Design and Performance Analysis of Nanoscale Content-Addressable Memories

Bryan Davis, Jose Principe, and Jose Fortes

Electrical and Computer Engineering, University of Florida, Gainesville, FL, 32611, USA

Abstract — This paper proposes a nanoscale content addressable memory (CAM) architecture. The type of CAM considered is adept at handling the irregular patterning of nanoscale components but requires an encoding scheme to be useful. We show how a randomly configured interface between the micro and nanoscales can accomplish this encoding. This new CAM eliminates some of the supporting microscale components required by the original crossbar architecture, at the cost of reduced storage capacity. If the projections made by Kuekes et al prove correct about the potential size of their memory, this CAM may be a valuable alternative in some applications.

Index Terms — Memory architecture, memory fault tolerance, molecular electronics, nanotechnology, semiconductor devices, stochastic logic circuits, stochastic systems.

I. INTRODUCTION

The promise of nanotechnology is redirecting the technology road map for CMOS based VLSI design as well as creating new trends in computer architecture. One of the fundamental challenges is how to interface the newly developed nanocomponents to the microworld of present computing devices. The chiasm occurs not only because of the scale mismatch, but also because conventional device design is achieved by miniaturizing a precise interconnect and device geometry (top down), while the present designs at the nanoscale are bottom up, and are unable to yield precise geometry or location of components. In this paper, we use a stochastic interface between nano and microscales, and well-established principles from Content Addressable Memories (CAMs) to quantify the specifications achieved with a stochastic interconnect for associative memories.

CAMs are a very well studied type of memory structure, very different from conventional computer (listing) memory. CAMs store globally (in the weights of the CAM) vectors of items, instead of one item per memory location. They do not require addressing because the content is retrieved by presenting one of the inputs used during training of the memory. We argue in this paper that the distributed characteristics of CAMs as well as their robustness to noise in the weights or inputs are a fundamental tool to link the nanoscale, characterized by millions of imprecise elements, to the well-organized and patterned microscale that is mastered in CMOS chip design.

In this paper, we propose a specific nanoscale CAM design, called NanoCAM. It uses a structure very similar to that used by Kuekes et al in [1]-[2], however, the design parameters and functional use of the structure are very different.

II. ARCHITECTURAL OVERVIEW

Instead of attempting to use the Molecular-Wire Crossbar Memory (MWCM) in [2] as a listing memory, we propose here to utilize the theory of associative memories. The main reason is the intrinsic error correcting properties of CAMs to defects in the matrix or noise in the inputs. The content-associative memory architecture described herein consists of a core nanoscale molecular wire crossbar memory (MWCM) [1], directly implementing the basic design of the LernMatrix described by Steinbuch. [3] To provide the microwire-to-nanowire interface and to provide for the encoding needed by the LernMatrix we use a microwire/nanowire grid. Finally, we will look at the problem of readout and a proposed solution that solves this problem simply. A graphical representation of the overall architecture is shown in Fig. 1.

A. An Implementation of the LernMatrix with the MWCM

The LernMatrix is a CAM that simply consists of a grid of wires with reconfigurable interconnects. The key (or address) is applied to say the horizontal nanowires and the value (or data) is read from the vertical nanowires. A key/value pair is stored in the CAM during learning by forming connections between all combinations of active key and value wires. When a key is applied during recall, we assume that each value nanowire connected during storage will have a current that flows through all of the junctions that connect to active key nanowires. If all keys have the same number of active nanowires, then this number could be used as a threshold to determine whether a value wire should be associated with this key. [1] According to G. Palm, the number of active wires should
be order the range of three to ten, depending on the size of the memory, which means the key and value must be sparse. [4]

Assuming that the nanowire-to-nanowire junction in the MWCM has a relatively high resistance, the MWCM is a perfect implementation of the LernMatrix. Of course, we are still left with two questions: How do we access the nanowires on the MWCM? And how do we solve the problem of sparsity inherent when using the LernMatrix?

For reasons presented in the paper, the keys and values applied to the microscale wires of the NanoCAM need to be presented as complimentary pairs, i.e. for each pair of microwires connected to the MNG, one must be active and the other must be inactive. In addition, the microwire farthest from the MWCM is used to drive the potential of the microwires high for the key side and low for the value side (or vice-versa) as in [3]. In operation, a nanowire will be active unless a microwire (other than the driver) connected to it is inactive.

C. Storing Data in the CAM

Data is stored into the CAM as a list of key-value pairs. During storage, the key and value MNGs perform identical functions (except that they use opposite polarity). Each takes a set of active microwires and converts that into a set of active nanowires. The result is that any microwire key/value will transform into a sparse nanowire key/value needed for the LernMatrix function to work.

Once the key and value have been applied to the microwires of the NanoCAM (as shown in Fig. 1) at an appropriate voltage difference (refer to [1]-[2],[5]) junctions are formed in the NanoCAM between all of the active nanowires.

