A CMOS Current-Mode Dynamic Programming Circuit

Terrence Mak, *Member, IEEE*, Kai-Pui Lam, H. S. Ng, Guy Rachmuth, and Chi-Sang Poon, *Fellow, IEEE*

***Abstract*—Dynamic programming (DP) is a fundamental algorithm for complex optimization and decision-making in many engineering and biomedical systems. However, conventional DP computation based on digital implementation of the Bellman–Ford recursive algorithm suffers from the "curse of dimensionality" and substantial iteration delays which hinder utility in real-time applications. Previously, an ordinary differential equation system was proposed that transforms the sequential DP iteration into a continuous-time parallel computational network. Here, the network is realized using a CMOS current-mode analog circuit, which provides a powerful computational platform for power-efficient, compact, and high-speed solution of the Bellman formula. Test results for the fabricated DP optimization chip demonstrate a proof of concept for this solution approach. We also propose an error compensation scheme to minimize the errors attributed to nonideal current sources and device mismatch.**

***Index Terms*—Bellman's equation, continuous-time formulation, current-mode circuit design, dynamic programming, fabrication and testing.**

I. Introduction

DYNAMIC programming (DP) is a powerful mathematical technique for making a sequence of interrelated decisions. The term DP describes the process of solving problems where one needs to find the best decision one after another [2]. It provides a systematic procedure for determining the optimal combination of decisions to maximize or minimize an objective function in recursive form, which is known as the Bellman–Ford algorithm [2]–[4].

DP circuits are important and have been employed in various engineering and biomedical applications as fundamental components for optimization and decision-making [4]. For example, a derivative of DP, namely reinforcement learning, is employed to model natural reward systems based on the ability of animals to learn appropriate actions in response to stimuli and associated rewards [2], [4]–[6]. DP circuits can also be employed in very large scale integration (VLSI) system design to enable run-time dynamic optimization. For example, inclusion of a DP network into network-on-chips (NoCs) has been recently proposed and shown to improve the network bandwidth via on-chip dynamic routing [7], [8]. DP circuits can also be integrated into energy harvesting circuits to enable dynamic power scheduling and utilization [9].

In conventional DP computation, the Bellman's equation [2] is evaluated iteratively and sequentially and such time-consuming sequential iterations result in substantial computational delay. The notion of "curse of dimensionality" as coined by Bellman in [10] refers to the vast computational effort required for the numerical solution of Bellman's equation when there is a large number of state variables that are subjected to the optimization objective function. Both the computational delay and hardware resources requirements grow explosively when the problem size increases. To accelerate the DP computation, a first-order ordinary differential equation (ODE) system was proposed by Lam and Tong [1] and can be employed to transform the sequential DP algorithm into a continuous-time parallel computational network which enables high-speed convergence for Bellman's optimality criterion. Here, a CMOS current-mode analog circuit is presented to provide a highly portable and low-power implementation of the proposed network architecture. Detailed analysis on computational speed, power consumption and network convergence is presented. Realization of a circuit with a reasonable size is demonstrated to exemplify the design principles. A procedure to test and validate the fabricated circuit is discussed. We have also investigated the error models and the results lead to a compensation scheme whereby the errors due to nonideal current source and device mismatch are minimized.

II. A Dynamic Programming Network

A. Dynamic Programming

The objective of DP is to find a set of actions that maximize the received reward in the long run based on the given model. Particularly, it can be formulated as a shortest path problem, in which the objective of the formulation is to find a set of optimal actions that lead to the shortest path under a model of an environment, which can be obtained *a priori* using probabilistic model estimation and identification.

The shortest path problem can be described as follows: Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ with $n = |\mathcal{V}|$ nodes, $m = |\mathcal{A}|$ edges, and a cost associated with each edge $u \longrightarrow v \in \mathcal{A}$, which

Manuscript received December 21, 2009; revised April 02, 2010; accepted May 13, 2010. This work was supported in part by the MOSIS educational program (research) for chip fabrication. The work of T. Mak was supported by the Croucher Foundation. The work of G. Rachmuth was supported by the U.S. National Defense Science and Engineering Graduate Fellowship. The work of C-S. Poon was supported by the National Institutes of Health under Grants EB005460 and RR028241. This paper was recommended by Associate Editor Manuel Delgado-Restituto.

T. Mak is with the School of Electrical, Electronic and Computer Engineering, Newcastle University, Newcastle upon Tyne, UK (e-mail: terrence.mak@ncl.ac.uk).

K.-P. Lam and H. S. Ng are with the Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong.

G. Rachmuth and C.-S. Poon are with the Harvard-MIT Division of Health Sciences and Technology, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2010.2052661

is denoted as $C_{u,v}$. The edge cost can be defined subject to different applications. The total cost of a path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the costs of its constituent edges: $\mathrm{Cost}(p) = \sum_{i=1}^{k} C_{i-1,i}$. The shortest path of G from i to j is then defined as any path p with cost that is $\min \sum_{i=1}^{k} C_{i-1,i}$ for all constituent edges v_i.

The Bellman–Ford algorithm [2]–[4] defines a recursive procedure in step k, to find the new estimate $\hat{V}_i(k)$ for the expected shortest path cost from s_i to the destination s_N using the previous estimates in step $k-1$, known as *dynamic programming*.

$$\hat{V}_i(k) = \min_{a \in \mathcal{A}(s_i)} [C_{i,j} + \hat{V}_j(k-1)]. \tag{1}$$

In addition, the optimal decisions at each node v_i that leads to the shortest path can be readily obtained from the argument of the minimum operator at the Bellman's equation as follows:

$$\mu^*(s_i) = \arg \min_{a \in \mathcal{A}(s_i)} [C_{i,j} + V_j^*] \tag{2}$$

where $\mu(s_i)$ is the optimal decision policy for the state s_i.

B. A Continuous-Time DP Network

The original DP computations proposed in [2]–[4] assume a digital platform for realizing Bellman's equation, which is applied iteratively until the optimal criterion is satisfied. By relaxing the assumption of a digital realization, the Bellman–Ford recursive DP can be realized using a continuous-time network with the aid of ODE formulation. The network has a parallel architecture, and can be used to derive DP solution through the simultaneous propagation of successive inferences. With close resemblance to the DP formulation on closed semiring, Lam and Tong introduced the Boolean Relation Inference Network (BRIN) [1] to solve a set of graph optimization problems with a continuous-time computational framework. This new class of inference network is inherently stable in all cases and it has been shown to be robust and with arbitrarily fast convergence rate. Therefore, the DP problem can be represented using a network structure with the adjacency matrix $\mathbf{A} = (a_{ij})$ where $a_{ij} = 1$ if $(i \rightarrow j) \in \mathcal{A}$ and, otherwise, $a_{ij} = 0$. A network of computational units $U(i,j)$ can then be constructed to represent the binary relations between node s_i and s_j and the outputs of the computational unit, $g(i,j)$, is defined as follows:

$$S_k(i,j) = \frac{1}{a_{ik}} [C_{i,k} + g(k,j)], \ \forall k \tag{3}$$

$$g(i,j) = \min_{\forall k} \{S_k(i,j)\}. \tag{4}$$

The computational units are interconnected resembling to the graph topology of the shortest path problem. The output of each unit represents the DP value of the corresponding node, which is the expected cost to the destination. Suppose that the "min" operator requires an infinitesimal time, δt, for evaluation. Also suppose that each computational unit $U(i,j)$ behaves dynamically as a first-order system, the whole network can be described by a set of differential equations as follows,

$$\frac{dV_i(t)}{dt} + \frac{1}{\lambda_i} V_i(t) = \frac{1}{\lambda_i} \min_{\forall k} \left\{ \frac{1}{a_{ik}} [C_{i,k} + V_k(t)] \right\}, \ \forall s_i \in \mathcal{V} \tag{5}$$

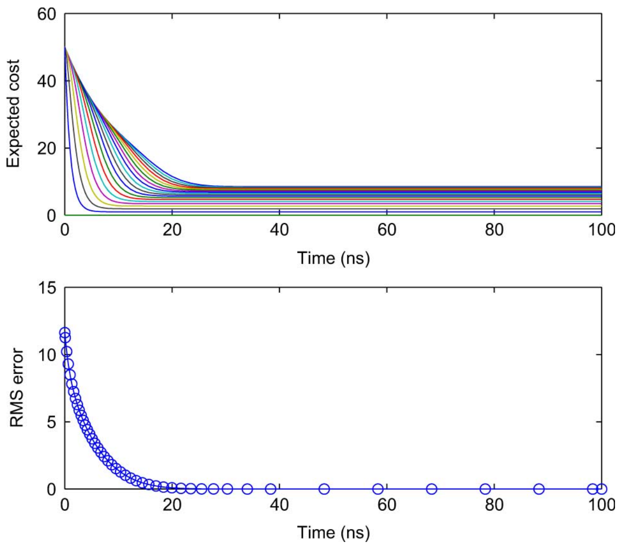


Fig. 1. Convergence of a DP network with 50 nodes in an array structure and the time constant, λ, is assumed to be 1 ns. Upper panel: Evolution of all DP values, $V_i(t), \ \forall s_i$, where s_i is the ith node in the DP network. Lower panel: The rms error of the overall DP values.

where $V_i(t)$ represents the approximated $\hat{V}_i$ in the continuous time domain, and λ_i is a first-order lag constant which is implementation-dependent. Notice that when $dV_i(t)/dt$ is approximated by the discrete-time finite difference $V_i(t+1) - V_i(t)$ and $\lambda_i = 1$, (5) reduces back to (1). When the ODEs has converged, the DP value V_i^* denotes the optimal shortest path solution and satisfies the Bellman's optimality criterion. Equation (5) provides a natural setting for analog circuit implementation, without the need of imposing sequential constraints as in the traditional digital Bellman–Ford algorithm in [4].

