Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2008

MSc THESIS

Single Electron Tunneling Based Building Blocks for Delay Insensitive Circuits

Saleh Safiruddin

CE-MS-2008-02

Abstract

The computer performance demands of the modern era are being met by CMOS technology thanks to continuous scaling. However, as fundamental physical limits are being approached alternative technologies are currently under investigation. Single Electron Tunneling (SET) technology is such a promising and elegant alternative. In SET technology, individual electrons can be utilized as computation vehicles, with the electron transport being controlled through the quantum tunneling effect. However, the tunneling of electrons is stochastic in nature resulting in indeterministic delays, which restricts the direct utilization of SET technology for the design of synchronous circuits. As an alternative, asynchronous circuits can be utilized. In this thesis we investigate the implementation of asynchronous computational paradigms in SET technology. We research two computational paradigms: delay-insensitive and fluctuation-based computations. The first paradigm is based on circuits that are insensitive to signal delays. They are thus free of any timing restrictions that may arise due to the SET stochastic behavior. The second paradigm goes one step further and allows its signals to be affected by fluctuations. For both paradigms we propose implementations for a set of basic building blocks, from which a universal set of primitives can be selected. Networks constructed from these delay-insensitive building blocks are presented and investigated. The fluctuation-based paradigm utilizes deterministic and fluctuation-driven blocks. We investigate the possibility of driving the signal fluctuations by means of thermal energy in SET circuits and demonstrate that the fluctuation-driven blocks and the deterministic blocks can coexist at the same temperature. All the proposed circuit topologies and networks are verified by means of simulations.

TU Delft
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Single Electron Tunneling Based Building Blocks for Delay Insensitive Circuits

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Saleh Safiruddin
born in Karachi, Pakistan

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Single Electron Tunneling Based Building Blocks for Delay Insensitive Circuits

by Saleh Safiruddin

Abstract

The computer performance demands of the modern era are being met by CMOS technology thanks to continuous scaling. However, as fundamental physical limits are being approached alternative technologies are currently under investigation. Single Electron Tunneling (SET) technology is such a promising and elegant alternative. In SET technology, individual electrons can be utilized as computation vehicles, with the electron transport being controlled through the quantum tunneling effect. However, the tunneling of electrons is stochastic in nature resulting in indeterministic delays, which restricts the direct utilization of SET technology for the design of synchronous circuits. As an alternative, asynchronous circuits can be utilized. In this thesis we investigate the implementation of asynchronous computational paradigms in SET technology. We research two computational paradigms: delay-insensitive and fluctuation-based computations. The first paradigm is based on circuits that are insensitive to signal delays. They are thus free of any timing restrictions that may arise due to the SET stochastic behavior. The second paradigm goes one step further and allows its signals to be affected by fluctuations. For both paradigms we propose implementations for a set of basic building blocks, from which a universal set of primitives can be selected. Networks constructed from these delay-insensitive building blocks are presented and investigated. The fluctuation-based paradigm utilizes deterministic and fluctuation-driven blocks. We investigate the possibility of driving the signal fluctuations by means of thermal energy in SET circuits and demonstrate that the fluctuation-driven blocks and the deterministic blocks can coexist at the same temperature. All the proposed circuit topologies and networks are verified by means of simulations.

Laboratory : Computer Engineering
Codenumber : CE-MS-2008-02

Committee Members :

Advisor:	Sorin Coţofană, CE, TU Delft
Chairperson:	Koen Bertels, CE, TU Delft
Member:	Jaap Hoekstra, ELCA, TU Delft
Member:	Stephan Wong, CE, TU Delft

Contents

List of Figures

List of Tables

Acknowledgements

I would firstly like to thank my parents. It was due to their continued efforts and sacrifices that I had the opportunities to further my education. I am where I am now because of them, and I'm very grateful for the love and support they have given me over the many years.

Also, I would like to thank all my three sisters, who have continually been there for me, and have always been ready to provide help in any way they could.

I would like to express my gratitude to my advisor Sorin Coţofană, who right from the start of my Masters project had a great deal of confidence in me and my work and remained to do so throughout. He has made my Masters project a very enjoyable experience and I have learned a great deal from him. I am also very thankful for all the time he put into reviewing my research papers and my thesis.

