EFFICIENT THRESHOLD ARCHITECTURES WITH BOUNDED FAN-INS FOR EXCLUSIVE-ORS

Feng Shi, Zhiyuan Yan, and Meghanad Wagh

Department of ECE, Lehigh University, Bethlehem, PA 18015
E-mails: {fes209, yan, mdw0}@lehigh.edu

ABSTRACT

As process technologies scale into nanometer region, new opportunities as well as challenges arise. Some emerging nanotechnology devices that are promising candidates to replace the CMOS technology are particularly suitable for threshold logic implementations. While most conventional Boolean gates, including AND, OR, NOT, NAND, and NOR, can be implemented by a single threshold gate, exclusive-OR (XOR) is an important exception. This represents a significant obstacle, since XORs are essential building blocks for all finite field arithmetic operations over GF(2^m), which in turn are used in various applications. In this paper, we propose efficient architectures with finite fan-ins for XORs based on threshold gates, and our architectures greatly outperform architectures obtained by first expressing an XOR based on other Boolean gates and then using their threshold logic implementations. Our work in this paper is novel in two aspects. First, in addition to two-input XORs, we also investigate multi-input XORs, because they are suitable for threshold logic implementation and are very instrumental in finite field arithmetic operations. Second, our architectures differ from previous implementations of XORs based on threshold logic in that our architectures assume bounded fan-in, which is critical to the reliability to nanotechnology devices.

***Index Terms*—** Nanotechnology, threshold logic, finite field

1. INTRODUCTION

According to the International Technology Roadmap of Semiconductors (ITRS) [1], the conventional CMOS technology has great challenge in further scaling. Although new materials and device structures can keep the CMOS scaling for the next several years, the CMOS scaling would reach the fundamental limits eventually. Some emerging nanotechnology, such as resonant tunneling diodes (RTDs), quantum cellular automata (QCA), and single electron transistors (SETs), have nanoscale structure and are promising candidates to replace the CMOS technology [2]. These new nanotechnology devices promise to have smaller feature size, higher speed and lower power consumption. Even at system level design they present two advantages. Firstly they easily realize threshold gates (see Fig. 1). Threshold gates are often more powerful than Boolean gates, and can implement complex Boolean functions with a single gate [3]. Thus the hardware complexity of larger systems implemented using nanotechnology tends to be a lot smaller. Secondly, the outputs of the threshold gates built with nanotechnology are self-latched. This provides a natural way of pipelining these systems in most signal processing applications.

Several applications dealing with real-valued signals have already been realized using nanotechnology-based threshold gates [4–7]. However, there is an equally important class of signal processing applications using finite fields, such as error correcting coding and cryptography [8]. Unfortunately, the applications using finite fields have not been realized using nanotechnology.

The main obstacle for the nanotechnology-based implementations of applications of finite fields of characteristic two, denoted as GF(2^m), is that they require exclusive-ORs (XORs) to realize all arithmetic operations over GF(2^m). Unlike most conventional Boolean gates such as AND, OR, NOT, NAND, and NOR, XOR cannot be realized as a single threshold gate. Thus the translation of a finite field architecture to nanotechnology merely by replacing a conventional gate with an appropriate combination of threshold gates becomes overly complex.

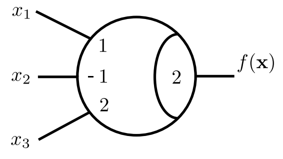
In this paper we address this obstacle by investigating efficient architectures of XORs based on threshold gates. The work in this paper is novel in two aspects. First, based on the CMOS technology, two-input XORs are the focus, whereas in this work multi-input XORs have been investigated. This is motivated by the applications as well as the properties of threshold gates. Many finite field arithmetic operations over GF(2^m), such as the polynomial basis multiplication [9,10] or the Massey-Omura (MO) multiplication [11], require multi-input XORs. Furthermore, multi-input XORs can take full advantage the more powerful threshold gates. Second, while the implementation of XORs based on threshold logic was investigated theoretically (see, for example, [3, 12]), these pre-

This work was supported in part by NSF under grants ECCS-0925890 and ECCS-1055877.

