jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2012, VOL. 58, NO. 1, PP. 49–54
Manuscript received December 31, 2011; revised March 2012. DOI: 10.2478/v10177-012-0007-5

Synthesis of Generalised Threshold Gates and Multi Threshold Threshold Gates

Maciej Nikodem, Marek A. Bawiec, and Janusz Biernat

***Abstract*—This paper presents synthesis algorithms for Generalised and Multi Threshold Threshold Gates. Both algorithms can be applied to generate circuit structures for arbitrary Boolean functions. We present gate's formal models, synthesis algorithms and complexity estimations of the resulting structures.**

***Keywords*—Threshold gates, synthesis, Boolean functions, MTTG, GTG.**

I. Introduction and Related Work

ACCORDING to the International Technology Roadmap for Semiconductors (ITRS) [1] complementary metal-oxide semiconductor (CMOS) circuits will satisfy the growing requirement for high performance for another 10–15 years. However, advances in nanoelectronics have already brought out a number of nanoscale devices which can be used in implementation of complex functions [2]. Devices such as resonant tunnelling diodes (RTDs), quantum cellular automata (QCA), carbon nanotubes, nanowires or single electron transistors demonstrate a negative differential resistance (NDR) property which can be exploited to implement logic gates. The focus on RTDs ensues from the fact that they allow to improve characteristics of both analog and digital electronic circuits in terms of switching frequencies and functional versatility. Moreover, RTDs are the most matured NDR devices used as building blocks of NDR-based logic circuits. A number of models and simulation methods has been suggested for RTDs [3]–[5]. Nevertheless, it's worth to note that other devices featuring NDR property can also be used as building blocks of electronic circuits that are capable of implementing complex Boolean functions in a single gate structure. For example, papers by Bhattacharya et al. and Le et al. [6], [7] depict the possibility of utilising such devices and present successful results of SPICE simulations for circuits implementing different Boolean functions.

The ability to implement threshold functions results from regions with positive and negative slope in device I-V characteristic (Fig. 1(C)). The most important parameter to describe such characteristic is the peak current I. Circuits made of NDR elements have a voltage divider structure composed of two (or more) serially connected NDR elements. Such circuits can have two stable operating points (Fig. 1(C)) and consequently can operate in one of two output states (LOW and HIGH).

Authors acknowledge support from Polish National Science Centre, grant no. N N516 451538.

M. Nikodem, M. A. Bawiec, J. Biernat are with Wrocław University of Technology, Institute of Computer Engineering, Control and Robotics, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland (e-mails: {maciej.nikodem, marek.bawiec, janusz.biernat}@pwr.wroc.pl).

The state of the device depends on the relation between peak currents of the load and driver elements – I_l and I_d. This relation is invariable (either $I_\mathrm{l} > I_\mathrm{d}$ or $I_\mathrm{l} < I_\mathrm{d}$), however, if NDR-transistor pair is connected in parallel to either load or driver element, then the state of the circuit can be controlled by external signal x_i (see Fig. 1(B)). For the circuit presented in Fig. 1(B) when input signal x_i is high then the sum of the peak currents of the driver and the additional NDR element I_i is greater than peak current of the load. This enforces the circuit to operate in LOW state. When x_i is low then the peak current I_i does not sum up with I_d that is now smaller than I_l forcing the circuit to operate in HIGH state. The additional NDR-transistor pair enables to control the relation between load and driver currents and consequently to implement an inverter function.

Modifications of the circuit's operating point are possible only when supply voltage is clocked in a four-phase scheme (Fig. 1(A)). In the first phase the supply voltage increases and the circuit evaluates its state based on the sum of peak currents of NDR elements located in upper (load) and lower (driver) branches of the circuit and the input signals; In the second phase the supply voltage is high and the state of the circuit is latched – changes of input signals do not influence the circuit state and output; Third phase resets the circuit while in fourth phase circuit awaits for changes of input signals. Device that operates in such a four-phase clocking scheme is called a monostable-bistable transition logic element (MOBILE) [8].

Circuits operating in MOBILE regime are capable of implementing threshold gates but can be also developed to implement more complex, non-threshold Boolean functions. This paper considers two such structures: Multi Threshold Threshold Gates (MTTGs) [9], [10] and Generalised Threshold Gates (GTGs) [2], [11] (Fig. 2). Both structures are feasible when RTDs and heterostructure field-effect transistors

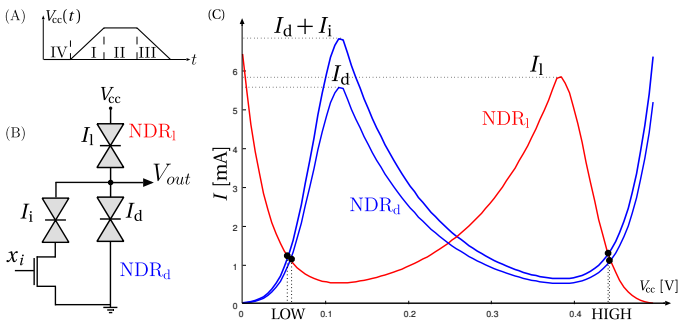

Fig. 1. Four-phase scheme of V_{cc} voltage (A), structure of an inverter gate (B), graphical solution for operating point of an inverter gate (C).

