SIMPLE PULSE ASYNCHRONOUS STATE MACHINES

Jeffrey Miller [1] *Woodward Yang* [2]
Division of Applied Sciences, Harvard University, USA
[1]miller@eecs.harvard.edu [2]woody@eecs.harvard.edu

ABSTRACT

Pulse computation is a hybrid of conventional analog and digital techniques which encodes and processes information in the time domain. In this paper we present a new technique for pulse computation with simple pulse-driven, asynchronous finite state machines. Asynchronous FSMs are robust and timescale-independent, and can be used as building blocks for constructing complex pulse processors. The notion of state within such processors is considerably simpler than in previous (neuromorphic) pulse computation elements. We describe two simple state machines, pulse matchers and pulse differencers, which we have implemented in CMOS 2μm VLSI. Finally, we discuss some simple image-processing applications to which these devices may be applied.

1. INTRODUCTION

Pulse computation [6, 10] is a novel analog/digital hybrid which uses the time domain to achieve large dynamic range, reasonable noise-insensitivity, and a new set of elementary operations with the possibility of elegantly performing many complex signal-processing tasks. Pulse streams, like digital signals, are binary-valued in voltage. Information is transmitted discretely by individual pulses. Pulse streams, however, are unclocked and pulses may arrive at arbitrary times; and pulses may be generated by analog processes within sensors, such as an integrate-and-fire driven by a continuously varying analog photocurrent.

This computational domain may hold the possibility of discovering silicon analogues to the enormously complex computational operations which the biological nervous system performs (see eg [7, 4, 8, 1].) For this reason, much attention has been paid in earlier work to accurate modelling of biological neural tissue.

Our starting point in this paper, however, is an exploration of more abstract elementary computational methods *in the pulse domain*. We propose a set of elementary operations, similar to the Boolean gates of digital logic, with which useful computations may be performed (robust comparison, simple arithmetic operations,) and which can easily be cascaded to form more complex processing structures.

Central to our processing paradigm is the incorporation of digital state as a useful, flexible abstraction of the complex characteristics of neural tissue. Many early neuromorphic procesing elements [4] did not incorporate a long-term state. Synaptic weightings [7] are one attempt to add trainable, very-long-term storage to the neuromorphic paradigm, but forfeit the flexibility of state possessed by, for example, a digital computer, in which a simply-described system state may change rapidly and discontinuously. The simple definition of state in digital computers makes it very easy to write programs and devise algorithms for information processing; in this paper we investigate a similar abstraction of state in the pulse, or neuromorphic, domain.

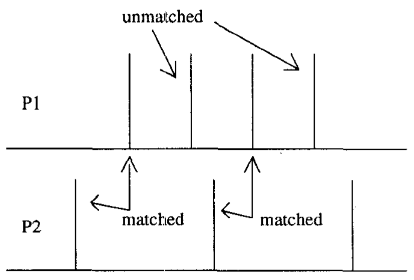

Figure 1. Matched and unmatched pulses.

2. MATCHING AND DIFFERENCING

The foundation of our devices is the operation of pulse matching. Given two signal lines P1 and P2, we can *match* the signals by pairing a pulse on P1 with a pulse on P2. A pulse on P1 is unmatched, by contrast, if another pulse arrives on it and there are no pulses on P2 during the interim time period. Figure 1 illustrates matched and unmatched pulses.

Now we can define two elementary functional units. A *matcher* has two outputs, O1 and O2, each of which pulses only if an matched pulse arrives on inputs P1 or P2 respectively. A *differencer* is a matcher's dual.

A state diagram for the pulse matcher is shown in figure 2. In contrast to a conventional FSM state diagram, state transitions for this machine occur when pulses (quick low-high-low voltage patterns) arrive on inputs P1 or P2. (Because a pulse is a time sequence of boolean inputs, a pulse-driven state machine actually consists of six other (hidden) states in addition to the two visible states; however the behavior of the state machine is expressed correctly, and more clearly, by means of pulse state transitions.)

In this device, *matched* input pulses produce pulse outputs. Output (a pulse on line O1 or O2) is shown as **[O1]** or **[O2]**; so if the machine is in P1-mode (state **S1**) and receives an input pulse on P1, no output is produced; but, if the pulse arrives instead on P2, an output pulse is produced on **O2** and the machine transitions to P2-mode (state **S2**.)