I would like to extend a special thanks to Wati Chaeron, who has throughout my entire Masters project kept my spirits up and has shared many joyful moments.

Saleh Safiruddin
Delft, The Netherlands
June 6, 2008

1 Introduction

"Progress in science depends on new techniques, new discoveries and new ideas, probably in that order."
- Sydney Brenner

"In science one must search for ideas. If there are no ideas, there is no science. A knowledge of facts is only valuable in so far as facts conceal ideas: facts without ideas are just the sweepings of the brain and the memory."
- Vissarion Grigorievich Belinskii

"Ce qui donne du courage, ce sont les idées."
(That which gives people courage are the ideas.)
- Georges Clemenceau

The demand for increased computing power has been unwavering throughout the past four decades. This has been due to mankind having developed an insatiable desire to perform increasingly complex and large calculations as fast as possible. With the advent of integrated circuit technology it became advantageous to use processors in a myriad of applications, ranging from simulating the real world, to robotics, and to data processing. Now, in many fields, greater processing power means direct advantages in terms of size, cost, and quality. The wide range utilization of processors has spurred on new ideas for applications for which still greater and faster computations are required. This also leads to more stringent constraints being put on devices and the processors behind them, it thus sustaining the demand for increased performance.

1.1 Problem Statement

The rate at which the performance of processors has increased has remained constant since the 1970's. In 1965, Gordon E. Moore formulated a prediction related to the development of integrated circuits, which he updated in 1975, effectively stating that the processing power would double every 18 months. This prediction still holds true till today, and is known as *Moore's Law* [21, 22].

CMOS technology emerged as the transistor technology which sustained the performance improvements predicted by Moore and today it is the dominant technology for digital circuits having won out against the rest. Out of the numerous ways by which CMOS technology gains improvement, scaling is the most important method. Continuous scaling to smaller dimensions has resulted in higher package densities, lower power dissipation, faster circuits, and lower cost.

It is generally accepted that eventually, CMOS technology will hit fundamental physical limits and scaling will no longer be possible. Currently, theoretical limits are being addressed, and it is expected that channel lengths of $22 - 32nm$ can be achieved within the next decade [2]. Scaling into the sub-$22nm$ range is predicted to be extremely difficult, if not even impossible, as undesired effects would dominate. The most notable of these effects is the quantum effect, which allows electrons to penetrate potential barriers by the so-called tunneling phenomena. This is already one the sources of error and power consumption due to leakage in current CMOS technology nodes.

With CMOS technology appearing to approach fundamental limits other alternative technologies are emerging. These include Molecular devices, Spintronics, Ferromagnetic logic, and Single Electron Tunneling technology [1].

Single Electron Tunneling (SET) technology appears to be a promising and elegant alternative, exhibiting very low power consumption and scalability. In SET technology calculations can be potentially performed by using singular or few electrons with electrons being controlled by the quantum tunneling effect. Thus, the quantum effect, which is actually a limiting factor in current CMOS technology, is utilized efficiently as the basis of computations. With single electron transport driving computations, the power consumption is reduced to very low levels.

To effectively utilize the unique features of the SET technology and the behavior of single electrons new and alternative architectures have been proposed. In recent years research has been conducted in achieving arithmetic computations by utilizing SET technology with Boolean gates and other SET specific novel building blocks. Two such examples are Electron Counting and Single Electron Encoding Logic [11, 19].

While these approaches are promising in terms of area and power consumption implementing synchronous circuits in SET technology is still a challenge. The tunneling mechanism on which SET technology is based is of a stochastic nature, arising from the probabilistic behavior of electrons, and results in undetermined gate delays. Additionally, at the nanometer scale, a synchronization signal, i.e., a clock, would take up an increasingly large proportion of the chip area as scaling is applied and would be difficult to route. Furthermore, background charge and fluctuations result in the unpredictable behavior of electrons. The realization of synchronous circuits in SET technology thus puts high limitations on the underlying technologies and requires aggressive error-correction and fault-tolerance schemes.

1.2 Objectives and Contributions

An effective way to deal with the problems related to implementation of sychronous circuits in SET technology, is to eliminate the requirement of a global clock altogether.