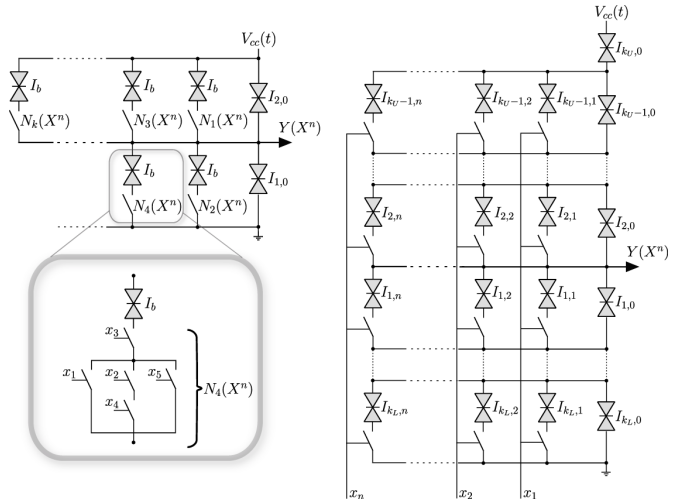

Fig. 2. General structure of GTG (left) and MTTG (right).

(HFETs) are used and the circuit operates in MOBILE regime [6], [12].

As presented in [9] the MTTG circuit composed of $n+1$ parallelly connected branches computes weighted sum of n binary inputs and quantises its result. Each branch of MTTG structure is constructed of an RTD-transistor pair where each RTD has different parameters (i.e. peak current) and is activated with one input signal. MTTG structure presented in [9] implements only some Boolean functions – precisely, any 2–input Boolean functions and single threshold functions of $n > 2$ variables (e.g. AND, OR).

GTG structure differs significantly since all but one RTD elements can have the same parameters (peak currents) and are activated by serial-parallel network of transistors implementing unate functions of up to n variables [2], [11], [13], [14]. Therefore GTG gate structure broadens capabilities of MTTG gates enabling to implement the same functionality by using smaller number of NDR elements and in less complex gate structure. This approach results in reduction of the number of branches that is now related to the complexity of the Boolean function implemented rather than the number of inputs. Our earlier paper [14] shows that using unate functions to activate circuit branches enables to implement any n-input Boolean function in a GTG circuit composed of at most $n+2$ branches.

Avedillo et al. [2], [9], [10] have proposed MTTG and GTG circuits structures and have shown that such circuits can implement some Boolean functions. However, those papers are focused on presenting some examples of NDR-based circuits, and did not deal with their general properties or synthesis methods. The first paper that has verified whether the NDR-based circuits can implement complex Boolean functions was presented by Berezowski [11]. He focused on GTGs assuming that all but one NDR devices have equal peak currents and proposed compact iterative model of such circuits (1).

Another paper by Avedillo et al. [15] explored both types of circuits, proposed logic models and presented that these gates can implement non-threshold functions. Research on MTTG was later continued by a number of authors (e.g. [16], [17]) that used different elementary MTTG gates to construct programmable logic elements capable of implementing Boolean functions of up to 4 variables. However, similarly to previous papers, they didn't deal with general properties, models and synthesis algorithms for MTTG and GTG structures.

II. Generalised Threshold Gate

A. Formal Model

Generalised threshold gate structure has several advantages over MTTG: (i) simpler structure of the circuit – it consists of less branches and NDR elements than the equivalent MTTG gates; (ii) in contrast to the MTTG identical NDR elements can be used that are activated by serial-parallel (SP) networks of transistors (iii) there is no need to use complementary transistor pair. Taking advantage of the second property Berezowski [11] proposed to implement Boolean functions in a GTG structure according to iterative formula

$$Y_l(\mathbb{X}^n) = \begin{cases} 0 & l = 0, \\ Y_{i-1}(\mathbb{X}^n) + N_i(\mathbb{X}^n) & \text{for odd } l, \\ Y_{i-1}(\mathbb{X}^n)\overline{N_i(\mathbb{X}^n)} & \text{for even } l. \end{cases} \quad (1)$$

An important contribution of the above formula is the ability to determine Boolean function implemented by means of the activation functions of every branch. Relying on formula (1) Berezowski has verified that no more than 4 unate functions $N_i(\mathbb{X}^n)$ are necessary to implement any logic function of at most 4 variables. This verification, however, has been done through exhaustive examination of all possible combinations of four input unate Boolean functions $N_i(\mathbb{X}^n)$, and so no general conclusions on features of GTG gates has been derived.

