Building multi-input RTD circuits under reliability constraints

Viswanath Annampedu and Meghanad D. Wagh
Department of Electrical and Computer Engineering
Lehigh University, Bethlehem, PA 18015
Email: via2, mdw0@lehigh.edu

Abstract

Threshold functions can be implemented in nanotechnology using the MOBILE architecture based on resonant tunneling diodes (RTDs). The operational reliability of the architecture is greatly dependent upon the number of RTDs used, due to the variance of the accumulated peak currents. This constrains the number of inputs to the threshold circuit. The number of inputs is also a measure of the area and interconnect complexity. This paper shows that a threshold function with n inputs can be implemented as a network of smaller threshold functions. Our implementation uses linear chains of identical threshold functions and is therefore easier to fabricate and can be easily pipelined for high throughput. The total input complexity of all the functions involved in our implementation is $O(nR^2)$, where R is the difference between the sum of all the positive weights and the threshold. This compares very favorably with the traditional decompositions of threshold functions which have a total input complexity of $O(2^n)$.

1 Introduction

Resonant tunneling diodes (RTDs) represent the most mature nanotechnology devices available to date for implementing new innovative applications [1, 2]. They employ quantum electron tunneling creating the effect of negative resistance. Two RTDs connected in series as shown in Fig. 1 make up the most basic logic element with two discrete logic states. This architecture is known as a *monostable-bistable transition logic element* (MOBILE) [3].

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, V_p, where the current peaks. As V_{SWP} is increased from 0 to more than $2V_p$, V_{out} settles to a low value (corresponding to a logic 0) if $I_A < I_B$ or to a high value (corresponding to a logic 1) if $I_A > I_B$. When the output settles to either of the two values, the current flowing through the circuit is very small.

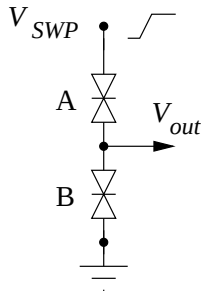

Figure 1. Schematic representation of a MOBILE made up of two RTDs in series.

By connecting multiple RTDs in parallel, one can create an effective peak current equal to the 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 an architecture based on this concept. In this figure, depending upon the corresponding HFETs, the RTDs with areas α_1, α_4, α_5, α_7 and α_8 contribute to the effective peak current in the top section of the MOBILE. Similarly, RTDs with areas α_2, α_3 and α_6 contribute to the bottom peak current. Consistent with the MOBILE behavior, the logic state of the output is determined by

$$V_{out} = \begin{cases} 1 & \text{if} \quad x_1\alpha_1 - x_2\alpha_2 - x_3\alpha_3 + x_4\alpha_4 + \\ & \quad x_5\alpha_5 - x_6\alpha_6 + x_7\alpha_7 + x_8\alpha_8 \geq T \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

[1]The monolithic integration of HFET and RTD is known as the three terminal hybrid Resonant Tunneling Transistor [4]

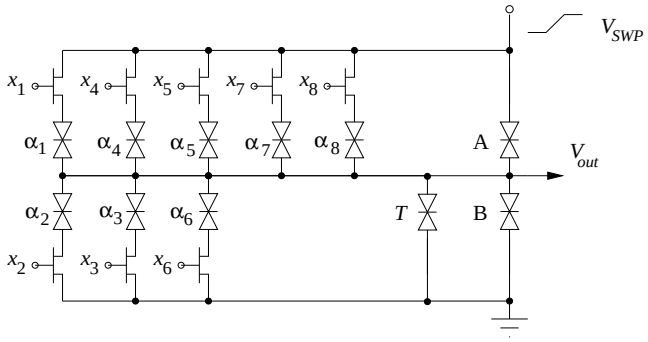

Figure 2. Implementing a threshold circuit using the MOBILE ideas. T and w_i denote the areas of the corresponding RTDs. RTDs A and B have unit areas.

Equation (1) is the equation for a typical threshold function[2]. In (1), constants α_1, $-\alpha_2$, $-\alpha_3$, ..., α_8 are the weights w_1, w_2, w_3, ..., w_8 of the corresponding inputs and T is the threshold. The symbolic representation of the threshold function characterized by (1) is shown in Fig. 3.

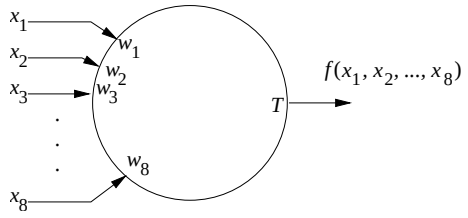

Figure 3. Representation of the threshold function $f(x_1, x_2, \ldots, x_8)$ with weights $w_1, w_2, \ldots, w_8$ and threshold T.

