pute the back-propagation error. There may be a processor associated with each layer or a specified cluster of neurons. These processors will also compute the necessary combinations of weights for the distributed arithmetic neurons and broadcast them to the dual ported memories. It may seem that calculating an average of 10 000–100 000 combinations at each new error update is a formidable task. It is indeed. But the network if desired may be trained off-line or simply modeled and the final weights loaded. Fortunately, there exist training methods which do not require an update at each pattern, but only at each epoch (or iteration) [8]. This method was chosen for the experiments.

## V. Conclusion

A hardware neural net was designed based on distributed arithmetic. The model used in the experiments was a three layer network with 10 neurons per layer at maximum (this was because of the 1K dual ported memories used). The numbers used were limited to 16 b, with an 8-b integer and 8-b fractional part. The LUT's were full length, but of the 16 b available for addressing at the linear output of the neuron (i.e., the sum output), only 13 were used—eight from the integer part and five from the fractional part. Because the sigmoid is absolute bounded by 1, only 8 b were used on the LUT output, simplifying the hardware and lowering the system cost. A TMS320E25 was preprogrammed as an error processor for the whole network. Recognition experiments were conducted, and while the correctness was slightly worse than when modeling was performed on an IBM AT with the floating point package from a high level language ("C"), a result was output roughly 48 cycles after a pattern was presented. At a clock rate of 10 Mhz that means one decision every 5 $\mu$s and this independently of the number of neurons in a layer as long as they are not shared.

## References

[1] D. Rumelhart and J. McClelland, *Parallel Distributed Processing.* Bradford Books, 1987.
[2] N. Botros, Z. Deiri, and I. Wattar, "Algorithm and instrumentation for isolated-word recognition" presented at the Math. Comput. Modelling Conf., England, 1990.
[3] M. Soucek, *Neural and Massively Parallel Computers.* 1989.
[4] *Proc. IEEE* (Special Issue on Neural Networks), Sept. 1990.
[5] *Proc. IEEE* (Special Issue on Neural Networks, Applications), Oct. 1990.
[6] C. Climaksaus, "Applications of neural nets," in *Neural Explorer User's Manual*, Neuralware Inc.
[7] M. Marchesi, G. Orlandi, F. Piazza, L. Pollonara, and A. Uncini, "Multilayer perceptrons with discrete weights," presented at the IEEE Neural Networks Conf., 1990.
[8] D. D. Cavaglia, M. Valle, and G. M. Bisio, "Effects of weight discretization on the backprop learning method: Algorithm, design, and hardware realization," presented at the IEEE Neural Networks Conf., 1990.
[9] D. Van den Bout *et al.*, "Scalable VLSI implementations for neural networks," *J. VLSI Signal Processing*, vol. 1, no. 4, Apr. 1990.
[10] A. Peled and B. Liu, "A new approach to the realization of nonrecursive digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, no. 6, Dec. 1973.
[11] F. J. Taylor, *Digital Filter Design Handbook.* New York: Marcel Dekker, 1983.
[12] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Mag.*, July 1989.
[13] V. Bochev "Hardware structures for real time digital processing of audio signals," Ph.D. dissertation, Technical University Sofia, 1990.
[14] D. J. Myers and R. A. Hutchinson, "Efficient implementation of piecewise linear activation function for digital VLSI neural network," *Electron. Lett.*, vol. 25, no. 24, Nov. 23, 1989.