To summarize, in P1-mode the matcher will produce a pulse output (on O2) if and only if it receives a pulse on P2, and will immediately transition to P2-mode. The device's

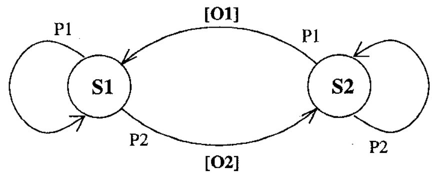

Figure 2. Matcher: Simplified State Diagram.

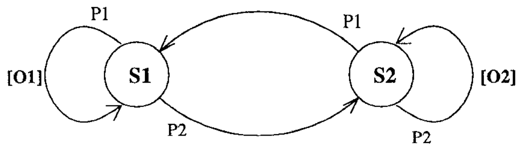

Figure 3. Differencer: Simplified State Diagram.

behavior in P2-mode is analogous. Therefore such a device outputs pulses only when matched input pulses are received.

We can think of the matcher as a model of a neuronal system with pulse-controlled refractory periods. The matcher consists of two seperate "axon segments," or pulse transmission lines: P1 to **O1**, and P2 to **O2**. When a pulse propagates through either line, the transmission of additional pulses on that line will be inhibited until a pulse arriving on the other line terminates the matcher's refractory period.

The precise function of a matcher is determined by the encoding method used. If value is encoded as local frequency density, for example, then the matcher's output signal is $\min(P1, P2)$.

This is only the beginning of possible applications. Matchers may also perform pattern recognition and comparison of arbitrary pulse trains robustly (without precise time-synchronization), qualitatively determining whether a time-section of one given pulse signal corresponds well with another. If the majority of pulses in the first signal can be matched to pulses in the second, and vice-versa, the two signals can be assumed to be very similar; even if they are not precisely in phase (coincident) and even if a few noise pulses appear, the two signals will still produce a strong matcher output. Signal comparison with this technique resembles the tapped-delay-line pulse-stream correlation used by Lazzaro [2] for auditory localization, with the matcher corresponding to the coincidence-detecting silicon neurons. We are currently investigating this signal-comparison process in detail.

Now consider the simplified state diagram of the differencer, in figure 3. If the machine is in 'P1-mode' (state **S1**), it will transition to 'P2-mode' (state **S2**) if a pulse is received on input P2, and remain in **S1** either if nothing happens, or if a pulse is received on input P1. Output (a pulse on line O1 or O2) is shown as **[O1]** or **[O2]**; so if the machine is in **S1** and receives a pulse on P1, it will output a pulse on O1. Therefore this device outputs pulses only when unmatched input pulses are received.

Again, we may think of the differencer as a simplifying abstraction of a biological neural system. Here the model is two coupled axon segments with cross-coupled inhibitory synaptic connections. A pulse on one segment will inhibit (cancel) the next pulse which attemps to pass along the other segment. Additional pulses on the first segment, however, may pass without impediment.

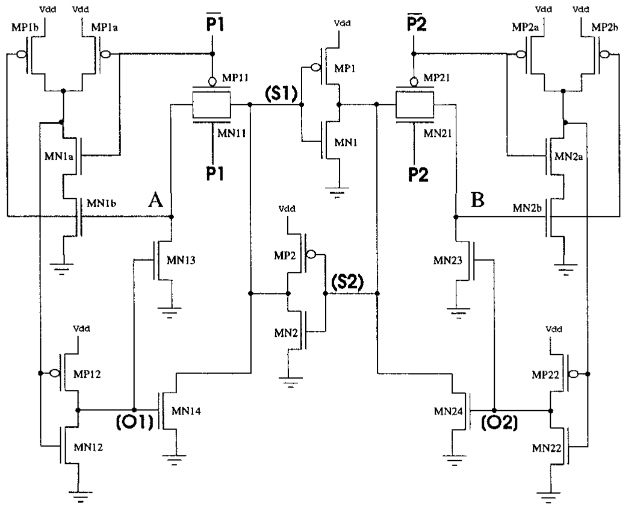

Figure 4. Matcher circuit.