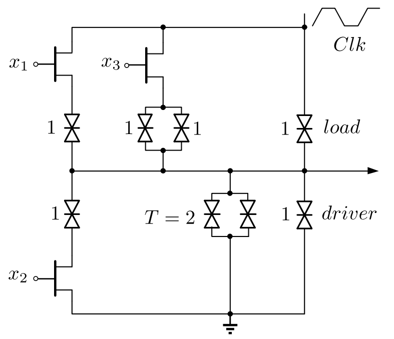
vious works usually did not account for the fan-in, whereas our work assumes a fan-in bound B. The fan-in is a critical factor for both reliability and performance. The reliability of the nanotechnology threshold gates decreases sharply with the fan-in. Also, threshold gates with large fan-ins tend to have slower switching speeds.

The main results in this paper are two classes of threshold architectures with bounded fan-ins of an n-input XOR. The first, called the *Boolean class*, expresses the XOR in a two-stage NAND circuit implemented through threshold gates. The second, referred to as the *majority class*, also has a two-level implementation and uses only generalized majority gates in the first level. Since one can implement an n-input XOR as a tree of two-input XORs, each of which can be expressed based on other Boolean gates and implemented by their threshold gates, we refer to this approach as *direct conversion* and use it as a basis for comparison. It turns out the architectures obtained by direct conversion are the same as Boolean class architectures with $B = 3$. Hence, our Boolean class architectures provide a variety of tradeoffs between hardware and time complexities beyond the direct conversion architectures. Our analysis results also show that the majority class performs better than the Boolean class as well as the architectures by direct conversion in both the hardware and time complexity, because the majority class takes better advantage of the more powerful nature of threshold gates.

The rest of the paper is organized as following. In Sec. 2, we introduce threshold logic and a nanotechnology device, resonant tunneling diode. Sec. 3 presents our threshold architectures for XORs. Sec. 4 computes and compares the complexities of various designs.



(a)



(b)

Fig. 1. (a) Threshold gate computing $f = [x_1, x_2, x_3; 1, -1, 2; 2]$; (b) RTD implementation.

2. BACKGROUND

2.1. Threshold logic

For $n \geq 1$, a threshold function f with n inputs $x_1, x_2, \cdots, x_n$ is a Boolean function whose output is determined by [3]

$$f(x_1, x_2, \cdots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq T \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where the real value w_i is called the *weight* of x_i and the real value T, the *threshold*. In this paper we represent this threshold function as $[x_1, x_2, \cdots, x_n; w_1, w_2, \cdots, w_n; T]$, and for simplicity sometimes denote it as $f = [\mathbf{x}; \mathbf{w}; T]$, where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{w} = (w_1, w_2, \cdots, w_n)$. The physical entity realizing a threshold function is called a threshold gate. Fig. 1(a) illustrates a threshold function $f = [x_1, x_2, x_3; w_1 = 1, w_2 = -1, w_3 = 2; T = 2]$, which corresponds to a Boolean expression $f = (x_1 + \overline{x_2}) \cdot x_3$, where "+" and "·" denote OR and AND, respectively.

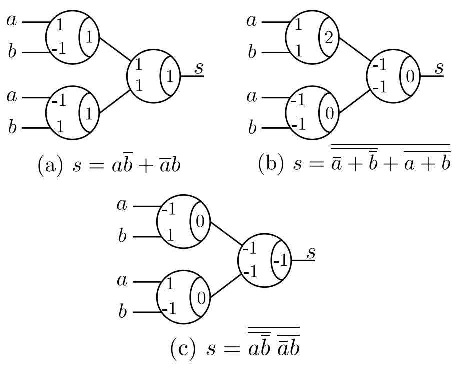
For the Boolean functions NOT and n-input AND, OR, NAND, and NOR, each corresponds to a single threshold function: $[x; -1; 0]$ is the NOT gate, $[\mathbf{x}; 1, 1, \cdots, 1; n]$ and $[\mathbf{x}; 1, 1, \cdots, 1; 1]$ are n-input AND and OR, respectively, $[\mathbf{x}; -1, -1, \cdots, -1; 0]$ and $[\mathbf{x}; -1, -1, \cdots, -1; 1-n]$ equal n-input NAND and NOR, respectively. Unfortunately, an XOR cannot be expressed as a single threshold function.