It is known that there are $2^{n^2/2}$ distinct threshold functions of n variables including AND, OR and NOT gates [5]. Thus any Boolean function can be expressed through one or more threshold functions and implemented by RTD/HFET circuits. In addition, some applications inherently have threshold solutions. For example, one can match two arbitrary n bit patterns within any given error tolerance with a single $2n$ input threshold function [6]. Such circuits are natural candidates for RTD/HFET implementations.

The operation of the threshold circuit implemented through the RTDs is based upon the comparison of the composite peak currents in the top and the bottom sections of the architecture. The value of each individual RTD peak current - being determined by the nano-range dimensions of the device - may have a large variance. The variance of the composite peak current is determined by all the conducting RTDs and, if sufficiently large, can create a wrong output logic value. Naturally, the reliability of an implementation is closely related to the number of RTDs contributing to the aggregate current [7]. Limiting the number of inputs also serves to limit the number of contributing RTDs and thereby ensuring a more reliable implementation. Further, it also bounds the maximum instantaneous peak current and hence the power, for a more robust operation.

This paper discusses partitioning threshold circuits with a large number of inputs into smaller threshold circuits so as to improve the overall reliability of the RTD/ HFET implementations. The size of the threshold circuits in the final implementation is governed by the trade-off between reliability and circuit complexity. We show that if small sets of inputs are processed in independent threshold circuits (called *fragments*), it is still possible to combine their outputs to realize the original threshold function. The architecture used for this recombination can be built using small threshold circuits (called *recombiners*) arranged as a binary tree. (See Fig. 5.) Further,each of these recombiners is shown to consist of one or more linear chains of small identical threshold circuits.

In Section 2 we propose a novel concept of *error*. This error is based on the inputs and can be used to define the output of a threshold function. We also provide an overview of our decomposition method and discuss the complexity of the fragments. In Section 3 we prove that the recombiners can be built out of threshold circuits. Section 4 is devoted to the design of recombiners, and in particular, to the decomposition of the recombiners into chains of identical threshold modules. Section 5 shows that the total input complexity of our architecture implementing an n input threshold function is $O(nR^2)$, where R is the difference between the sum of all positive weights and the threshold. This compares favorably with the complexity $O(2^n)$ of the classical decomposition [8]. Section 6 explores other nanotechnology implementation issues. Finally we provide our conclusions in Section 7.

2 Partitioning a large threshold circuit

As explained in Section 1, the reliability of a threshold function implemented as a RTD/HFET circuit can be improved by realizing it through multiple threshold functions, each with a small number of inputs. The classical method to achieve this is to decompose the threshold function into multiple threshold functions [8] as

$$f(x_1, x_2, \ldots, x_n) = f_1(x_2, x_3, \ldots, x_n) + x_1 f_2(x_2, x_3, \ldots, x_n). \quad (2)$$

[2]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 $\sum_{i=1}^{n} w_i x_i \geq T$ if and only if $f(x_1, x_2, \ldots, x_n) = 1$.

In (2), both $n-1$ variable functions f_1 and f_2 are threshold and variable x_1 is assumed to be positive. For negative variables, similar decomposition is possible. Since AND and OR gates are threshold functions, one can implement f as in Fig. 4. Equation (2) can be employed repeatedly to reduce the number of threshold inputs to any small value dictated by the technology. However, as is evident from Fig. 4, each time decomposition is used, the number of threshold functions double (not counting the AND and OR gates). If the reliability consideration limits the number of inputs to a threshold circuit to L ($< n$), then the final decomposition would have 2^{n-L} threshold circuits each with L inputs. Thus the total input complexity (i.e., the total number of inputs to all the threshold gates) of the classical decomposition is given by

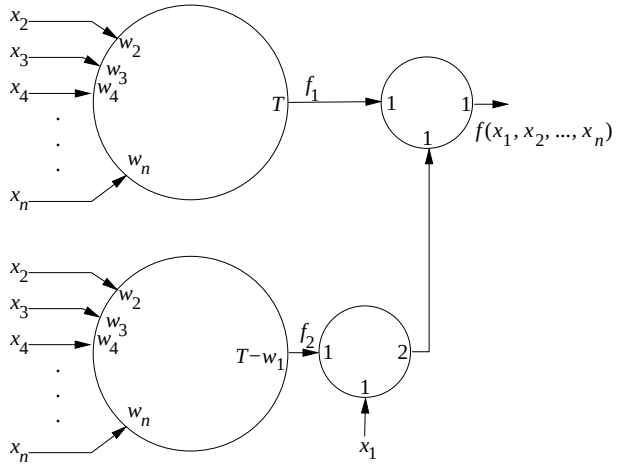

Figure 4. Classical decomposition of a threshold function.

