Single Electron Encoded Latches and Flip-Flops

Casper Lageweg, *Student Member, IEEE*, Sorin Coţofană, *Senior Member, IEEE*, and Stamatis Vassiliadis, *Fellow, IEEE*

***Abstract*—Single electron tunneling (SET) technology offers the ability to control the transport of individual electrons. In this paper, we investigate single electron encoded logic (SEEL) memory circuits, in which the Boolean logic values are encoded as zero or one electron charges. More specifically, we focus on the implementation of SEEL latches and flip-flops. All proposed circuits are verified by means of simulation using the SIMulation Of Nanostructures package. We first present a generic SEEL linear threshold gate implementation, from which we derive a family of Boolean logic gates. Second, we propose Boolean gate-based implementations of the RS latch, the D latch, and D flip-flop. Third, we propose threshold gate-based implementations of the same memory elements. Finally, we discuss the estimated area, delay, and power consumption of the Boolean gate-based and threshold gate-based implementations, and compare them with other SET-based memory elements.**

***Index Terms*—Coulomb blockage, digital circuits, logic circuits, single electron tunneling (SET), threshold logic circuits.**

I. Introduction

IT IS WIDELY known that the ever-decreasing feature size and the corresponding increase in the number of transistors per millimeters squared facilitated vast improvements in semiconductor-based designs. It is also understood that such improvement will eventually come to an end. There have been reports [1] suggesting that the MOS transistor itself cannot be shrunk beyond certain limits dictated by its operating principle. In order to ensure further feature size reduction, possible successor technologies with greater scaling potential such as single electron tunneling (SET) [2], [3] are currently under investigation [4]. SET circuits are centered around tunnel junctions, through which individual electrons can be transported in a controlled manner. Our current research focuses on the implementation of (digital) logic gates and memory elements in SET technology, using the SET tunnel junction's ability to control the transport of individual electrons. When applying the SET technology for the design of logic circuits, two main approaches have been suggested thus far.

The first approach is to assume that the tunnel junction operates as a switch, and use it to implement the SET equivalent of the MOS transistor [5], [6]. In this approach, charge is transported through an "open" SET transistor until the transistor's source and drain voltage are approximately equal. Although this has the advantage that existing MOS transistor-based designs can easily be ported to SET technology, it does not fully utilize the potential of the SET technology. The main disadvantage of this design style is that the current transport though an "open" transistor still consists of a large number of individual electrons "dripping" through the tunnel junctions. Given that electron tunneling is a sequential process, this is obviously a far slower process then the transport of just one single electron through the same junction.

The second approach, generally referred to as single electron logic, is to encode Boolean values directly as single electron charges. One approach in this direction, as first suggested in [7], is based on the physical transport of the charge from one gate to another, such that Boolean input signals consist of the presence of absence of the arriving charge. Another approach, as first suggested in [8], is based on scaling down the charge transport in SET transistor-based structures to a few electrons, and confining charge transport within individual gates. When charge transport is scaled down to just one electron, this approach leads to single electron encoded logic (SEEL) logic gates and memory elements, in which the Boolean logic values 0 and 1 are encoded as a net charge of zero and one electron charge only [8]. Typically, this charge is stored on the circuit's output node and the resulting voltage serves as input to the next gate, although a wireless SEEL logic family has also been suggested [9]. Our current research focuses on the implementation of logic gates and memory elements in SET technology that operate according to the SEEL paradigm.

In this paper, we investigate the implementation of SET-based SEEL memory elements. We investigate circuits consisting of SET tunnel junctions, capacitors, and voltage sources only, and we are primarily interested in the switching behavior that can be accomplished with such circuits. It is well known that the switching behavior of SET circuits is, amongst others, influenced by fabrication-technology dependent factors such as random background charge (charge noise). However, there is indication (see, e.g., [3]) that these effects may reduce or disappear entirely in the future. Therefore, and in order to keep the presentation as much as possible SET fabrication-technology independent, we ignore their effects on the switching behavior of circuits that we propose in this investigation. All our proposals are verified by simulation using the SIMulation Of Nanostructures (SIMON) simulation package [10].

The main contributions of the paper can be summarized as follows:

- presentation of a generic linear threshold gate (LTG) implementation and a family of Boolean logic gates derived from this gate;
- SET Boolean gate-based implementations of the reset–set (R–S) latch, D latch, and edge-triggered D flip-flop;

Manuscript received August 7, 2003; revised December 1, 2003. This work was supported in part by the Delft Interfacultary Research Program NanoComp.

The authors are with the Electrical Engineering Department, Delft University of Technology, Delft, The Netherlands (e-mail: C.R.Lageweg@ewi.tudelft.nl; S.D.Cotofana@ewi.tudelft.nl; S.Vassiliadis@ewi.tudelft.nl).

Digital Object Identifier 10.1109/TNANO.2004.828526

- proposals for threshold gate-based implementations of the same memory elements;
- estimates for area, delay, and power consumption of the Boolean logic gates, the Boolean gate-based memory elements, and the threshold gate-based memory elements, as well as a comparison with earlier proposals.

The remainder of this paper is organized as follows. Section II briefly presents the SET background theory, explaining the charge transport behavior appearing in SET circuits, as well as introducing a model for calculating the delay and power. In Section III, we present the generic LTG and the family of SEEL Boolean logic gates. Section IV investigates Boolean gate-based implementations of the R–S latch, D latch, and edge-triggered D flip-flop. In Section V, we propose threshold gate-based implementations of the same memory elements. Section VI discusses the estimated area, delay, and power consumption of these memory elements, and compares them with earlier proposals. This paper then presents some final remarks in Section VII.

II. Background of Single Electron Tunneling

A tunnel junction can be thought of as a leaky capacitor. The transport of charge through a tunnel junction is referred to as *tunneling*, where the transport of a single electron through a tunnel junction is referred to as a *tunnel event*. Electrons are considered to tunnel through a tunnel junction strictly one after another. We assume that all conditions are met such that charge quantization is observable ($E_C \gg E_Q$) and that tunnel events due to thermal energy can be ignored ($E_C \gg K_bT$). Under these conditions, the critical voltage V_c across a tunnel junction is the voltage threshold that is needed across the tunnel junction in order to make a tunnel event through this tunnel junction possible.

For calculating the critical voltage of a junction, we assume a tunnel junction with a capacitance of C_j. The remainder of the circuit, as viewed from the tunnel junction's perspective, has an equivalent capacitance of C_e. Given the approach presented in [11], we calculate the critical voltage V_c for the junction as

$$V_c = \frac{e}{2(C_e + C_j)}. \tag{1}$$

In the above equation, as well as in the remainder of this discussion, we refer to the charge of the electron as $q_e = 1.602 * 10^{-19}\ C$. Strictly speaking, this is incorrect, as the charge of the electron is, of course, negative. However, it is more intuitive to consider e as a positive constant for the formulas that determine whether or not a tunnel event will occur. We will, of course, correct for this when we discuss the direction in which the tunnel event takes place.

Generally speaking, if we define the voltage across a junction as V_j, and assuming the conditions stated above, a tunnel event will occur through this tunnel junction if and only if

$$|V_j| > V_c. \tag{2}$$

If tunnel events cannot occur in any of the circuit's tunnel junctions, i.e., $|V_j| < V_c$ for all junctions in the circuit, the circuit is in a *stable state*. For our research, we only consider circuits where a limited number of tunnel events may occur, resulting in a stable state. Each stable state determines a new output value resulting from the distribution of charge throughout the circuit.

The transport of an electron through a tunnel junction is a stochastic process. This means that we cannot analyze delay in the traditional sense. Instead, assuming a nonzero probability for charge transport ($|V_j| > V_c$), the switching delay t_d of a single electron transport can be calculated based on an error probability P_{error} that the desired transport did *not* occur as

$$t_d = \frac{-ln(P_{\text{error}})q_eR_t}{|V_j| - V_c} \tag{3}$$

where $R_t = 10^5\ \Omega$ is the tunnel resistance (though depending on the physical implementation this value is typically assumed). The error probability P_{error} will determine the reliability of the circuit. Given that the switching behavior is stochastic in nature, the error probability cannot be reduced to zero. It is, therefore, assumed that an error correction mechanism will be present in the form of hardware or data redundancy in order to achieve the desired accuracy.

When charge transport occurs through a tunnel junction, the difference in the total amount of energy present in the circuit before and after the tunnel event can be calculated by

$$\Delta E = E_{\text{final}} - E_{\text{initial}} = -q_e\left(|V_j| - V_c\right). \tag{4}$$

Therefore, the energy consumed by a single tunnel event occurring in a single tunnel junction can be calculated by taking the absolute value of ΔE. In order to calculate the power consumption of a gate, the energy consumption of each tunnel event is multiplied by the frequency of switching. The switching frequency, in turn, depends on the frequency at which the gate's inputs change and is input data dependent, as a new combination of inputs may or may not results in charge transport.