Certain threshold functions are of particular interest. An n-input threshold function with all unit weights and a threshold $\lfloor (n+1)/2 \rfloor$ is called a majority function. A threshold function with all unit weights but an arbitrary threshold is called a generalized majority function. Henceforth we denote a generalized n-input majority gate with a threshold k by t_k^n.

2.2. Resonant tunneling diode (RTD) technology

Among the new nanotechnologies, RTD is one of the most promising (see, e.g., [5]). Hence, in this work we focus on RTD implementations of threshold gates. RTD is a diode with resonant tunneling structure. It has a negative differential resistance, i.e., if one increases voltage across it, current through it initially increases and after reaching a certain peak, drops down to zero again. If two RTDs are tied in series and a control voltage across them is swept from low to high, then at the end, the RTD with the higher peak current bears all the applied voltage. The peak value of the current depends on the area of the RTD. By replacing each of these RTDs by multiple RTDs in parallel, and selectively adding them into the circuit with input variables, one can change the effective area of the top and bottom RTDs. The voltage at the junction of the two sets of the RTDs is thus decided by the comparison of the two sets of areas. Fig. 1(b) shows an RTD implementation of a threshold function $f = [x_1, x_2, x_3; 1, -1, 2; 2]$, where Clk is the control voltage, the threshold is composed of two RTDs, and the output composed of a *load* RTD and a *driver* RTD.

The fan-in of a threshold gate in RTD nanotechnology needs to be bounded for both reliability and performance.



(a) $s = a\bar{b} + \bar{a}b$
(b) $s = \overline{\overline{\bar{a}+\bar{b}} + \overline{a+b}}$
(c) $s = \overline{\overline{a\bar{b}}\,\overline{\bar{a}b}}$

Fig. 2. Threshold implementation of two-input XOR gate (a) SOP-type (b) NOR-type and (c) NAND-type. All three use threshold gates with a fan-in of only two.

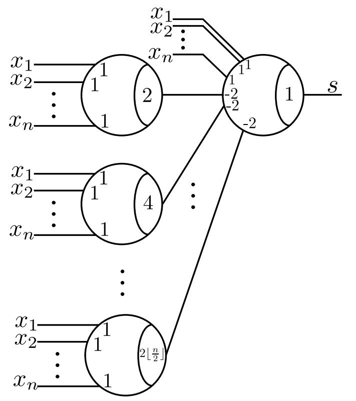


Fig. 3. Majority class implementation of $s = x_1 \oplus x_2 \cdots \oplus x_n$.

First, it is known that the reliability of an RTD threshold gate decreases sharply with the fan-in due to noise, fluctuation of supply voltage, and manufacture deviations (see [7] and the references therein). Second, the switching speed of RTDs depends on the ratio of load to drive peak currents [13]. The closer the ratio is to one, the slower the RTDs switch. Since a large gate with more inputs is more likely to produce a ratio closer to one than a small gate, it also suggests that the fan-in of an RTD threshold gate be bounded. A maximum fan-in of seven inputs was suggested for RTDs [5].

3. MULTI-INPUT XOR: THRESHOLD IMPLEMENTATION

In this section, we first present a traditional manner of implementing an n-input XOR, and then propose new designs.

3.1. Direct conversion

Although an XOR cannot be expressed as a single threshold function, one can first express an XOR based on other Boolean gates and then use their threshold logic implementations. We refer to this approach as direct conversion.

One can implement an n-input XOR through a binary tree of two-input XORs. A two-input XOR $s = a \oplus b$ can be expressed as $s = a\bar{b} + \bar{a}b$, $s = \overline{\overline{\bar{a}+\bar{b}} + \overline{a+b}}$, or $s = \overline{\overline{a\bar{b}}\,\overline{\bar{a}b}}$, and implemented as shown in Fig. 2. Among the three implementations in Fig. 2, the NAND-type implementation has the smallest hardware complexity and is therefore chosen in our implementation.

3.2. XOR with a small number of inputs

We consider two classes of architectures without considering the fan-in bound, which is valid when the number of inputs to an XOR is sufficiently small.

3.2.1. Boolean class of implementations

We can use Boolean algebra to express an n-input XOR based on two levels of NANDs. Since NANDs can be implemented as threshold gates, this provides a two-level implementation, called the *Boolean class* implementation.

