Reconfigurable approximate pattern matching architectures for nanotechnology

Viswanath Annampedu[a], Meghanad D. Wagh[b],*

[a]*Storage Products, Agere Systems Inc., Allentown, PA 18109, USA*
[b]*Department of Electrical and Computer Engineering, Lehigh University, Bethlehem, PA 18015, USA*

Abstract

Approximate pattern matching is comparing an input pattern with a target pattern with a specified error tolerance. This ability to compensate for real-world sensor errors makes approximate pattern matching an ideal choice for a wide range of applications. This paper shows that two n-bit patterns may be matched with a single stage nanotechnology architecture using $2n + 1$ unit area resonant tunneling diodes (RTDs) and one RTD of 1.5 times the unit area. This architecture is reconfigurable for any target pattern and any error tolerance. We analyze current and power characteristics of the architecture and develop configurations to minimize the same. The robustness of the architecture to variations in device characteristics is also investigated.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Nanotechnology; Resonant tunneling diode; Threshold logic; Reconfigurable architecture; Pattern matching; Error tolerance

1. Introduction

Pattern matching is a basic operation in many image recognition and image understanding systems. Such systems arise in a wide array of applications including visual inspections for quality control, fingerprint and bar-code recognition for security and robotic vision for intelligent manufacturing. Most of such applications demand high speed, flexibility, error tolerance, low complexity and low cost.

Nanotechnology with its high speed, small footprint and low-power requirement is a natural fit for such applications. A significant amount of research is already available in the technology of nanodevices. This paper puts focus on a novel application of these devices and bridges the gap between device physics and practice. In particular, it focuses on a nanotechnology architecture that compares an input pattern with a target pattern using a minimal number of resonant tunneling diodes (RTDs), each with the same unit area. Unlike other implementations, this architecture is not an assembly of basic Boolean gates and consequently has a minimal number of interconnects. Since interconnects limit the speed and power performance of nanoelectronics, this architecture suffers less from the ill effects of the nanowires. This architecture can take advantage of parallel data available from sensor systems such as charge coupled device (CCD) arrays. One can program the architecture for any target pattern to fit the application and for any error tolerance to compensate for inevitable sensor errors. Because of its error tolerance, we refer to this architecture as the *approximate pattern matching architecture* (APMA).

Since approximate pattern matching is recognizing a set of patterns, it may be accomplished by neural networks which are historically used for similar problems [1]. However, such networks require training and are much more complex than the deterministic architecture we propose here. Many other dedicated hardware solutions to the problem of pattern matching are also available in Refs. [2–6], but none of these are aimed towards nanotechnology.

Nanotechnology applications to digital logic, and in particular those using RTDs, have been studied by several

---

*Corresponding author. Tel.: +1 6107584142; fax: +1 6107586279.

E-mail addresses: vannampedu@agere.com (V. Annampedu), mdw0@lehigh.edu (M.D. Wagh).

researchers [7–10]. However, the problem of pattern matching has received relatively less attention. The nanopipelined correlator in [8] can be adapted to solve the problem of approximate pattern matching. This correlator compares two patterns with ExOR gates and then counts the number of mismatches using a multilevel adder. Error tolerant pattern matching can be accomplished by comparing the number of mismatches from this correlator with a preset error tolerance level. The APMA realized as a single stage nanoelectronic circuit was first presented by the authors in [11]. This paper further extends and generalizes the concepts presented therein. We present the current and power characteristics of the APMA and develop strategies to minimize the same. The resultant architecture has substantially less hardware complexity as compared with the solution adapted from the nanopipelined correlator [8].

This paper is organized as follows. In Section 2, we define threshold functions and show that the approximate pattern matching system can be viewed as a threshold function. Section 3 introduces RTDs and the concept of a monostable–bistable transition logic element (MOBILE) architecture. We build on the MOBILE ideas to implement the APMA using RTDs. We then analyze the architecture for its average current and power characteristics. Section 4 provides a power efficient APMA solution that exploits the availability of pattern complements in certain applications. This architecture can potentially reduce the peak currents in APMA by as much as $(n/2\varepsilon)$ where n and ε denote the pattern length and the error tolerance, respectively. Section 5 extends the architecture to allow for reconfigurability. This reconfigurable APMA can be programmed for any target pattern and any desired error tolerance. The architecture to match n-bit patterns uses $2n + 1$ RTDs of unit area and one RTD of 1.5 times the unit area. It can be programmed to take advantage of power efficient solution of Section 4. Section 6 investigates the robustness of APMA to device variations due to manufacturing. Finally, Section 7 presents the concluding remarks.

2. Pattern matching and threshold logic

A function $f(x_1, x_2, \ldots, x_n)$ is called a threshold function if one can determine weights $w_1, w_2, \ldots, w_n$ and threshold T (all real numbers) such that

$$\sum_{i=1}^{n} w_i x_i \geqslant T \quad \text{if and only if } f(x_1, x_2, \ldots, x_n) = 1. \quad (1)$$

We use the notation $\mathrm{TH}(\mathbf{x}; \mathbf{w}; T)$ to describe a threshold function with argument vector $\mathbf{x}$, weight vector $\mathbf{w}$ and threshold T. For example, $a + \overline{b}c = \mathrm{TH}(a, b, c; 2, -1, 1; 0.5)$. Note that these weights and thresholds are not unique.

Some basic gates such as AND, OR and NOT are threshold functions. But Exclusive-OR gate or simple functions such as $ab + cd$ or $ab + \overline{a}c$ cannot be realized as single threshold functions. In general, larger Boolean

functions are rarely threshold. However, we will show later (Theorem 1) that the Boolean function corresponding to the approximate pattern matching is a threshold function.