In addition to the switching error probability, as described in (3), there are two fundamental phenomena that may cause errors: thermally induced tunneling and cotunneling. Given a maximum acceptable switching error probability, we must ensure that both the thermal error probability, as well as the cotunneling error probability are of the same order of magnitude or less. For any temperature $T > 0$, there exists a nonzero probability that a tunnel event will occur through a junction (even if $|V_j| < V_c$). The error probability P_{therm} due to thermal tunneling can be described by a simple formula as

$$P_{\text{therm}} = e^{\frac{-\Delta E}{K_bT}}. \tag{5}$$

For a multijunction system in which a combination of tunnel events lead to a reduction of the energy present in the entire system, there exists a nonzero probability that those tunnel events occur simultaneously (even if $|V_j| < V_c$ for all individual tunnel junction involved). This phenomenon is commonly referred to as cotunneling [12], [13]. Although a detailed analysis of cotunneling is outside the scope of this paper, we remark that several means are available to reduce the cotunneling error probability. First, the ratio of cotunneling rate to desired tunneling rate can be reduced linearly by increasing the tunnel resistance R_t of the tunnel junctions involved in cotunneling. The main problem of this approach is that it also linearly

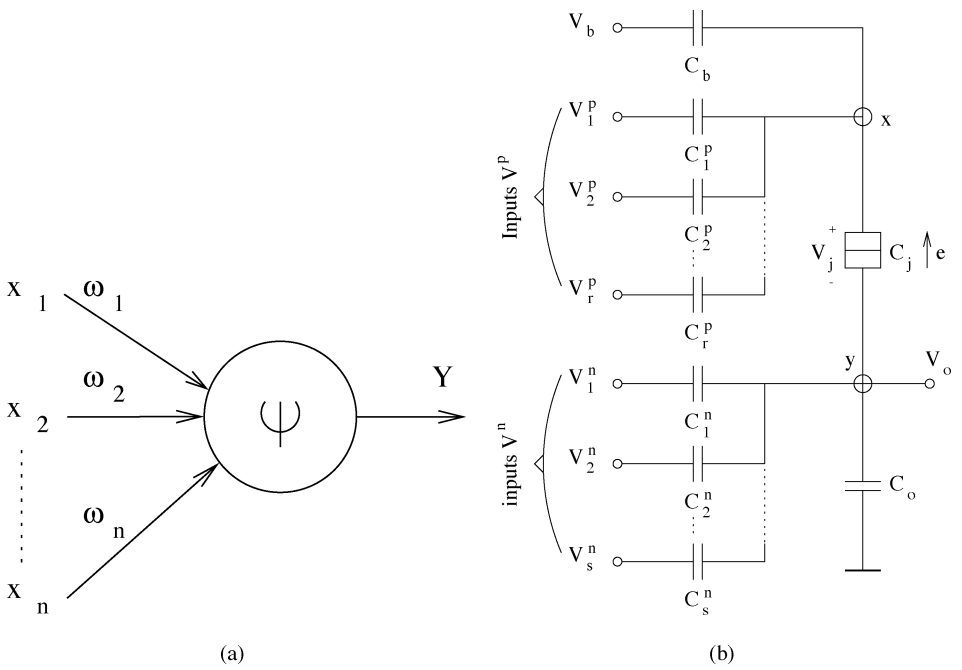

Fig. 1. LTG. (a) Gate symbol. (b) SET generic circuit implementation.

increases the switching delay, as stated in (3). Second, each of the individual junctions involved in the cotunneling process can be replaced by N junctions $(N > 1)$ separated by islands. Although such an approach results in an exponential decrease of the cotunneling probability, it also approximately results in a linear increase in the delay time as an electron must now tunnel through N times as many junctions as before. Third, resistors can be added between the SET circuit and supply voltage lines, as demonstrated in [14]–[16]. This method can reduce the cotunneling rate without significantly increasing the delay. This is due to the fact that the delay added by a resistor is on the RC scale. Thus, assuming, for example, $R = O(10^6)\ \Omega$ and $C = O(10^{-17})$ F, we find that the delay added by the resistor is $t_{RC} = O(10^{-11})$ s. Given that for the structures we propose in this paper the switching delay $t_d = O(10^{-9})$ s, the additional delay due to the cotunneling suppressing resistors would be negligible. Although the circuits discussed in the remainder of this paper do not contain such resistors, cotunneling suppressing resistors of appropriate value can be appended to our designs in order to reduce the cotunneling error to the acceptable error probability.

III. Building Blocks for Single Electron Logic

When designing memory elements in SET technology, the most straightforward implementation method is utilizing existing Boolean gate-based memory elements, as found in most textbooks on logic design (see, e.g., [17] and [18]). Such an approach, however, requires an SET-based family of Boolean logic gates. Thus, we first introduce a generic threshold gate scheme. We next present a family of Boolean logic, consisting of AND, OR, NAND, and NOR gates, which is derived from the generic threshold gate. The Boolean logic gates, as well as the threshold gates themselves, serve as building block for the R–S latch, D latch, a D flip-flop proposed in Sections IV and V.

A. Threshold Logic Gates

Threshold logic gates are devices that are able to compute any linearly separable Boolean function given by

$$Y = \operatorname{sgn}\{\mathcal{F}(X)\} = \begin{cases} 0, & \text{if } \mathcal{F}(X) < 0 \\ 1, & \text{if } \mathcal{F}(X) \geq 0 \end{cases} \tag{6}$$

$$\mathcal{F}(X) = \sum_{i=1}^{n} \omega_i x_i - \psi \tag{7}$$

where x_i are the n Boolean inputs and w_i are the corresponding n integer weights. The LTG (see Fig. 1(a) for the gate symbol) performs a comparison between the weighted sum of the inputs $\Sigma_{i=1}^{n} \omega_i x_i$ and threshold value ψ. If the weighted sum of inputs is *greater than or equal to* the threshold, the gate produces a logic 1. Otherwise the output is a logic 0. A generic threshold gate scheme [19], which is displayed in Fig. 1(b), has been proposed. The circuit operates as follows. The input voltages V^p (weighted by their input capacitors C^p) are added to V_j, while the input voltages V^n (weighted by their input capacitors C^n) are subtracted from V_j. The critical voltage V_c of the tunnel junction acts as the threshold value. The bias voltage V_b weighted by its input capacitor C_b adjusts the threshold value to the desired value. If the voltage V_j across the junction is less then V_c, no charge transport can occur and the circuit's output remains "low." If $|V_j| > V_c$, one electron tunnels through the junction (from node y to node x), resulting in a "high" output. This scheme can, therefore, be used as a basis for implementing LTGs with both positive and negative weights.

Such threshold gates, however, do not operate correctly in networks due to the passive nature of the circuit. It was found [20] that augmenting the output of each threshold gate with an SET buffer/inverter consisting of two SET transistors, as displayed in Fig. 2, results in correctly operating threshold gate networks. The buffer can also function as a standalone inverter

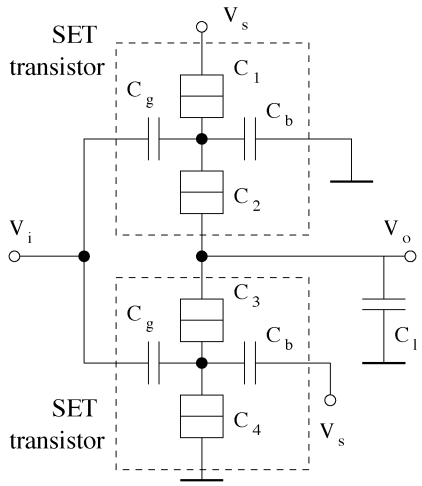

Fig. 2. SET buffer/inverter circuit implementation.

gate. Therefore, threshold gates can also be augmented with two cascaded buffers, such that both the normal and inverted output are available. Both the generic threshold gate and buffer operate in accordance with the SEEL paradigm, i.e., charge transport due to switching activity is limited to one electron. Note that the buffer/inverter can be augmented with strip resistors in order to suppress cotunneling. Referring to Fig. 2, this is achieved by adding strip resistors between junction 1 and V_s and between junction 4 and ground, as suggested in [16].

The implementations proposed in this paper are intended to be general, i.e., independent of a manufacturing process. Therefore, we only estimate the area of circuits in terms of the number of required circuits element. By circuit elements, we refer to the number of capacitors and tunnel junctions required for each implementations. In the remainder of this paper, we assume that Boolean input/output signals correspond with the following voltages: logic $0 = 0$ V, logic $1 = 0.1\ q_e/C$ V, where C acts as a unit for capacitance. For the circuit simulations, it is assumed that $C = 1$ aF, resulting in logic $1 = 16$ mV. For threshold gates, we assume the following circuit parameters: $C_j = 0.1\ C$, $V_b = 0.1\ e/C$. The remaining circuit parameters of the threshold gates depend on the required input weights and threshold value of specific gates.

For the buffer/inverter, the following circuit parameter ratios are assumed: $C_g = 0.5\ C$, $C_b = 4.25\ C$, $C_1 = C_4 = 0.1\ C$, $C_2 = C_3 = 0.5\ C$, $C_l = 9\ C$, $V_s = 0.1\ q_e/C$. Given the methodology described in Section II and assuming $C = 1$ aF, the calculated area, delay, and energy consumption per (output) switching of the buffer/inverter are summarized in Table I. Note that there exists a tradeoff between the gate's switching delay and corresponding switching error probability. If, for example, one chooses $P_{\text{error}} = 10^{-8}$, the corresponding delay would be $t_d = 0.41$ ns.

