Digital Memcomputing Machines

Massimiliano Di Ventra*, Fabio L. Traversa* Email: diventra@physics.ucsd.edu ftraversa@physics.ucsd.edu
*Department of Physics, University of California San Diego, La Jolla, CA 92093-0319 USA

***Abstract*—In this contribution we will discuss the *digital*, hence scalable, version of memcomputing machines. These are non-Turing machines that use memory to both process and store information at the same physical location. We will introduce their mathematical definition and provide as an example their implementation of an inverse three-bit sum gate using *self-organizable logic gates*, namely gates that organize dynamically to satisfy their logical propositions.**

A BRIEF SURVEY

We have recently introduced the concept of Universal Memcomputing Machines (UMMs) [1], as a collection of interconnected memprocessors (processors with memory) able to process and store information on the same physical location. UMMs are a class of non-Turing machines with distinct properties. In particular, they support intrinsic parallelism of the type in which all (mem-)processors act simultaneously on all the data and, through the *topology* of their network, they are able to compress more information than that allowed by their disjoint elements. We named this last property *information overhead* [1].

It was shown that these machines have the same computational power of non-deterministic Turing machines, but unlike the latter ones they can actually be fabricated. In Ref. [2], for instance, a machine able to solve the NP-complete version of the subset-sum problem was built using standard microelectronic components. However, that particular machine is fully analog and therefore difficult to scale up to an arbitrarily large number of processors without the control of noise.

On the other hand, UMMs are not limited to analog machines. Rather, one can define digital versions that map integers into integers and, as a consequence, are more robust against noise and thus easier to scale up like the modern digital computers.

The definition of digital UMMs (DMMs) is the eight-tuple

$$DMM = (\mathbb{Z}_2, \Delta, \mathcal{P}, S, \Sigma, p_0, s_0, F)\,, \tag{1}$$

where (although not strictly necessary) we consider the range of $\mathbb{Z}_2 = \{0, 1\}$. Generalization to any finite number of states is trivial. Δ is a set of functions

$$\delta_\alpha : \mathbb{Z}_2^{m_\alpha} \backslash F \times \mathcal{P} \to \mathbb{Z}_2^{m'_\alpha} \times \mathcal{P}^2 \times S\,, \tag{2}$$

where $m_\alpha < \infty$ is the number of memprocessors used as input of (read by) the function δ_α, and $m'_\alpha < \infty$ is the number of memprocessors used as output (written by) the function δ_α; $\mathcal{P}$ is the set of the arrays of pointers p_α that select the memprocessors called by δ_α and S is the set of indexes α; Σ is the set of the initial states written by the input device on the computational memory; $p_0 \in \mathcal{P}$ is the initial array of pointers; s_0 is the initial index α, and $F \subseteq \mathbb{Z}_2^{m_f}$ for some $m_f \in \mathbb{N}$ is the set of final states.

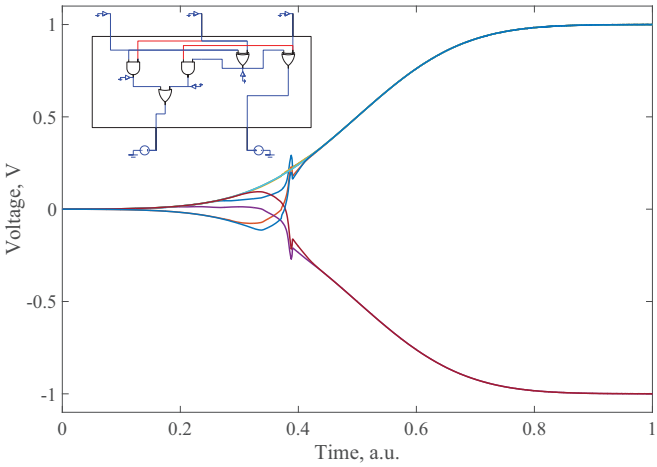

Fig. 1. Voltage as a function of time of all literals of the self-organizable logic circuit represented in the inset that performs the addition of three bits.

A practical realization of such machines can be done using a network of *self-organizable logic gates* [3]. These are boolean gates that self-organize dynamically (namely subject to an appropriate voltage or current bias) into the correct solution that satisfies the gate. In turn, these gates can be made using, e.g., resistors with memory (memristors).

A simple example is provided in Fig. 1. The circuit in the figure inset represents a three-bit adder. However, the inputs of this circuit are the voltage generators at the "output" of the adder. Therefore, feeding the circuit with the generators, the self-organizing gates (OR, AND and XOR) *dynamically* adjust to satisfy the boolean relations. In Fig. 1 the dynamic of all voltage points of all literals is shown. Initially, the generators are set to 0 and turn on gradually. During the transient the other voltages vary to find the equilibrium of the circuit that corresponds to the solution of the adder.

ACKNOWLEDGMENT

This work has been supported by the Center for Memory and Recording Research at UC San Diego.

REFERENCES

[1] F. L. Traversa and M. Di Ventra, "Universal memcomputing machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 2702 – 2715, 2015.
[2] F. L. Traversa, C. Ramella, F. Bonani, and M. Di Ventra, "Memcomputing NP-complete problems in polynomial time using polynomial resources and collective states," *Science Advances*, vol. 1, no. 6, p. e1500031, 2015.
[3] F. L. Traversa and M. Di Ventra, "Polynomial-time solution of prime factorization and np-hard problems with digital memcomputing machines," *submitted*, 2015.