Apart from presenting the first formal model of the GTG circuit Berezowski [11] brought in a concept of GTG operation where upper and lower branches are activated by turns, thus changing the output state of the gate. As for further analysis, with no loss of generality we can assume that $Y(0^n) = 0$. Then the activation of the upper branch (odd indexed) switches gate output to logic 1 while the lower branch (even indexed) switches the output to zero. Activation of the subsequent upper branch yields 1 again and so on. Since $Y(0^n) = 0$ then the $N_2(\mathbb{X}^n)$ function from the first lower branch (that forces gate output to 0) needs no activation for input vectors x that are in the off-set of $N_1(\mathbb{X}^n)$ (denoted as $\mathtt{0}(N_i(\mathbb{X}^n))$). This results from the fact, that $N_1(\mathbb{X}^n)$ switches $Y(X)$ function's output to 1 only for such vectors x that belong to $N_1(\mathbb{X}^n)$'s on-set (denoted as $\mathtt{1}(N_1(\mathbb{X}^n))$) while for all $x \in \mathtt{0}(N_1(\mathbb{X}^n))$ gate output remains zeroed (see Fig. 3). Consequently, $N_2(\mathbb{X}^n)$ can be chosen so that $\mathtt{1}(N_2(\mathbb{X}^n))$ is a subset of $\mathtt{1}(N_1(\mathbb{X}^n))$, since including other minterms in $N_2(\mathbb{X}^n)$ function does not influence the resulting function $Y(\mathbb{X}^n)$. The same property holds for other pairs of $N_i(\mathbb{X}^n)$, $N_j(\mathbb{X}^n)$ functions, i.e.:

$$\mathtt{1}(N_j(\mathbb{X}^n)) \subseteq \mathtt{1}(N_i(\mathbb{X}^n)) \quad (2)$$

for any $i < j$.

Observe, that (2) allows for two functions $N_i(\mathbb{X}^n)$ and $N_j(\mathbb{X}^n)$ to be equal. We propose to restrict functions $N_i(\mathbb{X}^n)$

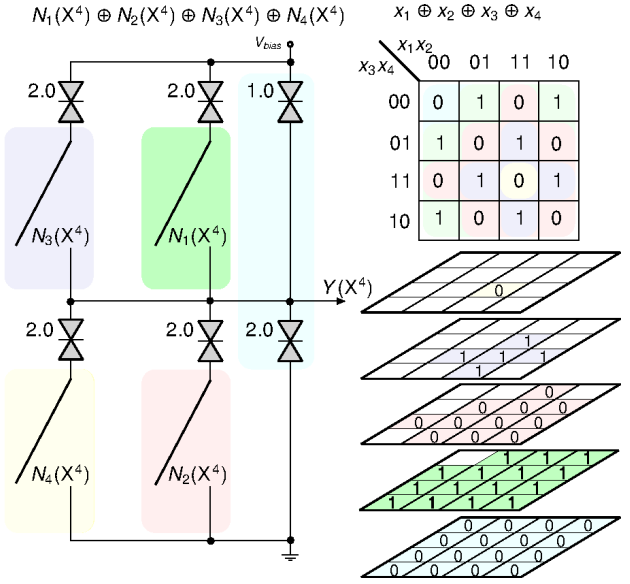

Fig. 3. Structure of GTG gate implementing XOR($\mathbb{X}^4$) function and Carnough-based interpretation of activation functions.

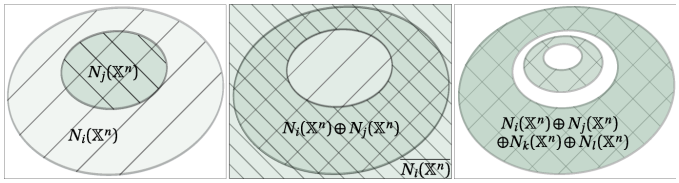

Fig. 4. Venn diagrams to present relations between activation functions $N_i(\mathbb{X}^n), N_j(\mathbb{X}^n), N_k(\mathbb{X}^n)$ and $N_l(\mathbb{X}^n)$ for any $i < j < k < l$.

further and assume that none of two activation functions are equal, i.e.:

$$1(N_j(\mathbb{X}^n)) \subset 1(N_i(\mathbb{X}^n)), \tag{3}$$

for any $i < j$.

The assumption mentioned above is significant for the model proposed in [11]. Consequently we obtain that for any two functions $N_i(\mathbb{X}^n)$, $N_j(\mathbb{X}^n)$ and $i < j$

$$N_i(\mathbb{X}^n)N_j(\mathbb{X}^n) = N_j(\mathbb{X}^n). \tag{4}$$

Further, assuming $i < j < k < l$ following equalities hold (($\mathbb{X}^n$) symbol is neglected to keep the equations compact):

$$\begin{aligned} N_i\overline{N_j} &= N_i\overline{N_j} + \overline{N_i}N_j = N_i \oplus N_j && (5)\\ N_i \oplus N_j + N_k &= (N_i \oplus N_j)\,\overline{N_k} + N_k\overline{N_i \oplus N_j} \\ &= N_i \oplus N_j \oplus N_k && (6)\\ N_i\overline{N_j} + N_k\overline{N_l} &= (N_i \oplus N_j) + (N_k \oplus N_l) \\ &= N_i \oplus N_j \oplus N_k \oplus N_l. && (7) \end{aligned}$$

We pass over formal derivations of equations (5)-(7) providing graphical interpretation instead (Fig. 4).

Following the relations given by (5)-(7) the model (1) for the m-branches GTG can be mapped to

$$Y(\mathbb{X}^n) = \bigcup_{i=1}^{\lfloor \frac{m}{2} \rfloor} N_{2i-1}(\mathbb{X}^n)\overline{N_{2i}(\mathbb{X}^n)} + \delta N_m(\mathbb{X}^n), \tag{8}$$

where $\bigcup$ denotes logic sum and δ equals 0 if m is even and 1 otherwise.