Threshold nature of some Boolean functions is easy to establish. For example, an n input AND gate is a threshold function $\mathrm{TH}(x_1, x_2, \ldots, x_n; 1, 1, \ldots, 1; n)$. Similarly an n input OR gate is $\mathrm{TH}(x_1, x_2, \ldots, x_n; 1, 1, \ldots, 1; 1)$. If a variable in a threshold function is complemented, the new function is still threshold, and its weights and threshold are related to those of the original function. In particular, the weight of the complemented variable gets negated, the threshold decreases by the same amount and the other weights remain unchanged. For example, since a Boolean function $abc = \mathrm{TH}(a, b, c; 1, 1, 1; 3)$, $\overline{a}bc = \mathrm{TH}(a, b, c; -1, 1, 1; 2)$ and $\overline{a}b\overline{c} = \mathrm{TH}(a, b, c; -1, 1, -1; 1)$. We use these properties of threshold functions in the following theorem.

Theorem 1. *A threshold function* $\mathrm{TH}(\mathbf{x}; \mathbf{w}; k - \varepsilon)$ *can identify a binary pattern* $\mathbf{x}$ *with* k *ones where* ε *is the specified error tolerance. The ith component of the weight vector is* $+1$ *if the corresponding component of* $\mathbf{x}$ *is* 1, *and* -1, *if it is* 0.

Proof. Let $\mathbf{x} = \langle x_0, x_1, \ldots, x_{n-1} \rangle$ represent the input vector which is to be matched against the binary target vector $\mathbf{p} = \langle p_0, p_1, \ldots, p_{n-1} \rangle$.

Define x^p for any Boolean variable x and binary constant p as

$$x^p = \begin{cases} x & \text{if } p = 1, \\ \overline{x} & \text{if } p = 0. \end{cases}$$

Then the Boolean function of $\mathbf{x}$ which is 1 only when $\mathbf{x} = \mathbf{p}$ is given by

$$f(\mathbf{x}) = x_1^{p_1} x_2^{p_2} \cdots x_n^{p_n}. \quad (2)$$

Note that the expression in (2) is an AND of the n inputs, with $n - k$ of them complemented. The discussion preceding the theorem suggests that $f(\mathbf{x})$ is a threshold function $\mathrm{TH}(\mathbf{x}; \mathbf{w}; T)$, where the ith component of the weight vector $\mathbf{w}$ is $+1$ if $p_i = 1$ and -1 if $p_i = 0$, and the threshold $T = n - (n - k) = k$.

When $p_j = 1$ but $x_j = 0$, the error in jth position reduces the weighed sum of inputs $\sum_{i=0}^{n} x_i w_i$ by 1 because $w_j = +1$. Similarly, when $p_j = 0$ but $x_j = 1$, the resultant error also reduces the weighed sum by 1 because $w_j = -1$. Thus, a total of ε errors in $\mathbf{x}$ decreases the weighed sum by ε irrespective of the type of errors. To tolerate up to ε errors, i.e., to have an output 1 in spite of the weighed sum decrease of up to ε, one should decrease the threshold by ε as the threshold function definition (1) suggests. □

To make the comparison between the weighed sum and the threshold in (1) more robust, the threshold should be set exactly in the middle of the maximum weighed sum of inputs when the output is 0 and the minimum weighed sum when the output is 1. Let T_L denote the largest value of the weighed sum when $\mathrm{TH}(\mathbf{x}; \mathbf{w}; k - \varepsilon) = 0$ and

T_{H}, the smallest value of the weighed sum when $\mathrm{TH}(\mathbf{x}; \mathbf{w}; k - \varepsilon) = 1$. As in the proof of Theorem 1, for every error, the weighed sum decreases by 1. Therefore, T_{H} corresponds to ε errors and T_{L} to $(\varepsilon + 1)$ errors. One then gets $T_{\mathrm{L}} = T_{\mathrm{H}} - 1$ and the optimal threshold as $T = (T_{\mathrm{L}} + T_{\mathrm{H}})/2 = (2T_{\mathrm{H}} - 1)/2 = T_{\mathrm{H}} - 0.5$. For example, the function $\mathrm{TH}(\mathbf{x}; 1, -1, -1, 1, 1, -1, 1, 1; 2.5)$ matches the 8-bit input pattern $\mathbf{x}$ with the target pattern 10011011 within an error tolerance of 2.

3. Nanotechnology implementation

RTDs and resonant tunneling transistors (RTTs) represent the most mature nanotechnology devices available to date for implementing new innovative applications [12,13]. RTDs and RTTs employ double barrier quantum well structures for tunneling electrons at discrete energy levels. The tunneling phenomena create the effect of negative resistance (see Fig. 1(a)) which may be exploited to build the basic logic element shown in Fig. 1(b). This architecture is known as a MOBILE [14] and is capable of providing two stable circuit states.

Behavior of the MOBILE is dependent upon the peak currents I_{A} and I_{B} of the two RTDs (which depend on the area of the RTD junctions) and the voltage where the current peaks, V_{p}. As V_{SWP} is increased from 0 to more than $2V_{\mathrm{p}}$, V_{out} settles to a low value (corresponding to a logic 0) if $I_{\mathrm{A}} < I_{\mathrm{B}}$ or a high value (corresponding to a logic 1) if $I_{\mathrm{A}} > I_{\mathrm{B}}$. When the output settles to either of the two values, the current flowing through the circuit is very small.

By connecting multiple RTDs in parallel, one can create an effective peak current equal to sum of the individual peak currents. Each individual peak current may be varied by using RTDs of varying areas. In addition, it may be switched in or out of the sum by employing a *heterostructure field effect transistor* (HFET) switch in series.[1] Positively weighed inputs may be placed in the top section (in parallel with the RTD A) and negatively weighed, in the bottom section, in parallel with RTD B. Thus, the MOBILE architecture allows one to create a weighed sum of the inputs. The comparison of two effective peak currents allows one to compare the weighed sum with a preset threshold. Fig. 2 shows the RTD/HFET implementation of $\mathrm{TH}(\mathbf{x}; 1, -1, -1, 1, 1, -1, 1, 1; 2.5)$ used to identify input pattern 10011011 within 2 errors.