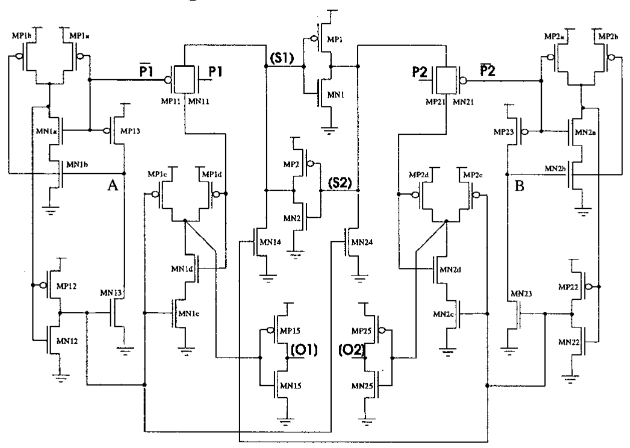

Figure 5. Differencer circuit.

3. IMPLEMENTATION

We simplified state diagrams by requiring that state transitions occur on pulses, not arbitrary individual clock edges. We can simplify the device implementation in the same way. Although states cannot be eliminated, we can use dynamic nodes to hold "transient states" that occur during individual pulses. This follows from our assumption that individual pulses are narrow ($< 1\mu$sec).

A circuit diagram of the matcher is shown in figure 4. The matcher is the simpler of the two devices to implement in silicon because we need not gate the pulse output based on the current state.

The base of the matcher is an SRAM cell constructed from cross-coupled inverters (transistors MP1, MP2, MN1, MN2) to hold the state. We have marked **(S1)** and **(S2)** on the diagram; **(S2)** high means the circuit is in P2-mode. In P2-mode, a pulse arriving on P1 first opens, then closes the left-side pass-gate (MP11, MN11). This charges the dynamic node A high. When P1 falls low again (at the end of the incoming pulse,) the output of the NAND gate formed by (MP1a, MP1b, MN1a, MN1b) then drops low; this signal is inverted by (MP12, MN12) onto O1. An output high on

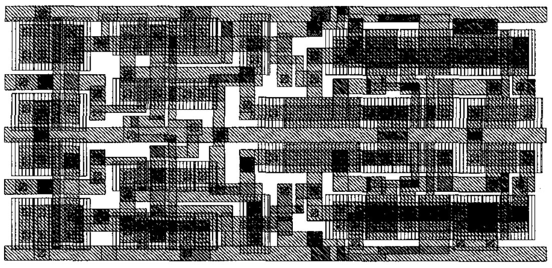

Figure 6. Layout of matcher cell.

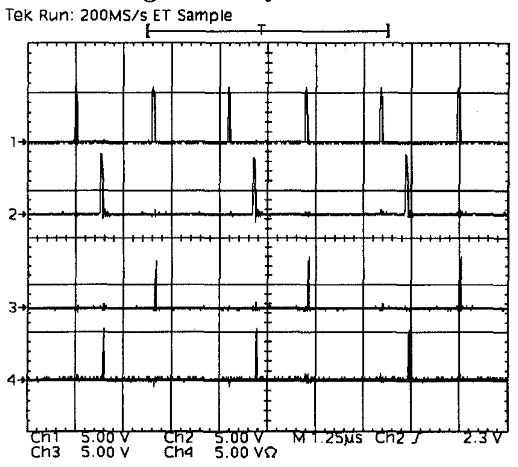

Figure 7. Oscilloscope traces from matcher. 1 and 2, inputs; 3 and 4, outputs

O1 discharges the dynamic node through MN13, causing the system, including O1, to reset. The result is a brief voltage pulse, **[O1]**, which also (through MN14) sets the SRAM to state **S1**. If the incoming pulse had instead been on P2, the right-side pass-gate (MP21, MB21) would have opened; but since the dynamic node B would not have been raised high no output nor change of state would have been produced.

For the differencer the circuit must be slightly more complex. Logic requirements force us to generate a pulse any time an input on either P1 or P2 appears. We use this reset pulse to set the new state of the machine. Then, depending on the old state, we gate the reset pulse to produce **(O1)** or **(O2)** if necessary. So, although the cross-coupled inverter remains, now the dynamic node A between MP13 and MN13 is set high for every input P1. When P1 falls low a reset pulse is produced by MP12 and MN12. An input P1 will always set P1-mode, since the reset pulse pulls **S2** low; but the output of the NAND gate made up of (MP1c, MP1d, MN1c, MN1d) will only go low during the reset pulse if the machine was *already* in P1-mode (**S1** high) when the prior state of the system was latched by the incoming P1. Finally an output inverter (MP15, MN15) restores the proper polarity of the outgoing pulse.

