currents, threshold voltages, time constants and connection weights has been proposed[5]. In this system, resistors are used for connection weights. Though the UHNM can apply and solve various combinatorial optimization problems, it is serious labor to reconfigure connection components manually one by one. So, it is necessary to make them software programmable.

In this research, we focus on an analog and digital mixed implementation, called 'Hybrid Architecture'. It is composed of two parts, a neuron part designed by analog circuits and a synapse part designed by digital circuits. Since we have confirmed that connection weights of the HNN can be expressed by only logical gate components, we propose a novel architecture that makes them software programmable by using an FPGA (Field Programmable Gate Array) device.

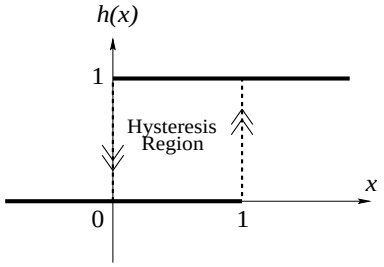


Figure 1: A normalized piecewise linear hysteresis. $h(x_i)$ is switched from 0 to 1 if x_i reaches the right threshold 1, and vice versa.

We investigate the efficiency of parallel computing of the FPGA synapse architecture by applying them to quantizers, four color problems and N-Queens problems.

2. Hysteresis Neural Network (HNN)

Normalized general state equations of the Hysteresis Neural Network can be described as follows,

$$\begin{cases} \lambda_i \frac{d}{d\tau} x_i &= -x_i - \sum_{j=1}^{N}\sum_{k=1}^{N} w_{ijk} y_j y_k + d_i, \\ y_i &= h(x_i) \qquad\qquad (i = 1, \cdots, N), \\ &= \begin{cases} 1 & \text{if} \quad x_i \geq 0, \\ 0 & \text{if} \quad x_i \leq 1, \end{cases} \end{cases} \tag{1}$$

where $\tau \in R$ is time, $x_i \in R$ is an internal state, $y_i \in \{0, 1\}$ is an output, $\lambda_i > 0$ is a time constant, $d_i \in R$ is an external input and $w_{ijk} \in R$ represents weight value between cell i and the product of the outputs of cell j and cell k. Especially at $j = k$, w_{ijk} represents weight value between cell i and cell j. $h(x_i)$ is a normalized piecewise linear hysteresis as shown in Figure 1. $h(x_i)$ is switched from 0 to 1 if x_i reaches the right threshold 1, and vice versa. We call the region between $0 < x < 1$ in this figure, 'Hysteresis Region'.

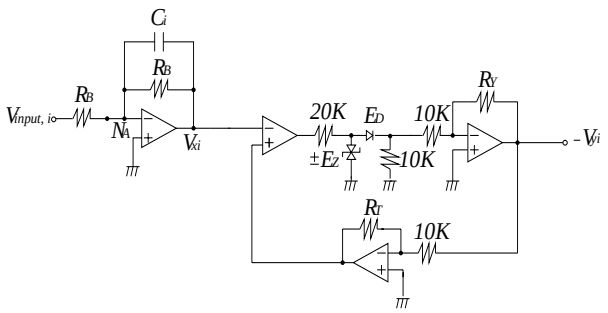


Figure 2: An implementation example of the UHNM. It consists of 4 OP-amps.

3. Universal Hysteresis Neural Module (UHNM)

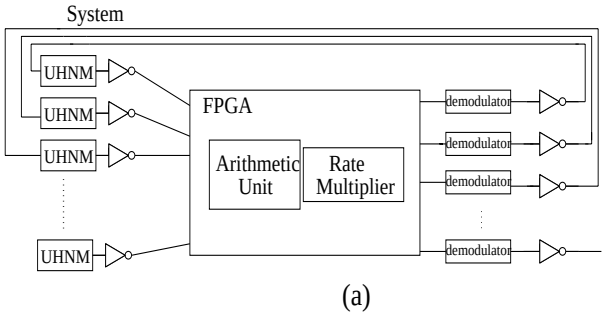
An Universal Hysteresis Neural Module which has five kinds of freedom, output voltages, input voltages and currents, threshold voltages, time constants and connection weights has proposed. An implementation example of the UHNM is shown in Figure 2. Applying KCL (Kirchhoff's Current Law) at node N_A, we obtain the circuit equations as

