Abstract—Intrusion Detection Systems (IDS) are designed to interpret intrusion attempts from incoming network traffic. As enhancing the cybersecurity capabilities of mobile products is becoming increasingly important, having IDS capability on these devices can bolster their security. The key operations in an IDS are essentially pattern matching operations. These can be computationally intensive and are one of the most power consuming functionalities in the system. The high computational and high power costs of IDS can prevent their use in mobile device. We present low power, high density memristor crossbar circuits to carry out the pattern recognition tasks in IDS efficiently. We demonstrate that neural network based memristor crossbar circuits can carry out the pattern matching tasks in IDS. Our evaluations show that the proposed approach can be five orders of magnitude more power efficient compared to current commodity processors.

Key words: Intrusion Detection System; string matching; neural networks; memristor crossbars.

I. INTRODUCTION

With the advent of newer technologies, security threats for computer networks have increased dramatically over the last decade. Cyber Security is a perennial problem for organizations such as banks, retail stores, and critical infrastructures like SCADA and power grids. Intrusion detection involves the task of detecting unauthorized and malicious access to the network system. Combating such intrusions is a vital aspect to be considered for stable, safe, and resilient cyberspace [1]. Intrusion Detection Systems (IDS) are designed to interpret intrusion attempts from incoming network traffic and squash their threats. As enhancing the cybersecurity capabilities of mobile products is becoming increasingly important, having IDS capability on these devices can bolster their security.

String matching algorithms are basic components used in intrusion detection that find occurrences of malicious signatures in incoming packets. String searching or string matching algorithms try to find the location where one or several strings (also called patterns) are found within a larger string or text. String matching has application in several areas including intrusion detection in network, bioinformatics, detecting plagiarism, information security, pattern recognition, document matching, and text mining [2].

The key operations in an IDS are essentially pattern matching operations. These can be computationally intensive and are one of the most power consuming functionalities in the system. The high computational and high power costs of IDS can prevent their use in mobile device. Novel circuits that can reduce the power consumption for pattern matching can enable the deployment of IDS on mobile devices.

Memristors [3] have received significant attention as a building block for low power, heavily dense circuit designs. These devices can have a large varying resistance range; for instance the device published in [4] has a high on state resistance ($R_{ON} \approx 125k\Omega$) and a very high resistance ratio ($R_{OFF}/R_{ON}=10^6$). Physical memristors can be laid out in a high density grid known as a crossbar [5] (Fig. 1). Using these devices will produce high density, extreme low-power, neuromorphic hardware that is capable of performing many multiply-add operations in parallel in the analog domain.

As embedded applications require high performance at extreme low power, memristor based circuits could potentially be used in these systems. Besides their area and computational efficiency, the non-volatile nature of memristors can reduce the static power consumption of these systems significantly.

Several recent studies examined memristor crossbar based systems for implementing linearly and non-linearly separable patterns. Hasan et al. [6] demonstrated circuits for the back propagation (BP) algorithm based training of a

![Fig. 1. (a) Memristor crossbar schematic and (b) memristor crossbar layout.](image-url)
cascaded set of memristor crossbars representing a multi-layered neural networks for implementing non-linearly separable functions.

This paper demonstrates low power memristor crossbar based circuits for carrying out the pattern recognition tasks needed in Intrusion Detection Systems (IDS). The rest of the paper is organized as follows: section II examines sneak path issue in memristor crossbar circuits. Section III examines memristor based linear separator designs and section IV reviews the memristor crossbar based IDS proposed in [11]. Section V presents our proposed memristor crossbar based IDS and section VI demonstrate experimental setup and results. Finally in section VII we summarized our work.

II. MEMRISTOR CROSSBAR BASED CIRCUITS

A. Sneak paths in Memory Circuits

Several studies [4,7] have proposed large memristor crossbar arrays as a high density memory design. The circuit diagram typically proposed for this type of memory is seen in Fig. 1. In this design each nanoscale memory element will occupy an area of just $4F^2$ [8] where $F$ is the feature size of the fabrication technique.

The schematic in Fig. 2 illustrates a potential problem with high density memristor crossbars. To read or write memory element at row 2 and column 2, voltages are applied across row 2 and/or column 2. The green colored path is the desired current path for reading the memristor. As there are no access transistors in this design, nothing is stopping current from flowing through other memristor devices (red colored paths). These alternate current flows are generally called sneak path currents [9] and can lead to potential read errors in a large crossbar.

We can observe that the column voltage $V_{o2}$ is a function of only the applied voltages, load resistance and cross-point resistances associated with the second column. As a result we can conclude that if voltages are applied to all the crossbar rows, their voltage drop across the load resistance is not affected by memristors corresponding to other columns. This makes analysis of large memristors simpler. This is not the case with memory circuits as many rows are left floating during reads.