# A Numerically Stable Pipeline Net VLSI Architecture for the Isomorphic Hopfield Model

Po-Rong Chang and Bao-Fuh Yeh

***Abstract*—This correspondence presents a reconfigurable pipeline net VLSI architecture for implementing Hopfield neural models. It is known that the Hopfield models involve computing the hyperbolic trigonometric functions which are hard to be realized by digital VLSI architectures. In order to tackle such difficulty, a useful isomorphic nonlinear mapping is introduced to convert those hyperbolic trigonometric nonlinear functions into the simple second-order polynomial functions. Moreover, the isomorphic formulation provides the higher ability to decompose the problem into several independent tasks which can be assigned to a number of processors. Handling the digital realizations on the Hopfield model, the previous attemps were to use the technique based on the first-order approximation called Euler's method which has poor numerical stability and large truncation error. To find a numerical solution with a prescribed accuracy, one of the promising approaches is a combination of both a single-step Runge–Kutta method and a multistep predictor-corrector method which has a larger stability interval and is particularly suitable for parallel computation. Since the mixed-type procedure requires data broadcasting, common VLSI architectures with fixed connections cannot offer such flexible connectivities. A pipeline net VLSI architecture which is a programmable two-level pipelined and dynamically reconfigurable systolic array would be adopted as the design platform. The pipelining period and block pipelining period of the proposed architecture have the computational orders of $O(1)$ and $O(n)$, respectively, where $n$ is the number of neurons.**

## I. Introduction

Artificial neural networks contain a large number of identical computing elements or neurons with specific interconnection strengths between neuron pairs [13]. The massively parallel processing power of neural network lies in the cooperation of highly interconnected computing elements. Analog VLSI implementations of the Hopfield network containing up to 512 neurons have been built with matrices of fixed resistors and nonlinear amplifiers fabricated on a single chip [6]. This task is made difficult by the large number of analog signals which much pass between chips and by the external parasitic capacitances which will distort the charging characteristics of the network and possibly cause erroneous results.

Due to the limitations of analog computing, digital simulation on neural networks would be the most promising approach to solve the difficulty. A lot of researchers [6], [1] apply the techniques based on the first-order approximation called Euler's method [10] to the simulation of neural networks. To find a numerical solution with a prescribed accuracy, it is recommended to use Ghoshal's fourth-order predictor-corrector multistep method [5], which is particularly suitable for parallel implementation. Unfortunately, this method is not self-starting. This is in contrast to the single-step Runge–Kutta methods where only the single initial condition is needed to start the computation, but may become weakly unstable in some unpredictable conditions. Hence the best way is to

Manuscript received January 22, 1992; revised November 15, 1992. The associate editor coordinating the review of this correspondence and approving it for publication was Prof. J. N. Hwang. This study was supported in part by the National Science Council, Republic of China, under Contract NSC 80-0404-E-009-77.

The authors are with the Department of Communication Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan, Republic of China.

IEEE Log Number 9207556.

use the five-point formula of Milne's Runge–Kutta method [12] to start the computations of Ghoshal's predictor–corrector method.

It is known that the Hopfield model involves computing the hyperbolic trigonometric functions which are hard to be realized by digital VLSI architectures. In order to tackle such difficulty, a useful isomorphic nonlinear mapping as discussed in Section II is proposed to convert those hyperbolic trigonometric nonlinear functions into the simple second-order polynomial functions. Moreover, the isomorphic formulation is specially suitable for parallel implementation. However, it could be shown that the time evolutions and the final solutions of both models are in the same results. Certainly, the mixed-type integration procedure is also suitable for evaluating the isomorphic model.

In this correspondence, the concept of a pipeline net architecture that is a programmable two-level pipelined and dynamically reconfigurable systolic array [9] would be adopted as our design principle. A reconfigurable pipeline net VLSI architecture including four processors, a $6 \times n$ shifter array and a programmable $12 \times 30$ routing network has been designed, based on the mixed-type integration procedure, for simulating the isomorphic Hopfield model. The pipelining period and block pipelining period of executing the integration algorithm on pipeline net architecture are characterized as $\alpha$ and $(n + 4)\alpha$, respectively, where $\alpha$ is the time required for performing a scalar addition and a scalar multiplication.