Using the above observation we can state the following:

Theorem 1: GTG gate composed of m branches with activation functions $N_1(\mathbb{X}^n), N_2(\mathbb{X}^n), \ldots, N_m(\mathbb{X}^n)$ such that $1(N_j(\mathbb{X}^n)) \subset 1(N_i(\mathbb{X}^n))$ for any $i < j$, implements Boolean function

$$Y(\mathbb{X}^n) = \bigoplus_{i=1}^{m} N_i(\mathbb{X}^n). \tag{9}$$

Proof: The proof is straightforward, as (9) follows from observations (5)-(7) and GTG model (8):

$$\begin{aligned} Y(\mathbb{X}^n) &= \bigcup_{i=1}^{\lfloor \frac{m}{2} \rfloor} (N_{2i-1}(\mathbb{X}^n) \oplus N_{2i}(\mathbb{X}^n)) + \delta N_m(\mathbb{X}^n) \\ &= \bigoplus_{i=1}^{\frac{m-\delta}{2}} (N_{2i-1}(\mathbb{X}^n) \oplus N_{2i}(\mathbb{X}^n)) \oplus \delta N_m(\mathbb{X}^n) \\ &= \bigoplus_{i=1}^{m} N_i(\mathbb{X}^n). && (10) \end{aligned}$$

■

B. Gate Synthesis

Synthesis of the GTG circuit is based on the observation that every Boolean function can be represented as an EXOR sum of unate functions $N_i(\mathbb{X}^n)$.

Theorem 2: Any Boolean function $Y(\mathbb{X}^n)$ can be represented as an EXOR sum of unate functions satisfying the assumption of (3).

Proof: Since 1 is a unate function thus it is enough to prove the theorem for functions $Y(\mathbb{X}^n)$ such that $Y(0^n) = 0$. For any such function there exists the smallest unate function $N_1(\mathbb{X}^n)$ such that $1(Y(\mathbb{X}^n)) \subseteq 1(N_1(\mathbb{X}^n))$ (at least $N_1(\mathbb{X}^n) = 1$). "The smallest" means that there is no other $N_1'(\mathbb{X}^n)$ such that $1(N_1'(\mathbb{X}^n)) \subset 1(N_1\mathbb{X}^n)$ and $1(Y(\mathbb{X}^n)) \subseteq 1(N_1'(\mathbb{X}^n))$. Consequently, $Y(\mathbb{X}^n)$ can be represented as:

$$Y(\mathbb{X}^n) = N_1(\mathbb{X}^n) \oplus Y_1(\mathbb{X}^n). \tag{11}$$

If $Y_1(\mathbb{X}^n)$ is unate then the theorem is proved. Otherwise we can find the smallest unate function $N_2(\mathbb{X}^n)$ such that $1(Y_1(\mathbb{X}^n)) \subseteq 1(N_2(\mathbb{X}^n))$ – such function always exists and differs from $N_1(\mathbb{X}^n)$ as $Y_1(\mathbb{X}^n)$ and $N_1(\mathbb{X}^n)$ have no common on-set minterms. Due to the same reason $1(N_j(\mathbb{X}^n)) \subset 1(N_i(\mathbb{X}^n))$ for every $i < j$ which means that unate functions get smaller (in terms of the number of on-set minterms) with each step. The procedure is thus finite and returns $Y_m(\mathbb{X}^n)$ which is a unate function, and so $N_m(\mathbb{X}^n) = Y_m(\mathbb{X}^n)$. Consequently, any Boolean function $Y(\mathbb{X}^n)$ can be represented as an EXOR sum of unate functions:

$$Y(\mathbb{X}^n) = \bigoplus_{i=1}^{m} N_i(\mathbb{X}^n), \tag{12}$$

that satisfy (3). ■

Two important observations follow from the Theorem 2: First, GTG circuit composed of m branches with activation functions $N_i(\mathbb{X}^n)$ that satisfy (3) can implement arbitrary Boolean function. Second, for a given Boolean function $Y(\mathbb{X}^n)$ we can synthesise GTG circuit by the iterated computation of the smallest unate functions (Alg. 1).

Algorithm 1 Synthesis of the GTG circuit

Require: n-variable Boolean function $Y(\mathbb{X}^n)$
Ensure: unate functions $N_i(\mathbb{X}^n)$
1: **if** $Y(0^n)=1$ **then** $Y(\mathbb{X}^n)=1\oplus Y(\mathbb{X}^n)$
2: set $i=1$,
3: find the smallest unate function $N_i(\mathbb{X}^n)$ covering $Y(\mathbb{X}^n)$,
4: **if** $Y(\mathbb{X}^n)=N_i(\mathbb{X}^n)$ **then** exit algorithm
5: calculate $Y_i(\mathbb{X}^n)$ such that

$$Y(\mathbb{X}^n)=N_i(\mathbb{X}^n)\oplus Y_i(\mathbb{X}^n),$$

6: **while** $Y_i(\mathbb{X}^n)\neq 0$ **do**
7: find the smallest unate function $N_{i+1}(\mathbb{X}^n)$ covering $Y_i(\mathbb{X}^n)$,
8: calculate $Y_{i+1}(\mathbb{X}^n)$ such thah

$$Y_i(\mathbb{X}^n)=N_{i+1}(\mathbb{X}^n)\oplus Y_{i+1}(\mathbb{X}^n),$$

9: set $i=i+1$,
10: **end while**

III. Multi-Threshold Threshold Gate

Similarly to the GTGs, the Multi Threshold Threshold Gates (MTTGs) can also implement complex Boolean functions drawing from negative differential resistance property. However, in contrast to GTG, the Boolean function implemented in MTTG structure, depends on the parameters of NDR elements, each of which is activated with a single transistor (not a SP network as for GTG).