B. Boolean Logic Gates

Given that the basic Boolean logic functions AND, OR, NAND, and NOR can be specified in the form of (6) and (7), we can implement the AND, OR, NAND, and NOR gates as instances of the generic threshold gate circuit (displayed in Fig. 1) augmented with a buffer/inverter (displayed in Fig. 2). In theory, the thresholds ψ are integer numbers. However, if the threshold, for example, is $\psi = i$ (i being an integer value), this implies that the gates function correctly for any value in the interval $i - 1 < \psi <= i$. In order to maximize robustness for variations in parameter values, we replace the threshold value $\psi = i$ by the average $\psi = i - (1/2)$. The correctness of the above can easily be verified. Consequently, the threshold equations of the two-input AND, OR, NAND, and NOR gates can be written as

$$\text{AND}(a,b) = \text{sgn}\{a + b - 1.5\} \tag{8}$$

$$\text{OR}(a,b) = \text{sgn}\{a + b - 0.5\} \tag{9}$$

$$\text{NAND}(a,b) = \text{sgn}\{-a - b + 1.5\} \tag{10}$$

$$\text{NOR}(a,b) = \text{sgn}\{-a - b + 0.5\}. \tag{11}$$

The threshold gate-based implementations of the Boolean gates all have the same basic circuit topology (for a general case, see Fig. 1) consisting of a bias capacitor C_b, a tunnel junction with capacitance C_j, and an output capacitor C_o. The AND and OR gates contain two input capacitors $C_1^p = C_2^p = 0.5\ C$ for positively weighted inputs, while the NAND and NOR gates contain two capacitors $C_1^n = C_2^n = 0.5\ C$ for negatively weighted inputs. Additionally, each of the threshold gates is augmented with an output buffer. Given that the buffer inverts its input, the logic function performed by the buffered threshold gate is the inverse of that performed by the threshold gate itself. For example, a buffered AND gate implements the NAND function. For the remainder of this discussion, when referring to a logic function such as AND, we imply the logic function performed by the entire gate (threshold gate + output buffer).

For the buffered Boolean logic gates, the following circuit parameter ratios are assumed: $C_b(\text{AND}) = 10.6\ C$, $C_b(\text{OR}) = 11.7\ C$, $C_b(\text{NAND}) = 13.2\ C$, $C_b(\text{NOR}) = 11.7\ C$, $C_j = 0.1\ C$, $C_1^p(\text{NAND},\text{NOR}) = C_2^p(\text{NAND},\text{NOR}) = 0.5\ C$, $C_1^n(\text{AND},\text{OR}) = C_2^n(\text{AND},\text{OR}) = 0.5\ C$, $C_o(\text{AND},\text{OR}) = 8\ C$, $C_o(\text{NAND},\text{NOR}) = 9\ C$. The circuit parameters for the buffer are as specified in Section III-A.

The buffered Boolean gates have been verified by means of simulation using the single electron device and circuit simulator SIMON [10]. The simulation results are depicted in Fig. 3. As can be observed, each of the gates correctly implements the specified logic function.

Given the circuit parameters ratios for the AND, OR, NAND, and NOR gate described above and assuming $C = 1$ aF, we calculated the area, delay, and energy consumption per (output) switching of each of the gates. The combined results are summarized in Table II. Note that there exists a tradeoff between the gates' switching delay and the corresponding switching error probability. If, for example, one chooses $P_{\text{error}} = 10^{-8}$ for the two-input AND gate, the corresponding delay would be $t_d =$

TABLE I
AREA, DELAY, AND POWER CONSUMPTION FOR BUFFER/INVERTER GATE

Gate	Area	Delay	Switching Energy
buffer/inverter	9 elements	0.022 $\lvert ln(P_{error}) \rvert$ ns	10.4 meV

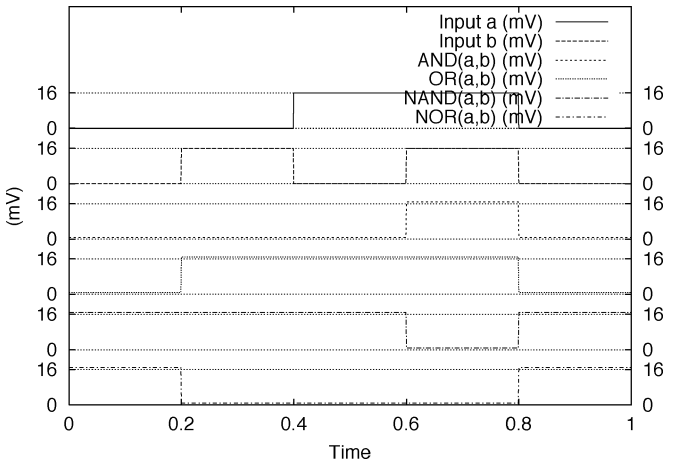

Fig. 3. Simulation results for buffered two-input AND, OR, NAND, and NOR gates.

TABLE II
AREA, DELAY, AND POWER CONSUMPTION FOR TWO-INPUT BUFFERED AND, OR, NAND, AND NOR GATE

Gate	Area	Delay	Switching Energy
2-input AND	14 elements	0.062 $\lvert ln(P_{error})\rvert$ ns	10.8 meV
2-input OR	14 elements	0.062 $\lvert ln(P_{error})\rvert$ ns	10.8 meV
2-input NAND	14 elements	0.080 $\lvert ln(P_{error})\rvert$ ns	10.7 meV
2-input NOR	14 elements	0.072 $\lvert ln(P_{error})\rvert$ ns	10.7 meV

1.1 ns. We emphasize that these gates were neither optimized for delay, nor for power consumption. Also, given the capacitor ratio's and voltage levels, which were used for the two-input buffered Boolean gates, reducing the unit for capacitance C by one order of magnitude has the effect of reducing delay by one order of magnitude (for a given switching error probability) while increasing power consumption by one order of magnitude.

IV. BOOLEAN GATE-BASED MEMORY ELEMENTS

Here, we investigate Boolean gate-based implementations of the R–S latch, D latch, and D flip-flop. Each of these implementations is based on the family of SEEL Boolean logic gates discussed in Section III-B and is discussed in detail below.

A. R–S-Latch Implementation

The R–S latch is a memory element with two inputs (R and S) and two outputs (Q and QN). The behavior of the R–S latch, summarized in Table III, is as follows. If R and S are both zero, the R–S latch holds the current output values. If $S = 1$ and $R = 0$, the output Q is set to $Q = 1$ (and $QN = 0$). If $R = 1$ and $S = 0$, the output Q is reset to $Q = 0$ (and $QN = 1$). The remaining input combination $R = S = 1$ should be avoided during normal operation, as for Boolean gates-based implementations, it typically results in unstable output values.

A Boolean gate-based implementation of the R–S latch usually consists of two cross-coupled gates that form a feedback loop. An R–S-latch implementation based on NOR gates is depicted in Fig. 4. The circuit operates as follows. When R and

TABLE III
FUNCTIONAL TRUTH TABLE OF THE R–S LATCH

R	S	Q	QN	Function
0	0	last Q	last QN	Hold current output values Q and QN.
0	1	1	0	Set output Q to 1 (and QN to 0).
1	0	0	1	Reset output Q to 0 (and QN to 1).
1	1	?	?	Unspecified / Forbidden input combination.

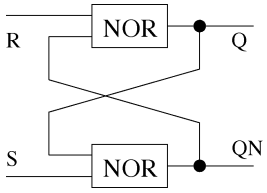

Fig. 4. Boolean gate-based R–S-latch implementation.

S are both zero, the two gates behave as chained inverters and form a bi-stable element (where $Q = 0$ and $QN = 1$, and $Q = 1$ and $QN = 0$ are stable states). If $R = 1$ while $S = 0$, the output of the upper NOR gate is forced to zero, resulting in $Q = 0$ and $QN = 1$ (similar for $S = 1$ and $R = 0$, resulting in $Q = 1$ and $QN = 0$). If $R = S = 1$, the output of both NOR gates is forced to zero. If both inputs are then dropped to zero simultaneously (forming a bi-stable element), the circuit either switches to $Q = 0$, $QN = 1$ or $Q = 1$, $QN = 0$, and may even oscillate between these two solutions. Given that this behavior is unpredictable, this input combination should be avoided.

A NOR gate-based R–S-latch implementation that consists of the buffered Boolean logic gates discussed in Section III-B has been verified by means of simulation (using SIMON) using the following circuit parameters: logic "0" = 0 V, logic "1" = $V_b = V_s = 16$ mV, $C_b = 11.7$ aF, $C_1^p = C_2^p = 0.5$ aF, $C_j = 0.1$ aF, $C_o = 9$ aF, $C_{g1} = C_{g2} = 0.5$ aF, $C_1 = C_4 = 0.5$ aF, $C_2 = C_3 = 0.1$ aF, $C_{b1} = C_{b2} = 4.25$ aF, $C_l = 9$ aF. Other simulator parameters include a tunnel resistance $R_t = 10^5\ \Omega$ and an operating temperature $T = 0$ K. The simulation results are depicted in Fig. 5. The first two bars represent the inputs R and S, and the bottom two bars represent the outputs Q and QN. Initially, the inputs are $S = 0$ and $R = 0$, while the outputs are $Q = 0$ and $QN = 1$. When the input S becomes one, the outputs are set to $Q = 1$ and $QN = 0$. These output values are memorized when S returns to zero. Likewise, when R becomes one, the outputs are reset to $Q = 0$ and $QN = 1$, which is memorized when R return to zero. Next, R and S are both set to one, as a result of which the outputs Q and QN are both set to zero. When R and S are then simultaneously set to zero, the simulator evaluates possible tunnel events until all are resolved, which, for this simulation, resulted in the displayed output values $Q = 1$ and $QN = 0$. We, therefore, conclude that the circuit correctly implements the behavior described above in Table III.