C. Convergence of the DP Network

The DP problem can be well represented using a system of ODEs as in the form of (5). The solution time of such network depends on the network convergent rate, compared with iteration cycles if implemented in the discrete, digital platform. Convergence of ODEs corresponds to the time required for the DP network to settle. This network settling time depends on a few system parameters, notably the network structure and the system time constant λ. There are other implementation related parameters, such as voltage and current, will also affect the network convergent rate, and will be discussed in Section IV. The network convergence rate can be investigated empirically using an ODE solver (the Matlab ODE solver, ode23, was employed in our experiment). While simulation time for solving the ODEs is not the actual physical timing of the circuit, it gives a theoretical estimate of the resolution time, and can be used for comparisons between different network settings and system parameters.

Fig. 1 shows the convergence of a DP network, which comprises 50 nodes in a linear topology. The network can be represented using 50 ODEs of the form of (5). The upper panel shows the evolution of the DP values, $V_i(t), \ \forall s_i \in \mathcal{V}$. Each curve is the output of a node in the network and the figure shows all the outputs as a family of curves from the DP network and the lower panel shows the rms error of these values compared to the optimal values. This example demonstrates the effectiveness of a

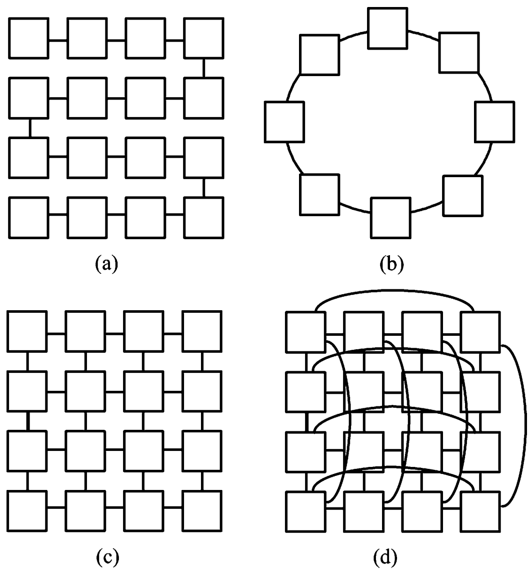


Fig. 2. Topological structure of DP networks, namely: (a) array; (b) ring; (c) mesh; and (d) torus.

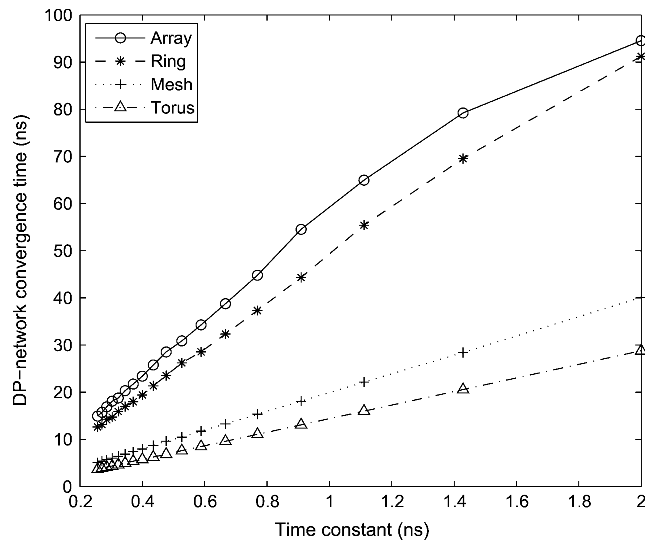


Fig. 3. Convergence time of DP networks with different time constants, λ and network structures (each with 100 nodes).

continuous ODE formulation for solving the DP problems. The network converges exponentially to the optimal solution. It is also interesting to note that the DP values are not converging at the same speed. This is because the propagation of DP values is constrained by the network topology.

Consider a network of 100 nodes with four different structures: array, ring, mesh, and torus (Fig. 2). In an array, nodes are arranged in a linear one-dimensional structure. In a ring, the two end nodes in an array are connected. In a mesh, nodes are interconnected as a two-dimensional network and each node is connecting to its four nearest neighbors. In a torus, the mesh network is folded into a three-dimensional structure. As shown in Fig. 3, the network convergence speed varies considerably with different network structures and is also affected by the system parameter λ. The network convergence time increases as the time constant, λ, increases.

Different network topologies facilitate network dynamics and convergence differently. Fig. 4 compares the convergence time

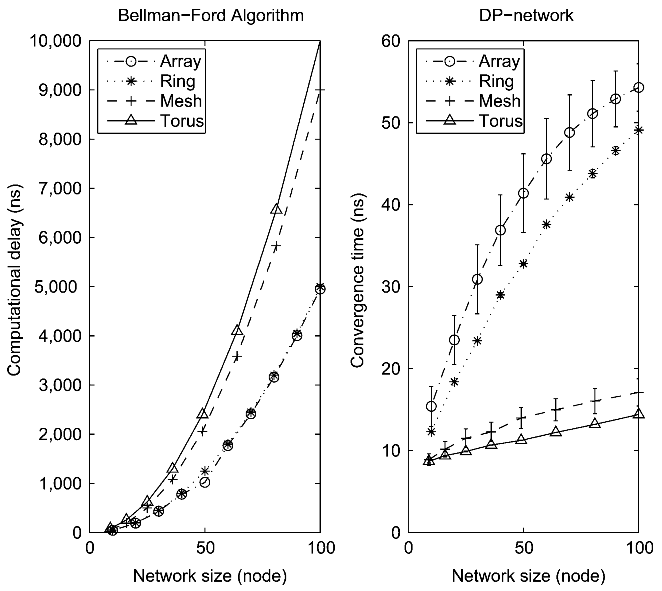


Fig. 4. Comparisons of computational delay between the sequential (left panel) and continuous-time parallel (right panel) implementations. Four different structures: array, ring, mesh and torus, and different network sizes are considered. The data point and error bar represent the mean and standard deviation of the convergence time, respectively.

of DP network for four different network structures with network size from 10 to 100 nodes as opposed to the computational delay of the sequential discrete-time Bellman–Ford algorithm. The network is considered converged when the rms error is less than 0.01. Since the network convergence time varies for different spatial location of the destination node in the network, the mean and standard deviations of the convergence time are reported based on Monte Carlo simulations with random path costs.

For all network topologies, the Bellman–Ford curves grow exponentially upward while the DP network curves level off in time. For the DP network, the array structure has the longest convergence time. This can be explained by the fact that the time required for information to propagate to all nodes in an array is dependent on the longest node-to-node distance in the network. The ring network has a faster convergence speed as the network distance is half of the array network distance. The convergence is substantially enhanced in a network structure with more connections, such as mesh or torus. Also, the convergence time grows much more slowly with the network size for such network structures. Interconnections in the network facilitate the interactions between computational units and, hence, enhance the convergence speed for DP networks. In contrast, for the Bellman–Ford algorithm the number of edges in the DP problem graph does not facilitate the computational speed but requires additional computational efforts. As can be seen from Fig. 4, structures with more complex edges, such as mesh and torus structures, cause longer computational delays in the Bellman–Ford algorithm.