## II. The Hopfield Model with Isomorphic Mapping

### A. The Original Hopfield Model

The Hopfield model [7], [8] is a popular model of continuous, interconnected $n$ nodes. Each node is assigned a potential, $u_i(t)$, $i = 1, \cdots, n$ as its state variable. Each node receives external input $I_i(t)$, and internal inputs from other nodes in the form of a weighted sum of firing rates $\Sigma_j T_{ij} g_j(u_j)$, where $g_i(\cdot)$ is a monotonically increasing bounded function converting potential to firing rate. The equations of motion are

$$\frac{du_i}{dt} = \sum_{j=1}^{n} T_{ij} v_j + I_i$$
$$v_i = g_i(u_i) \qquad (1)$$

for $i = 1, 2, \cdots, n$. The dynamic model (1) is highly nonlinear, coupled, and of a high dimension for a large number of nodes. As such it is difficult to analyze, simulate, and synthesize. Due to the saturation of $g_i(u_i)$, an acceptable solution for the firing rates $v_i$ may involve very high values for the potentials $u_i$ implying a large computational dynamic range.

### B. The Isomorphic Model

In order to derive the governing equations of motion characterized by firing rate, a useful isomorphic nonlinear mapping $\psi(\cdot)$ may be defined on the space of potentials and described as

$$\boldsymbol{v} \equiv \psi(\boldsymbol{u}). \qquad (2)$$

For the sake of simplicity, one may let $\psi(\cdot)$ be $\boldsymbol{G}(\cdot)$. And, the $i$th term in $\boldsymbol{v}$ is a function of $u_i$ only $v_i = g_i(u_i)$, $i = 1, 2, \cdots, n$. The procedure of transforming the original model (1) into its isomorphic form could be described as

$$\frac{dv_i(t)}{dt} \frac{dg_i(u_i(t))}{dt} = l_i(v_i(t)) \cdot \left[\sum_{j=1}^{n} T_{ij} v_j(t) + I_i(t)\right] \qquad (3)$$

for $i = 1, 2, \cdots, n$. In vector form,

$$\dot{\boldsymbol{v}}(t) = \boldsymbol{L}(\boldsymbol{v}(t)) \cdot [\boldsymbol{T}\boldsymbol{v}(t) + \boldsymbol{I}]. \qquad (4)$$

It is shown that the model (3) is isomorphic to (1) since the mapping $g_i(u_i)$ is one to one and onto for all $i = 1, 2, \cdots, n$. According to (3), the rate of change of the firing rate for the $i$th node is proportional to its external input $I_i(t)$, to a linear combination of the firing rates of other nodes $\Sigma_{j=1}^{n} T_{ij} v_j$. The summed input is shunted by the nonlinear function $l_i(v_i)$, which is specially suitable for parallel implementation.

The sigmoid nonlinear function $g_i(u_i)$ in (1) can be identified as

$$v_i = g_i(u_i) = \tfrac{1}{2}[1 + \tanh(\lambda_i u_i)],$$
$$\text{for } i = 1, 2, \cdots, n \qquad (5)$$

where $\lambda_i > 0$ is the gain scaling parameter of the nonlinear function for the $i$th neuron. The explicit formulation of the nonlinear function for the isomorphic model (3) can be derived as

$$l_i(v_i) = 2\lambda_i \cdot v_i(1 - v_i) \qquad (6)$$

for $i = 1, 2, \cdots, n$. The sigmoid nonlinearity of (5) of the original Hopfield model has the term tanh, which is hard to be implemented. For the isomorphic model, it is shown that the nonlinear function could be converted to a simple second-order polynomial, which is quite easy to be realized by digital VLSI architectures.

## III. Numerical Methods for the Isomorphic Hopfield Model

The limitations of analog computing have led researchers of neural networks to rely upon digital simulation. It is known that the Hopfield-type neural model could be formulated as the following initial value problem (IVP):

$$\dot{\boldsymbol{v}} = \boldsymbol{L}(\lambda, \boldsymbol{v}(t)) \cdot [\boldsymbol{T}\boldsymbol{v}(t) + \boldsymbol{I}]$$
$$= \boldsymbol{F}(t, \lambda, \boldsymbol{v}(t)) \quad \text{and} \quad \boldsymbol{v}(t_0) = \boldsymbol{v}_0. \qquad (7)$$