B. D-Latch Implementation

The D latch is a memory element with two inputs (D and L) and two outputs (Q and its complement QN). The behavior of

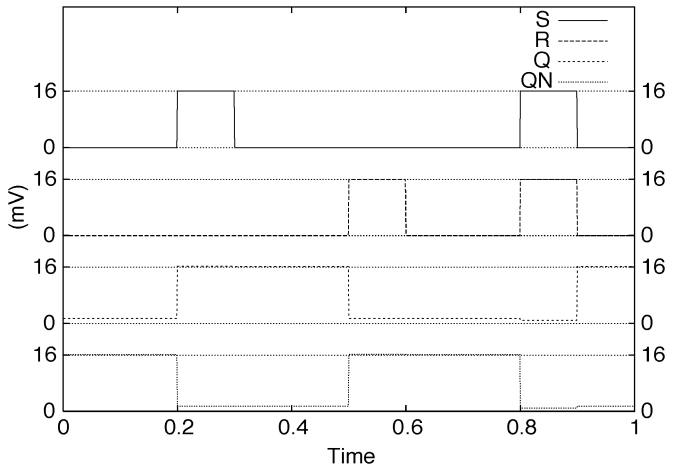

Fig. 5. Simulation results of Boolean gate-based R–S latch.

TABLE IV
FUNCTIONAL TRUTH TABLE OF THE D LATCH

L	D	Q	QN	Function
0	0	last Q	last QN	Hold current output values.
0	1	last Q	last QN	Hold current output values.
1	0	0	1	Transparent: Q=0 and QN=1
1	1	0	0	Transparent: Q=1 and QN=0

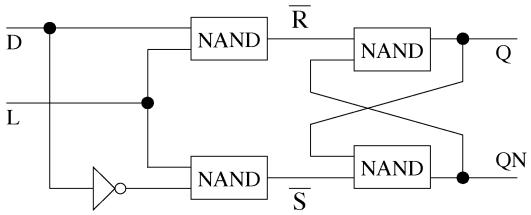

Fig. 6. Boolean gate-based D-latch implementation.

the D latch, summarized in Table IV, is as follows. If the input $L = 0$, the D latch holds the current output values. If the input $L = 1$, the latch is transparent and the output Q follows the input D (and QN follows the complement of D). Unlike the R–S latch, the D latch does not have an unspecified or forbidden input combination.

A possible D-latch implementation based on Boolean logic gates is depicted in Fig. 6. The circuit operates as follows. The cross-coupled NAND gates form an $\overline{R}$–$\overline{S}$ latch, where $\overline{R} = \overline{S} = 1$ corresponds with the hold function. When $L = 0$, the inputs of the $\overline{R}$–$\overline{S}$ are both one regardless of the value of D, and the $\overline{R}$–$\overline{S}$ latch holds its current output values. If $L = 1$, the inputs of the $\overline{R}$–$\overline{S}$ have complementary values, and the output Q becomes follows the value of D.

A D-latch implementation consisting of the buffered Boolean logic gates discussed in Section III-B has been verified by means of simulation (using SIMON) using the following circuit parameters: logic "0" $= 0$ V, logic "1" $= V_b = V_s = 16$ mV, $C_b = 10.6$ aF, $C_1^p = C_2^p = 0.5$ aF, $C_j = 0.1$ aF, $C_o = 9$ aF, $C_{g1} = C_{g2} = 0.5$ aF, $C_1 = C_4 = 0.5$ aF, $C_2 = C_3 = 0.1$ aF, $C_{b1} = C_{b2} = 4.25$ aF, $C_l = 9$ aF. Other simulator parameters include a tunnel resistance $R_t = 10^5\ \Omega$ and an operating temperature $T = 0$ K. Note that the inverter gate is a standalone buffer. The simulation results are depicted in Fig. 7. The first two bars represent the inputs D and L, and the bottom two bars represent the outputs Q and $\overline{Q}$. Initially, the inputs are $L = 0$ and $D = 0$, while the outputs are $Q = 0$ and $QN = 1$. When D becomes one, while L remains zero, the outputs remain unchanged. Once L is set to one, the output Q follows the values of D until L is dropped to zero. At that point, the last values of Q are memorized. The same can be observed when L becomes one again. We, therefore, conclude that the circuit correctly implements the behavior described above in Table IV.

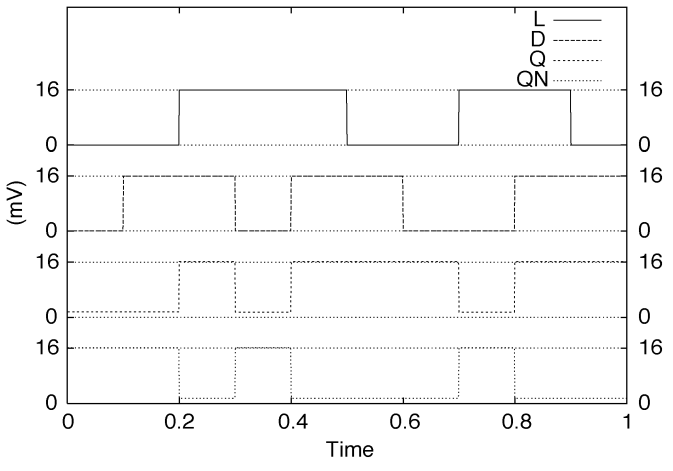

Fig. 7. Simulation results of Boolean gate-based D latch.

TABLE V
FUNCTIONAL TRUTH TABLE OF THE POSITIVE EDGE-TRIGGERED D FLIP-FLOP

CLK	D	Q	QN	Function
$0 \rightarrow 1$	0	0	1	Sample D: Q=0 and QN=1
$0 \rightarrow 1$	1	1	0	Sample D: Q=1 and QN=0
0	0	last Q	last QN	Hold current output values.
0	1	last Q	last QN	Hold current output values.
1	0	last Q	last QN	Hold current output values.
1	1	last Q	last QN	Hold current output values.

C. Edge-Triggered D Flip-Flop Implementation

An edge-triggered D flip-flop is a memory element with two inputs (D and CLK) and two outputs (Q and its complement QN). The behavior of the positive edge-triggered D flip-flop is as follows, and as summarized by Table V. When the input CLK transitions from 0 to 1 (a rising or positive edge on the time graph), the flip-flop samples the current value of D and copies this value to the output Q. For all other input combinations, including a negative-edge clock transition from 1 to 0, the circuit holds its current output values.

A possible implementation of the positive edge-triggered D flip-flop consists of combining two D latches, as depicted in Fig. 8. The circuits works as follows. The first latch is referred to as the master and the second one is referred to as the slave.

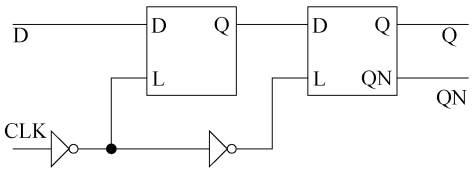

Fig. 8. D-latch-based positive edge-triggered D flip-flop.

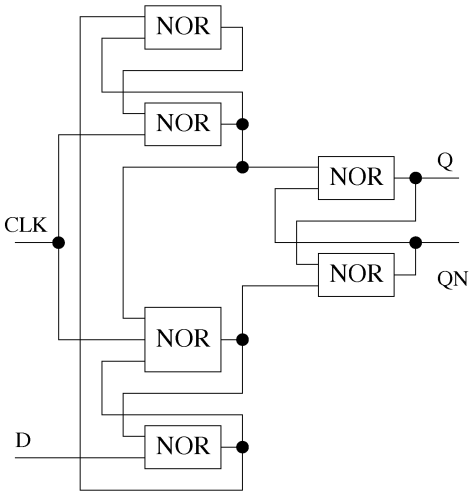

Fig. 9. Boolean gate-based negative edge-triggered D flip-flop.

When $\mathrm{CLK} = 0$, the master latch follows D, while the slave latch holds its current output values. When CLK becomes one, the master latch closes (holding its current output values) and transfers its output value Q to the slave latch, which is now transparent. However, since the output of the master latch is now constant, the slave's output remains constant regardless of the current value of D. If CLK then switches back to zero, the slave latch holds its current output values. The positive edge-triggered flip-flop can be modified into a negative edge-triggered design by removing the first inverter in the clock path.

In Section IV-B, we discussed the Boolean gate-based D latch. Given that this D-latch implementation requires four NAND gates and two inverters, the flip-flop depicted in Fig. 8 would require a total of eight NAND gates and six inverters. However, there are faster and smaller Boolean gate-based implementations that specifically make use of the R–S-latch "unstable" output values (such as $Q = 0$ and $QN = 0$ for the cross-coupled NOR gate-based R–S latch). One such implementation, realizing a negative edge-triggered D flip-flop, is based on three NOR gate-based R–S latches. This flip-flop implementation requires a total of five two-input NOR gates and one three-input NOR gate, as depicted in Fig. 9.

