# Mixed Signal Neural Circuits for Shortest Path Computation

Nasir Shaikh-Husin and Jack L. Meador
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752

## Abstract

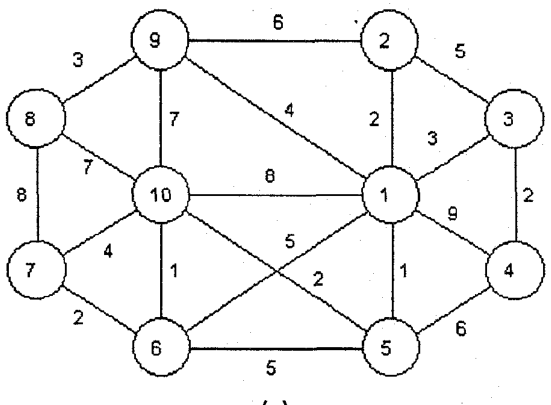
*The objective of the graphical shortest path problem is to discover the least cost path in a weighted graph between a given source vertex and one or more destinations. This problem class has numerous practical applications including data network routing and speech recognition. This paper discusses the hardware realization of a recurrent spatiotemporal neural network for single source multiple-destination graphical shortest path problems. The network exhibits a regular interconnect structure and uses simple processing units in a combination which is well suited for VLSI implementation with a standard fabrication process.*

## 1 Introduction

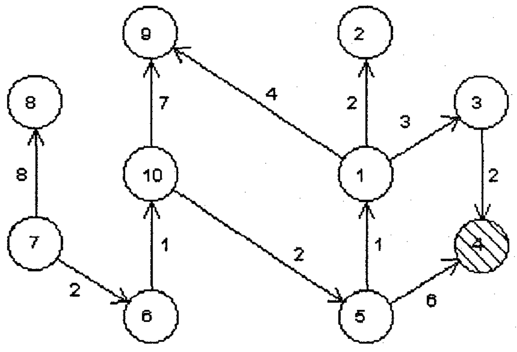
The determination of shortest paths in weighted graphs has numerous practical applications including telecommunications routing, transportation, robotic path planning, communications systems, and VLSI design automation. A special case of the problem class is to determine the shortest path from a single source to a single destination. Hopfield neural networks have been successfully used to solve this problem [1], [2], [3].

The single-source shortest path problem involves finding the shortest path from one source to *all* reachable destinations. Given a directed graph with edges assigned a number representing a cost associated with traversing that edge, the problem is to determine which edges should be traversed to get from a source node to all other nodes with least cost. A symmetric sample graph and the solution to the problem is shown in Fig. 1.

The single-source multiple-destination problem is computationally more difficult than the single-source single-destination problem and requires a dynamic programming optimization approach. The true cost of a single-source solution can be expressed as a sum of all source-destination path costs. There is no known way to express this cost as a quadratic expression over the simple adjacency matrix representation commonly used by Hopfield network graph optimization methods.



(a)



(b)

**Fig. 1. (a) Sample graph with symmetric costs. (b) Solution to sample graph. Node 7 is the source. The shaded node is the last one visited.**

A new approach using a recurrent neural network for solving single-source shortest path optimization has recently been proposed [4]. The new method exploits both temporal and spatial network properties in a manner fundamentally different than the Hopfield network approach. The traditional Hopfield approach combines conflicting costs and constraints into a single energy function that is minimized via gradient descent. The new spatiotemporal approach encodes constraints as a spatially distributed energy and costs as independent time delays incurred while network state follows a conservative trajectory. This approach yields a recurrent neural network guaranteed to converge to exact solutions of single-source shortest path problems. More importantly, the new approach can do this using a much simpler interconnect architecture: one which grows in size only with the square of the graph size. It is thus more easily implemented in VLSI for practical applications.