The purpose of this thesis is to explore the possibility of realizing hardware support for asynchronous computational paradigms in SET technology. Two asynchronous paradigms are investigated in this respect. The first one is utilizing Delay-Insensitive Circuits (DIC) [27]. The second one utilizes fluctuation-based calculations and requires so-called Brownian Circuits [24].

It has been proven that there exist groups of universal delay-insensitive functional building blocks for these two paradigms from which any conceivable circuit functionality can be realized [23, 16, 24].

Related to this investigation we address the following research questions:

- Can we design SET-based circuit implementations of building blocks for the delay-insensitive paradigm?
- Can we form large networks composed out of such SET-based building blocks?
- Can we design SET-based circuit implementations of building blocks for the Brownian circuit paradigm?
- Can fluctuation-based building blocks coexist with deterministic building blocks? If so, can we form large networks composed out of such SET-based building blocks?

In delay-insensitive systems, circuits function correctly regardless of any delays in gates or wires. This is very suitable for SET technology since with no synchronization signal required one of the largest drawbacks of SET technology, its stochastic behavior, can be tolerated. Calculations have to be performed based on arriving signals rather than on absolute values as it is the case with conventional logic. These signals can either be represented directly by physical 'tokens' moving around the circuit or by the transitions of line values.

Given that in SET technology the unique ability to control individual electrons exists, we aim to efficiently utilize the transport of single electrons to arrive at the desired functionality of each delay-insensitive building block.

Thus, for the Delay-Insensitive (DI) paradigm building blocks we utilize input and output voltage levels that result from the presence or absence of single electron charges. Input and output tokens are represented in terms of the transition of these input and output voltage levels. The outputs are controlled by means of the transport of electrons to and from output nodes. We utilize the input voltage levels to transport single electrons around the circuit such that the desired output voltage levels result. Based on the required functionality SET topologies are proposed that could potentially transport electrons around in a manner that would result in the correct output voltage levels.

For the Brownian circuits paradigm we represent input and output signals by a single electron charge. An input arriving at one of the building blocks is in the form of a quantized charge. An output generated by one of the building blocks is the same quantized charge. A charge is placed at the output based on the functionality of the block and charges arriving at the inputs. Based on the required functionality SET topologies are proposed that could potentially transport this charge around the circuit in a manner that would result in a charge available at the output.

The main contributions of this thesis are as follows:

- We proposed circuit topologies for the implementations in SET technology of the *C-Element*, the *Merge*, the *Sequencer*, the *Tria*, and the *Toggle*, which all function as DI building blocks.

- We proposed buffering techniques to enable network constructions using SET-based DI building blocks.
- We presented examples of networks using the proposed buffering techniques to connect SET-based DI building blocks together. These networks were verified by means of simulations.
- We proposed circuit topologies for the implementations in SET technology of the *Hub* and the *Conservative Join*, which constitute a universal set of building blocks for the Brownian circuit paradigm.
- We demonstrated that fluctuation-driven building blocks can coexist with deterministic building blocks.
- We proposed a method to form larger networks out of contructions using SET-based Brownian circuit building blocks.
- We presented an example of a network using the proposed method to connect the SET-based Brownian circuit buildings blocks together. The network was verified by means of simulations.

1.3 Overview

The organization of this thesis is as follows. In Chapter 2 the relevant background theory of SET technology is presented. In the same chapter the delay-insensitive computational paradigm is introduced and components which can form building blocks for such systems are presented and shortly described. The theory behind fluctuation-based computation is also presented. In Chapter 3 SET based implementations of the building blocks for the DI paradigm presented in Chapter 2 are presented and discussed. Buffering techniques are also introduced to enable the construction of larger networks, and subsequently, examples of networks which can be constructed from these building blocks are presented and discussed. In Chapter 4, the SET based implementations of the building blocks for the Brownian circuits paradigm are presented and discussed. The method by which to connect these building blocks together is introduced, and subsequently, examples of networks which can be constructed from these building blocks are presented and discussed. The two paradigms are then evaluated and a preliminary comparison is made between SET circuit implementations of a *NAND*-gate in each paradigm. Finally, the conclusions are presented in Chapter 5 along with recommendations for future research.