A. Formal Model

The output of the MTTG circuit depends on the relation between peak currents in the upper and the lower branches (Fig. 2) of the circuit. Both currents can be determined according to the laws of current flow. Consequently, MTTGs can implement Boolean functions by proper adjustment of peak currents of NDR elements ($I_{i,j}$) and controlling those elements by input signals x_i. Let $I_u(\mathbb{X}^n)$ and $I_l(\mathbb{X}^n)$ denote currents in the upper and the lower branches of the MTTG circuit respectively. Boolean function implemented in such MTTG structure is defined as

$$Y(\mathbb{X}^n)=\begin{cases}1 & \text{iff } I_u(\mathbb{X}^n)>I_l(\mathbb{X}^n)\\ 0 & \text{iff } I_u(\mathbb{X}^n)\leq I_l(\mathbb{X}^n)\end{cases}. \tag{13}$$

Folowing the serial connection of multiple levels of NDR-transistor pairs, the currents $I_u(\mathbb{X}^n)$ and $I_l(\mathbb{X}^n)$ can be determined as

$$\begin{aligned}I_u(\mathbb{X}^n)&=\min(I_1(\mathbb{X}^n),I_3(\mathbb{X}^n),\ldots,I_{l_u}(\mathbb{X}^n)),\\ I_l(\mathbb{X}^n)&=\min(I_2(\mathbb{X}^n),I_4(\mathbb{X}^n),\ldots,I_{l_l}(\mathbb{X}^n)),\end{aligned} \tag{14}$$

where odd/even indices refer to upper/lower branches of MTTG structure. Further, actual value of the current $I_j(\mathbb{X}^n)$ results from NDR-transistor pairs located in j-th level of the gate – peak currents of NDR elements and actual input signals x_i that activate them. Therefore,

$$I_j(\mathbb{X}^n)=I_{j,0}+\sum\nolimits_{i=1}^{n}I_{j,i}x_i. \tag{15}$$

Consequently, the Boolean function implemented in the MTTG gate equals

$$Y(\mathbb{X}^n)=\begin{cases}1 \text{ iff } \min(I_1(\mathbb{X}^n),I_3(\mathbb{X}^n),\ldots,I_{l_u}(\mathbb{X}^n))\\ \quad -\min(I_2(\mathbb{X}^n),I_4(\mathbb{X}^n),\ldots,I_{l_l}(\mathbb{X}^n))>0\\ 0 \text{ otherwise},\end{cases} \tag{16}$$

where $I_j(\mathbb{X}^n)$ are given by (15).

B. Gate Synthesis

Synthesis of an MTTG is more complex than that of a GTG as in its structure the value of the peak current for each NDR element needs to be found. For further analysis and with no loss of generality from now on we will deal with Boolean functions $Y(\mathbb{X}^n)$ such that $Y(0^n)=0$.

The proposed synthesis procedure follows from three facts: (i) every Boolean function $Y(\mathbb{X}^n)$ can be represented as a $\min/\max$ composition of threshold functions $F_i(\mathbb{X}^n)$; (ii) composition of $\min/\max$ functions can be transformed to a difference of two $\min$ functions each having a number of arguments; (iii) any threshold function $F_i(\mathbb{X}^n)$ can be represented by using corresponding hyperplane $H_i(\mathbb{X}^n)=\sum_{j=1}^{n}w_jx_j+T_i$ (w_j is a real-valued coefficient and T_i is a threshold) which separates function's on-set and off-set:

$$F_i(\mathbb{X}^n)=[H_i(\mathbb{X}^n)>0]=\begin{cases}0 & \text{iff } H_i(\mathbb{X}^n)\leq 0\\ 1 & \text{otherwise}\end{cases}. \tag{17}$$

The following theorem formally states the first fact.

Theorem 3: Any Boolean function $Y(\mathbb{X}^n)$ can be represented as a $\min/\max$ compound of threshold functions $F_i(\mathbb{X}^n)$ (F_i for short):

$$Y(\mathbb{X}^n)=\min(F_m,\max(F_{m-1},\min(\ldots(\min(F_2,F_1))\ldots))) \tag{18}$$

such that any two functions $F_i(\mathbb{X}^n),F_j(\mathbb{X}^n)$ satisfy:

$$\begin{aligned}&\min(F_i(\mathbb{X}^n),F_j(\mathbb{X}^n))=F_j(\mathbb{X}^n) \text{ if } i<j \text{ are odd},\\ &\min(F_i(\mathbb{X}^n),F_j(\mathbb{X}^n))=F_i(\mathbb{X}^n) \text{ if } i<j \text{ are even}.\end{aligned} \tag{19}$$