Let s denote the XOR of x_1 x_2, ..., x_n. One can express s in a sum-of-product (SOP) form. The NAND implementation of such a form is obtained simply by using a NAND to combine the literals in each product term and combining the outputs of these NANDs with another NAND. For example, a three-input XOR $s = a \oplus b \oplus c$ can be expressed as $s = \overline{\overline{\bar{a}bc}\,\overline{a\bar{b}c}\,\overline{ab\bar{c}}\,\overline{\bar{a}\bar{b}\bar{c}}}$. Note that there are 2^{n-1} product terms in the SOP expression of an n-input XOR. Since the last threshold gate has inputs from each of these terms, this implementation is possible only when the fan-in bound B satisfies $B \geq 2^{n-1}$.

3.2.2. Majority class of implementations

It is also possible to devise a two-level implementation of an n-input XOR so that all the gates in the first level are generalized majority gates [3, 12]. For an n-input XOR $s = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, let the generalized majority functions t_i^n's with even threshold i's be the intermediate variables. Then, $s = [x_1, x_2, \cdots, x_n, t_2^n, t_4^n, \cdots, t_{2\lfloor \frac{n}{2} \rfloor}^n; 1, 1, \cdots, 1, -2, -2, \cdots, -2; 1]$, which can be implemented as shown in Fig. 3. This implementation consists of two stages. The first stage is to compute how many pairs of ones there are in the inputs. The second stage subtracts all pairs of ones from n. The result is either one (odd number of ones) or zero (even number of ones).

The threshold gate of the second stage has the maximum fan-in amongst all the gates and it equals $\lfloor 3n/2 \rfloor$. Thus these designs are useful only when $B \geq \lfloor 3n/2 \rfloor$.

3.3. XOR with a large number of inputs

When the number of inputs for an XOR exceeds the fan-in bound, the implementations in Sec. 3.2 cannot be used di-

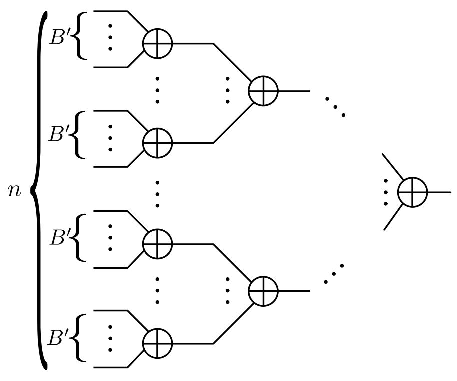


Fig. 4. n-input XOR realized as a tree of B'-input XORs, where $n = B'^l$ for some l.

rectly. Instead, an n-input XOR is realized using a tree of B'-input XOR gates as shown in Fig. 4. For given B and n, the height of the tree l is chosen to be the smallest number satisfying $n \le B'^l$, and B' is chosen so that no gate in the tree exceeds the fan-in bound B.

If the Boolean class implementation is used for the B'-input XORs, then from the results in Sec. 3.2.1, $B \ge 2^{B'-1}$, which gives $B' = 1 + \lfloor \log_2 B \rfloor$. Alternatively, if the majority class implementation is used, then from Section 3.2.2, $B \ge \lfloor 3B'/2 \rfloor$. This gives $B' = \lfloor (2B+1)/3 \rfloor$.

4. COMPLEXITY OF MULTI-INPUT XOR

We now investigate the hardware and time complexities of various designs presented in Section 3. Let a Boolean function of n variables be realized as a network of k threshold functions $f_i(\mathbf{x}) = [\mathbf{x}^i; \mathbf{w}^i; T_i]$ with $\mathbf{x}^i = (x_1^i, \cdots, x_{n_i}^i)$ and $\mathbf{w}^i = (w_1^i, \cdots, w_{n_i}^i)$ for $i = 1, \cdots, k$. The total area of the implementation, including the sum of areas of all the RTD gates as well as the *load* and the *driver* RTDs, is given by $A(n) = \sum_{i=1}^{k} (\sum_{j=1}^{n_i} |w_j^i| + |T_i|) + 2k$. The total number of gates in the design is given by $G(n) = k$. In addition, because the devices in nanotechnology are so small, the area occupied by interconnects between different devices is significant and hence needs to be accounted for. The number of interconnects $I(n)$ is given by $I(n) = \sum_{i=1}^{k} n_i$. Finally, we compute the latency $L(n)$ of a design by identifying the longest path from the input to the output.