## 2 Circuit Architecture

Fig. 2 shows the architecture of the least-cost path processor for a five-node problem. Basically, it consists of five rows of almost identical structure. Each row consists of four identical processing elements. The other component of the row is essentially an OR gate. If enabled, a processing element (PE) will try to fire (i.e. produce a HIGH output.) The PE that fires first in a given row (the winner) also causes the OR gate to output a HIGH signal. The OR output is fed back to all PEs in that row. This signal inhibits the rest of the PEs from firing, hence making sure that only a single PE fires in a particular row. With this mechanism, only a single incoming edge is allowed for each node in the final solution. The output of the OR gate is also routed to all the PEs in the corresponding column. Here, the signal acts as the enable flag; the PEs must wait for the signal before they can start their computation. PEs already inhibited by the OR gate in the respective row are of course precluded from firing.

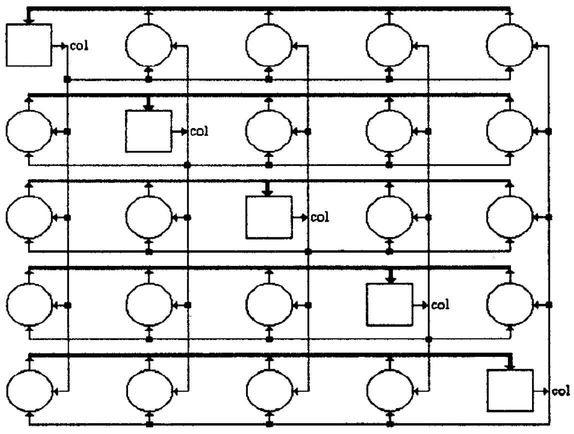


Fig. 2. Block diagram of least-cost path processor.

Each PE is made up of a 6-bit register, a D/A converter, a comparator, a latch, and an AND gate. Fig. 3 shows the block diagram of PE circuitry. The D/A converter is a 6-bit, current mode D/A converter. It consists of six scaled current mirrors, with an integrating capacitor as load. Each current mirror can be turned on or off by a PMOS switch. By regulating the total currents that flow, the time for the capacitor to discharge to a certain level can be varied. Indeed, this is how the cost of a path is encoded. The 6-bit register holds the digital input to the D/A converter, hence its content determines the cost assigned to the PE. As stated earlier, the column signal dictates when cost computation should begin, if not prevented by the row signal.

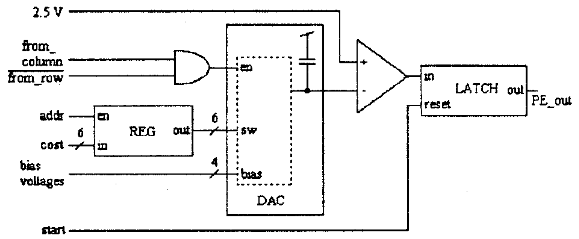


Fig. 3. PE block diagram.

The output of the D/A converter is connected to a comparator. The comparator output becomes HIGH when the output of D/A converter dips below 2.5 V. The latch circuit will catch the HIGH signal and will latch into it, signifying that the PE has fired. A global *start* signal is connected to the latch to make sure that all PEs are initialized to a LOW output.

## 3 Circuit Implementation

In this section, the circuits used for processor subsystems are described.

### 3.1 D/A Converter

The 6-bit current-mode D/A converter (Fig. 4) basically consists of six scaled current mirrors. Since accurate current scaling is essential, cascode current mirrors are used instead of the basic mirrors. The currents are scaled from a maximum of 16 mA to a minimum of 16 mA/$2^5$ = 0.5 mA. Each current mirror is connected in series to a PMOS transistor, which acts as a switch. If the gate of the PMOS transistor is LOW, then current is allowed to flow; otherwise no current will flow in the current mirror.