Proof: Assume $Y(\mathbb{X}^n)$ is not a threshold function such that $Y(0^n)=0$ and $\mathtt{1}(Y(\mathbb{X}^n)),\mathtt{0}(Y(\mathbb{X}^n))$ denote its on-set and off-set respectively. We can define a sequence of threshold functions $i=1,2,\ldots,m$

$$F_i(\mathbb{X}^n)=\begin{cases}t & \text{for } x\in\mathbf{X}_i=\mathbf{X}_{i-1}\cup\mathbf{S}_i,\\ \overline{t} & \text{otherwise},\end{cases} \tag{20}$$

where $\mathbf{X}_0=\emptyset$, $F_0(\mathbb{X}^n)=Y(\mathbb{X}^n)$, t equals 0 if i is odd and 1 otherwise, $\mathbf{S}_i$ is a nonempty subset of $\mathtt{0}(F_{i-1}(\mathbb{X}^n))\cap\mathtt{0}(Y(\mathbb{X}^n))$ and $\mathtt{1}(F_{i-1}(\mathbb{X}^n))\cap\mathtt{1}(Y(\mathbb{X}^n))$ for odd and even i respectively.

Note that the above definition guaranties that $F_i(\mathbb{X}^n)$s are threshold functions – $F_1(\mathbb{X}^n)$ is a threshold function since in boundary case $\mathbf{X}_1$ contains single minterm only. Further, $F_2(\mathbb{X}^n)$ is also a threshold function since it can be constructed from $F_1(\mathbb{X}^n)$ by moving one (possibly more) minterm from $F_1(\mathbb{X}^n)$'s off-set to its on-set and negating the output. A similar procedure can be applied to generate subsequent functions.

Functions (20) also satisfy condition (19) and their $\min/\max$ composition equals

$$\begin{aligned}&\min(F_m,(\max(F_{m-1},\min(\ldots,(\min(F_2,F_1))\ldots))))=\\ &\quad=\begin{cases}0 & \text{iff } x\in\mathbf{S}_1\cup\mathbf{S}_3\cup\mathbf{S}_5\ldots\\ 1 & \text{otherwise}\end{cases}\end{aligned} \tag{21}$$

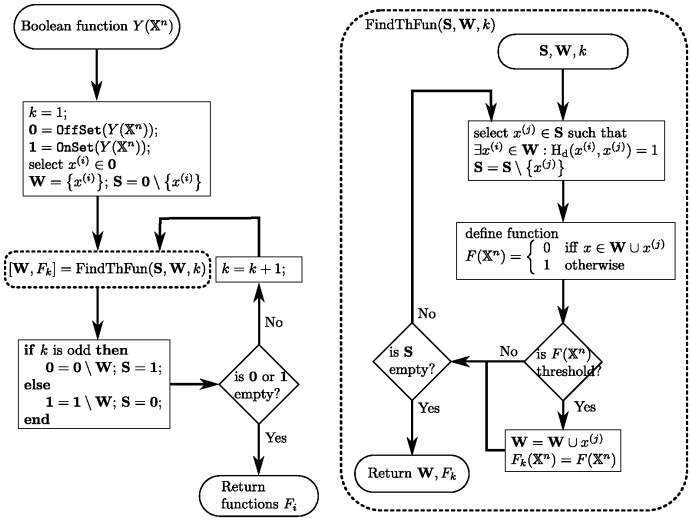

Fig. 5. Flow diagram providing an overview of the proposed threshold decomposition algorithm.

Since all sets $\mathbf{S}_i$ with odd indices are subsets of $\mathtt{0}(F_{i-1}(\mathbb{X}^n))\cap \mathtt{0}(Y(\mathbb{X}^n))$ thus no two of them have common minterms and the sum of all subsets equals $\mathtt{0}(Y(\mathbb{X}^n))$. Therefore,

$$\begin{aligned}&\min(F_m,(\max(F_{m-1},\min(\ldots,(\min(F_2,F_1))\ldots))))=\\&=\begin{cases}0 & \text{iff } x\in\mathtt{0}(Y(\mathbb{X}^n))\\ 1 & otherwise\end{cases}=Y(\mathbb{X}^n).\end{aligned}\tag{22}$$

■

Relying on the third fact we can represent $Y(\mathbb{X}^n)$ as a $\min/\max$ composition of hyperplanes $H_i(\mathbb{X}^n)$ representing threshold functions $F_i(\mathbb{X}^n)$. Precisely, (18) can be transformed to

$$\begin{aligned}&Y(\mathbb{X}^n)=\min(F_m,\max(F_{m-1},\min(\ldots(\min(F_2,F_1))\ldots)))\\&=[\min(H_m,\max(H_{m-1},\min(\ldots(\min(H_2,H_1))\ldots)))>0]\,.\end{aligned}\tag{23}$$

Composition of hyperplanes $H_i(\mathbb{X}^n)$ (23) can be transformed to a difference of two $\min$ functions. This is done through repeated application of two transformations:

$$\min(H_1,H_2)=\min(H_1+C,H_2+C)-\min(C)\tag{24}$$

$$\max(H_1,H_2)=\min(H_1+H_2)-\min(H_1,H_2).\tag{25}$$

where C is some nonzero function of $\mathbb{X}^n$.

Algorithm 2 Synthesis of the MTTG

Require: n-variable Boolean function $Y(\mathbb{X}^n)$
Ensure: peak currents for all the NDR elements in MTTG structure
1: Find threshold decomposition of function $Y(\mathbb{X}^n)$ suitable for $\min/\max$ representation.
2: Represent threshold functions $F_i(\mathbb{X}^n)$ with corresponding hyperplanes $H_i(\mathbb{X}^n)$.
3: Represent $Y(\mathbb{X}^n)$ as a $\min/\max$ composition of hyperplanes $H_i(\mathbb{X}^n)$.
4: Transform $\min/\max$ representation to difference of two $\min$.