In order to find a numerical solution with a prescribed accuracy, one of the commonly used numerical algorithms is the multistep method based on the predictor-corrector strategy [10] which has a larger stability interval and is particularly suitable for parallel computation. Ghoshal *et al.* [5] proposed that a fourth-order method employing a predictor and three correctors could be described as

$$P\colon\ \boldsymbol{v}^{[0]}_{k+3} = \boldsymbol{v}^{[3]}_{k-1} + h/3(8\boldsymbol{F}^{[0]}_{k+2} - 4\boldsymbol{F}^{[1]}_{k+1} + 8\boldsymbol{F}^{[2]}_{k})$$
$$\boldsymbol{F}^{[0]}_{k+3} = \boldsymbol{F}(t_{k+3}, \lambda, \boldsymbol{v}^{[0]}_{k+3})$$
$$C_1\colon\ \boldsymbol{v}^{[1]}_{k+2} = \boldsymbol{v}^{[3]}_{k-1} + h/8(3\boldsymbol{F}^{[0]}_{k+2} + 9\boldsymbol{F}^{[1]}_{k+1} + 9\boldsymbol{F}^{[2]}_{k} + 3\boldsymbol{F}^{[3]}_{k-1})$$
$$\boldsymbol{F}^{[1]}_{k+2} = \boldsymbol{F}(t_{k+2}, \lambda, \boldsymbol{v}^{[1]}_{k+2})$$
$$C_2\colon\ \boldsymbol{v}^{[2]}_{k+1} = \boldsymbol{v}^{[3]}_{k-1} + h/3(\boldsymbol{F}^{[1]}_{k+1} + 4\boldsymbol{F}^{[2]}_{k} + \boldsymbol{F}^{[3]}_{k-1})$$
$$\boldsymbol{F}^{[2]}_{k+1} = \boldsymbol{F}(t_{k+1}, \lambda, \boldsymbol{v}^{[2]}_{k+1})$$
$$C_3\colon\ \boldsymbol{v}^{[3]}_{k} = \boldsymbol{v}^{[3]}_{k-1} + h/24(9\boldsymbol{F}^{[2]}_{k} + 19\boldsymbol{F}^{[3]}_{k-1} - 5\boldsymbol{F}^{[3]}_{k-2} + \boldsymbol{F}^{[3]}_{k-3})$$
$$\boldsymbol{F}^{[3]}_{k} = \boldsymbol{F}(t_k, \lambda, \boldsymbol{v}^{[3]}_{k}). \qquad (8)$$

Note that the superscript $[i]$ means the result after the evaluation of the $i$th corrector, $1 \le i \le 3$, [0] means the predicted value, and [3] means the final corrected value, which can be output. The proposed algorithm could keep four processors busy, with each processor computing the predictor or one of the corrector formulas and then evaluating the function in terms of the predicted or corrected arguments.

One of the computational defects of the predictor-corrector methods is that the methods are not self-starting. This is in contrast to the single-step Runge–Kutta methods wbere only the single ini-