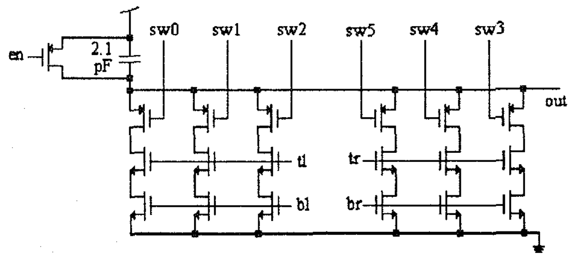


**Fig. 4. D/A converter circuit.**

To minimize circuit area, minimum sized transistors (i.e. W/L is 3 μm/2 μm) are used for the mirror that handles the minimum current (0.5 mA.) If normal scaling is used, the transistors in the mirror that handles the maximum current requires W/L ratio of 96 μm/2 μm. In order to reduce area, the six current mirrors are split into two sections, each with a set of three mirrors. The section that handles the larger currents (16, 8, and 4 μA) are biased directly from the external, 16 mA current source. The bias circuit is shown in Fig. 5. The external current source is then indirectly used to bias the other current mirror section. First, the external source is reduced by half. Although the currents at both ends of the bias circuit differs by a factor of two, the same W/L ratio is used. Specifically, for the current mirror on the right hand side, W/L ratio of 48 μm/2 μm is chosen for the transistors to conduct 16 mA; for the current mirror on the left, the same W/L ratio is used to facilitate current flow of 8 mA. With this scheme, the current mirrors that conduct 0.5, 1, 2, 4, 8, and 16 μA are assigned widths of 3, 6, 12, 12, 24, and 48 μm respectively.

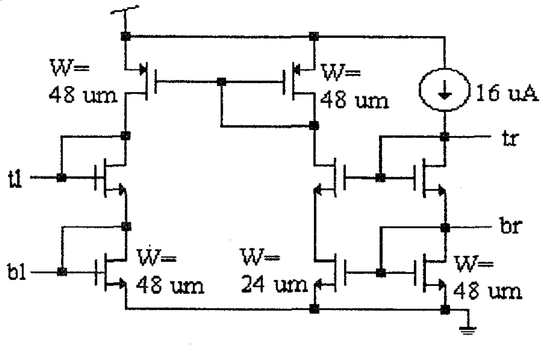


**Fig. 5. Circuit to bias all D/A converters.**

Depending on the signal at the gate of the respective PMOS switches, total currents that flow can be regulated from a maximum of 31.5 mA (when all six switches are on) to a minimum of zero current. A six-bit register is used to hold the bit patterns which control each D/A converter. Since each bit must be applied for quite a long time, a static register is required. A standard register circuit is used.

## 3.2 Latch

Fig. 6 shows the latch circuit. The function of the latch is to produce a permanent HIGH output when the comparator output becomes HIGH. An *inhibit* signal is connected to all latches in the same row to make sure only a single latch 'fires.' The *start* input is used to initialize all the latches to LOW.

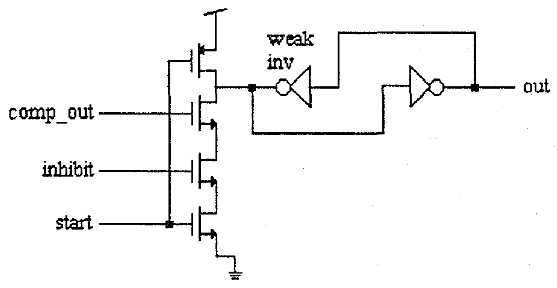


**Fig. 6. Latch circuit.**

## 4 Simulation Results

Cost computation is done in terms of time taken by the D/A converter to discharge load capacitor from VDD to 2.5 V. Since the load capacitor is 2.1 pF, ideally a PE will fire after time t = (2.5*2.1)/I μs, where I is total current in μA. Consequently, path costs has a reciprocal relationship with the current. Since the current is linearly proportional to register content (actually linear to the complement of register content,) the costs also has a reciprocal relationship with the value of the stored digits. Fig. 7 shows the relationship between register content and the associated path cost. For each code, both the ideal time to fire and the actual time from circuit simulation are plotted.

All simulations are done through Hspice program. To ensure simulation result is as accurate as possible, the Hspice netlist is obtained directly from the extraction of actual circuit layout. From Fig. 7, it is clear that the reciprocal relationship holds. Although not shown in Fig. 7, it is obvious that a PE assigned 111111 bits (digit 63) will never fire because no current is available to discharge the capacitor. The code can be assigned to any PE to indicate infinite cost.