2 Background

"There is nothing more practical than a good theory."
- Leonid Ilich Brezhnev

"It is now generally assumed that matter is not continuous but coarse grained, i.e., that matter is composed of atoms which are practically indivisible and are situated at very small but not infinitesimal distances apart."
- Harold Hilton (1903)

"The electron is not as simple as it looks."
- [Sir] William Lawrence Bragg

In this chapter some background theory on Single Electron Tunneling technology and Delay-Insensitive Circuits is provided. In Section 2.1 we present only the theory that is relevant for the functional behavior of SET devices and for an in depth discussion on the physics of SET devices the reader is referred to [28]. In Section 2.2 we present methods by which Delay-Insensitive Circuits communicate and transfer data. Also in this section we present a method by which delay-insensitive functionality is specified, called Trace Theory, and present a set of building blocks from which a universal computational set can be selected for the DI paradigm. In the last part of this section we introduce the theory describing Brownian motion. Finally, in Section 2.3 an overview of the chapter is presented as well as how we utilize the theory in the rest of the thesis.

2.1 SET Theory

Initially, the electron was treated in classic physics theory as a particle. In this theory, the electron was modeled as a moving charge following the then known laws of classical physics. According to this theory, if an electron approached an insulating potential barrier it would never cross that barrier as it wouldn't have enough energy to overcome the energy band gap and would simply reflect back. However, during the 1920's new insights were found about the characteristics of electrons by L. de Broglie [6] and Schrödinger, which culminated in the formulation of the Schrödinger wave equation. It was discovered that electrons didn't behave as simply as it was first thought. In some instances they behaved as waves. This behavior was described by the Schrödinger wave equation, and it is the basis for quantum mechanics. What this actually meant for the electron was that there was in fact a non-zero probability that the electron would tunnel through the potential barrier, but only if the electron's final energy state would be lower than it

was behind the potential barrier. This phenomenon is known as the quantum tunneling effect. The junction over which the electrons will or will not tunnel is called a quantum tunnel junction and this is a new circuit element, depicted in Figure 2.1. Tunnel junctions together with capacitors are the basic building blocks in SET based circuits. Since

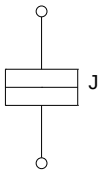


Figure 2.1: Circuit symbol for SET junction

electron tunneling is a quantum mechanical process, an electron's wave function extends through potential barriers, and the electron charge is spread over the capacitive islands in a SET circuit. If this effect was to prevail there would be no localized charges and computations using electrons would not be possible. To ensure that the charge of an electron is quantized on each specific island the tunnel junctions must have a sufficiently high tunneling resistance, so that the charging energy, also called the Coulomb energy, dominates over the quantum charge fluctuations. This can be expressed as:

$$\frac{{q_e}^2}{2 \cdot C_j} \cdot R_j \cdot C_j >> h => R_j >> h/{q_e}^2 = 25.8k\Omega,$$

where h is Planck's constant, C_j is the tunnel capacitance, and R_j is the tunneling resistance. Therefore, the resistance of all the tunneling junctions in this research is chosen as 100kΩ, which has also been used in previous studies, [4, 13, 12, 3, 19].

Whether a tunneling event occurs or not depends on whether the amount of free energy in the circuit after the tunneling event decreases or increases. If the free energy would decrease due to a tunneling event then that tunneling would occur after some time, bringing the circuit to a lower energy state. It is thus possible to calculate the tunneling behavior of electrons by determining the free energy before and after a potential tunneling event.

The method of determining whether tunneling occurs or not using free energy calculations, however, becomes increasingly complicated as the amount of circuit elements increases and it becomes prohibitive when performing calculations for circuits with an amount of elements required to create useful logic circuits. However, by introducing the concept of a critical voltage (Coulomb gap) the calculation of determining the possibility of a tunneling event occurring is reduced to the comparison of the applied voltage to a tunneling junction to its critical voltage. The change in free energy ΔE can be expressed as:

$$\Delta E = -q_e(|V_j| - V_c), \tag{2.1}$$

where V_j is the applied voltage over the tunneling junction and V_c is the critical voltage of the junction.