D. Computational Complexity of DP

The conventional DP algorithm is based on a scheme that recursively executes Bellman's equation for all the nodes in a network. This formulation is suitable and was designed specifically for a microprocessor or software-based implementation. The sequential routine can be mapped to a scheduled programme and carried out iteratively. However, the sequential and discrete formulation is computationally intensive and, thus, prolongs the delay for optimization and learning. For example, the Bellman–Ford algorithm [3], [11] runs in time $O(mn)$ where n is the number of nodes, $|\mathcal{V}|$, and m is number edges, $|\mathcal{A}|$, in a network [12]. For a fully connected graph, the number of edge $m = n(n-1)/2$, hence the computational complexity for the Bellman–Ford algorithm becomes $O(n^3)$. This vast complexity in algorithmic computation results in large computational delay and is the major cause of the "curse of dimensionality" problem.

The proposed DP network provides a parallel computation framework for solving the Bellman's equation. The convergence time to an optimal solution depends on the network topology, which determines the delay of information propagates within the network, and the delay of each computational unit. Each unit involves $O(|\mathcal{A}|)$ additions followed by a minimization where $|\mathcal{A}|$ is the number of adjacent edges. Hence, the solution time is $O(k|\mathcal{A}|)$ where k is the number of iterations that the (1) has been evaluated by each unit. In a software-based computation, k equals to the number of nodes in the network, thus $k = |\mathcal{V}|$, which guarantees that all nodes have been updated. However, in the DP network implementation with parallel execution, k will be determined by the network structure and $|\mathcal{A}|$ additions can be executed in parallel. Each computational unit can simultaneously compute the new expected cost for all neighboring nodes. The delays for addition and minimization are further reduced with analog realization which takes advantage of the Kirchhoff's laws for sum of currents and the characteristics of basic current-mirror circuits.

Therefore, the solution time will be the time for the updated value distributed to every node in the network. Consider a mesh network of n nodes with $\sqrt{n}$ rows and $\sqrt{n}$ columns. The longest path in this network, which is the network diameter, is $O(2\sqrt{n}-1)$, which is the computational complexity for a DP mesh network and the minimum time required for updating the expected costs at all nodes. In Fig. 4, the computational delay of the DP problem increases in a much slower rate when comparing to the Bellman–Ford algorithm. Thus, DP network provides a significant improvement on solution time when compared to the algorithmic iterative implementation of DP.

III. Current-Mode DP Circuits

The continuous-time formulation of a DP network provides an opportunity for analog circuit realization. Analog current-mode realization presents many outstanding characteristics for circuit implementation including wide-dynamic range [13] and ultralow-power dissipation [14]. Computational arithmetic can be mapped to physical characteristics of circuits utilizing current for number representation. The basic computational unit of the DP network involves minimum and addition arithmetic. These operations can be realized based on the Kirchhoff's law of sum of currents and the properties in current mirror circuit, as will be detailed in the following.

The Max or the Min operations, also known as the winner-take-all (WTA) and the loser-take-all (LTA) operations, respectively, are the key functional units in Bellman's equation to realize DP, as well as in the DP network. Many WTA/LTA circuit implementations have been proposed in the literature [15]–[20]. Various implementations to improve the circuit speed, precision, and power consumption have been proposed and schemes that are robust to device mismatch have been presented. In this paper, we used an LTA circuit based on Vittoz's work in [21]. The advantages of this circuit is its simplicity and low-power consumption. Results of a preliminary study using Vittoz's LTA circuit are reported in [22]. Extension of such circuit by employing LTA circuits in [15] and [16] to improve device mismatch tolerance and reduce power consumption will be considered in the future work.

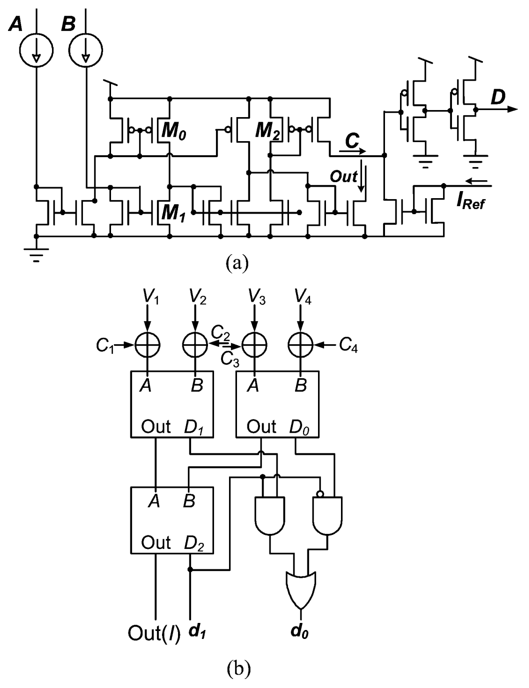


Fig. 5. Schematics for: (a) an analog minimizer circuit and (b) a 4-input DP computational unit.

The basic circuit schematic for a two-input DP computational unit is shown in Fig. 5. This circuit [Fig. 5(a)] realizes the Bellman's equation in (1) which utilizes an analog current-mode minimizer or an LTA circuit [21]. The minimizer circuit is further extended to realize addition from the Bellman's equation and to output the argument of the minimum, as shown in Fig. 5(b).

The design of a minimizer combines NMOS and PMOS transistors to carry out addition and subtraction of replicas of two currents, A and B, at current mirrors M_0 and M_1. $A-B$ is then used in a two-transistor arrangement to give $\max(A-B, 0)$ and, hence, blocking any negative input to 0. The output $\min(A, B)$ is obtained by subtracting A from $\max(A-B, 0)$. The output $\min(A, B)$ is mirrored and output at output Out. On the other hand, the value $\max(A-B, 0)$, which is mirrored at M_2 and output at node C, provides very useful signal on determining which input belongs to the shortest path. This value can be readily converted into a digital value by using a single stage algorithmic ADC to give the decision variable at output D. Other alternative designs of current-mode minimizers, such as LTA circuits in [23] and [24], provide higher precision but at the cost of circuit complexity.

The two input minimizer can be extended to handle multiple inputs minimization problems, as a two-stage binary-tree circuit design for a four-input DP unit (Fig. 5(b)). This design can be extended to a k-input multiple-stage binary-tree minimizer circuit. Note that the current-mode binary tree structures require many copying operations, whose number is proportional to $\log_2 k$. Each current mirror could be a source of mismatch error that accumulates at the tree output. Also, for large networks operating in subthreshold region, there is an additional gain in the mismatch errors. However, there are various techniques, such as in [15], [17] and [19], to enhance transistor mismatch tolerance and binary-tree Min/Max resolution, which left for future work to improve the accuracy and scalability of the DP network.

The overall decision variable can be obtained based on the decision output at each subcircuit. For example, the decision variable for a four-input minimizer can be represented by two bits, d_0, d_1 and d_1 is the most significant bit. These decision values can be obtained by simple logic as follows:

$$d_0 = D_0 D_2' + D_1 D_2 \tag{6}$$

$$d_1 = D_2 \tag{7}$$

where D_0, D_1 and D_2 are the decision values of each subcircuit.

The addition operator can be readily realized based on the Kirchhoff's current law, which states that the sum of currents flowing into that node is equal to the sum of currents flowing out of that node. Therefore, the summation in Bellman's equation can be simply realized by joining the $C_{i,j}$ input and V_j input at each computational unit, as shown in Fig. 5(b).

IV. Design Evaluations and Comparisons

The proposed circuit was studied using the SPICE-like Cadence Virtuoso Spectra circuit simulator based on a 90-nm technology model. The minimizer circuit is the core computational circuit in the DP network. Propagation delay and power consumption of this circuit have a large impact on the overall system. Fig. 6 shows the propagation delay and power consumption with respect to a wide range of input currents from micro to nano Amps. The propagation delay of a DP unit decreases exponentially from 100 ns to 1 ns with the currents. The power consumption is increasing inverse exponentially with the currents.

Fig. 7 shows the energy consumption per single DP operation with different current inputs. It is interesting to observe that for such a DP unit, there is an optimal energy consumption point for the current inputs and is in line with the ultralow-power circuit design reported in the literature [25], [26]. When the input current is smaller than the optimal operating point, the current leakage of CMOS transistors begins to dominate the power dissipation, whereas when the input current is larger than this point, the power dissipation of the operation dominates the energy consumption.

Fig. 8 shows the log-log plot of power consumption and network convergence time for four different network topologies: array, ring, mesh, and torus (all have the same number of computational units). A curve in the upper-right of the graph implies less power-speed efficiency (more power is required to obtain the same network convergence speed), and *vice versa*. The results show that a linear relationship between power and time can be found for all network structures. The result for a single DP unit is used as a reference for comparison. The mesh network shows the most power efficiency because the interconnects in a mesh structure have a shorter network diameter and, thus, enhance the network convergence speed. The ring structure is less power efficient than the array, due to the extra current mirrors required for establishing the extra interconnects. The additional current mirrors consume extra power. The torus structure requires even more current mirrors for the connectivity, and, therefore, is less power efficient.