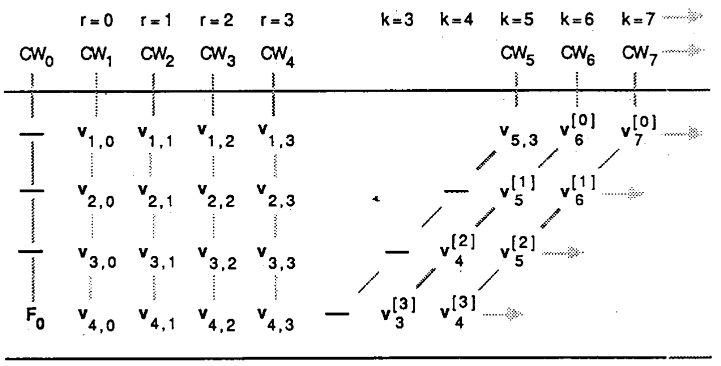


Fig. 1. The computational wavefront diagram of the proposed mixed-type integration method ESIIH. Note that the evaluations $F_{k,r} = F(t_k, \lambda, v_{k,r})$ and $F_k^{[i]} = F(t_k, \lambda, v_k^{[i]})$ are also performed after their associated $v_{k,r}$ and $v_k^{[i]}$ have been determined, respectively. $v_{k,r}$: an approximation to $v(t_k)$ of order $r + 1$ (or local error of $O(h^{r+2})$). $v_k^{[0]}$: the predicted value of $v(t_k)$. $v_k^{[i]}$: the $i$th corrected value of $v(t_k)$, $i = 1, 2, 3$. $r$: order index, $t_k$: the $k$th time instant. ——: No operation. $\mathrm{CW}_j$: the $j$th computational wavefront.

tial condition $v(t_0) = v_0$ is needed to start the computation. Unfortunately, [11] showed that the Runge–Kutta method will become weakly unstable in some unpredictable conditions. However, the five-point formula of the Milne's Runge-Kutta procedure (MRKP) is especially suitable for starting the Ghoshal's predictor-corrector method (GPCM). The procedures for the MRKP are as follows:

1) To begin, the following Euler formula below is used to predict solutions at four time steps as

$$v_{1,0} = v_0 + hF_0, \qquad v_{2,0} = v_0 + 2hF_0,$$
$$v_{3,0} = v_0 + 3hF_0, \qquad v_{4,0} = v_0 + 4hF_0.$$

2) The solution values can be iteratively improved using the following five-point formula [12] which is an order-improving recursion:

i) The first four approximations to $v(t_1)$, $v(t_2)$, $v(t_3)$, and $v(t_4)$ of order $m$:

For $r = 0$ step 1 until $(m - 2)$ do

$$\begin{aligned} v_{1,r+1} &= v_0 + \frac{h}{720}(251F_0 + 646F_{1,r} - 264F_{2,r} \\ &\quad + 106F_{3,r} - 19F_{4,r}) \\ F_{1,r+1} &= F(t_1, \lambda, v_{1,r+1}) \\ v_{2,r+1} &= v_0 + \frac{h}{90}(29F_0 + 124F_{1,r} + 24F_{2,r} + 4F_{3,r} - F_{4,r}) \\ F_{2,r+1} &= F(t_2, \lambda, v_{2,r+1}) \\ v_{3,r+1} &= v_0 + \frac{3h}{80}(9F_0 + 34F_{1,r} + 24F_{2,r} + 14F_{3,r} - F_{4,r}) \\ F_{3,r+1} &= F(t_3, \lambda, v_{3,r+1}) \\ v_{4,r+1} &= v_0 + \frac{2h}{45}(7F_0 + 32F_{1,r} + 12F_{2,r} + 32F_{3,r} + 7F_{4,r}) \\ F_{4,r+1} &= F(t_4, \lambda, v_{4,r+1}) \end{aligned} \tag{9}$$

End.

Each $v_{k,r+1}$ is an approximation $v(t_k)$ of order $r + 2$ (or local error of $O(h^{r+3})$), $1 \le k \le 4$.

ii) Observing the five-point formula [12], the approximation $v_{5,m-1}$ can be in terms of the first four approximations as

$$\begin{aligned} v_{5,m-1} &= v_0 + \frac{5h}{144}(19F_0 - 10F_{1,m-1} + 120F_{2,m-1} \\ &\quad - 70F_{3,m-1} + 85F_{4,m-1}) \\ F_{5,m-1} &= F(t_5, \lambda, v_{5,m-1}). \end{aligned} \tag{10}$$