The negative edge-triggered D flip-flop has been implemented using the buffered Boolean logic gates discussed in Section III-B. All circuit parameters of the two-input NOR gates are equal to those used in Section IV-A to implement the R–S latch. The three-input NOR gate was also derived as an instance of the generic threshold gate (as a three-input OR gate) and then augmented with a static inverting buffer. Given the same methodology as described for the two-input gates in Section III-B, we calculated $C_b(\text{three-input NOR}) = 12.3\,C$,

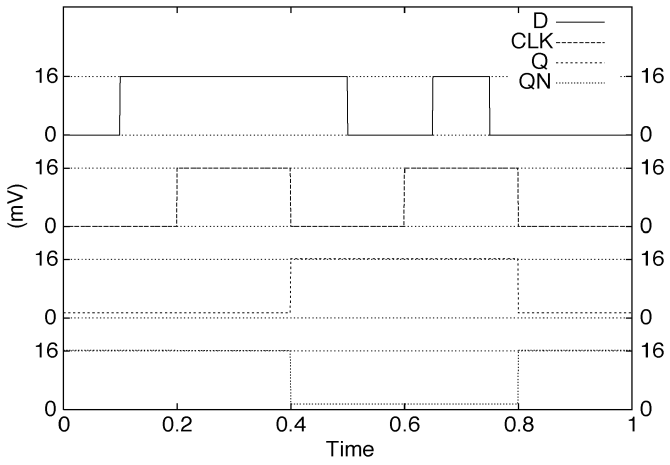

Fig. 10. Simulation result of Boolean gate-based negative edge-triggered D flip-flop.

while all other circuit parameters remain as used for the two-input NOR.

We have simulated the negative edge-triggered D flip-flop circuit using the simulation package SIMON. The simulation results are displayed in Fig. 10. Starting from the top of this figure, the first row represent the input data signal D. The second row represents the clock signal CLK. The third and fourth bars represent the two outputs Q and QN of the flip-flop. Initially, the inputs are $D = 0$ and $\mathrm{CLK} = 0$, while the outputs are $Q = 0$ and $QN = 1$. When the input D changes to one, the output remain unchanged. The same applied when the input CLK changes to one. The input D is sampled for the first time when CLK changes back to zero (a negative-edge), resulting in $Q = 1$ and $QN = 0$. These output values are memorized until D is sampled again during the next negative-edge transition of CLK, at which point the outputs become $Q = 0$ and $QN = 1$. We, therefore, conclude that the negative edge-triggered D flip-flop operates correctly, sampling the input D (and updating outputs Q and QN) only on the negative edge of the clock signal CLK.

V. Threshold Gate-Based Memory Elements

In previous sections, we have examined Boolean gate-based implementations of the R–S latch, D latch, and edge-triggered D flip-flop, and verified by means of simulation that they operate according to their specified logic function. However, each of the Boolean logic gates that were utilized for these implementations (except for the inverter) is derived from a generic threshold gate design. It is known that any Boolean logic function can also be realized by a network of threshold logic gates [21]. Moreover, when comparing the Boolean gate-based realization with the threshold gate-based realization of the same logic function, the threshold gate-based design has, at most, the same number of logic gates and the same network depth. Here, we, therefore, discuss threshold logic gate-based implementations of the same three memory elements that were implemented in Boolean logic gates in the above section: the R–S latch, D latch, and edge-triggered D flip-flop. Each of the utilized threshold gates has been derived from the generic threshold gate design.

A. R–S Latch

In Boolean logic, the logic function of the Boolean gate-based R–S-latch implementation discussed in Section IV-A can be expressed as

$$Q^{t+1} = \overline{R}S + \overline{R}Q^t \tag{12}$$

$$QN^{t+1} = \overline{S}R + \overline{S}QN^t. \tag{13}$$

Examining the Boolean gate-based R–S-latch implementation, we observe the following. The input combination $R = S = 1$ that results in unstable behavior is a side effect of cross-coupled NOR gate-based implementation. If we implement an R–S latch without cross-coupling gates, the problem regarding the unstable behavior will be resolved. For a Boolean gate-based design, this would results in an increase in the number of required gates. However, for a threshold gate-based design, the opposite occurs: an R–S latch can be implemented at the cost of one threshold gate only as

$$Q^{t+1} = \overline{R}S + \overline{R}Q^t + SQ^t \tag{14}$$

$$QN^{t+1} = \mathrm{NOT}\big(Q^{t+1}\big). \tag{15}$$

We first verify that (14) and (15) correctly implement the specified function of the R–S latch. When $S = R = 0$, we find $Q^{t+1} = Q^t$ (hold). When $S = 1$ and $R = 0$, we find $Q^{t+1} = 1$ (set). When $S = 0$ and $R = 1$, we find $Q^{t+1} = 0$ (reset). When $S = R = 1$, we find $Q^{t+1} = Q^t$ (hold). In all cases, the output QN is the complement of Q. Therefore, we conclude that these logic equations describe the behavior of an R–S latch without a forbidden input combination (both $R = S = 0$ and $R = S = 1$ correspond with the hold function).

Examining (14), we observe the following. A Boolean gate-based implementation of this equation costs three two-input OR gates and one three-input AND gate and would result in a logic network with a depth of two. However, a threshold gate-based implementation of this logic equation would require only one three-input threshold gate implementing

$$Q^{t+1} = \mathrm{sgn}\Big\{S + \overline{R} + Q^t - 2\Big\}. \tag{16}$$

The correctness of the above can be easily verified. Additionally, given that $\overline{R} = -R + 1$, we arrive at the final form

$$Q^{t+1} = \mathrm{sgn}\big\{S - R + Q^t - 1\big\}. \tag{17}$$

As stated earlier, the threshold gates derived from the generic threshold gate scheme require an output buffer for correct operation in a network structure. Given that the applied buffer inverts its input signal, we can place two inverter/buffers in series, such that both Q and its logic complement QN are available. We, therefore, propose an R–S-latch implementation that consists of a three-input threshold gate and two buffer/inverters, as displayed in Fig. 11. The threshold logic gates-based implantation has the additional advantage that it does not have a forbidden input combination. In this case, both $R = S = 0$ and $R = S = 1$ can be used to hold the current output values.

We have verified the proposed R–S-latch implementation by simulation using the single electron device and circuit simulator SIMON [10]. Simulation results were obtained using the following circuit parameters for the threshold logic gate: $C_b = 12.7\,C, C_1^p(\omega_1 = 1) = C_2^p(\omega_2 = 1) = 0.5\,C, C_1^n(\omega_3 = -1) = 0.4\,C$, $C_j = 0.1\,C$, $C_o = 8.6\,C$. For the buffer/inverter, the following circuit parameters were used: $C_{g1} = C_{g2} = 0.5$ aF, $C_1 = C_4 = 0.5$ aF, $C_2 = C_3 = 0.1$ aF, $C_{b1} = C_{b2} = 4.25$ aF, $C_l = 9$ aF. Other simulator parameters include a tunnel resistance $R_t = 10^5\ \Omega$ and an operating temperature $T = 0$ K. The simulation results are displayed in Fig. 12. Initially, the inputs are $S = 0$ and $R = 0$, while the outputs are $Q = 0$ and $QN = 1$. When the input S becomes one, the outputs are set to $Q = 1$ and $QN = 0$. These output values are memorized when S returns to zero. Likewise, when R becomes one, the outputs are reset to $Q = 0$ and $QN = 1$, which is memorized when R return to zero. Next, R and S are both set to one, as a result of which the outputs Q and QN remain unchanged as this input combination now also corresponds with the hold function. Likewise, when R and S are simultaneously set to zero, the output also remain unchanged. We, therefore, conclude that the circuit correctly implements the behavior of the R–S latch without the additional disadvantage of unstable behavior.

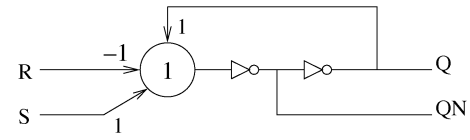

Fig. 11. Threshold gate-based R–S-latch implementation.

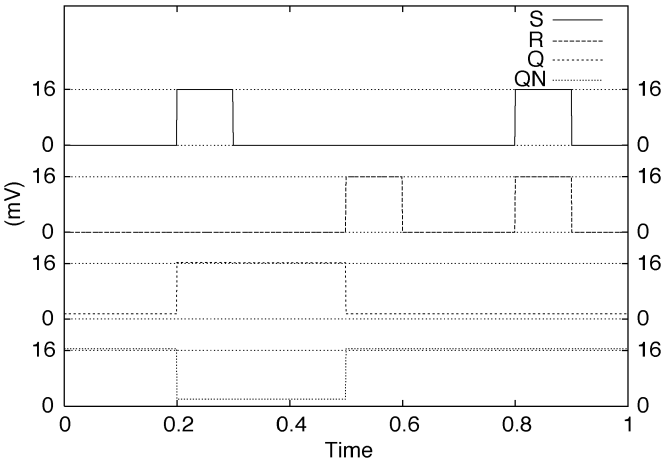

Fig. 12. Simulation results of threshold gate-based R–S latch.

B. D Latch

In Boolean logic, the logic function of the Boolean gate-based D-latch implementation (see Section IV-B) can be expressed as

$$Q^{t+1} = LD + \overline{L}Q^t \tag{18}$$

$$QN^{t+1} = \mathrm{NOT}\big(Q^{t+1}\big). \tag{19}$$

The above equations can easily be verified. If $L = 0$, then (18) becomes $Q^{t+1} = Q^t$ (hold). If $L = 1$, then (18) becomes $Q^{t+1} = D$ (transparent). In all cases, the output QN is the complement of Q.