To determine the critical voltage of a junction, the capacitance of the circuit as seen from the tunneling junction has to be determined. This is done by using the Thevenin

equivalent of the circuit. All voltage sources and charges on nodes are taken as zero and the voltage sources are replaced by grounded wires and the capacitive equivalent is then calculated. To clarify the procedure let us assume the circuit depicted in Figure

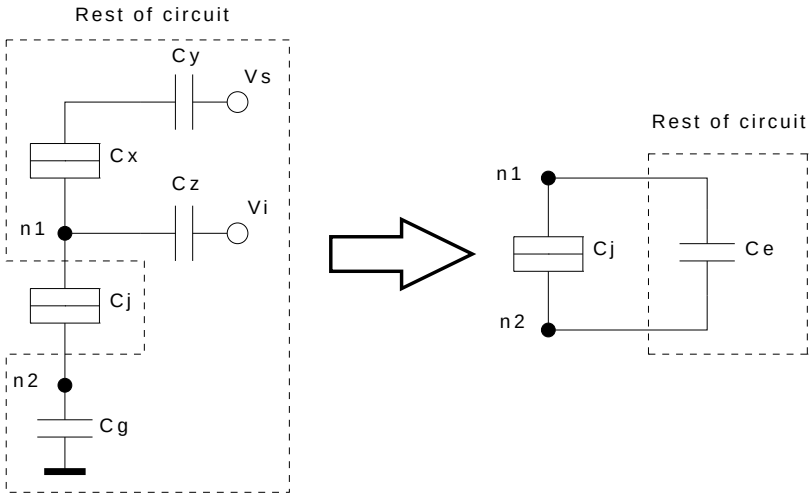


Figure 2.2: SET junction equivalent

2.2 and calculate the critical voltage for the tunnel junction with capacitance C_j. Using Kirchoff's laws, the capacitance between nodes $n1$ and $n2$ is calculated for the rest of the circuit. This results in the equivalent capacitance, C_e. For this example case C_e can be expressed as follows:

$$C_e = \frac{C_g\left(\frac{C_xC_y}{C_x+C_y} + C_z\right)}{C_g + \frac{C_xC_y}{C_x+C_y} + C_z}. \tag{2.2}$$

The critical voltage of the considered tunneling junction can then be expressed as:

$$V_c = \frac{q_e}{2(C_j + C_e)}, \tag{2.3}$$

where $q_e = 1.602 \cdot 10^{-19} C$, C_j is the capacitance of the junction, and C_e is the capacitive value of the remainder of the circuit as seen from the junction, calculated as in Equation (2.2). If $|V_j| \geq V_c$ then tunnel events will occur through the junction until the condition does not hold anymore, in which case the junction is said to be in Coulomb blockade. The tunneling rate can be expressed as:

$$\Gamma = \frac{|V_j| - V_c}{q_e R_j}.$$

To calculate the delay due to tunneling we have to use a probabilistic model, since tunneling is stochastic in nature. If we take P_{errror} to be the probability that no charge has been transported after t_d seconds then:

$$P_{error} = e^{\Gamma t_d}.$$

We can then express the minimum switching delay t_d as follows:

$$t_d = \frac{-\ln P_{error}}{\Gamma} = \frac{-\ln (P_{error}) q_e R_j}{|V_j| - V_c}. \tag{2.4}$$

In this thesis, consistent with previous research, P_{error} is taken to be 10^{-8}.

The second effect that has to be considered is that of thermal energy. If the thermal energy dominates over the charging energy, E_c, the quantization effects become again non-observable. For temperatures above $0K$ there is always a probability that an electron will tunnel through a junction even though the critical voltage condition is not met. The error probability due to tunneling caused by thermal effects can be expressed as: $P_{therm} = e^{\frac{-E_c}{k_B T}}$. In order to ensure that thermal effects do not dominate the charge quantization, the following condition should thus hold:

$$E_c = \frac{{q_e}^2}{2 \cdot C} >> k_B \cdot T,$$

where k_B is Boltzmann's constant and T is the absolute temperature.

2.2 Delay-Insensitive Circuits Theory

> *"Life is all about timing... the unreachable becomes reachable, the unavailable become available, the unattainable... attainable. Have the patience, wait it out. It's all about timing."*
>
> - Stacey Charter

For the past few decades great strides have been made with the conventional synchronous computing paradigm utilizing a global clock. However, in recent years asynchronous computation is attracting significant interest once again. As current technology scales down synchronous design becomes increasingly more difficult; wire delays become more prominent, clock-related power consumption becomes prohibitively large and the global clock wiring takes up an increasing proportion of the chip area. Additionally, with many new emerging technologies, which are candidates to replace the current CMOS technology, small size and robustness are issues which exclude the utilization of a global clock.