In order to compare the computational speed, power consumption and hardware area for our analog implementations, a digital counterpart of the analog circuit was implemented (see Fig. 9). The digital implementation provides a parallel and synchronous realization of the DP computation and this implemen-

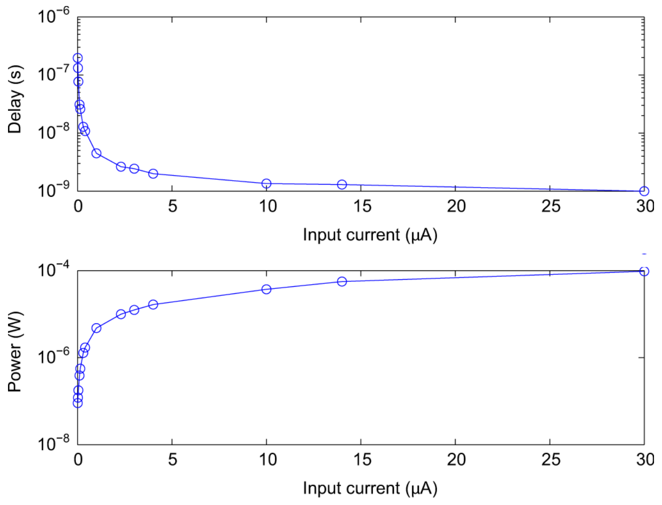


Fig. 6. Computational delay and power dissipation of a single DP computational unit with different current inputs. The current inputs correspond to the "A" and "B" inputs in Fig. 5(a).

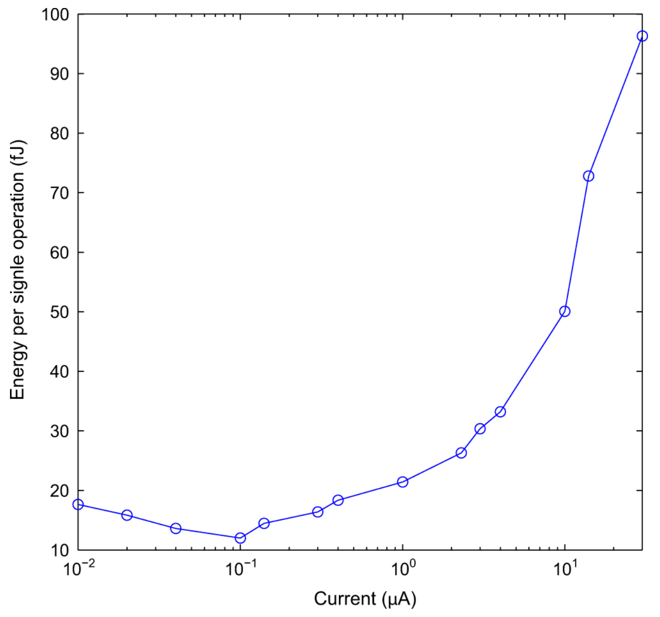


Fig. 7. Energy per single DP operation with different current inputs. The current inputs correspond to the "A" and "B" inputs in Fig. 5(a).

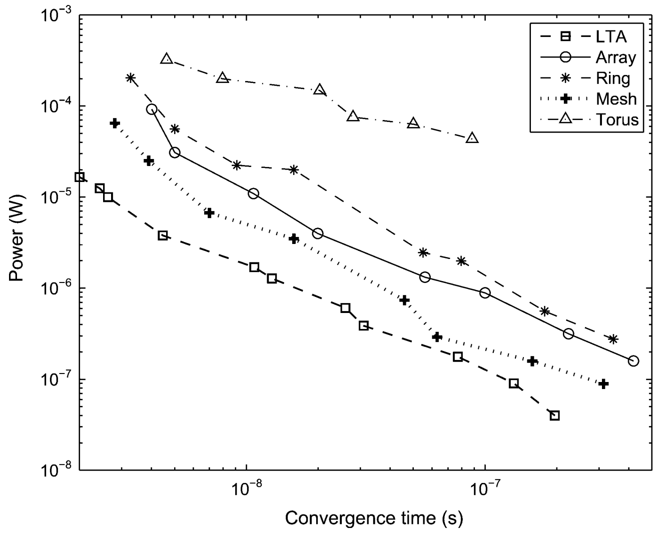


Fig. 8. Log-log plot of power consumption and network convergence time for different network structures.

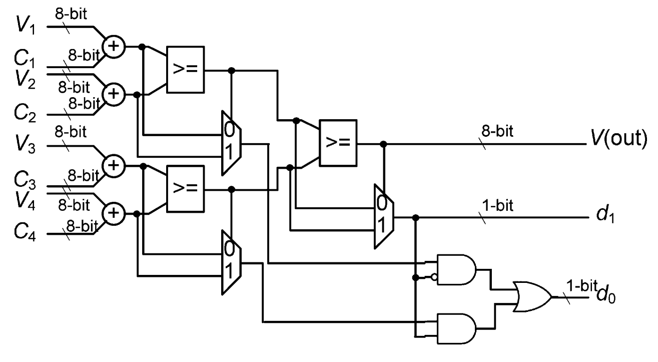


Fig. 9. Block diagram of digital implementation for a 4-input DP computational unit.

TABLE I
COMPUTATIONAL DELAYS OF PRIMITIVE FUNCTIONS USING ANALOG AND DIGITAL IMPLEMENTATIONS.

Operation	Delay (ps)		Energy (pJ)	
	Analog	Digital (8-bit)	Analog	Digital (8-bit)
Addition	32	244	1.51×10^{-5}	0.35
Minimum	69	278	8.07×10^{-4}	0.52

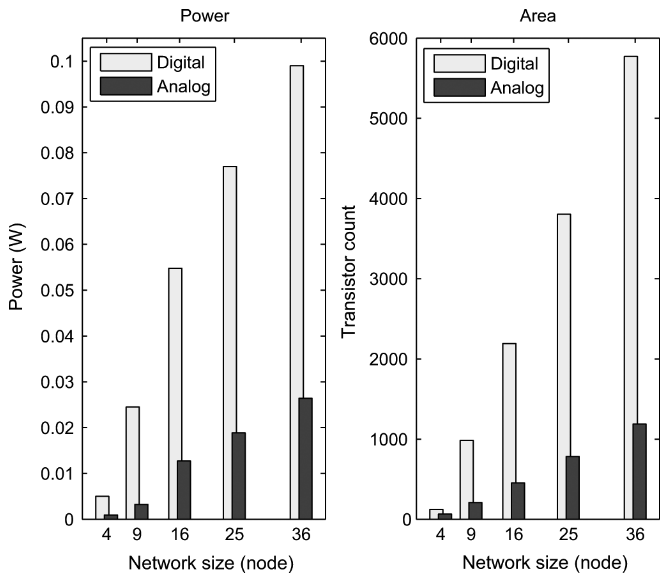


Fig. 10. Comparison between analog and digital implementation for DP networks on hardware area and power consumption. The digital implementation is realized using digital logics, which provides a conservative estimate on digital area and power consumption. Left: Power consumption comparison. Right: Transistor count comparison.

tation provides an optimistic speed estimate on digital realizations. Four 8-bit ripple carried adders were used for the addition circuit and the minimize circuit was implementing by using a subtractor and a multiplexer. The wordlength of the digital circuit is 8 bits to ensure a reasonable resolution. Because the digital circuit was constructed using the same technology model and implemented using CMOS transistors, a fair comparison for the speed, power consumption and area can be obtained.

With the utilization of device physics, analog realization provides a substantial improvement in speed and energy efficiency for primitive functions. The comparisons of delay and energy consumption between analog and digital implementations are shown in Table I. By exploiting the Kirchhoff's current law, analog realization of an adder is more than seven times faster than the digital implementation. More importantly, the analog adder is substantially energy efficient. It achieves more than four order of magnitudes more energy efficient than the digital adder. Similarly, the analog minimizer realized using current mirror can achieve more than four times speed-up and more than three orders of magnitude of energy saving when compared to the digital minimizer.

We compared the area by counting the number of transistors used between the two different circuits. The area comparison result is shown in Fig. 10. We compared the area of the analog circuit and its digital counterpart for different network size varying from 4 to 36. The analog circuit area is generally much smaller than its digital counter part. For smaller network, such as 4-node network, analog area is around half of the digital design (64 versus 124 for analog and digital respectively). For a large network such as 16-node network, analog area is only 20.6% of the digital area (453 versus 2192 for analog and digital, respectively).