4.1. Complexity of an XOR with a small number of inputs

The following two lemmas evaluate the complexity of our two classes of XOR designs in Sec. 3.2.

Lemma 4.1. *The complexity of an n-input Boolean class XOR with $n \le 1 + \lfloor \log_2 B \rfloor$ is given by $A(n) = 3(n+2)2^{n-2} + 1$, $G(n) = 2^{n-1} + 1$, $I(n) = (n+1)2^{n-1}$, and $L(n) = 2$.*

Proof. Since in this design there are only two stages, $L(n) = 2$. There are 2^{n-1} n-input gates in the first stage and one gate with 2^{n-1} inputs in the second stage. Hence, $G(n) = 2^{n-1} + 1$ and $I(n) = n2^{n-1} + 2^{n-1} = (n+1)2^{n-1}$.

The weights of all the inputs to all the gates in this design are either 1 or -1. The threshold of a gate implementing a product term with i variables complemented is $1 + i - n$. Note that i takes all possible odd values and there are $\binom{n}{i}$ distinct product terms corresponding to an i. The threshold of the gate in the last stage is 0. Thus the area of the design is $A(n) = 3(n+2)2^{n-2} + 1$. □

Lemma 4.2. *The complexity of an n-input majority class XOR with $n \le \lfloor (2B+1)/3 \rfloor$ is given by $A(n) = \lfloor n/2 \rfloor \lfloor (3n+10)/2 \rfloor + n + 3$, $G(n) = 1 + \lfloor n/2 \rfloor$, $I(n) = (n+1)\lfloor n/2 \rfloor + n$ and $L(n) = 2$.*

Proof. For this design, there are $\lfloor \frac{n}{2} \rfloor$ gates with n inputs in the first stage and one gate with $n + \lfloor \frac{n}{2} \rfloor$ inputs in the second stage. Hence, $G(n) = 1 + \lfloor n/2 \rfloor$, $I(n) = (n+1)\lfloor n/2 \rfloor + n$ and $L(n) = 2$.

All the gates in the first stage have a weight of 1 while n inputs of the gate in the second stage have weights 1 and the remaining $\lfloor n/2 \rfloor$ inputs have a weight of -2. The threshold of the i-th gate, $1 \le i \le \lfloor n/2 \rfloor$, has a weight $2i$, while the gate in the second stage has a weight of 1. Thus one gets $A(n) = \lfloor n/2 \rfloor \lfloor (3n+10)/2 \rfloor + n + 3$. □

According to Lemma 4.2, the area and interconnect complexities of an n-input XOR increase quadratically with n.

4.2. Complexity of an XOR with a large number of inputs

When the number of inputs to an XOR is large, one uses the designs in Sec. 3.3. In this case the XOR is implemented through a tree of B'-input XORs as illustrated in Fig. 4. For mathematical convenience, assume that $n = B'^l$ for some l. That is, the tree is complete with height $l = \log_{B'} n$. Thus the implementation involves $(B'^l - 1)/(B' - 1)$ B'-input XORs.

Since B' depends explicitly on the fan-in bound B, we use B as a parameter in the complexities. For example, instead of $A(n)$, we will now use the notation $A(n, B)$.

If the B'-input XORs use a Boolean class implementation, then as shown in Sec. 3.3, $B' = 1 + \lfloor \log_2 B \rfloor$. Using the complexity from Lemma 4.1 for each B'-input XOR, the complexity of an n-input XOR with the fan-in bound B is thus given by

$$A(n, B) = (n-1)[3(\lfloor \log_2 B \rfloor + 3)2^{\lfloor \log_2 B \rfloor - 1} + 1] \,/\, \lfloor \log_2 B \rfloor,$$

$$G(n, B) = (n-1)[(2^{\lfloor \log_2 B \rfloor} + 1)] \,/\, \lfloor \log_2 B \rfloor, \quad \text{and}$$

$$I(n, B) = (n-1)[(\lfloor \log_2 B \rfloor + 2)2^{\lfloor \log_2 B \rfloor}] \,/\, \lfloor \log_2 B \rfloor.$$

The latency of this network, $L(n, B) = 2\lceil \log_{\lfloor 1 + \log_2 B \rfloor} n \rceil$.