$$(L+2)2^{n-L}. \tag{3}$$

The total input complexity is a measure of the circuit complexity [9]. In addition to the large $O(2^n)$ total input complexity, the classical decomposition also suffers from the fact that the L inputs x_{n-L+1} to x_n that are not used in the decomposition have a fanout[3] of 2^{n-L}. Fanout of other input variables x_i, $1 \le i \le n-L$, is equal to 2^i. This large fanout implies excessive load on the circuits or sensors providing these inputs.

In this paper we propose an alternate decomposition of the threshold function in which each input x_i is applied to a small number of threshold functions, i.e., each input variable has a small fanout. We call the threshold circuits to which inputs are applied as *Fragments* and the threshold circuits which combine the outputs of the fragments into the final output of the function as the *Recombiners*. This scheme is shown in Fig. 5.

We assume that the weights and threshold are integers. One may scale the weights and threshold to achieve this. This is also consistent with nanotechnology since all RTD areas are multiples of some unit area. We also assume that the weights and threshold have been minimized.

Note that a threshold function output is determined by the comparison of a weighed sum of its inputs $\sum w_i x_i$ with the threshold, T. Unfortunately, the weighed sum of the inputs to each fragment lies between the sum of the negative weights of these inputs to the sum of the positive weights. Thus each fragment may produce the sums within different bounds. As a result, the task of combining these sums and then comparing it with the threshold would be very difficult.

In this paper, we demonstrate a different approach. Rather than computing and passing on the weighed sums by each fragment, we compute and pass on the *error*. Error E for the function f is defined as

$$E = K - \sum_{i=1}^{n} w_i x_i, \quad \text{where} \quad K = \sum_{\substack{i=1 \\ w_i>0}}^{n} w_i \tag{4}$$

Let R denote the constant $K-T$, where T is the threshold. Clearly, $R \ge 0$; otherwise the function would always be 0. Further, $E \ge 0$ because K is the largest value of $\sum w_i x_i$. The non-negative character of E is ideal for information transfer between the fragments and recombiners. Recall that the function $f(x_1, x_2, \ldots, x_n) = 1$ if and only if $\sum_{i=1}^{n} w_i x_i \ge T$. But from (4), this condition is equivalent to

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if } E \le R \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

In the proposed decomposition, we compute the error of each fragment j based on the set of indices S_j that determines its inputs as

$$E_j = K_j - \sum_{i \in S_j} w_i x_i, \quad \text{where} \quad K_j = \sum_{\substack{i \in S_j \\ w_i>0}} w_i. \tag{6}$$

As before, each $E_j \ge 0$ and

$$\begin{aligned} \sum_j E_j &= \sum_j K_j - \sum_j \sum_{i \in S_j} w_i x_i \\ &= K - \sum_{i=1}^{n} w_i x_i = E. \end{aligned} \tag{7}$$

Thus the errors within individual fragments give the total error E which is needed to establish the value of the function by (5). We use the recombiners to add E_js together. Since one is only interested in comparing E to R, a fragment only

[3] Fanout of x_i is the number of threshold circuits it is applied to.

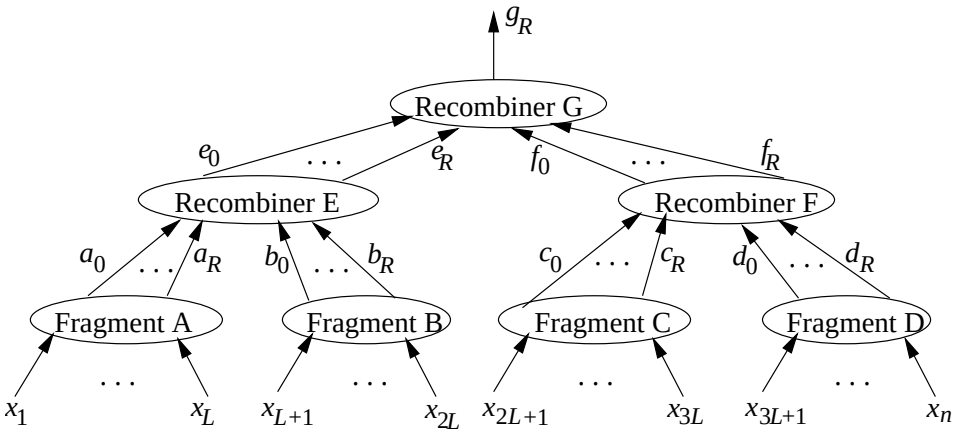

Figure 5. A threshold function of n variables and threshold T partitioned into four fragments and recreated by a tree of recombiners. R equals sum of all positive weights minus T.