Comparing circuit area based on transistor count may not be accurate because transistors used in an analog circuit are generally larger than those in a digital circuit. However, this size advantage is partially compensated for by the reduced routing found in analog circuits. Therefore, the area results presented above is probably optimistic in favor of the analog implementation. More accurate comparisons are presented below for the circuit layouts.

We also compared the power consumption between the analog and its digital implementations (Fig. 10). The power analysis was performed using SPICE, in which the average power dissipation is computed in conjunction with a transient analysis. The circuit was injected with a set of random test vectors and was allowed to run for one section for the transient analysis. The analog design generally consumes significantly less power than its digital counterpart. Even for a small mesh network, 4-node network, the analog circuit consumes 19% of the power that the equivalent digital circuit consumes (0.95 mW versus 5 mW for analog and digital circuits respectively). For a larger network, 16-node network, the analog circuits consumes

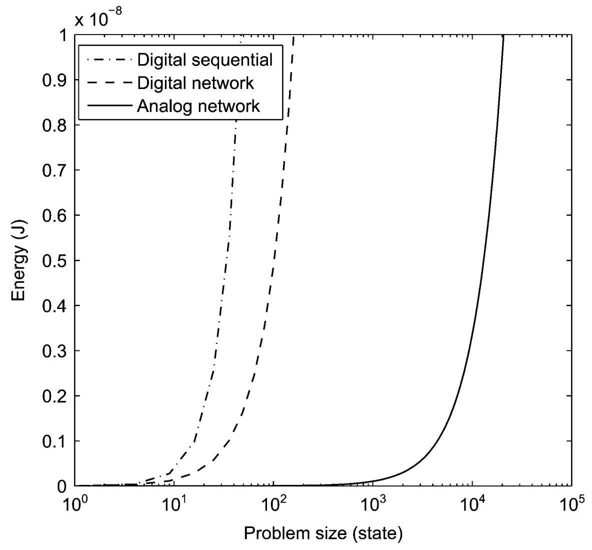


Fig. 11. Energy consumption for the three different implementation approaches: sequential implementation of the Bellman–Ford algorithm, digital and analog DP networks.

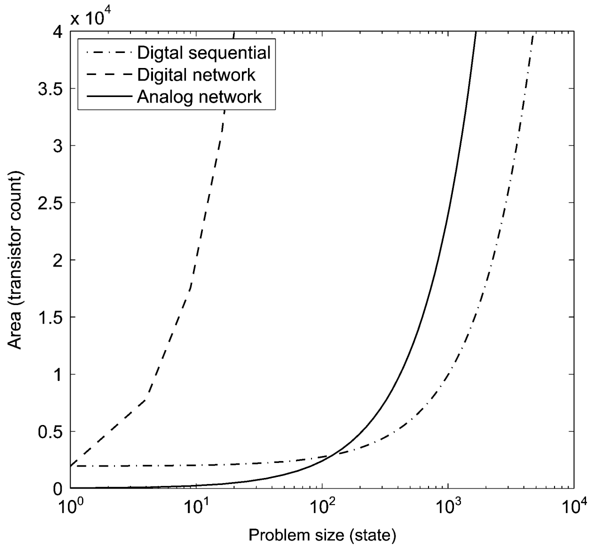


Fig. 12. Hardware area consumption in terms of transistor count for the three different implementation approaches: sequential implementation of the Bellman–Ford algorithm, digital and analog DP networks.

23% of the power that the digital circuit consumes (12.7 mW versus 54.8 mW for analog and digital circuits respectively).

Estimates of energy consumption and silicon area utilization for solving large-scale DP problems are presented in Figs. 11 and 12. We estimated the energy consumption and the silicon area for digital sequential realization of the Bellman–Ford algorithm, digital parallel realization of the DP network, and analog realization of the continuous-time DP network.

The digital sequential realization results are based on a sequential implementation of DP algorithm using VLSI logics, since VLSI logic implementation is highly optimized in area and power and thus is a reasonable upper bound for the microprocessors, in which sequential instructions are carried out. The digital sequential implementation comprises a single computational unit as shown in Fig. 9 for realizing Bellman's equation. Additional registers and memory access to the storage are required to access the path costs and update the estimates when

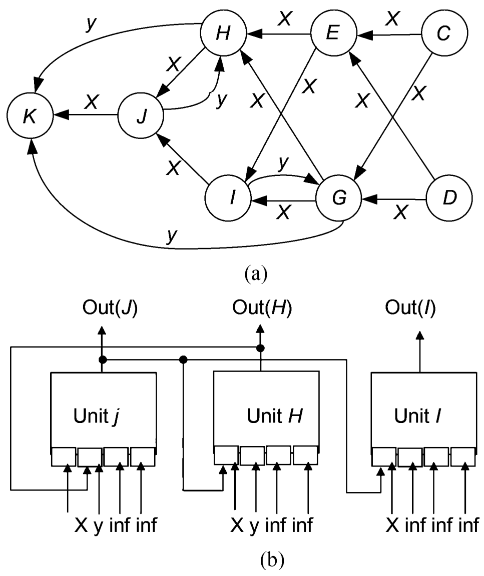


Fig. 13. (a) A DP network for solving an 8-node shortest path problem. (b) An example of interconnecting three computational units.

the problem is scaled up. Microprocessors with advanced architectures, such as multithreaded processors and parallel instructions processors, can provide a smaller execution delay and larger area and their performance can vary between that of the digital sequential and digital network in Figs. 11 and 12.

In terms of energy utilization, the analog DP network outperforms the two other approaches by a large margin. The analog DP network can handle a much larger problem size, three to four orders of magnitudes more, than the microprocessor and digital circuit approaches in the picojoule energy level. For hardware area utilization, microprocessor consumes the least area because of the sequential computation. The analog DP network consumes less transistors than microprocessor when the problem size is small, e.g., less than a few hundreds state variables. For large-scale problems, the analog approach consumes just marginally more area than the microprocessor due to the effective utilization of transistors and interconnects. Digital network consumes two orders of magnitude more area than its analog counterpart due to the substantial area-expensive digital logics.

V. Design Fabrication and Testing

A DP circuit with reasonable size is realized using CMOS current-mode circuit to exemplify the design principles. Fig. 13(a) is an eight-node directed graph, for which the shortest paths from the nodes J, H, I, G, E, D, C to the destination node K can be computed using from a DP network comprising of seven computational units. The circuit for a four-node subgraph $\{K, J, H, I\}$ is shown in Fig. 13(b). Node J has two outgoing paths: a direct one to the destination K with cost $d_{JK} = x$, and another one with cost $d_{JH} = y$ to node H. For the remaining sites, a high current value "infin" (e.g., the upper limit of the operation range) is inputted to disable the site.

For simplicity and without loss of generality, we chose to take a systematic approach in considering two cost variables (x and y) irrespective of the graph size. The idea is to consider a class of problems using x for all the feedforward paths (prewired within

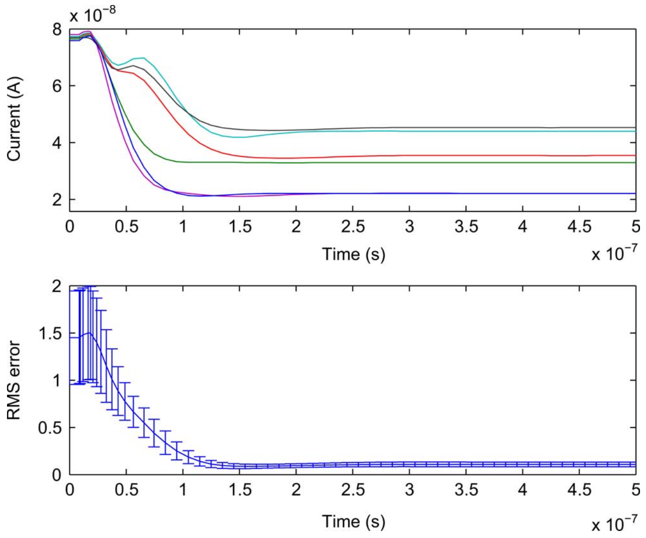


Fig. 14. Simulation results of the circuit for solving the shortest path problem in Fig. 13. (upper) Evolution of current outputs, which represent the expected value, from each computational unit in the network. (lower) RMS error of the overall current outputs. The error bar shows the standard deviation of the error.

the chip), while providing some y feedback loops to investigate the decision changes made by the computational units for verification.

Fig. 14 shows an example of the SPICE simulation results for solving the eight-node shortest path problem in Fig. 13. The evolution of current outputs from each computational units were shown at the upper panel while the average relative error of the computation is shown in the lower panel. The circuit converges rapidly, as the relative error decreases to less than 5% in 100 ns.