To check proper circuit operation, a simulation for solving a sample problem was run. Fig. 8(a) shows the cost assignments. PEs in row 1 are assigned infinite cost

(63) because node 1 is the source. PEs in row 2 and column 2 are assigned infinite cost because node 2 does not participate in this problem. Using Fig. 7, appropriate code for PEs are shown in Fig. 8(b). The simulation result is shown in Fig. 9, which shows that the processor works as expected. Since node 1 is the starting node, all PEs in column 1 begins cost computation immediately after *start* signal is active (not shown.) Since the PE in row 4 has the lowest cost, it fires first. When this PE fires, the PE in row 5, column 4 begins its computation and now competes with that in column 1. This is indicated by the discharging at the D/A converter output for the respective PEs (signals X2_306 and X2_159) respectively. The simulation correctly shows that the former PE wins. Finally, the PE in row 3, column 1 fires. Hence, the solution to the sample problem is from node 1 to node 4 to node 5 and from node 1 to node 3.

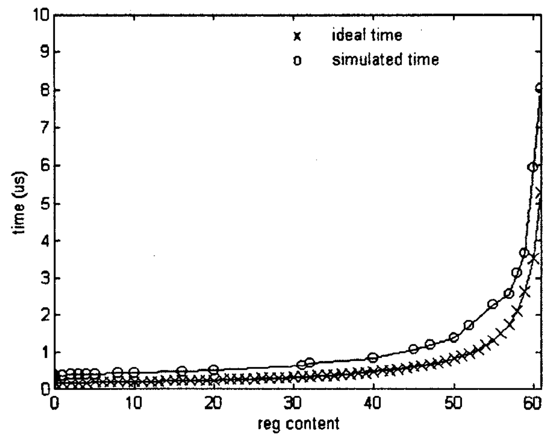
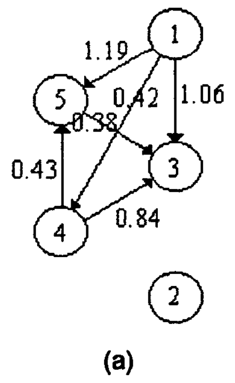


**Fig. 7. Plot of time to fire against register content.**

## 5 Conclusion

A least cost path processor based on recurrent spatiotemporal network has been designed. The circuit structure is very regular, hence works very well for VLSI implementation. The current implementation is targeted for a standard 2 micron fabrication technology. With a 40 pin, MOSIS tiny chip, a processor for solving a 5 node graphical shortest path problem will fit within 1350 x 1200 micron$^2$ bounding box. Hspice simulations shows that this processor functions correctly. Future work will focus on improving circuit accuracy and solving larger problems.



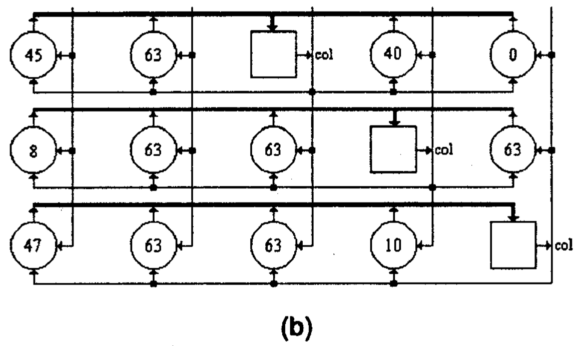


**Fig. 8. (a) Sample problem. Node 1 is the source node. No connection signifies infinite cost. (b) PE code assignments for solving sample problem. Other PEs (in rows 1 and 2) are assigned code 63.**

**References**

[1] Mehmet Ali, M.K. and Faouzi, K. 1993. Neural Networks for Shortest Path Computation and Routing in Computer Networks, *IEEE Trans. on Neural Networks*, 4:6, 941-953.

[2] Cavalieri, A., Di Stefano, A. and Mirabella, O. 1994. Optimal Path Determination in a Graph by Hopfield Neural Network, *Neural Networks*, 7:2, 397-404.