needs to transfer the value of E_j if it is less than or equal to R. We use $R+1$ outputs from each fragment to carry E_j. The tth output from a fragment, say p_t, $0 \leq t \leq R$ is defined as

$$p_t = \begin{cases} 1 & \text{if} \quad \text{error in that fragment} \leq t \\ 0 & \text{if} \quad \text{error in that fragment} > t \end{cases} \tag{8}$$

Clearly each output p_t of fragment j is a threshold function. It uses inputs x_i with weights w_i, $i \in S_j$ and a threshold of $K_j - t$. Note that p_ts of different ts are related. In particular, one has

$$p_i p_j = p_i \quad \text{if} \quad i < j. \tag{9}$$

3 Recombiner as a threshold function

We combine the outputs of the fragments by using a binary tree of recombiners. Each recombiner outputs the combined error in all the fragments feeding into it. As before, we are only interested in the error values between 0 and R and can output these through $R+1$ lines defined in the same manner as in (8). Let p_i, $0 \leq R$ and q_i, $0 \leq R$ be the $2R+2$ inputs to a recombiner from its two descendants. The tth output of the recombiner, s_t, is based on the $2t+2$ inputs p_i, $0 \leq i \leq t$ and q_i, $0 \leq i \leq t$ and is given by the Boolean expression

$$s_t(p_0 : p_t;\ q_0 : q_t) = \sum_{i=0}^{t} p_i q_{t-i}, \qquad 0 \leq t \leq R. \tag{10}$$

Relation (10) may be justified by noting that the product $p_i q_{t-i}$ is 1 only if the left descendant of the recombiner has at most i errors and the right, at most $t-i$ errors. Thus each term of the summation (10) accounts for a case when the total number of errors are t or less. Each term is 0 if combined error from the two descendants is greater than t. We now state the most important property of the function s_t.

Theorem 1 *Function s_t defined by (10) is a threshold function.*

Proof. We prove this theorem by induction on t. Note first that $s_0(p_0;\ q_0) = p_0 q_0$ is clearly threshold. Similarly, using (9), function s_1 can be simplified as

$$\begin{aligned} s_1(p_0 : p_1;\ q_0 : q_1) &= p_0 q_1 + p_1 q_0 \\ &= p_0 p_1 q_1 + p_1 q_0 q_1 \\ &= p_1 q_1 (p_0 + q_0). \end{aligned} \tag{11}$$

Equation (11) shows that s_1 is a threshold function because it is built by multiplying a threshold function $p_0 + q_0$ by two *new* variables p_1 and q_1.

Now assume that s_{t-2} is a threshold function. We will then prove that s_t is threshold. Using (9) we can write (10) as

$$\begin{aligned} s_t(p_0 : p_t;\ q_0 : q_t) &= \sum_{i=1}^{t-1} p_i p_t q_{t-i} q_t + p_0 p_t q_t + p_t q_0 q_t \\ &= p_t q_t (\sum_{i=1}^{t-1} p_i q_{t-i} + p_0 + q_0) \\ &= p_t q_t (\sum_{i=0}^{t-2} p_{i+1} q_{t-1-i} + p_0 + q_0) \end{aligned}$$

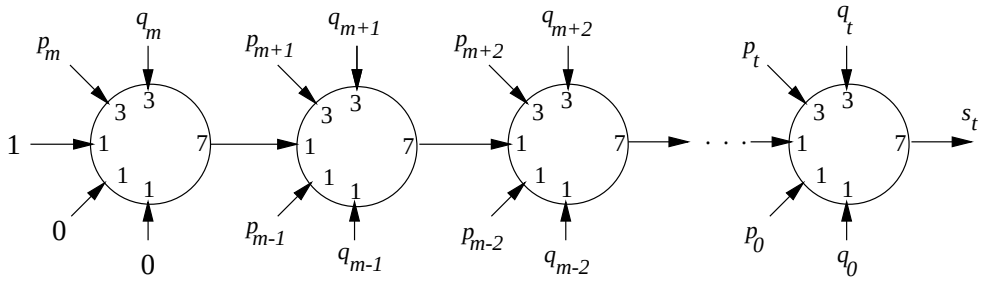

Figure 6. Recursive implementation of s_t with 5 input threshold modules for even t. $m = t/2$.