$$\left\{ \begin{array}{rcl} R_B C_i \frac{dV_{x_i}}{dt} & = & -V_{x_i} + V_{input_i}, \\ V_{y_i} & = & H(V_{x_i}), \\ & = & \left\{ \begin{array}{rl} E_A \frac{R_Y}{10K} & \text{if} \quad v_i \leq \frac{E_A R_Y R_T}{10K\,10K}, \\ 0 & \text{if} \quad v_i \geq 0, \end{array} \right. \\ E_A & = & E_Z - E_D, \qquad (i = 1, \cdots, N), \end{array} \right. \tag{2}$$

where V_{x_i} is a capacitor voltage which corresponds to an internal state, t is time, N is the number of cells, V_{y_i} is a voltage of which corresponds to a cell's output, E_Z is a zener voltage of reversely connected zener diodes, E_D is a forward biased threshold voltage of diode and V_{input_i} is a sum total of input voltage including an external input.

4. Hybrid Architecture

In this paper, we propose an analog and digital mixed system, called 'Hybrid Architecture'. It is composed of two parts, a neuron part designed by analog circuits and a synapse part designed by digital circuits. In the previous work, because we used discrete components, resistors and wires to express connection weights, we couldn't make connection weights software-programmable. But, if connection weights can be expressed by only logical gate components, it is possible to make them software-programmable by using a digital integrated device like an FPGA (Field Programmable Gate Array). This architecture has some advantages. One is that various synapse circuits can be re-written using the FPGA device. Another is that mathematical multiplying operations between outputs can be easily implemented.



3-bit rate multiplier

weight	states 0	1	2	3	4	5	6	7
000								
001					1			
010			1				1	
011			1		1		1	
100		1		1		1		1
101		1		1	1	1		1
110		1	1	1		1	1	1
111		1	1	1	1	1	1	1

(b)

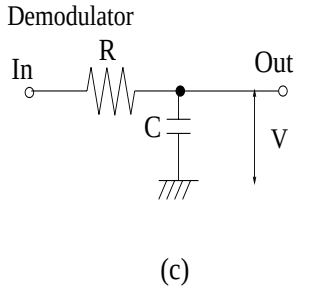


Figure 3: (a) The system of Hybrid Architecture. It consists of UHNMs, an FPGA and demodulators. (b) To illustrate the function of a rate multiplier, the multiplication table for 3-bit case is shown. (c) This CR integrator circuit is the demodulate circuit.

Figure 3(a) shows the proposed system. Summations of outputs are calculated by arithmetic unit in the FPGA device. Then, by using a rate multiplier, PDM(Pulse Density Modulation) signals are generated following to weight value w_{ijk}. Actually, a 4-bit rate multiplier is used in this system. The behavior of a rate multiplier is illustrated in Figure 3(b) using a 3-bit case for brevity. A rate multiplier is a counter and its state transits to the next state following to the input signal. Each binary bit of a given weight specifies at which states the output pulses are generated. When the LSB is on, an output pulse is generated at the fourth state. When the second bit is on, output pulses are generated at the second and at the sixth states. When the MSB is on, they are generated at all of the odd states. Figure 3(c) shows a simple demodulator which demodulates the PDM signal to analog voltage. Because our UHNM outputs inverted voltage and inputs inverted voltage, level converters which are actually only inverters are required for the interface between UHNMs and the FPGA.

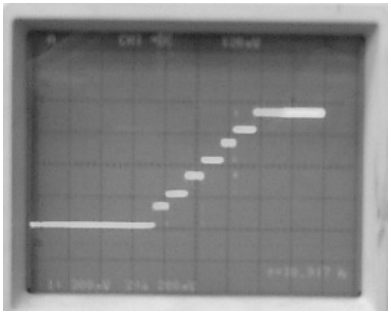


Figure 4: This waveform is an output state on 3-bit HNN quantizer. As an input signal, triangular wave is used. Its frequency is 10Hz and its amplitude is ±15 voltage.