If on the other hand, the B'-input XORs use a majority class implementation, then as shown in Sec. 3.3, $B' =$

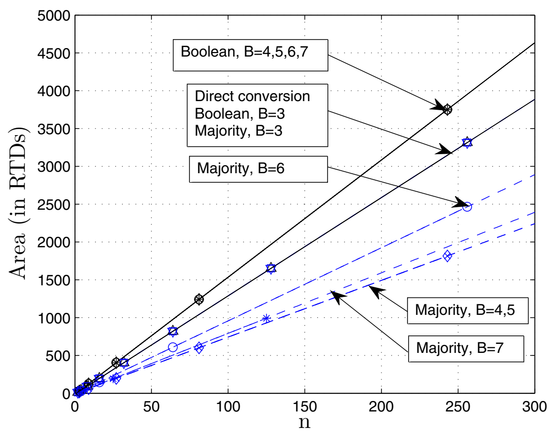


Fig. 5. Area complexity of multiple-input XOR.

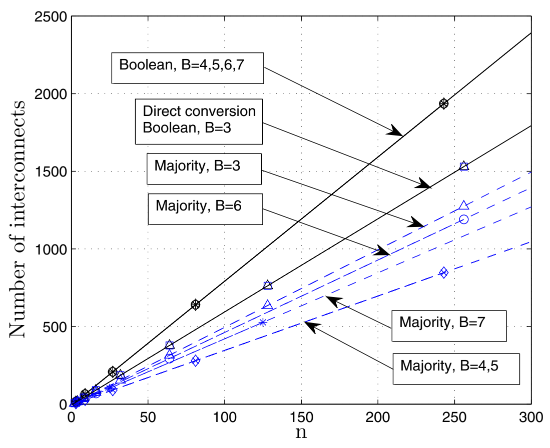


Fig. 7. Number of interconnects of multiple-input XOR.

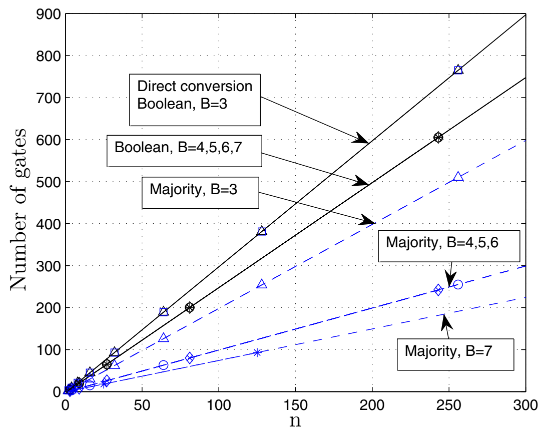


Fig. 6. Number of gates of multiple-input XOR.

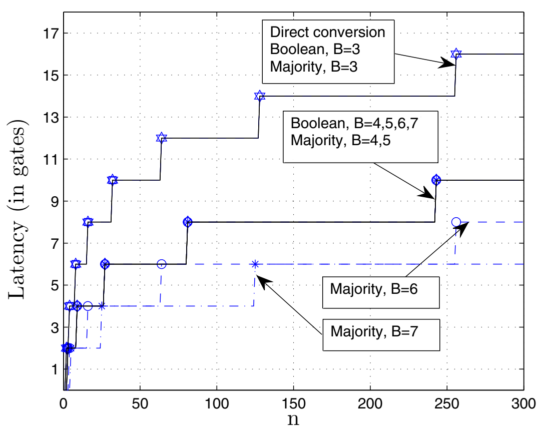


Fig. 8. Latency of multiple-input XOR.