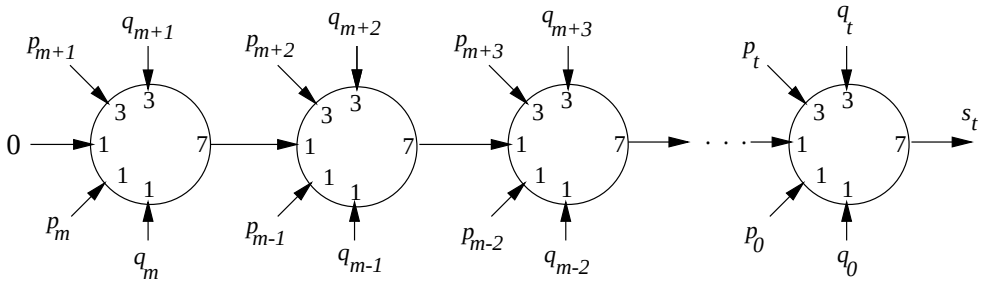

Figure 7. Recursive implementation of s_t with 5 input threshold modules for odd t. $m = \lfloor t/2 \rfloor$.

The summation term in the above equation is the expression for s_{t-2} with some change of variables. In particular,

$$s_t(p_0 : p_t;\ q_0 : q_t) = p_t q_t(s_{t-2}(p_1 : p_{t-1};\ q_1 : q_{t-1}) + p_0 + q_0) \quad (12)$$

Equation (12) shows that function s_t can be built from a threshold function s_{t-2} by adding two *new* variables p_0 and q_0 and multiplying the result by two *new* variables p_t and q_t. Function s_t is therefore threshold. ∎

4 Recombiner implementation

Theorem 1 shows that each s_t is a threshold function of $2t + 2$ variables. But since t may be as large as R, implementation of s_t as a single threshold function might not be reliable in nanotechnology. This section focuses on implementations of s_t with multiple (and if possible, identical) threshold functions, each with a small number of inputs.

One may implement s_t directly from its definition (10) using $t + 1$ two input ANDs and a $t + 1$ input OR, all implemented as threshold circuits. However, as will be shown later, the total input complexity of this solution is worse than the solution presented here. Further, such an implementation has poor layout efficiency.

The implementations we provide here are recursive and are a direct consequence of (12). They consist of chains of identical threshold modules. The length of the chain is dependent upon t but the modules themselves are identical for all ts. To obtain these chains, we repeatedly apply (12) to decrease the index of s_t. In particular, after i applications of (12), function s_t can be expressed in terms of function $s_{t-2i}(p_i : p_{t-i};\ q_i : q_{t-i})$. Let $m = \lfloor t/2 \rfloor$. Since the function index goes down by 2, each time (12) is applied, s_t reduces to $s_0(p_m;\ q_m)$ for an even t and to $s_1(p_m : p_{m+1};\ q_m : q_{m+1})$ for an odd t. Thus s_t can be implemented by first building s_0 (or s_1 for odd t) and then building successively higher indexed s_i from that using (12). Note that to get s_{i+2} from s_i, one needs to add to it two *new* variables and multiply the result with two other *new* variables. Thus s_{i+2} may be created from s_i by using a threshold function. Also because the same equation (12) is repeatedly used, the resultant threshold modules are all identical. This discussion is summarized by the following procedure to compute s_t. Let $m = \lfloor t/2 \rfloor$.

$$\text{Set } f_m = \begin{cases} p_m q_m & \text{if } t \text{ is even} \\ p_{m+1} q_{m+1}(p_m + q_m) & \text{if } t \text{ is odd} \end{cases}$$

Let $f_i = (f_{i+1} + p_i + q_i)p_{e-i}q_{e-i} \quad i = m-1, m-2, \ldots, 0$. Finally set $s_t = f_0$.

This procedure directly yields the implementation of s_t shown in Figs. 6 and 7. Note that each implementation has $\lfloor t/2 \rfloor + 1$ identical blocks connected in series for any t.