Threshold decomposition is a crucial part of the MTTG synthesis algorithm (Alg. 2) as the resulting threshold functions must satisfy (19). To achieve this we propose an iterative decomposition procedure (Fig. 5) which searches for linearly separable sets of minterms (function *FindThFun()*). Checking for linear separability follows from the fact that any Boolean function of up to 8 variables that is 2-assumable is also a threshold function [18] (for functions of more than 8 variables linear programming problem can be solved instead). Efficiency is the benefit of checking 2-assumability as only some minterms may violate this property thus reducing the number of checks required [18]. To further simplify the algorithm, decomposition procedure generates set of linearly separable minterms $\mathbf{W}$ by choosing one minterm $x^{(j)}$ at a time from the set of available minterms $\mathbf{S}$ and verifying if function

$$F_i(\mathbb{X}^n)=\begin{cases}0 & \text{iff } x\in\mathbf{W}\cup x^{(j)}\\ 1 & \text{otherwise}\end{cases}\tag{26}$$

is a threshold function. Every execution of *FindThFun()* function takes previously found set of minterms $\mathbf{W}$ as input thus ensuring that successive functions $F_i(\mathbb{X}^n)$ satisfy (19).

TABLE I
CAPABILITIES OF MTTG AND GTG GATES IN TERMS OF THE NUMBER OF INPUTS AND THRESHOLDS OF BOOLEAN FUNCTIONS THEY CAN IMPLEMENT

	MTTG			GTG	
	No. of inputs	No. of thresholds		No. of inputs	No. of thresholds
[2]	≤ 3	≤ 2	[11]	≤ 4	≤ 4
[10]	≤ 3	≤ 3			
[16]	≤ 3	1			
[17]	≤ 4	≤ 4			
Our	≤ 8	≤ 8	Our	any	any

IV. CONCLUSION

We presented two synthesis algorithms that can be used to synthesise arbitrary Boolean functions. Both algorithms were evaluated and the complexity of the resulting gate structures and synthesis algorithms themselves was analysed. Complexity of both synthesis algorithms grows exponentially with the number of input variables n. For MTTGs and Boolean functions of more than 8 variables complexity increases even more as additional linear programming problems need to be solved. Nevertheless large computational complexity, these are the first synthesis algorithms proposed in literature that allow to synthesise arbitrary Boolean functions of up to 8 variables efficiently. Table I compares capabilities of MTTG and GTG gates presented in literature with our results. Table II gives complexity estimations of the resulting MTTG and GTG gates, which implement Boolean functions of up to 4 variables.

TABLE II
COMPLEXITY OF MTTG AND GTG GATES FOR BOOLEAN FUNCTIONS OF UP TO 4 VARIABLES

	No. of input variables	No. of NDR elements	No. of transistors
GTG	1	3	1
	2	$3\leq\ldots\leq 4$	$2\leq\ldots\leq 3$
	3	$3\leq\ldots\leq 5$	$3\leq\ldots\leq 7$
	4	$3\leq\ldots\leq 6$	$4\leq\ldots\leq 32$
MTTG	1	3	1
	2	$4\leq\ldots\leq 9$	$2\leq\ldots\leq 6$
	3	$5\leq\ldots\leq 16$	$3\leq\ldots\leq 12$
	4	$6\leq\ldots\leq 25$	$3\leq\ldots\leq 20$

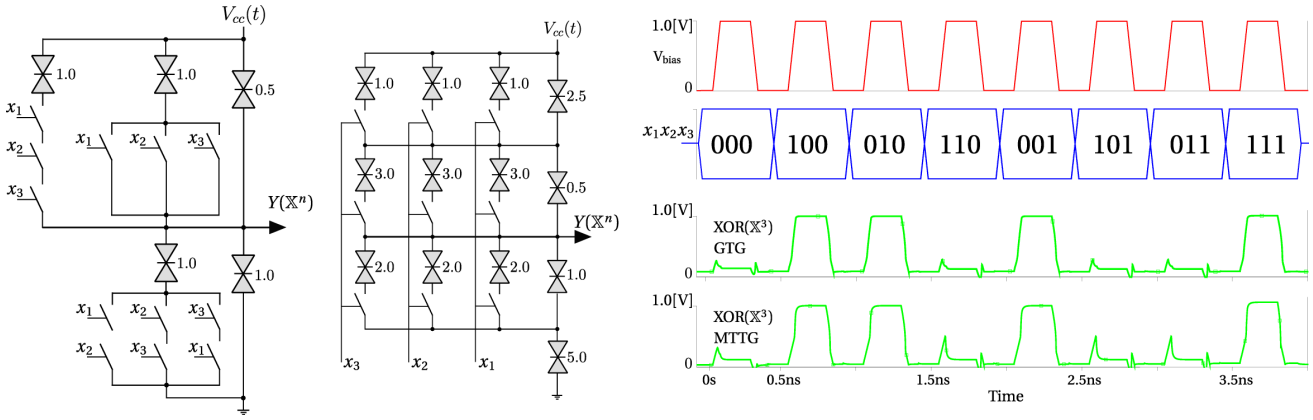

Fig. 6. GTG and MTTG (left) gate structures that implement $Y(\mathbb{X}^n) = x_1 \oplus x_2 \oplus x_3$ function and transient simulation from SPICE software (right).