5. Basic experimentation: Quantizer

5.1. 3-bit Quantizer

In this paper, in order to investigate our proposing architecture, at first we apply this system to 3-bit quantizer. The system becomes stable and gives an output of the quantizer when the internal states of all cells reach their equilibrium points. This system can be described as follows,

$$\begin{cases} \lambda_i \frac{d}{dt} x_i &= -x_i - \sum_{j=1}^{3} w_j y_j + d_i, \\ y_i &= h(x_i) \qquad (i = 1, 2, 3), \\ w_i &= 2^{i-1}, \\ \lambda_{i+1} &> \lambda_i, \end{cases} \tag{3}$$

where d, λ_i and w_i are inputs, time constants and connection weights respectively.

5.2. Experimental results

We implement a 3-bit quantizer using three UHNMs and the FPGA which has 208 I/O pins and 200,000 gates and is named spartanII by xilinx. Here, it is necessary to set up connection weights as the following ratios, {1,2,4}. The connection weights are expressed not by the ratio of resistances but by the ratio of frequencies. As an input signal of the quantizer, a triangular wave is used. Its frequency is 10Hz and its amplitude is ±15 voltage. We set parameters of demodulator as follows, R=10K Ω, C=100pF. Total equivalent gate count for the design of 3-bit quantizer is 202.

Figure 4 shows results of 3-bit quantizer. This system doesn't work with high speed. But, the purpose of this experimentation is to confirm exact operation of our hybrid architecture. From this experimental result, 3-bit quantizer with UHNMs and proposing hybrid architecture is effectively working.

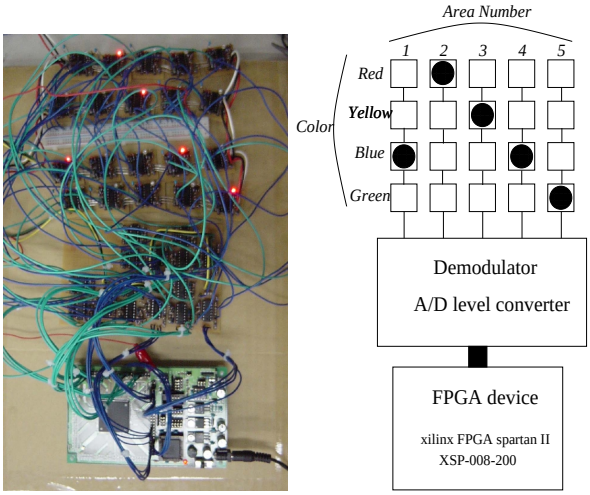


Figure 5: A top view of the four color problem solver. It consists of 20 UHNMs, the FPGA device and demodulators. The system becomes stable and one feasible solution is obtained.

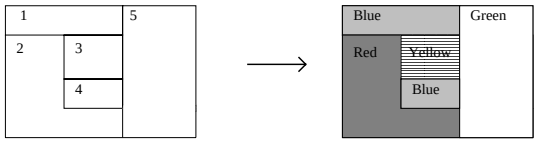


Figure 6: An example plane map for experimentations of this research. Right figure is one feasible solution.

6. Application: Four Color Problems

6.1. Numerical Model

Next, we focus on four color problems. Four color problem is a problem that if four colors are given, it can be colored that it is not the same color as the adjoined areas with a plane map.

To apply the HNN to four color problems, we have to construct a system including constrains to become stable only when all constraints are satisfied. Equations of the HNN for the problem with N areas can be given as follows,

$$\begin{cases} \lambda_{ij} \dot{x}_{ij} &= -x_{ij} + p, \\ p &= (1 - \sum_{k=1}^{4} y_{kj}) - y_{ij} \sum_{k=1}^{N} A_{ik} y_{kj} + d_{ij}, \\ y_{ij} &= h(x_{ij}) \end{cases} \tag{4}$$

where A_{ij} is the information that which areas are adjoined. If i-th area and j-th area are adjoined, A_{ij} is 1. Otherwise A_{ij} is 0.

6.2. Experimental Results

We give an example plane map for the experiment as shown in Figure 6.