If each threshold circuit is limited to only 3 inputs, then one may further decompose each threshold circuit with five

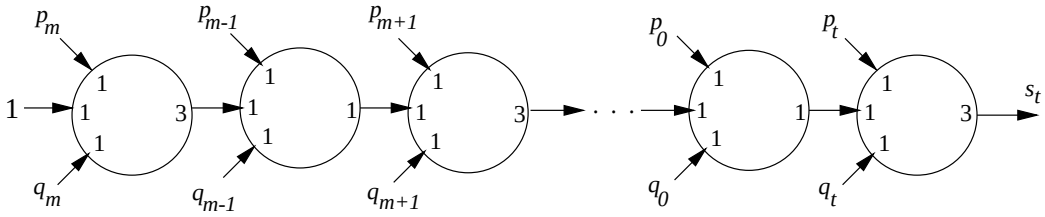

Figure 8. Recursive implementation of s_t with 3 input threshold modules for even t. $m = t/2$.

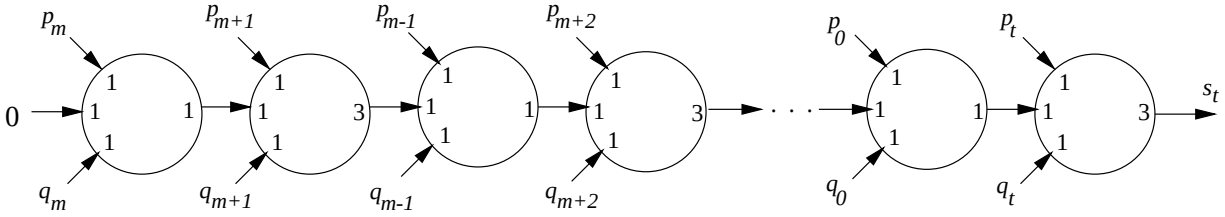

Figure 9. Recursive implementation of s_t with 3 input threshold modules for odd t. $m = \lfloor t/2 \rfloor$.

inputs into two circuits with 3 inputs each. The resultant implementations are shown in Figs. 8 and 9. Note that these implementations consist of $t+1$ blocks in series, alternating between only two types of threshold circuits for any t.

5 Total input complexity

We now compute the total input complexity (i.e., the total number of inputs to all the threshold gates) of the architecture proposed here.

Let $I(t)$ denote the total input complexity of computing output s_t of a recombiner using threshold modules with 5 inputs. From Figs. 6 and 7, one gets

$$I(t) = \begin{cases} 5(t/2) + 2 & \text{if } t \text{ is even} \\ 5\lfloor t/2 \rfloor + 4 & \text{if } t \text{ is odd} \end{cases} \tag{13}$$

Besides allowing for efficient layout, the decomposition of s_t obtained here also lowers the total input complexity. For example, implementation of s_{10} as proposed in Fig. 6 has a total input complexity of 27. With a direct implementation (based on (10)) using AND/OR gates (limited to 5 inputs each), this complexity rises to 35, about 30% higher. Further, our implementation uses identical modules for every s_t which is an advantage for manufacturing.

To obtain the complexity of a recombiner, recall that it needs to compute functions s_i, $0 \leq i \leq R$ (see Fig. 5). Thus the total input complexity, I_C, of a recombiner for generating all the outputs is given by

$$I_C = \begin{cases} (5R/2+2) + (5\lfloor (R-1)/2 \rfloor + 4) + \\ \quad (5(R-2)/2+2) + \ldots + 2 \text{ for even } R \\ \\ (5\lfloor R/2 \rfloor + 4) + (5(R-1)/2+2) + \\ \quad (5\lfloor (R-2)/2 \rfloor + 4) + \ldots + 2 \text{ for odd } R. \end{cases}$$

This equation can be simplified to

$$I_C = \begin{cases} 1.25R^2 + 3R + 2 & \text{for even } R \\ 1.25R^2 + 3R + 1.75 & \text{for odd } R \end{cases} \tag{14}$$