There are many possible benefits of implementing asynchronous circuits [5, 10], and within the context of SET technology these become more pronounced.

- *No clock skew* - The problem of *clock skew* is eliminated altogether. The clock skew becomes even more of a concern as feature size decreases, so not having to deal with it for SET implementations is very desirable.

- *No clock circuitry* - All the additional area required for the clock circuitry is saved. For SET technology this would account for a significant proportion of the area. Furthermore, at the nanometer scale the fabrication of a reliable global clock circuit would be difficult.

- *Lower power consumption and dissipation* - For very high component densities the power dissipation becomes extremely large. Without having a continuously oscillating signal spread over the entire circuit, i.e., the clock, switching unused circuitry, a lot of power is saved. Additionally, with asynchronous circuits normally only the parts of the circuit that are relevant for a computation involve transitions and draw power.

- *Average instead of worst-case performance* - A synchronous circuit has to accommodate the slowest part of the circuit by switching only when all possible computations are complete, resulting in a worst-case delay. An asynchronous circuit on the other hand, can sense the moment a computation is complete and so has the potential to exhibit average-case performance.

- *Technology migration and modularity* - Asynchronous components can be designed independently and simply combined, as long as basic design constraints are met. Also, since performance depends on critical components, these can be migrated to better technologies for performance improvements, leaving the rest of the design as is.

- *Robustness to physical variations* - Any changes in physical properties or environment resulting in changes in delays of the circuitry can dramatically affect synchronous design but they have no influence on the correctness of asynchronous circuits operation.

- *Lower interference and noise production* - Synchronous circuits generate peaks in the radio frequency transmission due to the circuitry switching at the same frequency. Asynchronous circuits cause a more distributed spectrum of radio frequency transmission, since the switching is not coordinated at one frequency, resulting in significantly lower interference and electromagnetic noise.

Circuits which contain no global clock can be classified based on the kind of delay assumptions are made [5]. The most general category is called *asynchronous circuits* with no global clock, but timing assumptions being made for both circuits and the interaction with the environment. Another class includes *Speed-independent* circuits which can have gate delays but assume no wire delay. *Self-timed* circuits contain regions having delay-insensitive communication but are assumed to have no, or well-bounded wire delay internally and contain *speed-independent* circuits or circuits requiring local timing assumptions. A *delay-insensitive* circuit is one where gate and wire delays do not affect the correct functioning of the circuit. This is the most robust class with no timing assumptions being made about the arrival of signals, as long as the signal does eventually arrive.

Considering that in SET technology tunneling is a stochastic process and that tunneling times are unknown, and taking into account all of the above mentioned advantages of asynchronous circuits, one can observe that it is most advantageous to implement SET circuits which function in a *delay-insensitive* manner.

2.2.1 Delay Insensitive Communication

Since we are dealing with circuits where it is not possible to differentiate between a value having arrived but being the same value, or the value simply being late, we have to use signal *arrival* as the basis of computations. A component should thus always wait for the arrival of a signal before proceeding. A signal arrival is defined as a change of state or value of a line and it is referred to as a *token*. This is in contrast to the conventional *Boolean* logic where absolute signal levels are utilized.

In conventional *token-based* DI communication, input line value *transitions* are utilized to represent *tokens*. The lines have two states, '0' and '1', and a transition from '0' → '1' or '1' → '0' represents a *token*. An alternative, more natural way to represent *tokens* is by means of physical elements which can travel around a circuit. Turning our attention to SET technology we can see that using electrons to represent *tokens* is very intuitive.

Handshaking Protocols

When components communicate with each other, components sending *tokens* will need to know when a *token* has been received by another component so the rest of the computation can proceed. Therefore, some form of handshaking has to be implemented. This is usually achieved by using *request* and *acknowledge* signaling protocols. The most widely used schemes are *2-phase handshaking* and *4-phase handshaking*. In Figures 2.3 and 2.4 the sequences of signal transitions for a 2-phase and a 4-phase handshaking protocol are presented, respectively, where the *Master* requests the *Slave* to perform a certain action. The *Master* sends a signal transition or *token* on the *request* line, after which the *Slave* performs the action and sends a signal transition back on the *acknowledge* line, once the action is complete. The process then repeats itself. The only difference between the two protocols is that in 4-phase handshaking the values of the *request* and *acknowledge* lines are returned to '0' ready for the next cycle while in 2-phase handshaking the values remain '1' so in the next cycle the transition from '1' → '0' is used. The 4-phase handshaking may seem to require more communication time but the encoding and decoding of the signals at either side are much simpler than for 2-phase handshaking, and a choice has to be made depending on the computations being performed.