Complexity is estimated as the number of transistors and NDR elements necessary to implement the gate.

In general an MTTG requires $(k+1)n$ transistors and $(k+1)(n+1)$ NDR elements to implement n-variable Boolean function that has k threshold functions in its threshold decomposition. There is a significant difference between both circuits when the number of NDR elements is compared – for GTGs the number of elements is proportional to the minimal number of thresholds in threshold decomposition of a given Boolean function. In case of MTTGs two factors influence the number of NDR elements: the number of thresholds and the number of input variables.

The SPICE software has been used to verify our synthesis algorithms. Figure 6 presents GTG and MTTG gates that implement $Y(\mathbb{X}^n) = x_1 \oplus x_2 \oplus x_3$ function and the result of their transient simulation.

The algorithm for synthesising MTTG and GTG gates for arbitrary Boolean functions proposed in this paper give the possibility to further explore properties of NDR-based gates and construct more complex and powerful electronic circuits. Possible future applications include synthesis of complex multiple input multiple output functions and powerful programmable logic elements.

REFERENCES

[1] "The international technology roadmap for semiconductors," International roadmap committee, Tech. Rep., 2009.

[2] M. Avedillo, J. Quintana, and H. Pettenghi, "Logic Models Supporting the Design of MOBILE-based RTD Circuits," *2005 IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05)*, pp. 254–259, 2005.

[3] J. N. Schulman, H. J. De Los Santos, and D. H. Chow, "Physics-based RTD current-voltage equation," *IEEE Electron Device Letters*, vol. 17, no. 5, pp. 220–222, May 1996.

[4] J. P. Sun, G. Haddad, P. Mazumder, and J. Schulman, "Resonant tunneling diodes: models and properties," *Proceedings of the IEEE*, vol. 86, no. 4, pp. 641–660, apr 1998.

[5] Z. Yan and M. J. Deen, "New RTD large-signal DC model suitable for PSPICE," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 2, pp. 167–172, February 1995.

[6] M. Bhattacharya, S. Kulkarni, A. Gonzalez, and P. Mazumder, "A prototyping technique for large-scale RTD-CMOS circuits," in *Proceedings of the 2000 IEEE International Symposium on Circuits and Systems*, vol. 1, May 2000, pp. 635–638.

[7] J. Le, L. Pileggi, and A. Devgan, "Circuit simulation of nanotechnology devices with non-monotonic I-V characteristics," in *International Conference on Computer Aided Design, 2003. ICCAD-2003.*, November 2003, pp. 491–496.

[8] K. Maezawa, T. Akeyoshi, and T. Mizutani, "Functions and applications of monostable-bistable transition logic elements (mobile's) having multiple-input terminals," *Electron Devices, IEEE Transactions on*, vol. 41, no. 2, pp. 148 –154, feb 1994.

[9] M. J. Avedillo, J. M. Quintana, H. Pettenghi, P. Kelly, and C. Thompson, "Multi-threshold threshold logic circuit design using resonant tunnelling devices," *Electronics Letters*, vol. 39, no. 21, pp. 1502–1504, Oct. 2003.

[10] J. M. Quintana, M. J. Avedillo, and H. Pettenghi, "Rtd-based compact programmable gates," in *IEEE International Joint Conference on Neural Networks*, July 2004, pp. 2637–2640.

[11] K. Berezowski, "Compact binary logic circuits design using negative differential resistance devices," *Electronic Letters*, vol. 42, no. 16, pp. 902–903, 2006.

[12] J. I. Bergman, J. Chang, Y. Joo, B. Matinpour, J. Laskar, N. M. Jokerst, M. A. Brooke, B. Brar, and E. B. III, "RTD/CMOS nanoelectronic circuits: thin-film InP-based resonant tunneling diodes integrated with CMOS circuits," *IEEE Electron Device Letters*, vol. 20, no. 3, pp. 119–122, March 1999.

[13] M. Bawiec and M. Nikodem, "Boolean logic function synthesis for generalised threshold gate circuits," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09. New York, NY, USA: ACM, 2009, pp. 83–86.

[14] M. A. Bawiec and M. Nikodem, "Generalised threshold gate synthesis based on and/or/not representation of boolean function," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, ser. ASPDAC '10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 861–866.

[15] M. J. Avedillo, J. M. Quintana, and H. Pettenghi, "Self-latching operation of mobile circuits using series-connection of rtds and transistors," *IEEE Transactions on Circuits and Systems*, vol. 53, no. 5, pp. 334–338, May 2006.

[16] Y. Wei and J. Shen, "Novel universal threshold logic gate based on RTD and its application," *Microelectronics Journal*, vol. 42, no. 6, pp. 851–854, 2011.

[17] Y. Zheng and C. Huang, "Complete Logic Functionality of Reconfigurable RTD Circuit Elements," *IEEE Transactions on Nanotechnology*, vol. 8, no. 5, pp. 631–642, Sep 2009.

[18] S. Muroga, *Threshold logic and its application*. John Wiley & Sons, 1971.