We can do similar analysis for the recombiners using threshold modules with 3 inputs. Let the total input complexity of computing s_t in this case be denoted by $I'(t)$. An inspection of Figs. 8 and 9 gives

$$I'(t) = 3t + 2 \tag{15}$$

As before, even in this case, one needs to compute all the functions s_i, $0 \leq i \leq R$ in each recombiner (see Fig. 5). Thus the input complexity, I'_C, of a recombiner generating all the required outputs using 3 input modules is given by

$$I'_C = 1.5R^2 + 3.5R + 2. \tag{16}$$

We now compute the total input complexity of our complete architecture shown in Fig. 5. Note that each fragment has $R+1$ independent threshold circuits. Thus the total input complexity of all the fragment threshold circuits is

$$I_F = n(R+1). \tag{17}$$

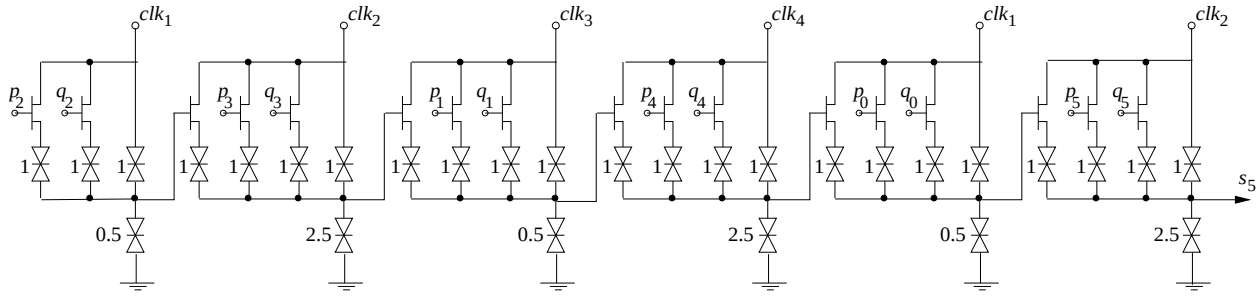

Figure 10. A nanotechnology implementation of output s_5 of a recombiner.

Further, if each fragment has at most L inputs, there will be $\lceil n/L \rceil$ fragments and $\lceil n/L \rceil - 1$ recombiners in the architecture. Of these, the final recombiner has only one output s_R. Thus using the 5 input modules in recombiners , the input complexity of the entire architecture is given by (13), (14) and (17) as:

$$I_F + (\lceil n/L \rceil - 2)I_C + I(R). \tag{18}$$

Equation (18) evaluates to

$$\begin{array}{l} (1.25\lceil n/L \rceil - 2.5)R^2 + (3\lceil n/L \rceil + n - 3.5)R + \\ \quad (2\lceil n/L \rceil + n - 2) \quad \text{for even } R \\ (1.25\lceil n/L \rceil - 2.5)R^2 + (3\lceil n/L \rceil + n - 3.5)R + \\ \quad (1.75\lceil n/L \rceil + n - 2) \quad \text{for odd } R. \end{array} \tag{19}$$

On the other hand, if only 3 input threshold modules are used in the recombiner, then the total input complexity of the architecture is

$$(1.5\lceil n/L \rceil - 3)R^2 + (3.5\lceil n/L \rceil + n - 4)R + (2\lceil n/L \rceil + n - 2) \tag{20}$$

From (19) and (20) one can see that the total input complexity of our architecture is $O(nR^2)$ which compares very favorably with the complexity $O(2^n)$ of the classical decomposition given in (2).

6 Nanotechnology implementation

The architectures developed in the earlier sections are well suited for reliable nanotechnology implementations. They use networks of threshold circuits which can be easily realized using MOBILE based architectures discussed in Section 1. The small number of inputs to each individual threshold circuit implies that the variance of the accumulated peak currents within a MOBILE is small and therefore the operation of the MOBILE is less prone to errors.

Fig. 10 shows implementation of a typical output s_5 of the recombiner using RTD/HFET nanotechnology devices. One may note that this figure shows the implementation of the threshold network in Fig. 9 for $t = 5$.

To make the comparison between the weighed sum of the inputs to an individual circuit and its threshold 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 the output is 0 and T_H, the smallest value of the weighed sum when the output is 1. In all the modules used in the decomposition of a recombiner one can see that $T_H = T$ and $T_L = T - 1$, where T is the threshold of the module. Thus in case of these modules, the optimal threshold to be used for robustness is $T = (T_L + T_H)/2 = T - 0.5$. This is illustrated in Fig. 10.