D. Reading out Values from the Grid

During readout, a key applied to the microwires is transformed by the key MNG into a set of active key nanowires. These active key nanowires are capable of supplying a current through MWCM junctions to the value nanowires. In order to readout the values from the grid, we need to perform two functional tasks: First, we need to perform the LernMatrix operation of decoding the currents on the value nanowires into the most likely set of nanowires. Then we must find the microwire value most likely to encode into that set of value nanowires. A technique described next to do this can be used that works on the principle that the microwires connecting to the value MNG are complementary.

While learning a particular key-value pair, the nanowires selected by the value-side MNG are fully connected to the key-side nanowires in the crossbar memory. The value-side nanowires that were selected are, by design, not connected to any microwire that inactive during training of this key-value pair. Thus, any current that may flow in this microwire is due to connections formed in the MWCM by extraneous items stored in memory. On the other hand, a microwire that was active during training of a particular key-value pair is likely to be connected to at least some of the nanowires selected during learning. The current flowing through the selected value-side nanowires, which are fully connected to the
selected key-side nanowires, will be relatively large. Hence, we would expect that a microwire that was active during the learning of a particular key-value pair would draw more current than a microwire that was inactive. Therefore, the method used to choose which one of a complementary pair of nanowires on the value side corresponds to a given key is to choose the microwire that draws the most current. This current must be measured while all of the other value-side microwires are disconnected.

### III. SIMULATION

The performance of this architecture was simulated in Matlab, by making the following assumptions about the electrical properties of the device: 1) the resistance of the crossbar memory diodes dominates all other resistances; 2) the resistance of the driver/nanowire junction greater than or equal to the microwire/nanowire diode junction; 3) the reverse-bias diode current is 0. The keys and values were chosen from uniform and independent distributions.

In applications using nanotechnology, a high error rate in the components of the memory should be suspected. In this simulation, the defective crossbars are chosen randomly. We can then calculate the performance of the memory in terms of its ability to store information. Using Shannon’s theory, binary data stored in memory has an information content measured in bits. The amount of information stored is a function of the total number of bits stored and the probability of bit error ($P_{err}$) during recall.

In Table 1, we choose design parameters that maximize the number of information bits recalled for a given number of crossbar junctions and a given defect rate. It turns out that the NanoCAM is able to store the most information when a large amount of data is stored in the CAM, but with a high error rate.

<table>
<thead>
<tr>
<th>MWCM junctions</th>
<th>MWCM Defect rate</th>
<th>Capacity (bits)</th>
<th>$P_{err}$</th>
<th>Information (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>90,000</td>
<td>0%</td>
<td>4000</td>
<td>35%</td>
<td>243</td>
</tr>
<tr>
<td>90,000</td>
<td>10%</td>
<td>4000</td>
<td>38%</td>
<td>167</td>
</tr>
<tr>
<td>90,000</td>
<td>25%</td>
<td>4000</td>
<td>43%</td>
<td>56</td>
</tr>
<tr>
<td>300,000</td>
<td>0%</td>
<td>7000</td>
<td>32%</td>
<td>700</td>
</tr>
<tr>
<td>300,000</td>
<td>10%</td>
<td>7000</td>
<td>34%</td>
<td>512</td>
</tr>
<tr>
<td>300,000</td>
<td>25%</td>
<td>7000</td>
<td>40%</td>
<td>210</td>
</tr>
<tr>
<td>900,000</td>
<td>0%</td>
<td>9900</td>
<td>27%</td>
<td>1547</td>
</tr>
<tr>
<td>900,000</td>
<td>10%</td>
<td>9900</td>
<td>31%</td>
<td>1094</td>
</tr>
<tr>
<td>900,000</td>
<td>25%</td>
<td>9900</td>
<td>36%</td>
<td>574</td>
</tr>
</tbody>
</table>

### IV. CONCLUSION

After running the simulations, it is clear that the storage capacity is much smaller than a conventional listing memory implemented in nanoscale components. There are some clear advantages using this memory, however: In order to use this memory as a listing memory, we are require additional hardware—namely a conventional RAM on the address lines in order to map a contiguous address space into a highly disjoint address space that is usable by the memory. The associative architecture described herein does not require additional hardware or an address-scanning phase. It is also tolerant of defects that occur during operation because of its distributed nature. These advantages may outweigh the disadvantage of poor performance if the memory sizes become very large, e.g. $10^6 \times 10^6$ or larger, because the disadvantage of poor performance becomes less of a limitation with increase in memory size, and the advantages become more important. For example, the testing and configuration becomes more time consuming as the memory size increases, as does the size of the external memory needed to store the addresses for the memory proposed by Kuekes et al.

Much more work is needed in this area of research, particularly on the readout of the data, but we believe that the content addressable framework outlined here together with the stochastic error modeling seems very promising to model the micro to nanoscale interface.

### ACKNOWLEDGEMENT

This work was partially supported by NSF grant EIA 0135946.

### REFERENCES