[3] Rauch, H.E. and T. Winarske, Neural Networks for Routing Communication Traffic, *IEEE Cont. Syst. Mag.*, pp. 26-30, Apr. 1988.

[4] Meador, J. L., "Spaciotemporal Neural Networks for Shortest Path Optimization," in *Proc. Of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 801-804, 1995.

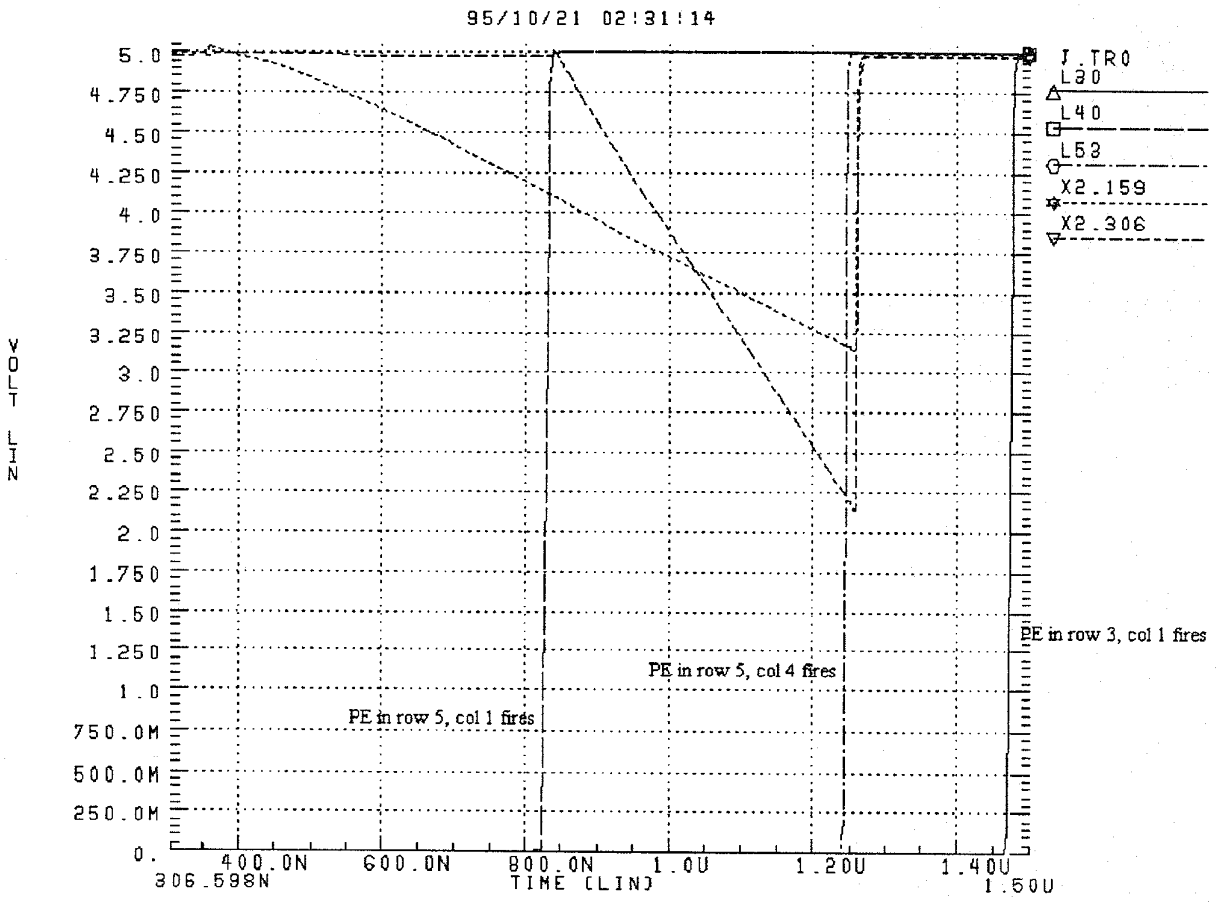


Fig. 9. **Hspice output showing proper operation of least cost path processor.**