Examining (18) in detail, we observe that this equation (in the form of $Y = A \cdot B + \overline{A} \cdot C$) cannot be implemented by a single threshold gate. A single threshold gate can only make a linear separation between points in the input space. Fig. 13 depicts the set of possible input combinations for (A, B, C) as a cube. The

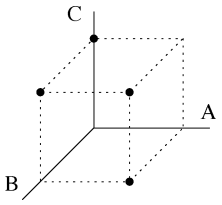

Fig. 13. Solution space of $Y = A \cdot B + \overline{A} \cdot C$.

combinations that result in $Y = 1$ are displayed as solution points. In order to be a linearly separable solution space, all solution points should be separable from the remaining points by a single plane. As this cannot be realized, functions in the form of $Y = A \cdot B + \overline{A} \cdot C$ cannot be implemented by a single threshold gate.

Given that at least two threshold gates are required to implement the logic function stated in (18), we split this equation in two parts, which each can be implemented as a single threshold gate as

$$A = LD \tag{20}$$

$$Q^{t+1} = A + \overline{L}Q^t. \tag{21}$$

One can verify the validity of this by straightforward substitution. Each of these two equations can now be specified as a single threshold equation

$$A = \text{sgn}\{L + D - 2\} \tag{22}$$

$$Q^{t+1} = \text{sgn}\{2A - L + Q^t - 2\}. \tag{23}$$

The conversion of the Boolean logic equations into their threshold logic counterparts can be verified as follows. The first equation only results in $A = 1$ when both L and D are one and, thus, implements a Boolean logic AND. Likewise, the second equation only results in $Q^{t+1} = 1$ when $A = 1$ or when both L and Q^t are one. As stated earlier, the threshold gates derived from the generic threshold gate scheme require an output buffer for correct operation in a network structure. Given that the applied buffer inverts its input signal, we modify the threshold equation of A such that it calculates $\overline{A} = \text{sgn}\{-L - D - 1\}$. In this way, the combined result of the threshold gate and its buffer/inverter is a buffered gate that calculates A. Thus, we propose the D-latch implementation depicted in Fig. 14.

We have verified the proposed D-latch implementation by simulation. For $tlg1$, we used the following circuit parameters: $C_b = 13.2\ C$, $C_1^n(\omega_1 = -1) = C_2^n(\omega_2 = -1) = 0.5\ C$, $C_j = 0.1\ C$, $C_o = 8\ C$. For $tlg2$, we used $C_b = 13.1\ C$, $C_1^p(\omega_1 = 2) = 1\ C$, $C_2^p(\omega_2 = 1) = 0.5\ C$, $C_1^n(\omega_3 = -1) = 0.4\ C$, $C_j = 0.1\ C$, and $C_o = 8.6\ C$. For the buffer/inverter, the same circuit parameters were utilized as in Section V-A. The simulation results are displayed in Fig. 15. As can be observed, the output Q follows D, while $L = 1$, and retains its last value while $L = 0$, thus correctly implementing the D latch.

C. Edge-Triggered D Flip-Flop

A common implementation of an edge-triggered D flip-flop is a cascade of two D latches, as depicted in Fig. 8, such that the output Q of the first latch serves as the input D of the second latch. The input L of the first latch is connected to NOT(CLK), while the input L of the second latch is connected directly to

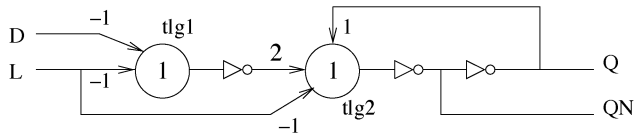

Fig. 14. D-latch implementation.

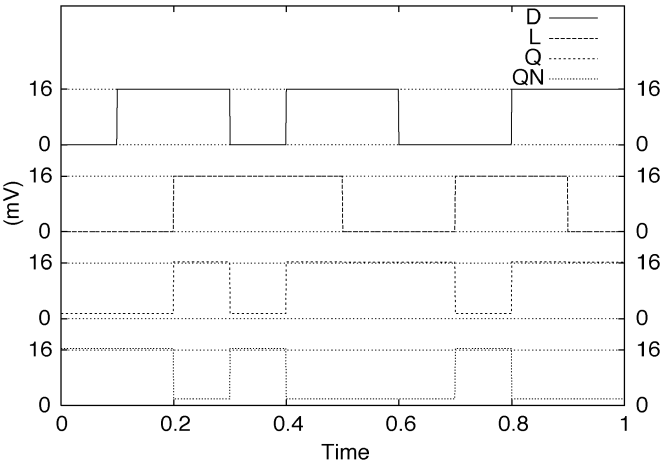

Fig. 15. Simulation results of threshold gate-based D latch.

CLK. This results in a positive edge-triggered D flip-flop. Note that a negative edge-triggered D flip-flop can be implemented by exchanging the L input connections of the two D latches.

Examining Fig. 8, we observe that the L input signal of the first D latch operates on the inverted clock signal NOT(CLK). If we adjust the threshold-based implementation of the D latch such that it implements the Boolean function $Q^{t+1} = \overline{L} \cdot D + LQ^t$ instead of the function specified by (18), we require one less inverter block. The resulting threshold gate-based equations are

$$A = \text{sgn}\{L + D - 1\} \tag{24}$$

$$Q^{t+1} = \text{sgn}\{2A + L + Q^t - 1\}. \tag{25}$$

Given that for threshold equations $\overline{L} = -L + 1$, one can verify the correctness of the above equations. As the applied buffer inverts its input signal, we again modify the threshold equation of A such that it calculates $\overline{A} = \text{sgn}\{L - D\}$. In this way, the combined result of the threshold gate and its buffer/inverter is a buffered gate that calculates A. Resulting, we propose the threshold logic gate-based positive edge-triggered D flip-flop implementation depicted in Fig. 16. Note that QM is the output of the first D latch.

We have verified the proposed D flip-flop implementation by simulation using the SIMON simulator. For $tlg1$, we used the following circuit parameters: $C_b = 12.2\ C$, $C_1^p(\omega_1 = 1) = 0.5\ C$, $C_1^n(\omega_2 = -1) = 0.4\ C$, $C_j = 0.1\ C$, $C_o = 8.6\ C$. For $tlg2$, we used $C_b = 11.7\ C$, $C_1^p(\omega_1 = 2) = 1\ C$, $C_2^p(\omega_2 = 1) = C_3^p(\omega_3 = 1) = 0.5\ C$, $C_j = 0.1\ C$, and $C_o = 9\ C$. The circuit parameters for $tlg3$ and $tlg4$, as well as those of the buffer/inverter, are equal to those used for the D latch described in Section V-B. The simulation results are displayed in Fig. 17. The output QM of the first D latch is only displayed for reference. Initially, the output Q is zero. The first rising edge of CLK samples $D = 1$, resulting in $Q = 1$. This value is retained until the second rising edge samples $D = 0$, resulting in $Q = 0$.

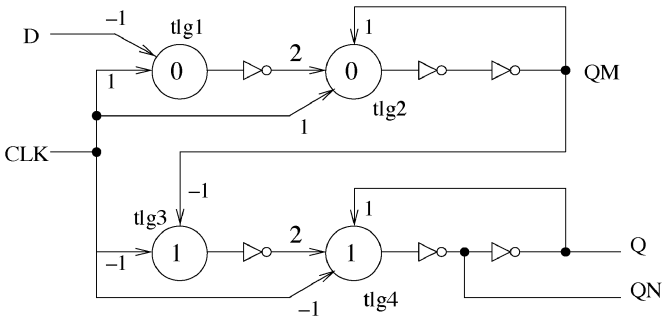

Fig. 16. Positive edge-triggered D flip-flop implementation.

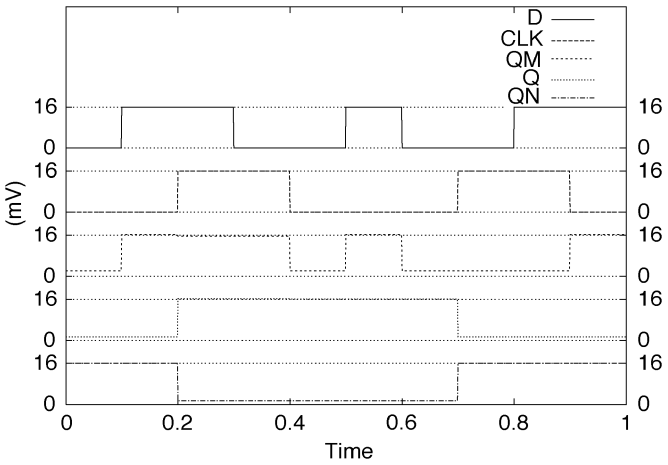

Fig. 17. Simulation results of threshold gate-based positive edge-triggered D flip-flop.

VI. Discussion

Here, we present the estimated area, delay, and switching energy consumption of the memory element implementations discussed in Sections IV and V. The estimated total area of the implementations is expressed in circuit elements (capacitors and junctions). Given our aim to remain fabrication-technology independent, we cannot estimate the exact layout dimensions. As such, the estimated area only serves as a comparison metric of the different implementations.