We implemented matcher and differencer circuits in CMOS VLSI (MOSIS 2 μm nwell) for testing. The differencer cell measures 192 μm by 72 μm; the matcher is 145 μm by 72 μm. Both devices function correctly (see figs 7,8) and have been demonstrated to perform all of the operations outlined above.

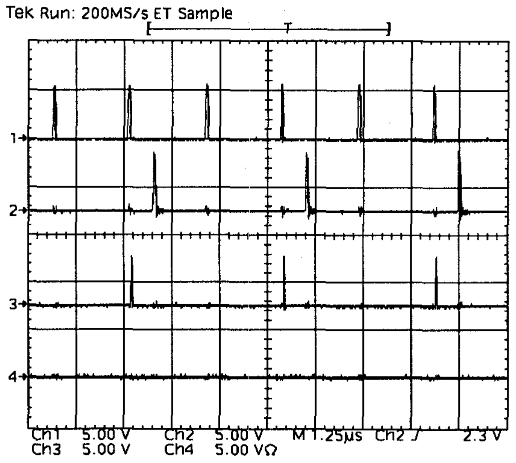

Figure 8. Oscilloscope traces from differencer. 1 and 2, inputs; 3 and 4, outputs.

4. COMBINING PULSE STATE MACHINES

Matcher and differencer are the most elementary of binary pulse state machines. More complicated devices with more numerous states or inputs have also been designed. Yet it is useful to think of matcher and differencer as building blocks which can be chained together, like Boolean logic gates, to form complicated devices easily.

Let's consider a straightforward local frequency encoding, to illustrate some simple combinations of pulse asynchronous state machines. We'll imagine that we have pulse trains generated by a photosensor [9] in which pixel pulse frequency is proportional to light intensity.

A differencer alone, when connected to adjacent pixels P1 and P2, forms an edge detector. The differencer's two outputs give the signed magnitude of the difference in average pulse frequencies of the inputs.

By combining edge detectors, we can look for peaks or valleys. For example, we might connect differencers between adjacent pixels P1 and P2, P2 and P3. P2 is a peak if it is more intense than either P1 or P3. Attach a matcher to the appropriate outputs of the differencers, as shown in the figure, and the resulting single output will pulse only if there is an intensity peak present, with an average frequency which reflects the strength of the peak:

$$f_o = \begin{cases} \min\{(f_2 - f_1), (f_2 - f_3)\} & f_2 > f_1, f_2 > f_3 \\ 0 & \text{otherwise} \end{cases}$$

We have also shown one possible state diagram of the peak detector (fig 10) to indicate the complexity of the function, and the simplifying effect of constructing large devices with 'building block' PASMs. Finally, figure 11 shows an oscilloscope trace of the peak detector in operation, in the case that a frequency peak is present. As expected, the output line pulses, indicating the presence of a peak.

5. CONCLUSION

We have presented a robust, extendable, generalizable paradigm for pulse processing with pulse asynchronous state machines. Two simple devices, differencer and matcher, have been shown to perform useful elementary processing operations and implemented in VLSI. These devices may be cascaded to perform arbitrarily complex tasks; one example, image feature detection, has been discussed.

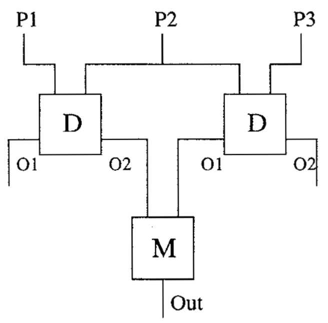

Figure 9. Peak detector: block diagram.