B. Minimizing Impact of Sneak Paths

It is important to note that during the read process in memory circuits, a read voltage is applied only to the desired row, while other rows are kept floating or grounded. Sneak paths problem can introduce read errors in memory circuits. Sneak path can essentially be eliminated by applying voltages to all the rows in a crossbar. Consider the memristor crossbar shown in Fig. 3 where voltages are applied to all the rows. Then for the second column we get

$$\frac{V_{o2}}{R_L} = \sum_{i=1}^{5} \frac{V_i - V_{o2}}{R_{i2}}$$

Or,

$$V_{o2} = \left( \frac{1}{R_L} + \sum_{i=1}^{5} \frac{1}{R_{i2}} \right)^{-1} \left( \sum_{i=1}^{5} \frac{V_i}{R_{i2}} \right)$$

Fig. 2. The schematic for a high density unconstrained crossbar. It displays the target memristor path (green) as well as two possible alternate current paths (red).

Fig. 3. Accessing multiple rows of a memristor crossbar simultaneously.
III. MEMRISTOR CROSSBAR BASED NEURAL NETWORK IMPLEMENTATION

A. Neuron Circuit

We can implement neurons utilizing memristor crossbars. The schematic in Fig. 4 shows the memristor based neuron circuit used in this paper. This example shows a neuron having three data inputs and one bias input. These inputs are represented by voltages of appropriate magnitudes. Each input signal and its complemented signal (same magnitude but opposite polarity) are applied to a column of memristors and at the end of the column a CMOS inverter is connected to evaluate the neuron output. Two memristors, one connected to input signal and another connected to corresponding inverted signal, represent a single synaptic weight of positive or negative value. If the conductance of the memristor connected with input signal is greater than the conductance of the memristor connected with the corresponding inverted signal then that pair of memristors represent a positive weight. If the conductance of the former memristor is less, then the pair represents a negative weight.

\[
\text{If for a neuron, } x_i \text{ is the input and } w_{j,i} \text{ is the corresponding synaptic weight, then } \text{DP}_j = \sum_i x_i w_{j,i} \text{(see Fig. 4) and the neuron output is } y_j = h(\text{DP}_j), \text{ where } h \text{ is the activation function. In our approach we are approximating the activation function using CMOS inverters by applying } V_{dd} = 1V \text{ and } V_{ss} = -1V. \text{ Fig. 5(a) shows the inverter circuit and Fig 5(b) shows its transfer curve. When inputs are applied to the neuron, a value proportional to } \text{DP}_j \text{ appears at the first inverter input. This implies that all the multiplications and summation are carried out in parallel. This approach provides a very efficient way of implementing the activation function in terms of power, speed, and circuit components.}
\]

B. Linearly Separable Classifier Design

We have utilized the memristor crossbar based neuron circuit in Fig. 4 to implement linearly separable functions. All 16 minterms for 4 input functions have been implemented in a single crossbar (Fig. 6). Table I shows the 16 four inputs linearly separable functions used for training. Each neuron was trained to learn one linearly separable function. We used an 11×16 crossbar array (Fig. 6) to classify the 16 four input linearly separable minterms.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>16 Four Inputs Linearly Separable Functions Used For Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>m0 = a’b’c’d’</td>
<td>m4 = ab’c’d’</td>
</tr>
<tr>
<td>m1 = a’b’c’d’</td>
<td>m5 = ab’c’d’</td>
</tr>
<tr>
<td>m2 = a’b’cd’</td>
<td>m6 = a’bcd’</td>
</tr>
<tr>
<td>m3 = a’bcd’</td>
<td>m7 = a’bcd’</td>
</tr>
</tbody>
</table>

Training requires update of the conductance of the memristors in the crossbar. If a voltage greater than the memristor write threshold is applied across a memristor, its conductance increases or decreases depending on the polarity of the device (determined by the fabrication process). To make desirable changes in memristor conductances, we need to apply a voltage of appropriate magnitude and polarity for a suitable duration across the memristor [10]. A detailed SPICE simulation of the crossbar that considered wire resistance showed that the circuit was able to correctly classify the linearly separable functions. Fig. 7 shows the training curve for learning 16 four input minterms.
IV. PREVIOUS WORK ON MEMRISTOR BASED IDS

Bontupalli et al. [11] proposed the implementation of a string matching circuit utilizing memristor crossbars for IDS. A string matching operation is similar to matching a minterm of Boolean variables. Neuron circuits similar to the circuit presented in Fig. 4 were used for string matching. In that system, logic low was represented by -1 V and logic high was represented by +1 V. To implement any four input minterm, the weights associated with normal inputs (not complemented) were maximum positive value (w=1/R\text{on}) and weights associated with complemented inputs were maximum negative value (-w). The summation of inputs multiplied by their corresponding weights would be 4w. If any input other than the input satisfying the minterm is applied, the summation of inputs multiplied by weights would be less than or equal to 2w. The bias input and weight were set such that when the summation of inputs multiplied by weights was greater than 3w, the neuron gave a logic high output and otherwise it gave a logic low output. Fig. 8 shows a neuron implementing the minterm ab’c’d.