A CMOS chip for the DP network was fabricated by MOSIS using the AMI 1.5 μm process. Table II compares the two technologies in terms of transistor sizes, supply voltage, power dissipation, and speed. Although the analysis presented in previous section was carried out based on the 90-nm technology model, the general trends of power dissipation and delay are also in line with the 1.5 μm technology in our fabricated circuit. The transistor length is 4 μm and the width is 28 μm. Once circuit operation was verified, optimal layout techniques were used to reduce circuit sizes. The total chip area of a 8-node DP network is $0.6\ \mathrm{mm} \times 1.4\ \mathrm{mm}$. A photograph of the completed design is shown in Fig. 15. What you can distinguish in this photo is the DP computational units corresponding for the shortest-path problem described in Fig. 13. For chip testing, a large number of voltage-controlled current sources were built in-house using matched $100\ \mathrm{K}\Omega/10\ \mathrm{K}\Omega$ resistor pairs for the differential inputs of an operational amplifier (LM324). In dealing with the 4-node subgraph problem, a total of 12 current sources ($3x$, $2y$, and 7 "infin") were used; while the 8-node graph problem required 28 current sources ($11x$, $4y$, and 13 "infin"). Current measurements in the μA range were made using a transimpedance configurable analog module (CAM) of the Anadigm field-programmable analog array (FPAA). Appropriate calibrations were also performed on both the current-measurement and the current-source setup using an HP 3245A universal source.

Dynamic tests were performed on individual 4-input DP units of our fabricated chip using a varying input on one site, while

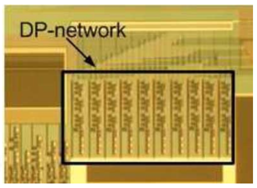


Fig. 15. Photomicrograph of fabricated DP network circuit.

TABLE II
PHYSICAL PARAMETERS OF THE TWO TECHNOLOGICAL MODELS

Technology	STMicro 90nm	AMI 1.5μm
pMOS size (W/L)	540nm/270nm	27um/4.5um
nMOS size (W/L)	540nm/270nm	13.5um/4.5um
Supply voltage	1V	3V
Power[1]	1.2uW	0.39mW
Delay[2]	0.1ns	1.5ns

[1] A standard 2-stage inverter was used to estimate the power dissipation.
[2] The delay refers to the signal propagation time between injected signal and the output of a 2-stage inverter.

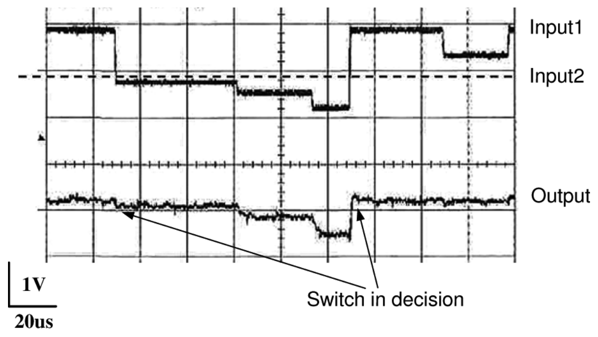


Fig. 16. Dynamic test of the minimizer circuit in the fabricated DP network. The "Output" curve shows the minimizer output of the two input curves, "Input1" and "Input2."

keeping the other site inputs constant. A typical example is provided in Fig. 16 for DP unit J, showing a case for the varying the inputs and the measured J output: Input 1 is 30 μA, Inputs 3 and 4 are dedicated as reference inputs (labeled Input 2) voltage-controlled at 2.3 V (to give 11.3329 μA). The DP-circuit performs well, giving the minimum output (labeled OUT) as those Input 1 that are smaller than Input 2. The output J is shown to behave like a first-order lag (including that of the lag of the current-measurement device) with respect to step changes. The dynamic tests are useful for selecting the functional DP units of the fabricated chip for subsequent network connection; and more detailed timing measurements can be performed to deduce the computational delay of the circuits.

Static tests were performed on two graph problems specified by Fig. 13(a): a smaller 4-node graph with a connected 3 DP-unit array for $\{J, I, H\}$, and an 8-node graph with a 7 DP-circuit array for $\{J, I, H, G, E, D, C\}$. In each static test, the y-inputs and "infin"-inputs were fixed while the x-inputs vary over a range of values. The various measurable DP unit outputs were then measured at a set of x-input values, and were compared with that obtained from theoretical analysis and prediction. For the 4-node subgraph $\{K, J, H, I\}$ of Fig. 13(a), the J and I outputs are expected to be $\min\{x, 2y\}$ and $\min\{2x, x+2y\}$, respectively. A very nice experimental curve is obtained in Fig. 17, showing that both J and I are able to make the necessary decision changes at around 1 μA (which matches well

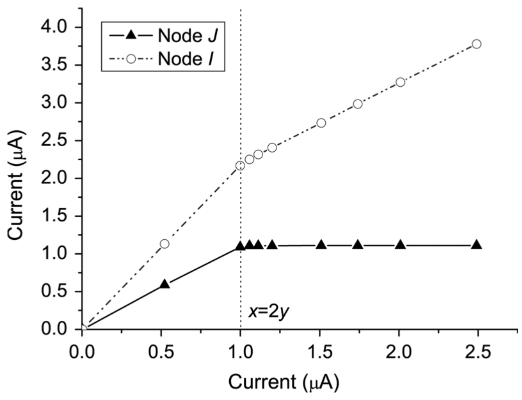


Fig. 17. Static test for 4-node problem.

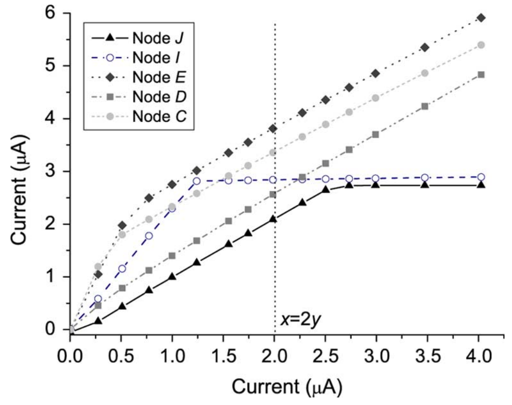


Fig. 18. Static test for 8-node problem.

with the value of two times the y current setting at $0.5011\ \mu\mathrm{A}$). The curve slopes are also useful to convey information, i.e., J becomes flat after $x > 2y$ because its minimum path cost is $2y$, and I's slope switches from 2 to 1 because the minimum path costs for I are $2x$ (for $x \leq 2y$) and $x + 2y$ (for $x > 2y$), respectively.

Referring to the 8-node graph of Fig. 13(a), the expected outputs of J and I are obtained as $\min\{x, 2y\}$ and $\min\{2x, x + 2y, 2y\}$, respectively. Fig. 18 shows the output curves for nodes $J,\ I,\ E,\ D$, and C; and it is interesting to compare the J and I curves of Figs. 17 and 18. Because of an additional y feedback loop at node I, I makes the correct decision change near $x = y$, and J makes the change near $x = 2y$. They both have a minimum path cost close to $2y$, but for two different paths, i.e., $I \longrightarrow G \longrightarrow K$ versus $J \longrightarrow H \longrightarrow K$. These results are as expected (within the prescribed error tolerance for analog circuits) from the problem specification and validate the network performance. The expected output values of the other nodes can readily be obtained from further analysis, e.g., $\min\{3x, x + y\}$ for E, and $\min\{4x, x+y\}$ for D. The curves for $E,\ D$, and C show the expected rise with a slope of 1 for large x; but their initial slopes for small x are less than the predicted values due to measurement errors and transistor mismatch.

Noise and device mismatch are inherent with the analog circuit and these errors can be attributed to the current source noise and device mismatch. Transistor mismatch has been long known to be a critical problem to analog VLSI precision due to fabrication variation [27] in device dimension (e.g., length/width ratio), current level, and transistor separation. To analyze the mismatch error (apparently due to the output current mirrors) of the DP unit of our fabricated chip, we assume that the output current is αI instead of I, where α is close to 1, which can be achieved by minimizing the impact of mismatching with a large width of current mirror, e.g., $27\ \mu\mathrm{m}$ in our fabricated circuit.

For node I, when $2x = 2y$ as x increases for a fixed y, we have $\alpha_1 \cdot (x + \alpha_{12} \cdot x) = \alpha_2 \cdot (y + \alpha_{21} \cdot y)$ because each will involve current going through a minimum of two minimizers that may have different α's. Hence, x is approximately $(\alpha_2/\alpha_1) \cdot y$ as the α's are assumed to be close to 1.