The estimates for switching delay and energy consumption strongly depend on the chosen unit capacitance C. In this paper, we assume a unit capacitance $C = 1$ aF. Although this choice is arbitrary, it is a commonly assumed value that is also used in other publications on SET-based logic (see, for example, [22]). There is also a tradeoff between the switching delay and corresponding switching error probability. When, for example, a larger switching error probability is acceptable or a smaller unit capacitance is selected, the delay of the individual Boolean and threshold logic gates can be dramatically reduced. The switching delay as a function of the switching error probability P_{error} is depicted in Fig. 18(a). Note that the gate delay has been normalized for the error probability $P_{\text{error}} = 10^{-8}$. Likewise, Fig. 18(b) depicts the switching delay as a function of the unit capacitance C and is normalized for $C = 10^{-18}$ F. From Fig. 18(a) and (b), one can, for example, deduce that, compared to the normalized values, choosing $P_{\text{error}} = 10^{-6}$ and $C = 0.5$ aF results in $t_d = 0.38$.

We can estimate the area, switching delay, and switching energy consumption of the Boolean gate-based memory element implementations discussed in Section IV by adding the results obtained for the individual gates. The area, switching delay, and switching energy consumption of the inverter gate and the two-input buffered Boolean logic gates are presented in Table sI and II, respectively. Given this approach, the obtained results for the Boolean gate-based implementations are presented in Table VI. For the threshold gate-based implementations, a similar approach is applied. We first calculate the area, switching delay, and switching energy consumption of the individual threshold gates (using the same methodology as applied for the Boolean gates). We then add the results of the individual gates in order to obtain the area delay and power consumption of the threshold gate-based implementations, which are presented in Table VII. The delay calculations of the R–S latch and D latch are based on the critical path from the inputs to the output. The delay calculations of the D flip-flop are based on the critical path from the clock input CLK to the outputs. The switching energy calculations are based on all individual gates of the memory elements switching their output values.

When comparing the Boolean and threshold gate-based implementations of the individual memory elements, we observe the following. The threshold logic-based implementations of the three memory elements each require less circuit elements to implement then their Boolean gate-based counterparts. Also, the threshold gate-based implementations result in less delay, while still consuming less power then their Boolean counterparts. This, in general, suggests that, for SEEL, threshold gate-based implementations of logic circuit are a promising alternative to Boolean gate-based implementations.

As stated earlier in Section II, in addition to the switching error probability, there are two fundamental phenomena that may cause errors: thermally induced tunneling and cotunneling. Given a maximum acceptable switching error probability, we must ensure that both the thermal error probability, as well as the cotunneling error probability, are of the same order of magnitude or less. The thermal error probability is a function of the operating temperature T and the energy per switching event ΔE. Given our choice for the unit capacitance $C = 10^{-18}$, we find that $\Delta E = O(10^{-3})$ eV for each individual gate. Assuming, for example, a maximum acceptable thermal error probability $P_{\text{therm}} = 10^{-8}$, we find a maximum operating temperature $T = 6.3$ K. This suggests that, for room-temperature operation, we require at least $C = 10^{-19}$ F, as well as further design optimizations. Likewise, we must ensure that the cotunneling error probability is $O(10^{-8})$ or less. Given the experimental results with strip resistors reported in [16], we can assume that a strip resistor $R_s = 10^5\ \Omega$ is sufficient to achieve this accuracy. The addition of such a strip resistor would not result in a significant increase of the delay. Given our choice of circuit parameters and assuming an ideal resistor, the capacitive load in parallel with the strip resistor is of $O(10^{-19})$ F. Hence, the added RC delay would only be $t_{RC} = O(10^{-14})$ s. We can, therefore, assume that delay added by the strip resistor can be neglected.

Earlier proposals for SET-based memory elements can be divided in two main categories. The first category consists of memory cell designs based on the quantum dot or floating gate

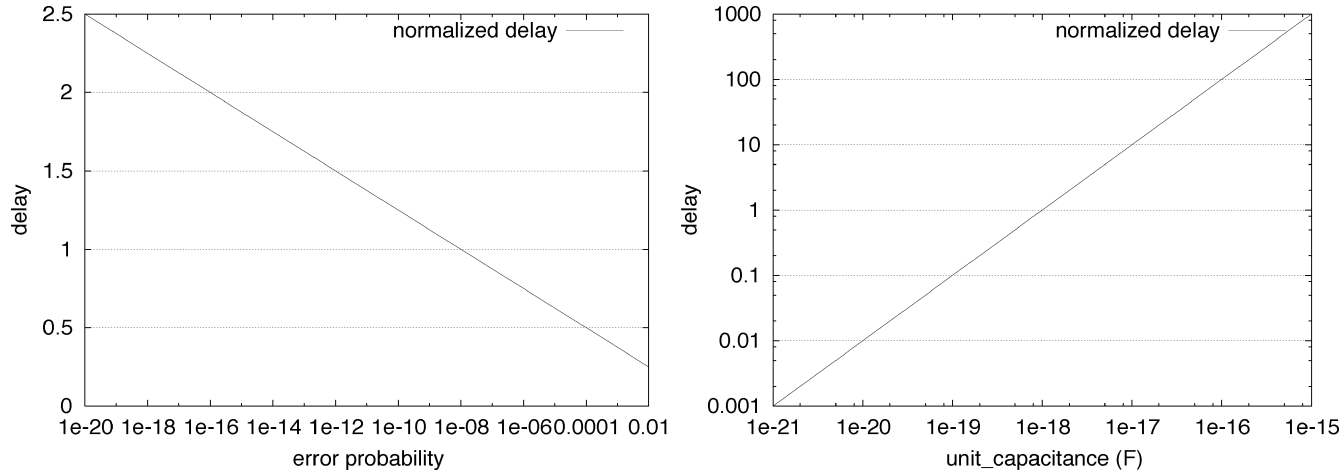

Fig. 18. Normalized gate delay: (a) versus error probability and (b) versus unit capacitance.

TABLE VI
AREA, DELAY, AND POWER CONSUMPTION OF BOOLEAN GATE-BASED IMPLEMENTATIONS

Circuit	Area	Delay	Switching Energy
R-S latch	28	0.14 $\lvert ln(P_{error})\rvert$ ns	21.4 meV
D latch	65	0.26 $\lvert ln(P_{error})\rvert$ ns	53.2 meV
D flip-flop	85	0.22 $\lvert ln(P_{error})\rvert$ ns	64.2 meV

TABLE VII
AREA, DELAY, AND POWER OF THRESHOLD GATES-BASED IMPLEMENTATIONS

Circuit	Area	Delay	Switching Energy
R-S latch	24	0.098 $\lvert ln(P_{error})\rvert$ ns	21.1 meV
D latch	38	0.16 $\lvert ln(P_{error})\rvert$ ns	31.9 meV
D flip-flop	76	0.16 $\lvert ln(P_{error})\rvert$ ns	63.8 meV

principle (see, for example, [23]–[25]). Most designs can be thought of as a three-terminal device. The gate terminal can be utilized to tunnel charge to or from the quantum dot. The presence or absence of this charge changes the conductivity between the source and drain connection. These designs typically transport larger amounts of charge, require a reset (dynamic memory), and require signal amplification. Such designs are intended for large-scale memory arrays, such as dynamic random access memory (DRAM), and cannot be compared with the proposed memory elements. The second category is based on multiple tunnel junctions (MTJs) (see, for example, [26]–[28]). An MTJ consists of a chain of tunnel junctions and can be through of as a two-terminal device. MTJ designs typically operate on a three-phased control input. If the control input is set to "enable," the data input determines if charge transport occurs through the first junction of the MTJ. If charge transport does occur, it results in a chain of tunnel events, until the charge has been transported through all junctions. Once the charge transport sequence has been completed, the control input can switch to a "memorize" voltage (commonly 0 V) and no further charge transport can occur. The control signal must switch a "reset" voltage in order to reset the MTJ after each utilization. Charge transport through the MTJ can also be limited to one electron, realizing single electron memory. A delay of 10 ns has been reported for an MTJ-based DRAM cell with similar sized capacitors as applied in our examples [27]. An MTJ-based majority gate with an "enable" duration of 3.7 ns and a logic cycle time of 15–20 ns has also been reported [22]. Comparing MTJ-based proposals with our proposals, we can conclude the following. The MTJ-based design require a comparable number of circuit elements (ten elements for a two-input majority gate [22] versus 14 elements for a two-input Boolean or threshold gate). However, MTJ-based designs require more complicated control logic (three control voltage levels versus a single dc voltage) then our designs. Finally, the proposed threshold gate-based memory elements have less delay then reported MTJ-based designs, while realizing more complex functionality (latch and flip-flop versus DRAM cell and majority gate).

VII. CONCLUSIONS

SET technology offers the ability to control the transport of individual electrons. In this paper, we have investigated SEEL memory circuits in which the Boolean logic values are encoded as zero or one electron charges. More specifically, we focused on the implementation of SEEL latches and flip-flops. All proposed circuits were verified by means of simulation using the simulation package SIMON. We first presented a generic SEEL LTG implementation from which we derived a family of Boolean logic gates. Second, we proposed Boolean gate-based implementations of the R–S latch, D-latch, and D flip-flop. Third, we proposed threshold gate-based implementations of the same memory elements. Finally, we discusses the estimated area, delay, and power consumption of the Boolean and threshold gate-based implementations, and compared them with other SET-based memory elements.