The procedure for evaluating the proposed mixed-type integration method could be illustrated in Fig. 1. The $j$th computational wavefront $\mathrm{CW}_j$ represents the $j$th computational activities. At $\mathrm{CW}_0$, $F_0 = F(t_0, \lambda, v_0)$ is evaluated to provide the necessary information to the successive computations. The first-order approximations for the first four points $v(t_1)$, $v(t_2)$, $v(t_3)$, and $v(t_4)$ and their corresponding $F_{1,0}$, $F_{2,0}$, $F_{3,0}$, and $F_{4,0}$ are evaluated parallely along the computational wavefront of $\mathrm{CW}_1$. From $\mathrm{CW}_2$ to $\mathrm{CW}_5$, an order-improving procedure has been executed for evaluating the approximations $v_{k,r}$ to the first five points and their corresponding $F_{k,r}$, $1 \le k \le 5$ parallely along a particular wavefront. After the starting conditions have been determined, the computational activities (the evaluations of $v_k^{[i]}$ and $F_k^{[i]}$) of the GPCM along the 45° dash lines would sweep from $\mathrm{CW}_6$ to a time step in which the termination conditions of that the state in the isomorphic Hopfield system is near the thermodynamic ground state. The resultant approximations would be pumped out from the leftmost side of $k = 3$ to the right side along the fourth row of the computational wavefronts. The mixed-type integration procedure for the isomorphic Hopfield model can be described in [2].

Digital implementations of neural networks have been constructed to offer a nature, and well-understood technology, flexibility, scalability, and accuracy much better than those of analog implementations. In this correspondence, the concept of a pipeline net architecture which is a programmable two-level pipelined and dynamically reconfigurable systolic array [9] would be adopted as our design principle. Basically, a pipeline net is made of multiple functional pipelines, programmable routing or crossbar networks, and a set of data registers. Each functional pipeline is used to execute its assigned operations. The programmable routing networks are used to provide dynamic connecting paths among multiple functional pipelines and registers. Therefore, local connections necessary in a systolic array are no longer a structural constraint in