From Fig. 18, the ratio (α_2/α_1) is roughly estimated to be 1.3. For node J with $x = 2y$, $\alpha_1 \cdot x = 2\alpha_2 \cdot y$ will give an x approximately equal to $(\alpha_2/\alpha_1) \cdot 2y$ (which is around $2.6y$). Errors in the initial slopes can then be attributed to similar reasons, because $3x$ will involve at least three minimizers and give $\alpha_1 \cdot (x + \alpha_{12} \cdot (x + \alpha_{13} \cdot x)) < 3x$.

The most direct but expensive method to deal with transistor mismatch is to reduce the gradient in fabrication variation as much as possible to give $\alpha = 1$. Other approaches using online calibration [28], [29] in learning devices look promising for our future work.

VI. Error Models and Compensation

The experimental data of the chip testing (Fig. 18) provided a clue on the error models for deriving compensation procedure to obtain more accurate computation. Based on a theoretical ODE emulation of the analog chip, two error models were derived to yield simulation curves matching the experimental data. The first model takes account of input errors in the current sources x (where 11 of them were needed for the 8-node network). As only one highly precise HP 3245A was available to maintain y at 1 uA, the current sources for x were constructed in-house using simple circuitry with voltage calibration techniques (i.e., all the x were not actually measured but were obtained from calibrated voltage settings). To account further for variability, the following error model for x was assumed:

$$x_p = x + kx \tag{8}$$

where x_p is the actual x used in chip testing assuming an offset $k \in N(\mu, \sigma^2)$ with a normal distribution of mean μ and standard deviation σ. Particularly, we assume a 10% offset in our error model.

The other error model takes account of transistor mismatch for the seven minimizers (for nodes C, D, E, G, H, I, J) in the chip. Through extensive evaluation for different αs (some with nonlinear dependence on x_p), the following choices:

$$\alpha_C = 1.2 * \left(\frac{1 - 1}{(x_p + 2)^2} \right) \tag{9}$$

$$\alpha_D = 1.2 * \left(\frac{0.95 - 1}{(x_p + 1.5)} \right) \tag{10}$$

$$\alpha_E = \alpha_I = 1.02 * 1.2 \tag{11}$$

$$\alpha_G = \alpha_H = \alpha_J = 1.2 \tag{12}$$

were found to yield simulation curves closely resembling the experimental data. A reverse compensation procedure could

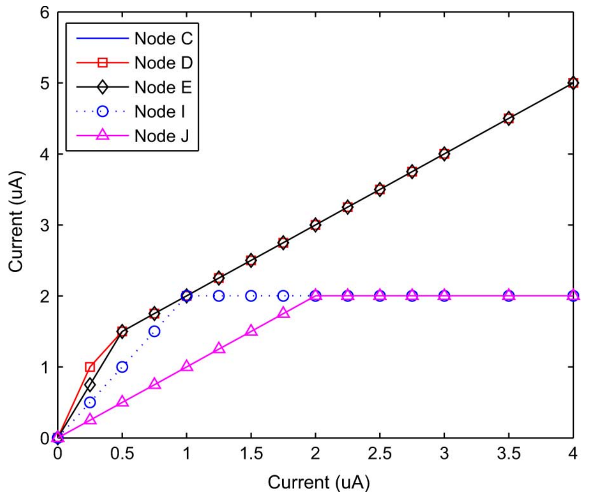


Fig. 19. The current outputs, which correspond to the optimal expected values of the DP computation in Fig. 13, from the DP circuit in ideal current sources and no device mismatch.

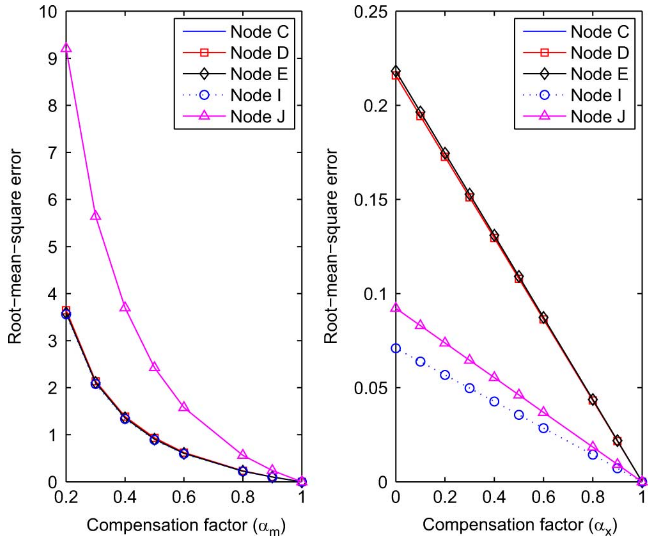


Fig. 20. RMS errors of the DP values from all computational units against the device mismatch and current source errors compensation factors, α_m and α_x, respectively.

readily be obtained to give the ideal curves (with no errors) for the chip outputs (Fig. 19).

The error margin and sensitivity of our compensation are further studied based on rms network output errors. Using a simple compensation factor, α_x for x_p, and α_m for a minimizer output (where $\alpha_x = \alpha_m = 1$ for perfect compensation, and their values decrease for less perfect compensation)

Fig. 20 shows that the sensitivity of the rms output errors due to errors in compensated inputs x_p is much less than that for compensated transistor mismatch. The sensitivity is also observed to be heterogeneous, indicating variability with respect to different network outputs.

VII. Conclusions

The computational delay inherent in dynamic programming (DP) can be greatly reduced by exploiting the parallelism in interconnection networks and utilizing the device physics for high-speed analog computation. The continuous-time DP network transforms the DP problem into a system of ODEs and provides a parallel platform for accelerating the computational speed. We have shown that the primitive computational units in the DP network can be realized by a CMOS current-mode analog circuit. The resulting analog DP circuit outperforms the microprocessor realization by four orders of magnitude in large-scale problems. More importantly, such analog DP network approach reduces the computationally intensive digital realization to a rapid converging network with linear computational complexity. Thus, it provides an opportunity to mitigate the large computational delay when solving large-scale DP problems as posed in the “curse of dimensionality.” Also, we have shown that the energy efficiency of the DP network significantly outperforms the conventional realization platforms, such as microprocessors and digital circuits. The circuit has been fabricated and tested. Procedures for dynamic and static tests are also discussed and successfully demonstrated to validate such DP circuits.

We have also investigated the computational errors attributed to nonideal current sources and device mismatches, and an error compensation model is proposed to tackle the noise and precision issues. Specifically, nonideal current sources and device mismatches could introduce computational errors in the numerical representation of current-mode circuit. Through thorough testing and analysis on the fabricated circuit, an error compensation model has been developed. Precision due to mismatch and input current offset can be handled with voltage calibrations and the error model. Further, the device mismatch issue in analog circuit design has been well studied [30]–[32] and different layout techniques [33], models [30], [31], and compensation schemes [34] were reported to improve accuracy and yield, and to avoid overdesign. Incorporation of the device matching compensation schemes to improve the computational accuracy in DP circuit will be our future work.

Finally, it should be noted that the computation of the expected costs in the value function generally does not require high accuracy as the important outputs of the circuit are the optimal decisions. There is typically a large margin, defined herein as the difference between the expected cost values, when dealing with the comparison between input costs. This margin provides a reasonable error tolerance to the dynamic noise in the circuit and thus the DP network is intrinsically robust. Furthermore, it has been suggested that the DP problem can be formulated as a convex optimization problem [35], [36], such that local perturbations of the values in the circuit may still lead to a suboptimal solution. Other interesting implementation challenges to be studied in the future include improving the analog DP network performance when solving large-scale DP problems, effective distribution of path costs from the memory to the network, and integration of DP circuit as a modular block into other system-on-chip or NoC platforms.

References

[1] K. Lam and C. Tong, “Closed semiring connectionist network for the bellman-ford computation,” *IEE Proc. Comput. Digit. Tech.*, vol. 143, pp. 189–195, 1996.
[2] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton Univ. Press, 1957.
[3] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.
[4] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[5] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[6] S. Haykin, *Neural Networks—A Comprehensive Foundation*, Second ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[7] T. Mak, P. Sedcole, P. Cheung, W. Luk, and K. Lam, "A hybrid analog-digital routing network for NoC dynamic routing," in *Proc. IEEE Int. Symp. Networks-on-Chip (NoC)*, 2007, pp. 173–182.

[8] T. Mak, P. Cheung, W. Luk, and K. Lam, "A DP-network for optimal dynamic routing in network-on-chip," in *Proc. ACM Int. Conf. Hardware/Software Codesign Syst. Synthesis (CODES)*, 2009, pp. 119–128.

[9] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *ACM Trans. Embedded Comput. Syst.*, vol. 6, no. 4, pp. 32–70, 2007.

[10] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton Univ. Press, 1961.