Note that because all the input weights are ± 1 (Theorem 1), RTDs corresponding to each input have the same area. This minimum area will be called the unit area. To reduce the number of RTDs, the RTD implementing the threshold and the RTD B may be replaced by a single RTD of area 3.5. Further, if the threshold is negative, then the threshold RTD may be placed in the top section rather than the bottom. We will refer to the architecture in Fig. 2 as the APMA.

---

[1]The monolithic integration of HFET and RTD is known as the three terminal hybrid RTT [15].
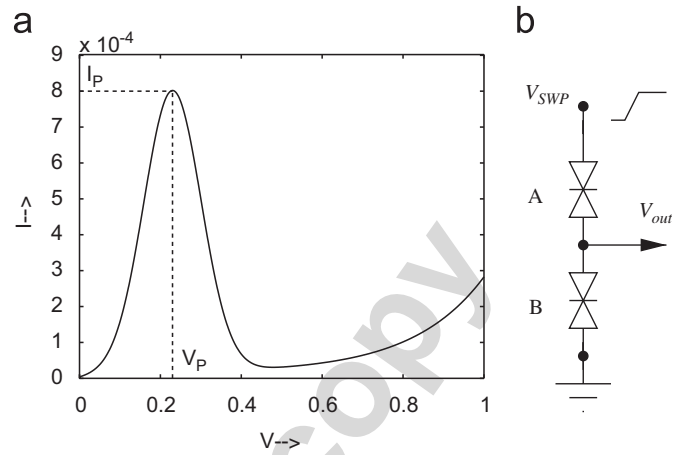

Fig. 1. (a) The I–V curve of a resonant tunneling diode (RTD) showing the negative resistance and (b) schematic representation of a MOBILE obtained by connecting two RTDs in series.
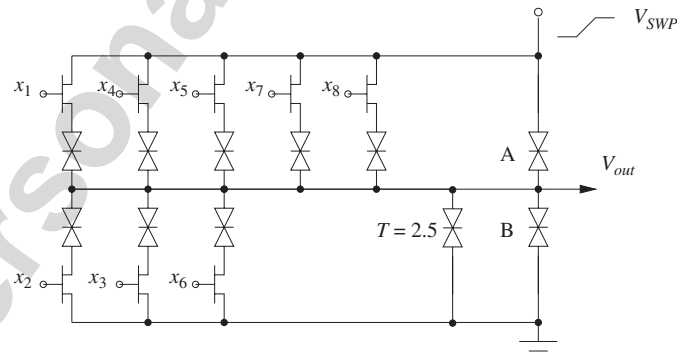

Fig. 2. Approximate pattern matching architecture (APMA) to match an input $\langle x_1, x_2, \ldots, x_n \rangle$ with the target pattern 10011011 tolerating up to 2 errors. Note that the area of the threshold RTD is 2.5 times that of all other RTDs.

We now analyze the current and power characteristics of the APMA in Fig. 2. Since the top and bottom section RTDs are in series, the maximum instantaneous current in the APMA is the minimum of the effective peak current of the top RTDs and the bottom RTDs. This peak current also affects the peak power drawn by the architecture. Fig. 3(a) shows the typical profiles of instantaneous current and power in an APMA. One can see that the current and the power curves peak at around the same V_{SWP} voltage values. The peak current through the APMA strongly influences the power drawn by it. When the peak current through the architecture is increased, the V_{SWP} voltage at which the current and power profiles reach their maximums also increases by a small amount. Thus the peak power increases faster than the peak current. One can see this easily from Fig. 3(b) which shows two cases with different peak currents through the APMA. In both cases, the effective peak current of the bottom RTDs is smaller than that of the top RTDs and therefore will determine the profile of the
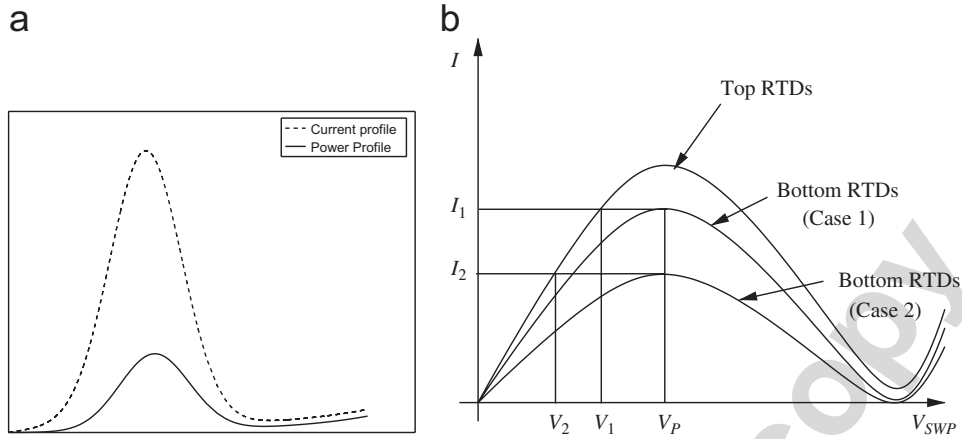
Fig. 3. (a) Instantaneous current and power drawn by the approximate pattern matching architecture (APMA) of Fig. 2 (vertical axis) as it is swept with V_{SWP} (horizontal axis) and (b) effective I–V curves of the top and the bottom RTDs in Fig. 2 and determination of the voltage V_{SWP} at which the architecture draws the maximum current.

current through the APMA. In the first case, when the bottom current peaks at voltage V_P across it, top RTDs have a voltage V_1 across them because they carry the same effective current as the bottom RTDs. Thus the value of the voltage across the APMA at that time is $V_P + V_1$. In the second case with a lower bottom current peak, the voltage across the top RTDs is only V_2 when the current through the APMA reaches its maximum. Thus the voltage across the APMA at the current peak is $V_P + V_2$. For simplicity we assume that the power and current peaks occur at the same sweep voltage. Thus, the peak powers in the two cases are $P_1 = I_1(V_P + V_1)$ and $P_2 = I_2(V_P + V_2)$. This clearly shows that the power peak is dependent upon the lower of the two peak currents in the APMA. Also note that $(P_1/P_2) > (I_1/I_2)$ indicating that if the peak current through the APMA increases, the peak power increases at a faster rate. Thus, understanding the behavior of peak current is crucial to reduce the power consumption of the APMA.