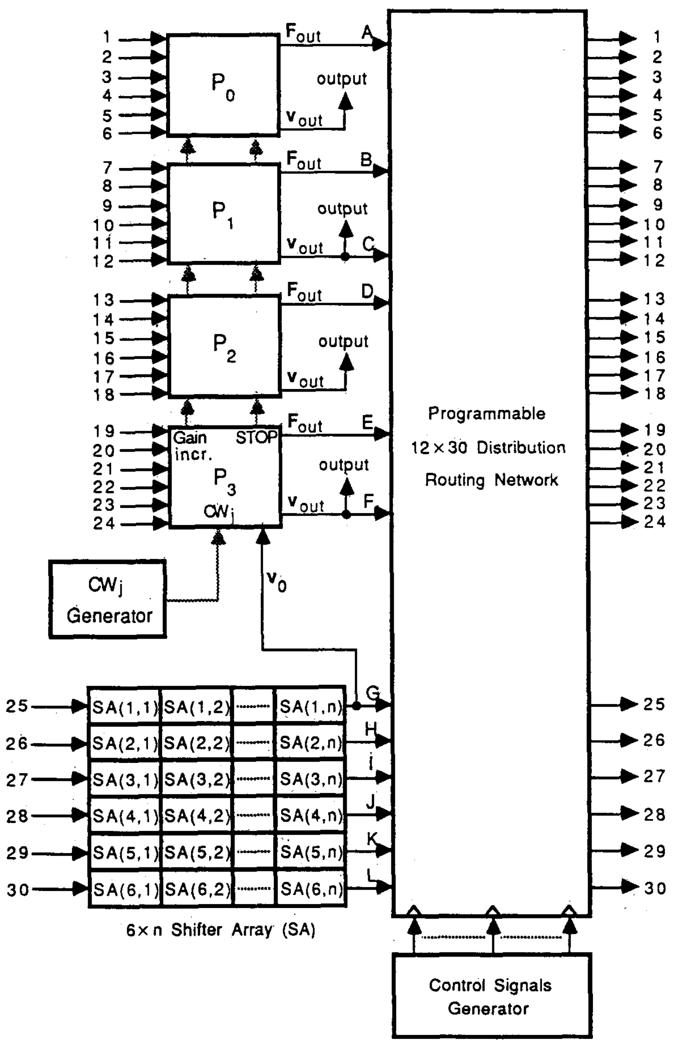


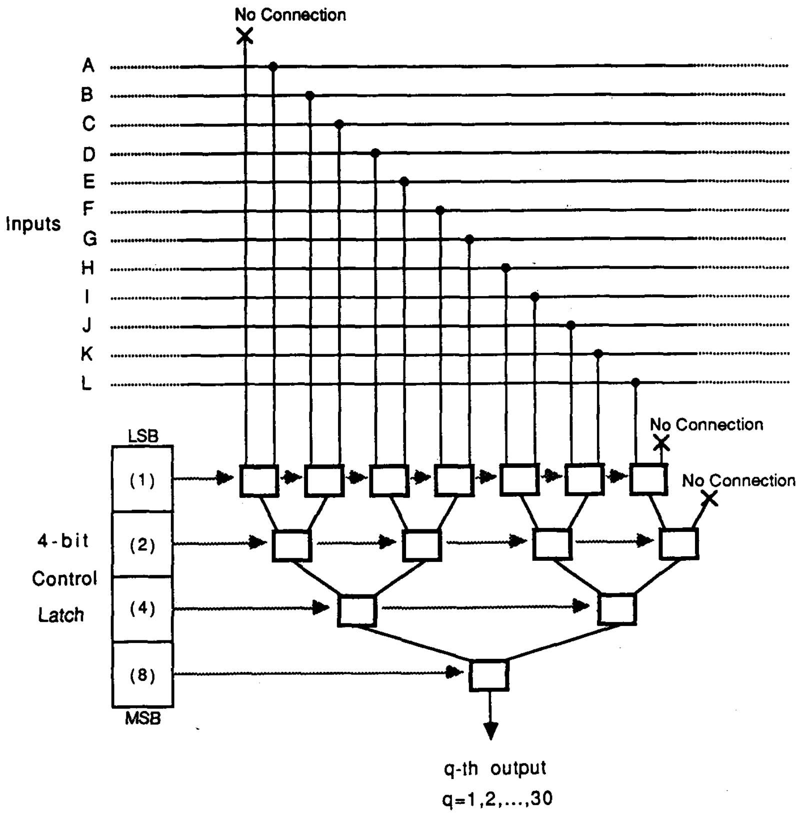
Fig. 2. The reconfigurable pipeline net architecture for computing the algorithm ESIIH.

a pipeline net. However, the systolic flow of data through the pipeline net is preserved. In a pipeline net, noncompute delay buffers can be inserted at any data path in order to handle the problem of delay matching.

The design of implementing the algorithm ESIIH, as illustrated in Fig. 2, is made of four two-dimensional functional pipelines, a programmable $12 \times 30$ distribution routing network, and a $6 \times n$ shifter array. Functional pipelines (or processors) $P_0$, $P_1$, $P_2$, and $P_3$ are assigned to deal with the operations of procedure GPCM, respectively. Meanwhile, the starting conditions should be generated by the MRKP before executing the GPCM. More details about executing MRKP on the pipeline net would be summarized in the procedure RPNAE and described in [2]. After the starting conditions have been set up, the pipeline net would start performing the GPCM and the desired solutions of $v_k^{[2]}$, $k \geq 3$ would be then pumped out from the output port $v_{\text{out}}$ of $P_3$ sequentially. A programmable $12 \times 30$ distribution routing network is applied to exchanging the information and data among the predictor and the correctors. It is possible to be realized by crossbar network [3]. However, [4] showed that each crosspoint contributes capacitance which limits the speed and size of the network. As shown in Fig. 3, the solution presented here consists of connecting the rows to each column with a tree structure as opposed to a direct connection. More details about the 3-$\mu$m CMOS VLSI implementations of the binary tree crossbar network are described in [4].