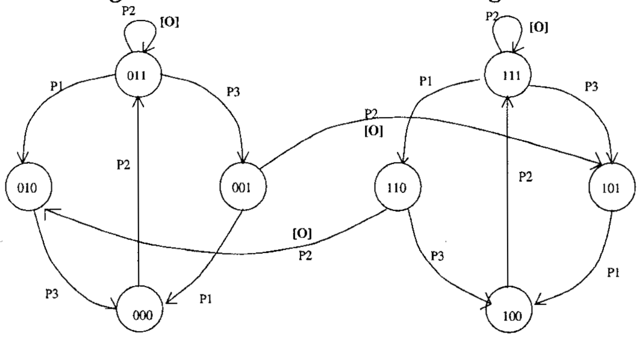

Figure 10. Peak detector: state diagram.

ACKNOWLEDGEMENTS

This work is supported in part by an NSF Young Investigator Award (MIP-92-57964), an Army Research Office Young Investigator Award (DAAL03-92-G-0375), a grant from the Joint Services Electronics Program (N00014-89-J-1023), and a gift from Hamamatsu Photonics.

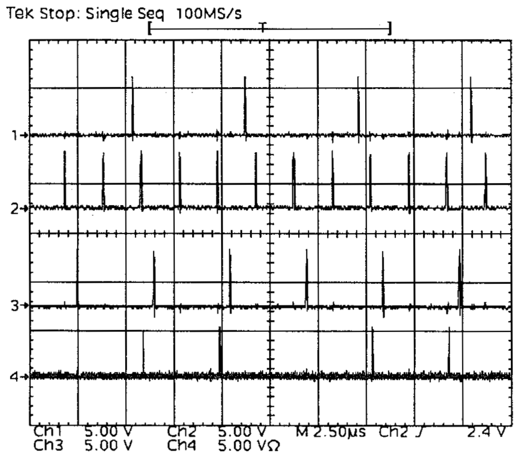

Figure 11. Oscilloscope traces from peak detector. Shows peak detection. 1, 2, 3, inputs; 4, output

REFERENCES

[1] Douglas R, Mahowald M., Mead C. "Neuromorphic Analog VLSI". *Annu. Rev. Neurosci.* **18**:255-81, 1994.

[2] Lazzaro, J., Mead, C. "A silicon model of auditory localization." *Neural Computation* **1**: 41-70, 1989.

[3] Mahowald M. "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function." Ph. D. Thesis, Computation and Neural Systems, California Institute of Technology. 1992.

[4] Mead C. *Analog VLSI and Neural Systems.* Reading, MA: Addison-Wesley, 1989.

[5] Mortara A., Vittoz E., Venier P. "A Communication Scheme for Analog VLSI Perceptive Systems." *IEEE J. Solid-State Circuits.* **30**:660-669, 1995.

[6] Murray A. , Tarassenko L. *Analogue Neural VLSI* London: Chapman & Hall, 1994.

[7] Northmore D., Elias J. "Temporal Processing in Artificial Dendritic Tree Neuromorphs". *Proceedings of The Twenty-ninth Conference on Information Sciences and Systems* pp. 345-349. Johns Hopkins University, 1995.

[8] Sarpeshkar R., Bair W., Koch C. "Visual motion computation in analog VLSI using pulses". *Neural Information Processing Systems 5* Eds. S Hanson, J Cowan, C Giles. pp 781-788. San Mateo: Morgan Kaufman, 1993.

[9] Yang W. "A Wide-Dynamic-Range, Low-Power Photosensor Array". *ISSCC Digest of Technical Papers,* pp.230-231, 1994.

[10] Zaghloul M.E. , Meador J.L., Newcomb, R.W., eds. *Silicon Implementations of Pulse Coded Neural Networks* Boston, MA: Kluwer Academic Publishers, 1995.

SIMPLE PULSE ASYNCHRONOUS STATE MACHINES

Jeffrey Miller [1] *and Woodward Yang* [2]

Division of Applied Sciences, Harvard University, USA
[1]miller@eecs.harvard.edu [2]woody@eecs.harvard.edu

Pulse computation is a hybrid of conventional analog and digital techniques which encodes and processes information in the time domain. In this paper we present a new technique for pulse computation with simple pulse-driven, asynchronous finite state machines. Asynchronous FSMs are robust and timescale-independent, and can be used as building blocks for constructing complex pulse processors. The notion of state within such processors is considerably simpler than in previous (neuromorphic) pulse computation elements. We describe two simple state machines, pulse matchers and pulse differencers, which we have implemented in CMOS 2μm VLSI. Finally, we discuss some simple image-processing applications to which these devices may be applied.