A similar circuit for matching longer strings was designed. This study showed that as we increase the length of the string to be matched, the magnitude of the voltage across the load resistor decreases. In a single column crossbar, we can match 112 bits or 14 characters successfully.

V. PROPOSED ARCHITECTURE FOR IDS

If we can train the crossbar for matching a particular string using as high resistance as possible, it will enable lower power operations. Training for matching large strings is not efficient as we need to train with all possible strings of that length.

Fig. 6. Schematic of memristor crossbar circuit for training 16 four input minterms.

Fig. 7. Training curve of 16 four input minterms.

Fig. 8. Neuron implementing minterm ab’c’d.

An advantage of this implementation technique is that it does not require any neuron training circuitry. It only needs circuitry for programming the memristor crossbar for detecting a desired string. The disadvantage of this design is that it requires extra memristors (50% of the memristors are associated with constant inputs). During string matching the memristor crossbar consumes a significant amount of dynamic power as 50% of the memristors are at the low resistance R\text{on}.

Fig. 9. Combining smaller substring matching systems for matching longer substring.
In this study, we designed memristor crossbar based circuits for smaller substring matching following the approach used for learning linearly separable functions described in subsection III(b). Combining these smaller crossbar circuits, longer substring will be matched. This approach is illustrated in Fig 9, where three crossbars have the same number of outputs. These crossbars will be trained such that neuron $i$ of crossbars 2 or 3 will give a logic high output if neuron $i$ of the previous crossbar is high and its own input is matched with the desired substring.

To match a longer string, parts of the string will be matched by multiple versions of the circuit shown in Fig. 9. An AND operation will have to be carried out on those outputs. The AND function can be implemented by training a memristor crossbar based neuron.

In this paper we have examined SNORT [12] based IDS. SNORT does protocol analysis and content matching for detecting malicious packets. The basic operation in SNORT is string matching in the packet header and content.

For searching the desired string in a binary text of length $n$, we need to take a substring of length $m$ from the text starting at position $i$ (where $i=1, 2, \ldots, n-m+1$) and apply it to the crossbar inputs. A logic high output of the neuron indicates a match. Once a match is found, the output of the corresponding neuron will be latched in a flip-flop. Fig. 10 shows the memristor crossbar based tile design for string matching which internally uses the circuit shown in Fig. 9. The overall architecture of the SNORT based IDS is shown in Fig. 11, which utilizes multiple tiles shown in Fig. 10.
VI. EXPERIMENTAL SETUP AND EVALUATIONS

The technique used in [11] requires $2n$ memristors to match a string of length $n/2$. The power consumed by the memristor crossbar for matching $m$ strings of length $n/2$ is $4mnV^2/(5.33*R_{on})$ (see Fig. 12). The proposed approach requires $n$ memristors to match a string of length $n/2$. The power consumed by the memristor crossbar for matching $m$ strings of length $n/2$ is $mnV^2/R$ (see Fig. 13). We observed that the average memristor resistance after training is around 1M$\Omega$, which makes our proposed approach 6 times more power efficient than the previous approach.

We have examined 857 network packets (with a maximum payload size of 1480 bytes and 20 bytes for header) and 12 snort rules for our analysis. These packets were processed on an Intel E8400 system using the SNORT application and had an execution time of 0.65 seconds. These 12 snort rules had 30 distinct string matching including both header (9) and body (21) of packets such as protocol, source address, destination address, source port, destination port and contents. Table II shows a comparison between the Intel system and the proposed system for executing the mentioned dataset (this is for the detection engine of SNORT).

<table>
<thead>
<tr>
<th>System</th>
<th>Area (mm$^2$)</th>
<th>Power (W)</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel E8400</td>
<td>107</td>
<td>65</td>
<td>0.65</td>
</tr>
<tr>
<td>Memristor based</td>
<td>0.02</td>
<td>0.00026</td>
<td>0.025</td>
</tr>
</tbody>
</table>

VII. CONCLUSION

In this paper we have demonstrated a memristor crossbar based circuit for string matching which is more power efficient compared to earlier work. We utilized this circuit to design a low power network Intrusion Detection System. Our results show that memristor based systems can be five orders of magnitude more power efficient compared to Intel systems for SNORT based IDS.

REFERENCES