Theorem 2 gives the average peak value of this instantaneous current in terms of the design parameters n, k and ε and the number of matching errors e. In order to simplify the discussion, we will ignore the threshold adjustment of 0.5 and the currents through RTDs A and B in this theorem.

Theorem 2. *The peak current flowing through an APMA averaged over all patterns with e mismatches is given by*

$$I_{\max} = I_P(\max(k,\varepsilon) - \max(e,\varepsilon) + e(n-k)/n), \tag{3}$$

where I_P is the peak current of a unit area RTD, k is the number of 1's in the target pattern and ε is the error tolerance.

Proof. We separate the proof into the following four cases.

Case 1: $k < \varepsilon$ *and* $e \leqslant \varepsilon$. In this case, the output of the circuit is expected to be 1. Thus, the sum of the individual RTD peak currents in the bottom section must be less than that at the top and will give $I_{\max}$. The only inputs in this

section are those that correspond to 0's in the target pattern. Thus, each mismatch in these positions contributes a current I_P to $I_{\max}$. Since the threshold, $k - \varepsilon$, is negative, the threshold RTD is in the top section and does not contribute to $I_{\max}$. Let i denote the number of mismatches in the bottom section. Average peak current $I_{\max}$ is then given by

$$I_{\max} = I_P \left[\frac{1}{{}^n C_e} \sum_{i=0}^{e} i^k C_{e-i}{}^{n-k} C_i \right]. \tag{4}$$

Note that the averaging in (4) is achieved by summing over a typical case of i mismatches out of $n - k$ positions in the bottom section and the remaining $e - i$ mismatches in the k top positions. ${}^n C_e$ represents the total number of mismatch vectors.

By using ${}^{n-k} C_i = ((n-k)/i)^{n-k-1} C_{i-1}$, one can simplify (4) as

$$\begin{aligned} I_{\max} &= I_P \left[\frac{(n-k)}{{}^n C_e} \sum_{i=0}^{e} {}^k C_{e-i}{}^{n-k-1} C_{i-1} \right] \\ &= I_P \left[\frac{(n-k)}{{}^n C_e}{}^{n-1} C_{e-1} \right] \\ &= I_P[e(n-k)/n]. \end{aligned} \tag{5}$$

Case 2: $k \geqslant \varepsilon$ *and* $e \leqslant \varepsilon$. In this case also, the RTD currents in the bottom section of the architecture determine $I_{\max}$. However, the threshold being positive, the threshold RTD resides in this section and contributes a peak current of $I_P(k - \varepsilon)$ to $I_{\max}$. Thus the equation for $I_{\max}$ would be identical to (4) except for this additional term. The total $I_{\max}$ in this case is obtained as

$$I_{\max} = I_P[e(n-k)/n + k - \varepsilon]. \tag{6}$$

Case 3: $k \geqslant \varepsilon$ *and* $e > \varepsilon$. In this case, the output of the circuit is expected to be 0 and therefore $I_{\max}$ is determined by the RTD currents in the top section. The threshold RTD is now in the bottom section. With i mismatches in

the top section, there will be $(k - i)$ RTDs in that section that will contribute to I_{max}. The I_{max} equation can thus be shown to be

$$I_{max} = I_P \left[k - \frac{1}{{}^nC_e} \sum_{i=0}^{e} i^k C_i^{\ n-k}C_{e-i} \right]. \tag{7}$$

Simplifying (7) as in case 1, we get

$$I_{max} = I_P[k - (ke)/n].$$

Case 4: $k < \varepsilon$ *and* $e > \varepsilon$. This is similar to case 3 except that the threshold RTD of area $(\varepsilon - k)$ is in the top section and therefore contributes a constant peak current of $I_P(\varepsilon - k)$ to I_{max}. Thus we get

$$I_{max} = I_P[\varepsilon - (ke)/n]. \qquad \square$$

Note that the peak current in Fig. 2 is also affected by the RTDs A and B as well as the threshold adjustment of 0.5. Since one of the two unit area RTDs, A or B, always lies in the section that determines I_{max}, their contribution can be accounted for by adding I_P to (3). Similarly, the threshold decrease of 0.5 can be viewed as an increase in the error tolerance ε by 0.5. The expression for I_{max} that takes these factors into account can be written as

$$I_{max} = I_P(\max(k, \varepsilon + 0.5) - \max(e, \varepsilon + 0.5) \\ + e(n - k)/n + 1). \tag{8}$$

4. Power efficient APMA

In practice it is important to minimize the number of conducting RTDs in the APMA. Clearly this implies reduction in the peak current requirements. Further, as discussed with reference to Fig. 3, the peak current has a strong influence on peak power drawn by the APMA. Finally, a smaller number of conducting RTDs implies a smaller variance of the effective currents in the top and bottom sections of the APMA. This improves the operational reliability of the APMA [16].

Unfortunately in the APMA, the threshold RTD is always ON (see Fig. 2). We now show that if the complement signals $\langle \overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, \overline{x}_5, \overline{x}_6, \overline{x}_7, \overline{x}_8 \rangle$ are also available, then the architecture can be modified to substantially reduce the peak current. This modification is based on the following two lemmas.