$\lfloor(2B+1)/3\rfloor$. Using the complexity from Lemma 4.2 for each B'-input XOR, the complexity of an n-input XOR is thus given by

$$A(n,B) = \frac{n-1}{\lfloor\frac{2B-2}{3}\rfloor}(\lfloor\frac{B}{3}\rfloor^2+\lfloor\frac{2B+16}{3}\rfloor\lfloor\frac{B}{3}\rfloor+\lfloor\frac{2B+10}{3}\rfloor),$$

$$G(n,B) = \frac{n-1}{\lfloor\frac{2B-2}{3}\rfloor}\cdot(\lfloor\frac{B}{3}\rfloor+1), \quad \text{and}$$

$$I(n,B) = \frac{n-1}{\lfloor\frac{2B-2}{3}\rfloor}[\lfloor\frac{2B+4}{3}\rfloor\lfloor\frac{B}{3}\rfloor+\lfloor\frac{2B+1}{3}\rfloor].$$

Finally, the latency in this case is given by $2\lceil\log_{\lfloor(2B+1)/3\rfloor} n\rceil$. Note that all three parameters characterizing the hardware complexity, $A(n,B)$, $G(n,B)$ and $I(n,B)$, are linearly proportional to B and for a constant B, have an order $O(n)$.

In Tab. 1, we give the closed-form expressions of area, numbers of gates and interconnects, and latency of three classes of n-input XOR with respect to different B.

For a small n, the fan-in bound is not an issue and an XOR can be implemented in two stages. The area and the numbers of gates and interconnects of the Boolean class XORs are given by exponential functions of n. In contrast, the area and number of interconnects of the majority class XORs are given by quadratic functions of n.

For a large n, an n-input XOR is decomposed into a tree of B'-input XORs to satisfy the fan-in bound. The area, numbers of gates and interconnects, and latency of the three classes of XORs are compared in Figs. 5, 6, 7, and 8, respectively, where the expressions under the conditions $B \leq 2^{n-1}$ and $B \leq \lfloor\frac{3n}{2}\rfloor$ in Tab. 1 are depicted for $B = 3, 4, \cdots, 7$. Though the closed-form expressions in Tab. 1 are derived for some discrete values $n = B'^l$, we assume the expressions are valid for all n with $B \leq 2^{n-1}$ or $B \leq \lfloor\frac{3n}{2}\rfloor$. All three classes of XORs have linear complexity. The direct conversion has the same hardware complexities and latency as the Boolean class with $B = 3$, which implies that the

Table 1. Comparison of hardware complexities and latency for two classes of n-input XORs.

Class		Area	Number of Gates	Number of interconnects	Latency
Direct conversion		$13(n-1)$	$3(n-1)$	$6(n-1)$	$2\lceil \log_2 n \rceil$
Boolean	$B > 2^{n-1}$	$3(n+2)2^{n-2}+1$	$2^{n-1}+1$	$(n+1)2^{n-1}$	2
	$B \leq 2^{n-1}$	$\frac{n-1}{\lfloor \log_2 B \rfloor} \cdot [3(\lfloor \log_2 B \rfloor +3) \cdot 2^{\lfloor \log_2 B \rfloor -1} + 1]$	$\frac{n-1}{\lfloor \log_2 B \rfloor} \cdot (2^{\lfloor \log_2 B \rfloor} + 1)$	$\frac{n-1}{\lfloor \log_2 B \rfloor}(\lfloor \log_2 B \rfloor +2)2^{\lfloor \log_2 B \rfloor}$	$2\lceil \log_{\lfloor \log_2 2B \rfloor} n \rceil$
Majority	$B > \lfloor \frac{3n}{2} \rfloor$	$\lfloor \frac{n}{2} \rfloor \lfloor \frac{3n+10}{2} \rfloor + n + 3$	$\lfloor \frac{n}{2} \rfloor + 1$	$(n+1)\lfloor \frac{n}{2} \rfloor + n$	2
	$B \leq \lfloor \frac{3n}{2} \rfloor$	$\frac{n-1}{\lfloor \frac{2B-2}{3} \rfloor}(\lfloor \frac{B}{3} \rfloor^2 + \lfloor \frac{2B+16}{3} \rfloor \cdot \lfloor \frac{B}{3} \rfloor + \lfloor \frac{2B+10}{3} \rfloor)$	$\frac{n-1}{\lfloor \frac{2B-2}{3} \rfloor} \cdot (\lfloor \frac{B}{3} \rfloor + 1)$	$\frac{n-1}{\lfloor \frac{2B-2}{3} \rfloor}[\lfloor \frac{2B+4}{3} \rfloor \lfloor \frac{B}{3} \rfloor + \lfloor \frac{2B+1}{3} \rfloor]$	$2\lceil \log_{\lfloor \frac{2B+1}{3} \rfloor} n \rceil$