Figure 5 shows a top view of the four color problem solver. And this hardware solver consists of 20 UHNMs, the FPGA device and demodulators. In this problem, 4-bit rate multiplier is used and we set parameters of the demodulator as follows, R=10KΩ, C=100pF. The demodulator consists of demodulate circuits and A/D level converter which interface to connect UHNMs with an FPGA. Total equivalent gate count for the design is 2,691. In Figure 5, the system becomes stable and one feasible solution is obtained.

7. Application: N-Queens Problem

7.1. Numerical Model

Finally, we focus on N-Queens Problems. In the N-Queens problems, N chess queens must be placed on a square chessboard composed of N rows and N columns, in such a way that they do not attack each other. Considering the constraints of N-Queens problem, following equation of the HNN can be described,

$$\begin{aligned}\lambda_{ij}\dot{x_{ij}} &= -x_{ij} + (1 - \sum_{m=1}^{N} y_{im}) + (1 - \sum_{m=1}^{N} y_{mj}) \\ &\quad -y_{ij}(\sum_{1\le i-k,j-k\le N} y_{i-k,j-k} \\ &\quad + \sum_{1\le i-k,j+k\le N} y_{i-k,j+k}) + d_{ij}, \qquad (5)\end{aligned}$$

The system becomes stable when that corresponds with a solution of the problem and the output vector is always unstable in the other case.

7.2. Experimental Result

Figure 7 is a top view of 4-Queens problem solver. And this hardware solver consists of 16 UHNMs, the FPGA device and demodulators. In this problem, 4-bit rate multiplier is used and we set parameters of demodulator as follows, R=10KΩ and C=100pF. Total equivalent gate count for the design is 2,853. In Figure 7, the system becomes stable and one feasible solution is obtained.

8. Conclusions

In this paper, we propose a hybrid architecture for the UHNM. Our proposed architecture makes connection weights of the modules software programmable by using FPGA devices.

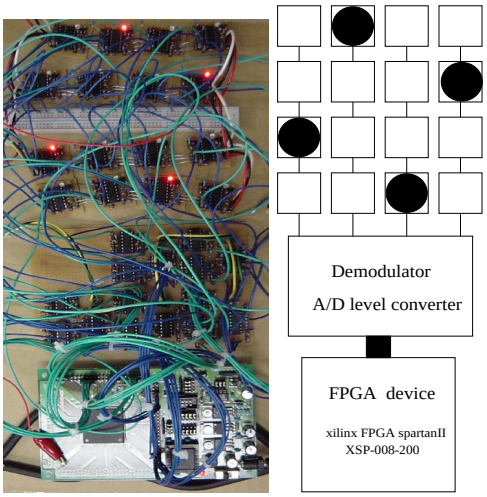


Figure 7: A top view of 4-Queens problem solver. It consists of 16 UHNMs, the FPGA device and demodulators. The system becomes stable and one feasible solution is obtained.

In order to investigate our proposing architecture, we apply them to quantizers, four color problems, N-Queens problems and we obtain an effective result.

In this research, we only checked operation of this system. In future problem, we will evaluate the performance of this system.

References

[1] T.Saito, M.Oikawa and K.Jin'no, "Dynamics of hysteretic neural networks," in Proc. of IEEE/ISCAS'91, pp.1469-1472, Singapore, 1991.5.

[2] T.Nakaguchi, K.Jin'no and M.Tanaka, "Hysteresis Neural Networks for N-Queens Problems," in IEICE Trans. Fundamentals, vol.E82-A, No.9, pp.1851-1854, 1999

[3] T.Nakaguchi, K.Jin'no and M.Tanaka, "Hysteresis Neural Networks for Solving Traveling Salesperson Problems," in Proc. of IEEE/ISCAS 2000, III-153, Geneva, Switzerland, 2000.5.

[4] T.Nakaguchi, K.Jin'no and M.Tanaka, "Hardware Combinatorial Optimization Problems Solver by Hysteresis Neural Networks," in Proc. of IEEE/ISCAS 2001, W09-Sky3-0.2, Sydney, Australia, 2001.5.

[5] R.Aoki, T.Nakaguchi, T.Otake and M.Tanaka, "A Implementation of Universal Hysteresis Neural Module" in Proc. of ECCTD'01, pp.III 409-412, Espoo, Finland, 2001.8.