Lemma 3. *In a threshold function, an input x with weight w and an input 1 with weight −w can be replaced with a single input $\overline{x}$ with weight −w.*

Proof. The weighed sum of the inputs x and 1 in the first case is $(x - 1)w$. But the arithmetic value of $\overline{x}$ equals $1 - x$. Therefore, the same weighed sum will result from only one input $\overline{x}$ with weight $−w$. $\square$

To illustrate Lemma 3, consider the architecture of Fig. 2. Ignoring the 0.5 adjustment, the threshold RTD in the bottom section can be emulated by a parallel

connection of three unit area RTD/HFETs with inputs 1. Recall that inputs in the bottom section can be considered to have weights −1 and those in the top, weights +1. Thus, each combination of an input x_i in the top section with an input of 1 in the bottom section can be replaced with a single input of $\overline{x}_i$ in the bottom section. The result of this modification is shown in Fig. 4. Note that in this form, the pattern matching architecture uses one RTD per input bit, two MOBILE RTDs and a 0.5 area RTD for threshold adjustment which may be combined with RTD A. This minimum RTD architecture will henceforth be referred to as the *power efficient* APMA.

When the threshold is negative, the constant input will be in the top section. This corresponds to inputs 1 with a positive weight in the threshold function. We can deal with this case in similar manner using the following lemma.

Lemma 4. *In a threshold function, an input x with weight −w and an input 1 with weight w can be replaced with a single input $\overline{x}$ with weight w.*

Proof. As in the proof of Lemma 3, the weighed sum of the inputs x and 1 in the first case is $(1 - x)w$. The same weighed sum results from only one input $\overline{x}$ with weight w. $\square$

As a consequence of Lemmas 3 and 4 and the discussion following Lemma 3, we have the following theorem.

Theorem 5. *If the complement of the input pattern is also available, then one can design an APMA for an arbitrary target pattern and any error tolerance such that no RTD is constantly ON except RTDs A, B and the threshold adjustment RTD of value 0.5.*

The procedure to design the power efficient APMA can be described as follows:

(1) If $k \geqslant \varepsilon$, place ε inputs corresponding to 1's in the target pattern in the top section. Place the complements of the remaining $(k - \varepsilon)$ inputs corresponding to 1's of the target pattern in the bottom section. Place all the inputs corresponding to 0's of the target pattern in the bottom section.
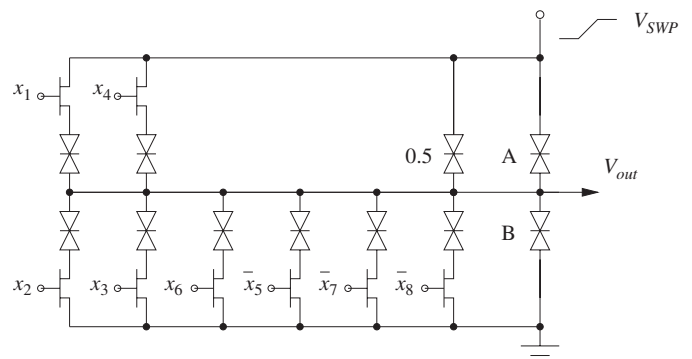

Fig. 4. Power efficient APMA to match an input $\langle x_1, x_2, \ldots, x_n \rangle$ with the target pattern 10011011 tolerating up to 2 errors. Note that the RTD with area 0.5 times that of all other RTDs may be merged with RTD A.

(2) If $k < \varepsilon$, place all the inputs corresponding to 1's of the target pattern in the top section. Place the complements of the $(\varepsilon - k)$ inputs corresponding to 0's of the target pattern in the top section as well. Keep the rest of the inputs corresponding to 0's of the target pattern in the bottom section.

(3) Add one MOBILE RTD to each section (RTDs A and B) and a 0.5 area RTD in the top section (which may be combined with the RTD A).

One can see that while the APMA (Fig. 2) used as many as $(n + |k - \varepsilon - 0.5|)$ RTDs (apart from the RTDs A and B), the power efficient APMA (Fig. 4) uses only $(n + 0.5)$ RTDs. In realistic situations, one would expect the number of ones in the target pattern, $k \approx n/2$ and the error tolerance, $\varepsilon \ll n$. Thus using the power efficient APMA described here reduces the complexity of the pattern matching from approximately $1.5n$ to n RTDs.

Another consequence of the power efficient APMA is a substantial reduction in the average peak current as described in the following theorem.

Theorem 6. *With the same assumptions and notation of Theorem 2, the peak current averaged over all patterns with e mismatches in the power efficient APMA is given by*

$$I_{\max}^* = I_P(\min(\varepsilon, e)(n - \max(\varepsilon, e))/n). \tag{9}$$

Proof. As before, we partition the proof in to the following four cases.

Case 1: $k < \varepsilon$ *and* $e \leqslant \varepsilon$. In this case, the circuit outputs a 1 implying that the top section has the larger peak current. Thus $I_{\max}^*$ is determined by the RTDs in the bottom section. Each of these RTDs correspond to 0's in the target pattern and therefore conducts only if there is a mismatch in the corresponding bit. Assuming i errors within these bottom $(n - \varepsilon)$ positions and $(e - i)$ errors in the top ε positions, one gets the average peak current as

$$I_{\max}^* = I_P \left[\frac{1}{{}^{n}C_e} \sum_{i=0}^{e} i \, {}^{\varepsilon}C_{e-i} \, {}^{n-\varepsilon}C_i \right]. \tag{10}$$

By using procedure similar to the one used to simplify (4), one gets

$$I_{\max}^* = I_P[e(n - \varepsilon)/n]. \tag{11}$$

Case 2: $k \geqslant \varepsilon$ *and* $e \leqslant \varepsilon$. In this case also, $I_{\max}^*$ is determined by the currents in the bottom section. However, this section now consists of $(n - k)$ inputs corresponding to 0's in the target pattern and $(k - \varepsilon)$ complemented inputs corresponding to 1's of the target pattern. A mismatch in any of these $(n - \varepsilon)$ inputs implies that the corresponding RTDs conduct. Thus the $I_{\max}^*$ expression in this case is identical to (11).