[11] R. Bellman, "On a routing problem," *Q. Appl. Math.*, vol. 16, pp. 87–90, 1958.

[12] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.

[13] C. Toumazou, F. J. Lidgey, and D. G. Haigh, *Analogue IC Design: The Current-Mode Approach*. London, U.K.: Inst. Electr. Eng., 1993.

[14] C. Mead and M. Ismail, *Analog VLSI Implementation of Neural Systems*. New York: Springer, 1989.

[15] A. Fish, V. Milrud, and O. Yadid-Pecht, "High-speed and high-precision current winner-take-all circuit," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 52, no. 3, pp. 131–135, Mar. 2005.

[16] A. Demosthenous, S. Smedley, and J. Taylor, "A CMOS analog winner-takes-all network for large-scale applications," *IEEE Trans, Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 3, pp. 300–304, Mar. 1998.

[17] T. Serrano and B. Linares-Barranco, "A modular current-mode high-precision winner-take-all circuit," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 2, pp. 132–134, Feb. 1995.

[18] J. Ramirez-Angulo, J. Molinar-Solis, S. Gupta, R. Carvajal, and A. Lopez-Martin, "A high-swing, high-speed CMOS WTA using differential flipped voltage followers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 8, pp. 668–672, Aug. 2007.

[19] R. Dlugosz and T. Talaska, "Low power current-mode binary-tree asynchronous min/max circuit," *Microelectron. J.*, vol. 41, no. 1, pp. 64–73, 2010.

[20] B. Wilamowski, D. Jordan, and O. Kaynak, "Low power current mode loser take-all circuit for image compression," in *NASA Symp. VLSI Design*, 2000, pp. 7.6.1–7.6.8.

[21] E. Vittoz, "Analog VLSI signal processing: Why, where, and how?," *J. VLSI Signal Process.*, vol. 8, pp. 27–44, 1994.

[22] T. Mak, K.-P. Lam, H. S. Ng, G. Rachmuth, and C.-S. Poon, "A current-mode analog circuit for reinforcement learning problems," in *IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2007, pp. 1301–1304.

[23] N. Donckers, C. Dualibe, and M. Verleysen, "A current-mode CMOS loser-take-all with minimum function for neural computations," in *Proc. IEEE Symp. Circuits Syst.*, 2000, vol. 1, pp. 415–418.

[24] T. Serrano-Gotarredona and B. Linares-Barranco, "A high-precision current-mode WTA-MAX circuit with multichip capability," *IEEE J. Solid-State Circuits*, vol. 33, pp. 280–286, 1998.

[25] B. Mishra and M. Al-Hashimi, "Subthreshold FIR filter architecture for ultra low power applications," in *Proc. 18th Int. Workshop PATMOS 2008*, 2008, vol. 5349, Lecture Notes in Computer Science, pp. 1–10.

[26] A. Wang and A. Chandrakasan, "A 180-mv subthreshold fft processor using a minimum energy design methodology," *IEEE J. Solid State Circuits*, vol. 40, no. 1, pp. 310–319, Jan. 2001.

[27] T. Serrano-Gotarredona and B. Linares-Barranco, "Systematic width-and-length dependent CMOS transistor mismatch characterization and simulation," *Analog Integr. Circuits Signal Process.*, vol. 21, pp. 271–296, 1999.

[28] M. Figueroa, S. Bridges, and C. Diorio, "On-chip compensation of device-mismatch effects in analog VLSI neural networks," in *Proc. Neural Inf. Process. Syst. Conf.*, 2005, vol. 17, pp. 441–448.

[29] C. Diorio, D. Hsu, and M. Figueroa, "Adaptive CMOS: From biological inspiration to systems-on-a-Chip," *Proc. IEEE*, vol. 90, no. 3, pp. 345–357, Mar. 2002.

[30] C. Michael and M. Ismail, "Statistical modeling of device mismatch for analog mos integratedcircuits," *IEEE J. Solid-State Circuits*, vol. 27, pp. 154–166, 1992.

[31] K. Papathanasiou, "A designer's approach to device mismatch: Theory, modeling, simulation techniques, scripting, applications and examples," *Analog Integr. Circuits Signal Process.*, vol. 48, no. 2, pp. 95–106, 2006.

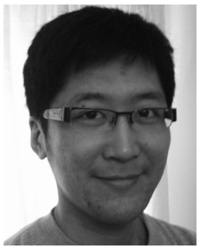
[32] J. Croon, W. Sansen, and H. Maes, *Matching Properties of Deep Sub-Micron MOS Transistors*. New York: Springer-Verlag, 2005.

[33] A. Hastings, *Art of Analog Layout*. Upper Saddle River, NJ: Prentice-Hall, 2005.

[34] P. Leary, "Practical aspects of mixed analogue and digital design," in *Analogue-Digital ASICs: Circuit Techniques, Design Tools and Applications,* , ser. IEE Circuits and Systems. London, U.K.: Peter Peregrinus Ltd., 1991, pp. 213–238.

[35] A. Rantzer and M. Johansson, "Piecewise linear quadratic optimal control," *IEEE Trans. Autom. Control*, vol. 45, no. 4, pp. 629–637, Apr. 2000.

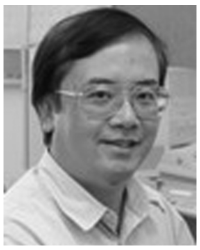
[36] A. Rantzer, "Dynamic programming via convex optimization," *Proc. IFAC World Congr.*, pp. 491–496, 1999.



Terrence Mak (S'05–M'09) received the B.Eng. and M.Phil. degrees in systems engineering from the Chinese University of Hong Kong in 2003 and 2005, respectively, and the Ph.D. degree from Imperial College London in 2009.

During his Ph.D. research, he worked as a Research Engineer Intern in the Very Large Scale Integration (VLSI) group at Sun Microsystems Laboratories, Menlo Park, CA. He also worked as a Visiting Research Scientist in the Poon's Neuroengineering Laboratory at Massachusetts Institute of Technology, Cambridge. He joined the School of Electrical, Electronic and Computer Engineering at Newcastle University, U.K., as a Lecturer in 2010. His research interests include field-programmable gate array architecture design, network-on-chip, reconfigurable computing, and VLSI design for biomedical applications.

Dr. Mak was the recipient of both the Croucher Foundation Scholarship and the U.S. Naval Research Excellence in Neuroengineering in 2005. In 2008, he served as the Cochair of the U.K. Asynchronous Forum, and in March 2008 he was the Local Arrangement Chair of the Fourth International Workshop on Applied Reconfigurable Computing.



Kai-Pui Lam received the B.Sc. (Eng.) degree in electrical engineering from the University of Hong Kong in 1975, the M.Phil. degree in electronics from the Chinese University of Hong Kong (CUHK) in 1977, and the D.Phil. degree in engineering science from Oxford University, U.K., in 1980.

He is a Professor in the Department of Systems Engineering and Engineering Management of CUHK. His research has focused on using field-programmable gate array in bioinformatics and neuronal dynamics and on intradaily information in financial volatility forecasting.

H. S. Ng received the B.Sc. degree in applied computing from Hong Kong Baptist University in 1994 and the M.Phil. and D.Phil. degrees in systems engineering and engineering management in 1996 and 2010, respectively, from the Chinese University of Hong Kong.

He presently works in the Hong Kong SAR Government as an Analyst/Programmer. His current research interests include the application of genetic fuzzy system to financial market trading, intraday prediction and modeling, and analog and very large scale integration implementation and optimization problems.

Guy Rachmuth received the B.S. degree (*summa cum laude*) in biomedical engineering, with a focus on medical device electronics design, from Boston University, Boston, MA, and the Ph.D. degree in biomedical engineering, with a focus on neuroengineering, from Harvard University, Cambridge, MA, in 2006.

His work was performed at the Massachusetts Institute of Technology (MIT) under the Harvard/MIT Division of Health Sciences and Technology (HST) where he continues his role as a Research Affiliate. While at MIT as a Postdoctoral Fellow, he launched NeuroAnalogICs, a start-up company in the neuroengineering space.

Chi-Sang Poon (F'XX) received the Ph.D. degree from the University of California, Los Angeles.

He was Visiting Scientist at Biologie Fonctionnelle du Neurone, CNRS, France, in 1994. He is Principal Research Scientist, Harvard-MIT Division of Health Sciences & Technology, Massachusetts Institute of Technology, Cambridge. He is on the Editorial Board of *Behavioral and Brain Functions*. His main research area is in neurophysiology and neuroengineering. He has also made many contributions in signal processing, system biology, and social sciences.

Dr. Poon is a Fellow of the American Institute for Medical and Biological Engineering (AIMBE).