## IV. Conclusion

A reconfigurable pipeline net VLSI architecture has been designed based on the mixed-type integration procedure, for simulating the isomorphic Hopfield model. Due to the difficulty of realizing the Hopfield model, an isomorphic model is introduced to reduce the complexity of its digital implementations. To ensure the numerical solution with a prescribed accuracy, a mixed-type integration algorithm based on combining a single-step Runge–Kutta method and a multistep predictor-corrector method is applied to the digital simulation of an isomorphic model, resulting in numerical stability. A reconfigurable pipeline net VLSI architecture is proposed to implement the mixed-type integration algorithm. Basically, the architecture is made of four processors, a programma-



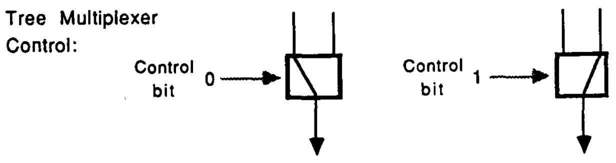


Fig. 3. The binary tree crossbar design of the programmable 12 × 30 distribution routing network.

ble 12 × 30 routing network, and a 6 × $n$ shifter array which are assigned to deal with the main operations of the integration algorithm, data routing, and synchronization, respectively. The pipelining period and block pipelining period of executing the integration algorithm on pipeline net architecture are characterized as $\alpha$ and $(n + 4)\alpha$, respectively.

## References

[1] D. E. Van den Bout and T. K. Miller, III, "A digital architecture employing stochasticism for the simulation of Hopfield neural nets," *IEEE Trans. Circuits Syst.*, vol. 36, no. 5, pp. 732–738, May 1989.

[2] P. R. Chang and K. S. Hwang, "A numerical stable VLSI architecture for digital simulation of neural networks," Tech. Rep. Dep. Commun. Eng., National Chiao-Tung University, Hsin-Chu, Taiwan, 1991.

[3] C. Y. Chin and K. Hwang, "Packet switching networks for multiprocessors and data flow computers," *IEEE Trans. Comput.*, vol. C-33, pp. 991–1003, Nov. 1984.

[4] M. Cooperman, A. Paige, and R. W. Sieber, "Broad-band video switching," *IEEE Commun. Mag.*, pp. 26–30, Dec. 1989.

[5] S. K. Ghoshal, M. Gupta, and V. Rajaraman, "A parallel multistep predictor-corrector algorithm for solving ordinary differential equations," *J. Parallel Distributed Computing*, vol. 6, pp. 636–648, 1989.

[6] H. Graf, L. Jackel, R. Howard, B. Straughn, J. Denker, W. Hubbard, D. Tennant, and D. Schwartz, "VLSI implementations of a neural network memory with several hundreds of neurons," in *Proc. Amer. Inst. Physics Conf. 151*, 1986, pp. 414–419.

[7] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci., U.S.A.*, vol. 81, pp. 3088–3092, 1984.

[8] J. Hopfield and D. Tank, "Neural computations of decisions in optimization problems," *Biolog. Cybern.*, vol. 52, pp. 141–152, 1985.

[9] K. Hwang and Z. Xu, "Multipipeline networking for compound vector processing," *IEEE Trans. Comput.*, vol. 37, no. 1, pp. 33–47, 1988.

[10] M. V. Mascagni, "Numerical methods for neuronal modeling," in *Methods in Neuronal Modeling*, C. Koch and I. Segev, Eds. Cambridge, MA: M.I.T. Press, 1989, pp. 439–484.

[11] W. L. Miranker and W. M. Liniger, "Parallel methods for the numerical integration of ordinary differential equations," *Math. Comput.*, vol. 21, pp. 303–320, July 1967.

[12] J. B. Rosser, "A Runge-Kutta for all seasons," *SIAM Rev.*, vol. 9, pp. 417–452, July 1967.

[13] M. Takeda and J. W. Goodman, "Neural networks for computation: Number representations and programming complexity," *Appl. Opt.*, vol. 25, pp. 3033–3046, Sept. 1986.