Case 3: $k \geqslant \varepsilon$ *and* $e > \varepsilon$. In this case, the output of the circuit is expected to be 0 and therefore $I_{\max}^*$ is determined by the RTD currents in the top section. The only inputs in the top section are those corresponding to 1's in the target pattern. Assuming $e - i$ mismatches out of ε position in

that section, only $(\varepsilon - e + i)$ RTDs contribute to $I_{\max}^*$. The $I_{\max}^*$ is therefore given by

$$I_{\max}^* = I_P \left[\frac{1}{{}^{n}C_e} \sum_{i=0}^{e} (\varepsilon - e + i) \, {}^{\varepsilon}C_{e-i} \, {}^{n-\varepsilon}C_i \right]$$
$$= I_P[\varepsilon(n - e)/n].$$

Case 4: $k < \varepsilon$ *and* $e > \varepsilon$. This case is similar to case 3 except that the top section now has k inputs corresponding to 1's of the target pattern and $(\varepsilon - k)$ complemented inputs corresponding to the 0's of the pattern. An error in any of these ε top inputs implies turning off of that RTD. Thus the expression for $I_{\max}^*$ will be exactly the same as in that of case 3. □

Note that $I_{\max}^*$ expression in Theorem 6 can be modified to include the currents through RTDs A, B and the RTD of area 0.5 as was done in (8).

One can compare the peak current of the APMA shown in Fig. 2 with that of the power efficient APMA shown in Fig. 4. Assume realistically that $k \approx n/2$, $e \approx n/2$ and $\varepsilon \ll n$. Theorems 2 and 6 then show that the peak current of the APMA is $I_{\max} \approx I_P(n/4)$ and that of the power efficient APMA is $I_{\max}^* \approx I_P(\varepsilon/2)$, a quantity substantially lower than $I_P(n/4)$.

5. Reconfigurable approximate pattern matching

A disadvantage of the APMA and the power efficient APMA is that they are married to specific target patterns and predetermined error tolerances. We now develop an architecture shown in Fig. 5 that can be programmed for any target pattern and arbitrary error tolerance. We refer to this architecture as the *reconfigurable* APMA.

In the reconfigurable APMA, we allocate two RTD/HFETs corresponding to each bit of the target pattern, one at the top and one at the bottom, to provide for the bit to be either 1 or 0. Since all weights are ± 1, all the RTDs used have unit area. Clearly, only one RTD/HFET out of the two allocated is used for each input bit for any target pattern. All the unused RTD/HFETs are used to emulate the threshold. Assume that the target pattern has k ones. Then the corresponding k inputs will be applied in the top section and the remaining $n - k$, in the bottom section. Thus, $n - k$ and k inputs are unused in the top and the bottom sections, respectively. Recall that the threshold T satisfies $-(n - k) - 0.5 \leqslant T \leqslant k - 0.5$ and if positive, is placed in the bottom section and if negative, in the top. The 0.5 portion of the threshold can be implemented through a RTD of that area at the top as shown in Fig. 5. Thus, a negative threshold, T, $-(n - k) \leqslant T < 0$, can be implemented through the $n - k$ unused RTDs in the top section and a positive threshold, T, $0 < T \leqslant k$, can be similarly implemented through the k unused RTDs in the bottom section. Thus, even though the threshold can take $2n + 1$ possible values from $-n$ (when $k = 0$ and $\varepsilon = n$) to
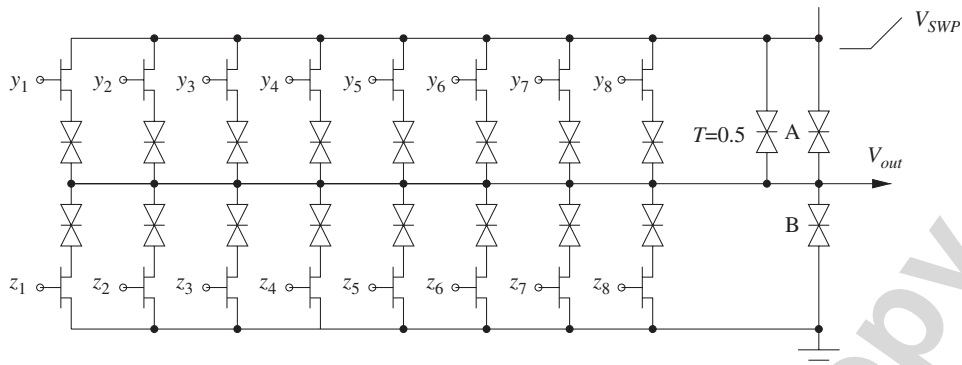
Fig. 5. A reconfigurable approximate pattern matching architecture (APMA) to match two 8-bit patterns with any error tolerance from 0 to 8. Note that the threshold RTD has half the area as that of all other RTDs and may be combined with RTD A.

Table 1
Programming illustrations of reconfigurable APMA of Fig. 5 to match an input $\langle x_1, x_2, \ldots, x_n \rangle$ with the target pattern 10011011 under error tolerance ε

ε	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
0	x_1	0	0	x_4	x_5	0	x_7	x_8	1	x_2	x_3	1	1	x_6	1	1
1	x_1	0	0	x_4	x_5	0	x_7	x_8	0	x_2	x_3	1	1	x_6	1	1
3	x_1	0	0	x_4	x_5	0	x_7	x_8	0	x_2	x_3	0	0	x_6	1	1
6	x_1	1	0	x_4	x_5	0	x_7	x_8	0	x_2	x_3	0	0	x_6	0	0

Table 2
Programming illustrations of reconfigurable APMA of Fig. 5 to match an input $\langle x_1, x_2, \ldots, x_n \rangle$ with the target pattern 10011011 under various error tolerances ε while minimizing the power

ε	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
0	0	0	0	0	0	0	0	0	$\overline{x}_1$	x_2	x_3	$\overline{x}_4$	$\overline{x}_5$	x_6	$\overline{x}_7$	$\overline{x}_8$
1	x_1	0	0	0	0	0	0	0	0	x_2	x_3	$\overline{x}_4$	$\overline{x}_5$	x_6	$\overline{x}_7$	$\overline{x}_8$
3	x_1	0	0	x_4	x_5	0	0	0	0	x_2	x_3	0	0	x_6	$\overline{x}_7$	$\overline{x}_8$
6	x_1	$\overline{x}_2$	0	x_4	x_5	0	x_7	x_8	0	0	x_3	0	0	x_6	0	0

$+n$ (when $k = n$ and $\varepsilon = 0$), our architecture allows us to implement it through only the n unused RTDs of unit area.