ACKNOWLEDGMENT

The authors would like to thank Prof. Y. V. Nazarov for useful discussions, as well as the anonymous reviewers of this paper's manuscript.

REFERENCES

[1] Y. Taur, D. A. Buchanan, W. Chen, D. Frank, K. Ismail, S. Lo, G. Sai-Halasz, R. Viswanathan, H. Wann, S. Wind, and H. Wong, "CMOS scaling into the nanometer regime," *Proc. IEEE*, vol. 85, pp. 486–504, Apr. 1997.

[2] A. Korotkov, "Single-electron logic and memory devices," *Int. J. Electron.*, vol. 86, no. 5, pp. 511–547, 1999.

[3] K. Likharev, "Single-electron devices and their applications," *Proc. IEEE*, vol. 87, pp. 606–632, Apr. 1999.

[4] "Technology roadmap for nanoelectronics,", [Online]. Available: http://www.cordis.lu/esprit/src/melna-rm.htm, published on the Internet by the Microelectronics Advanced Research Initiative (MELARI NANO), a European Commission (EC) Information Society Technologies (IST) program on Future and Emerging Technologies, 1999.

[5] A. Korotkov, R. Chen, and K. Likharev, "Possible performance of capacitively coupled single-electron transistors in digital circuits," *J. Appl. Phys.*, vol. 78, pp. 2520–2530, Aug. 1995.

[6] J. R. Tucker, "Complementary digital logic based on the "Coulomb blockade"," *J. Appl. Phys.*, vol. 72, no. 9, pp. 4399–4413, Nov. 1992.

[7] K. K. Likharev and V. Semenov, "Possible logic circuits based on the correlated single-electron tunneling in ultrasmall junctions," in *Int. Superconductive Conf. Extended Abstracts*, 1987, p. 182.

[8] Y. N. Nazarov and S. V. Vyshenskii, "SET circuits for digital applications," in *Single-Electron Tunneling and Mesoscopic Devices*. ser. Electron. Photon., H. Koch and H. Lubbig, Eds. Berlin, Germany: Springer-Verlag, 1992, vol. 31, pp. 61–66.

[9] A. Korotkov and K. Likharev, "Single-electron-parametron-based logic devices," *J. Appl. Phys.*, vol. 84, no. 11, pp. 6114–6126, Dec. 1998.

[10] C. Wasshuber, H. Kosina, and S. Selberherr, "SIMON—A simulator for single-electron tunnel devices and circuits," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 937–944, Sept. 1997.

[11] C. Wasshuber, "About single-electron devices and circuits," Ph.D. dissertation, Elect. Eng. Dept., Tech. Univ. Vienna, Vienna, Austria, 1998.

[12] D. V. Averin and A. A. Odintsov, "Macroscopic quantum tunneling of the electric charge in small tunnel junctions," *Phys. Lett. A*, vol. 140, no. 5, pp. 251–257, Sept. 1989.

[13] D. V. Averin and Y. V. Nazarov, "Virtual electron diffusion during quantum tunneling of the electric charge," *Phys. Rev. Lett.*, vol. 65, no. 19, pp. 2446–2449, Nov. 1990.

[14] S. V. Lotkhov, H. Zangerle, A. B. Zorin, and J. Niemeyer, "Storage capabilities of a four-junction single-electron trap with an on-chip resistor," *Appl. Phys. Lett.*, vol. 75, no. 17, pp. 2665–2667, Oct. 1999.

[15] A. B. Zorin, S. V. Lotkhov, H. Zangerle, and J. Niemeyer, "Coulomb blockade and cotunneling in single electron circuits with on-chip resistors: Toward the implementation of the R pump," *J. Appl. Phys.*, vol. 88, no. 5, pp. 2665–2670, Sept. 2000.

[16] S. V. Lotkhov, S. A. Bogoslovsky, A. B. Zorin, and J. Niemeyer, "Operation of a three-junction single-electron pump with on-chip resistors," *Appl. Phys. Lett.*, vol. 78, no. 7, pp. 946–948, Feb. 2001.

[17] R. Katz, *Contemporary Logic Design*. Redwood City, CA: Benjamin/Cummings, 1994.

[18] J. Wakerly, *Digital Design: Principles and Practices*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2001.

[19] C. Lageweg, S. Coţfană, and S. Vassiliadis, "A linear threshold gate implementation in single electron technology," in *IEEE Computer Society VLSI Workshop*, Apr. 2001, pp. 93–98.

[20] ——, "Static buffered SET based logic gates," in *2nd IEEE Nanotechnology Conf.*, Aug. 2002, pp. 491–494.

[21] S. Muroga, *Threshold Logic and Its Applications*. New York: Wiley, 1971.

[22] T. Oya, T. Asai, T. Asai, T. Fukui, and Y. Amemiya, "A majority-logic device using and irreversible single-electron box," *IEEE Trans. Nanotechnol.*, vol. 2, pp. 15–22, Mar. 2003.

[23] S. Banerjee, S. Huang, and S. Oda, "Operation of nanocrystalline-silicon-based few-electron memory devices in the light of electron storage, ejection and lifetime characteristics," *IEEE Trans. Nanotechnol.*, vol. 2, pp. 88–92, June 2003.

[24] K. Yano, T. Ishii, T. Sano, T. Mine, F. Murai, T. Hashimoto, T. Kobayashi, T. Kure, and K. Seki, "Single-electron memory for giga-to-tera bit storage," *Proc. IEEE*, vol. 87, pp. 633–651, Apr. 1999.

[25] L. Guo, E. Leobandung, and S. Chou, "A silicon single-electron transistor memory operating at room temperature," *Science*, vol. 275, pp. 649–651, 1997.

[26] I. Karafyllidis, "Design and simulation of a single-electron random-access memory array," *IEEE Trans. Circuits Syst. I*, vol. 49, pp. 1370–1375, Sept. 2002.

[27] Z. Durrani, A. Irvine, and H. Ahmed, "Coulomb blockade memory using integrated single-electron transistor/metal–oxide–semiconductor transistor gain cells," *IEEE Trans. Electron Devices*, vol. 47, pp. 2334–2339, Dec. 2000.

[28] K. Katayama, H. Mizuta, H. Muller, D. Williams, and K. Nakazato, "Design and analysis of high-speed random access memory with coulomb blockade charge confinement," *IEEE Trans. Electron Devices*, vol. 46, pp. 2210–2216, Nov. 1999.

Casper Lageweg (S'00) was born in Haarlem, The Netherlands. He received the M.Sc. degree in electrical engineering from the Delft University of Technology (T.U. Delft), Delft, The Netherlands, in 1998, and is currently working toward the Ph.D. degree in computer engineering ath T.U. Delft.

He was with Hewlett-Packard Laboratories, Bristol, U.K. He is currently with the Computer Engineering Laboratory, T.U. Delft. His research interests include nanotechnology, nanoelectronics, SET, logic design, computer arithmetic, computer architecture, integrated circuits, and physical design.

Sorin Coţofană (M'97–SM'00) was born in Mizil, Romania. He received the M.S. degree in computer science from the Politehnica University of Bucharest, Bucharest, Romania, in 1984, and the Ph.D. degree in electrical engineering from Delft University of Technology (T.U. Delft), Delft, The Netherlands, in 1998.

For a decade, he was with the Research and Development Institute for Electronic Components (ICCE), Bucharest, Romania, where he was involved with structured design of digital systems, design rule checking of integrated-circuit layout, logic and mixed-mode simulation of electronic circuits, testability analysis, and image processing. He is currently an Associate Professor with the Electrical Engineering Department, T. U. Delft. His research interests include computer arithmetic, parallel architectures, embedded systems, nanotechnology, reconfigurable computing neural networks, computational geometry, and computer-aided design.

Stamatis Vassiliadis (M'86–S'92–F'97) was born in Manolates, Samos, Greece.

He is currently a Chair Professor with the Electrical Engineering Department, Delft University of Technology (T.U. Delft), Delft, The Netherlands. He has also served on the electrical engineering faculties of Cornell University, Ithaca, NY, and the State University of New York (SUNY), Binghamton, NY. For a decade, he was with the Advanced Workstations and Systems Laboratory, IBM, Austin TX, the Mid-Hudson Valley Laboratory, Poughkeepsie, NY, and the Glendale Laboratory, Endicott, NY. While with IBM, he was involved in numerous projects regarding computer design, organizations, and architectures and the leadership to advanced research projects. A number of his design and implementation proposals have been implemented in commercially available systems and processors including the IBM 9370 model 60 computer system, the IBM POWER II, the IBM AS/400 models 400, 500, and 510, Server models 40S and 50S, the IBM AS/400 Advanced 36, and the IBM S/390 G4 and G5 computer systems. Six of his patents have been rated with the highest patent ranking at IBM, and in 1990, he was awarded the highest number of patents at IBM (70). His research interests include computer architecture, embedded systems, hardware design and functional testing of computer systems, parallel processors, computer arithmetic, neural networks, fuzzy logic and systems, and software engineering.

Dr. Vassiliadis is a member of the IEEE Computer Society. He was the recipient of numerous awards including 24 levels of Publication Achievement Awards, 15 levels of Invention Achievement Awards, and an Outstanding Innovation Award for Engineering/Scientific Hardware Design in 1989.