Boolean class includes the direct conversion as a special case and hence provides more tradeoffs between hardware and time complexities. From Fig. 5, both the Boolean class and majority class XORs have the same area when $B = 3$, the majority class XOR is more area efficient than the Boolean class XOR when $B = 4, 5, 6, 7$. From Figs. 5, 6, and 7, the numbers of gates and interconnects of the majority class XOR are smaller than those of the Boolean class XOR for any B. For example, when $B = 7$, the area and the numbers of gates and interconnects of Boolean class XOR are about twice of those of majority class XOR with the same number of inputs. For any given B, the latency of the majority class XOR is smaller than that of the Boolean class XOR. In summary, for $B = 3, 4, \cdots, 7$, the majority class XOR is more efficient than the Boolean class XOR.

5. CONCLUSION

Implementations of XORs are critical to all finite field arithmetic operations over GF(2^m). In this paper, we focus on efficient XOR implementations in RTD nanotechnology and propose two classes of threshold architectures of n-input XORs with a fan-in bound B. By a detailed analysis, we show that the proposed majority class outperforms the Boolean class implementation as well as the direct conversion in terms of both hardware and time complexities. The proposed architectures also account for the fan-in issue of RTD nanotechnology, which is crucial for both the reliability and performance. In future work, we plan to extend our work to efficient architectures of finite field multiplications in nanotechnology.

6. REFERENCES

[1] [Online], "International technology roadmap for semiconductors," available at http://www.itrs.net/Links/2009ITRS/Home2009.htm.

[2] D. Goldhaber-Gordon, M. S. Montemerlo, J. C. Love, G. J. Opiteck, and J. C. Ellenbogen, "Overview of nanoelectronic devices," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 521–540, April 1997.

[3] S. Muroga, *Threshold Logic and Its Applications*. New York: WILEY-INTERSCIENCE, 1971.

[4] V. Annampedu and M. D. Wagh, "Reconfigurable approximate pattern matching architectures for nanotechnology," *Microelectronics*, vol. 38, pp. 430–438, 2007.

[5] C. Pacha, U. Auer, C. Burwick, P. Glosekotter, A. Brennemann, W. Prost, F. Tegude, and K. F. Goser, "Threshold logic circuit design of parallel adders using resonant tunneling devices," *IEEE Trans. VLSI Systems*, vol. 8, no. 5, pp. 558–572, October 2000.

[6] C. Pacha and K. Goser, "Design of arithmetic circuits using resonant tunneling diodes and threshold logic," in *Proc. of the 2nd Workshop on Innovative Circuits and Systems for Nanoelectronics*, Delft, NL, Sep. 1997, pp. 83–93.

[7] Y. Sun and M. D. Wagh, "A fan-in bounded low delay adder for nanotechnology," in *Proc. of 2010 NanoTech Conf., vol. 2*, Anaheim, CA, July 2010, pp. 83–86.

[8] B. Sunar and C. L. Koc, "Mastrovito multiplier for all trinomials," *IEEE Trans. Computers*, vol. 48, no. 5, pp. 522–527, May 1999.

[9] E. D. Mastrovito, "VLSI architectures for computations in Galois field," Ph.D. Dissertation, Linkoping University, 1991.

[10] J. Deschamps, J. L. Imana, and G. D. Sutter, *Hardware Implementation of Finite-Field Arithmetic*. The McGraw-Hill Companies, 2009.

[11] J. L. Massey and J. K. Omura, "Computational method and apparatus for finite field arithmetic," *US Patent No. 4,587,627, to OMNET Assoc., Sunnyvale CA, Washington, D.C.: Patent and Trademark Office*, 1986.

[12] R. C. Minnick, "Linear-input logic," *IRE Trans. Electronic Computers*, vol. EC-10, no. 1, pp. 6–16, March 1961.

[13] K. Maezawa, "Analysis of switching time of monostable-bistable transition logic elements based on simple model calculation," *Jpn. J. Appl. Phys.*, vol. 34, no. 2B, pp. 1213–1217, February 1995.