This discussion leads to the following programming procedure for the reconfigurable APMA:

(1) Apply input x_i in the ith column; in the top section if the ith bit of the target pattern is 1, otherwise in the bottom section.
(2) If $\varepsilon \leqslant k$, apply logic 0 to any ε unused inputs in the bottom section and 1 to the rest. Apply 0 to all the unused inputs in the top section.
(3) If $\varepsilon > k$, apply 0 to all unused inputs in the bottom section. Apply 1 to any $\varepsilon - k$ unused inputs in the top section and 0 to the rest.

This procedure ensures that the number of inputs tied to logic 1 is as small as possible to achieve the specified error tolerance. This implies minimum power requirements in conformity with Theorem 2. Table 1 provides examples of programming the reconfigurable APMA of Fig. 5 to match

input patterns $\langle x_1, x_2, \ldots, x_8 \rangle$ with the target pattern 10011011 for various error tolerance values.

If the complement of the input pattern, $\langle \overline{x}_0, \overline{x}_1, \ldots, \overline{x}_{n-1} \rangle$ is also available, then one can program the reconfigurable APMA to take advantage of this to further minimize the power. In particular, Lemmas 3 and 4 allow one to modify any configuration such that no RTD is constantly ON. Consider, for example, the configuration used for $\varepsilon = 3$ in Table 1. Here, input pair $z_7 = 1$ and $y_7 = x_7$ may be replaced with the pair $z_7 = \overline{x}_7$ and $y_7 = 0$ using Lemma 3. Similar transformation can also be applied to z_8 and y_8. In general, any time $z_i = 1$, one can instead set $z_i = \overline{x}_i$ and $y_i = 0$. Similarly, from Lemma 4, any time $y_i = 1$, one can instead set $y_i = \overline{x}_i$ and $z_i = 0$. Thus, the final configuration is guaranteed never to have any RTD constantly conducting. Table 2 illustrates programming of the reconfigurable APMA of Fig. 5 for power efficient operation.

The concept of reconfiguration assumes that once the architecture is programmed for a target pattern and an error tolerance, it would be used for a large number of

input patterns before it is reprogrammed. This assumption is consistent with most real-world pattern recognition applications and implies that the configuration delay does not impact the throughput of the architecture. Reconfigurability can be achieved in nanotechnology using techniques such as crossbars [17,18]. Alternately, one can implement multiplexer circuits to choose the appropriate inputs to be applied to each y_i and z_i. By integrating these circuits with the one in Fig. 5 in a pipeline fashion, one can obtain a high throughput, flexible architecture for approximate pattern matching.

The circuits described in this paper were simulated and verified for functionality in MATLAB and HSPICE using RTD models in [19].

6. Robustness to device variations

The architectures presented in earlier sections are attractive from the point of view of manufacturing because all RTDs, except one, have identical areas. However, in reality, it is impossible to fabricate identical devices. The variation in the areas of RTDs results in variation in their peak currents. In this section we analyze the effect of this on the probability of incorrect detection.

The RTD peak currents can be modeled as independent Gaussian random variables [16]. Let I_P and σ^2 denote their mean and variance. A detection error occurs when $N_T > N_B$ but $I_T < I_B$, or when $N_T < N_B$ but $I_T > I_B$, where N_T and N_B represent the number of top and bottom RTDs that are ON, I_T, the sum of top peak currents, and, I_B, the sum of bottom peak currents. By applying standard error analysis one can then show that the probability of incorrect detection, P_e, is given by

$$P_e = Q\left(\frac{|N_T - N_B + 0.5| I_P}{\sigma\sqrt{N_T + N_B + 2.25}}\right), \qquad (12)$$

where the Q function is defined as $Q(\alpha) = (1/\sqrt{2\pi}) \int_\alpha^\infty e^{-x^2/2}\, dx$. The constant 2.25 in (12) represents variance from the unit RTDs A and B and the threshold RTD of value 0.5. Note that the Q function is a monotonically decreasing function. Thus, the worst case P_e is obtained when the argument of the Q function in (12) is minimum. By careful analysis of the different scenarios, one can show that the worst case P_e for the reconfigurable APMA of Fig. 5 is given by $Q(0.5 I_P/(\sigma\sqrt{2k + 3.25}))$ when $\varepsilon \leqslant k$ and $e = \varepsilon + 1 \leqslant n - k$. (The symbols are as defined in Theorem 2.) Similarly, for the power efficient reconfigurable APMA, the worst case $P_e = Q(0.5 I_P/(\sigma\sqrt{2\varepsilon + 3.25}))$ when $\varepsilon \leqslant (n-1)/2$ and $e = \varepsilon + 1$. It is worth noting that for the power efficient reconfigurable APMA, the worst case P_e is *independent* of both the target pattern and its length n and depends only on the error tolerance designed into the system. Thus, the maximum pattern size is not constrained by the RTD area variations for the power efficient APMA.