In order that the series connection of threshold modules works as desired, one needs to ensure that when a MOBILE is swept, its inputs are constant. This calls for a four phased clock shown in Fig. 11. The four clocks in the figure are ap-

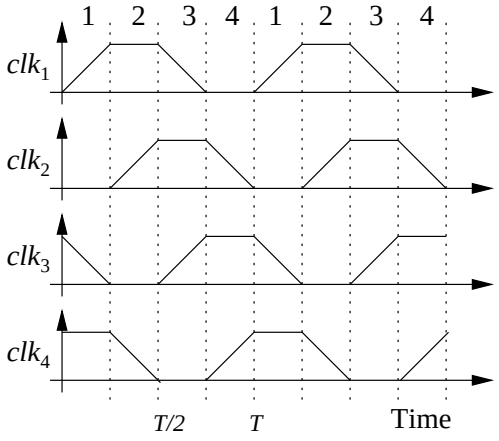

Figure 11. A four phase clocking scheme to drive a serial connection of MOBILES such as the one in Fig. 10

plied to consecutive modules as shown in Fig. 10. During

the first phase of clock, clk_1 rises and the first MOBILE in the series computes its output. This output is latched as clk_1 is held high during phase two, while at the same time, a rising clk_2 allows the second MOBILE to use this input for its computation. During the third phase of clock, clk_2 is held high resulting in the output of the second MOBILE to be latched. In this phase, a rising clk_3 allows the third MOBILE to create its correct output using this latched value. Output of the third MOBILE is then similarly used by the fourth MOBILE in the fourth phase using the rising clk_4. This process continues with all the modules. Note that each clock rises in one phase, allowing computation, is held steady in the next phase, allowing latching of its outputs, falls in the following phase to reset and remains low for the last phase for the inputs to change. The complete architecture thus behaves as a four stage pipeline [10] with a throughput of one recombiner calculation per clock.

7 Conclusions

It is known that RTD/HFET nanotechnology circuits can efficiently implement threshold logic. Implementations of small threshold functions are generally more reliable than large functions because they limit the worst case peak currents and current fluctuations in the circuit.

This paper presents a new strategy to decompose multi-input threshold functions into networks of threshold functions, each with a small number of inputs. The resultant realizations are expected to have a high reliability. The decomposition leads to a network of serially connected identical threshold modules which is easy to pipeline for high throughput. The total input complexity of our decomposition of an n input threshold function is only $O(nR^2)$, where R equals the difference between the sum of all the positive weights of the threshold function and the threshold. This compares favorably with the total input complexity $O(2^n)$ of classical threshold function decomposition.

References

[1] K. Goser and C. Pacha, "System and circuit aspects of nanoelectronics," in *24th European Solid-State Circuits Conf.*, (The Hague, NL), pp. 18–29, Sep. 1998.

[2] K. Nikolic, D. Berzon, and M. Forshaw, "Relative performance of three nanoscale devices – CMOS, RTDs and QCAs – against a standard computing task," *Nanotechnology*, vol. 12, pp. 38–43, March 2001.

[3] T. Akeyoshi, K. Maezawa, and T. Mizutani, "Weighted sum threshold logic operation of MOBILE (monostable-bistable transition logic element) using resonant-tunneling transistors," *IEEE Elec. Dev. Lett.*, vol. 14, pp. 475–477, Oct. 1993.

[4] C. Pacha, K. Goser, A. Brennemann, and W. Prost, "A threshold logic full adder based on resonant tunneling transistors," *24th European Solid-State Circuits Conf.*, pp. 428–431, 1998.

[5] J. Håstad, "On the size of weights for threshold gates," *SIAM J. Discrete Math*, vol. 7, pp. 484–492, Aug 1994.

[6] V. Annampedu and M. D. Wagh, "Approximate pattern matching in nanotechnology," in *Proc. of Nanotech 2006*, vol. 3, (Boston, MA), pp. 316–319, May 7–11 2006.

[7] W. Prost, U. Auer, F.-J. Tegude, C. Pacha, K. F. Goser, G. Janssen, and T. van der Roer, "Manufacturability and robust design of nanoelectronic logic circuits based on resonant tunnelling diodes," *Int. J. Circ. Theor. Appl.*, vol. 28, pp. 537–552, 2000.

[8] G. S. Glinski and C. K. Yue, "Decomposition of n-variable threshold function into p-variable threshold functions, where $p < n$," tech. rep. 63-10, Dept. of EE, Univ. of Ottawa, 1963.

[9] R. Zhang, P. Gupta, L. Zhong, and N. K. Jha, "Threshold network synthesis and optimization and its application to nanotechnologies," *IEEE Trans. on Comp. Aided Design*, vol. 24, pp. 107–118, Jan 2005.

[10] P. Gupta and N. K. Jha, "An algorithm for nanopipelining of RTD-based circuits and architectures," *IEEE Trans. on Nanotechnology*, vol. 4, pp. 159–167, Mar 2005.