Fig. 6 shows the worst case P_e as a function of ε for two typical values of σ/I_P reported in Ref. [16]. Assume that a probability of incorrect detection of 10^{-5} is acceptable in a
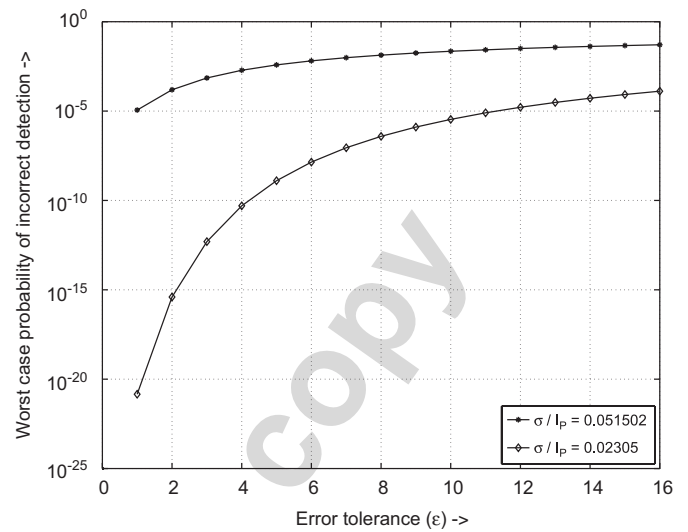

Fig. 6. The relationship between the worst case probability of incorrect detection and the error tolerance for the case of the power efficient reconfigurable APMA.

pattern matching application which is designed to tolerate up to 10% sensor errors (i.e., $\varepsilon/n = 0.1$). The plots in Fig. 6 then show that as the technology (σ/I_P) improves from 0.052 to 0.023, the pattern size handled by a power efficient APMA increases from 10 to 110 bits. When the process technology stabilizes in the future, σ/I_P will be even smaller and the architectures given here will be applicable to even larger patterns.

7. Conclusion

This paper provides a novel nanotechnology architecture for approximate pattern matching. It uses a fixed configuration of $2n + 1$ RTDs of unit area and one RTD with 1.5 times the unit area. It can be programmed for any n-bit target pattern and for any error tolerance between 0 and n. We have also shown that if the complements of the input pattern are also available (or are separately computed), then the architecture can be programmed to reduce the average current (and consequently the power) by a factor of $(n/2\varepsilon)$ where ε is the error tolerance. We have also investigated the effect of the RTD area variability on the probability of incorrect detection by our architecture. The efficiency of the new architecture can be attributed to its direct design in terms of threshold logic.

References

[1] C.M. Bishop, Neural networks and their applications, Rev. Sci. Instrum. 65 (6) (1994) 1803–1832.

[2] S. Pramanik, C.T. King, A hardware pattern matching algorithm on a dataflow, Comput. J. 28 (3) (1985) 264–269.

[3] R. Sastry, N. Ranganathan, K. Remedios, Casm: a VLSI chip for approximate string matching, IEEE Trans. Pattern Anal. Mach. Intell. 17 (8) (1995) 824–830.

[4] A.N. Du, B.X. Fang, X.C. Yun, M.Z. Hu, X.R. Zheng, Comparison of stringmatching algorithms: an aid to information content security,

in: Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xian, 2003, pp. 2996–3001.

[5] M. Giraud, D. Lavenier, Weighted finite automata in hardware for approximate pattern matching, Technical Report, EDAA Ph.D Forum, DATE '04, France, February 2004.

[6] V. Lohweg, C. Diederichs, D. Müller, Algorithms for hardware-based pattern recognition, EURASIP J. Appl. Signal Process. 12 (2004) 1912–1920.

[7] C. Pacha, K. Goser, Design of arithmetic circuits using resonant tunneling diodes and threshold logic, in: Proceedings of the Second Workshop on Innovative Circuits and Systems for Nanoelectronics, Delft, The Netherlands, 1997, pp. 83–93.

[8] P. Mazumder, S. Kulkarni, M. Bhattacharya, J.P. Sun, G.I. Haddad, Digital circuit applications of resonant tunneling devices, Proc. IEEE 86 (4) (1998) 664–686.

[9] T. Akeyoshi, H. Matsuzaki, T. Itoh, T. Waho, J. Osaka, M. Yamamoto, Applications of resonant-tunneling diodes to high-speed digital ICs, in: Proceedings of the 11th International Conference on Indium Phosphide and Related Materials (IPRM'99), 1999, pp. 405–410.

[10] W. Prost, U. Auer, J. Degenhardt, A. Brennemann, C. Pacha, K.F. Goser, F.-J. Tegude, A depth-2 full-adder circuit using the InP RTD/ HFET MOBILE, in: Proceedings of the Indium Phosphide and Related Materials Conference (IPRM'01), 2001, pp. 5045–5046.

[11] V. Annampedu, M.D. Wagh, Approximate pattern matching in nanotechnology, in: Proceedings of Nanotech 2006, vol. 3, Boston, MA, 2006, pp. 316–319.

[12] K. Goser, C. Pacha, System and circuit aspects of nanoelectronics, in: The 24th European Solid-State Circuits Conference, The Hague, The Netherlands, 1998, pp. 18–29.

[13] K. Nikolic, D. Berzon, M. Forshaw, Relative performance of three nanoscale devices—CMOS, RTDs and QCAs—against a standard computing task, Nanotechnology 12 (1) (2001) 38–43.

[14] T. Akeyoshi, K. Maezawa, T. Mizutani, Weighted sum threshold logic operation of MOBILE (monostable–bistable transition logic element) using resonant-tunneling transistors, IEEE Electron Device Lett. 14 (10) (1993) 475–477.

[15] C. Pacha, K. Goser, A. Brennemann, W. Prost, A threshold logic full adder based on resonant tunneling transistors, in: The 24th European Solid-State Circuits Conference, 1998, pp. 428–431.

[16] W. Prost, U. Auer, F.-J. Tegude, C. Pacha, K.F. Goser, G. Janssen, T. van der Roer, Manufacturability and robust design of nanoelectronic logic circuits based on resonant tunnelling diodes, Int. J. Circuit Theory Appl. 28 (2000) 537–552.

[17] Y. Chen, G.-Y. Jung, D.A.A. Ohlberg, X. Li, D.R. Stewart, J.O. Jeppesen, K.A. Nielsen, J.F. Stoddart, R.S. Williams, Nanoscale molecular-switch crossbar circuits, Nanotechnology 14 (4) (2003) 462–468.

[18] G. Snider, P. Kuekes, R.S. Williams, CMOS-like logic in defective, nanoscale crossbars, Nanotechnology 15 (8) (2004) 881–891.

[19] Definition of software interfaces, ANSWERS Technical Report (autonomous nanoelectronic systems with extended replication and signalling), January 1999.