Data Transfer

Data transfer can be achieved by utilizing the above mentioned handshaking protocols, in combination with data encoding schemes with which data can be passed in a delay-insensitive manner [26].

Conceptually, the simplest method is the *One-Hot* code, where n lines are used to transmit n unique messages. Figure 2.5 depicts a Sender and a Receiver module communicating with the *One-Hot* code. Each of the 4 lines, *I0* to *I3*, represents a message which the Receiver can decode.

4-phase handshaking is the most appropriate protocol to apply for the *One-Hot* coding scheme, because the receiver then only has to deal with decoding n different line

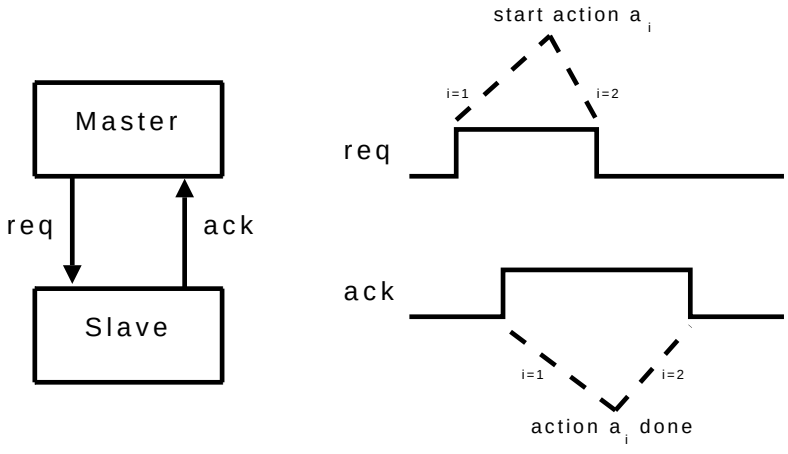


Figure 2.3: 2-phase handshaking protocol

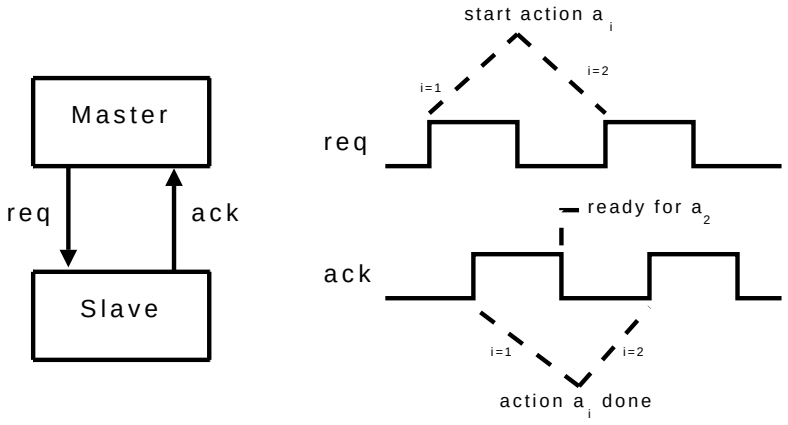


Figure 2.4: 4-phase handshaking protocol

states, whereas with 2-phase handshaking there would be 2^n input combinations to accommodate.

The most widely used coding technique is the *Dual-rail* code, where two tracks are used to represent one bit of data. One track is used to signal a '0' value and the other track is used to signal a '1' value. The receiver can thus know exactly when a bit of data has been transmitted and can then send an acknowledge signal back. Figure 2.6 presents a Sender and Receiver module communicating with the *Dual-rail* code. Data are sent using pairs of wires for each bit, *I0(0)* and *I0(1)* for the first bit, and so on. The Receiver waits until a transition has occurred on one wire in each pair, then performs the action and sends back an *acknowledge* signal. *2n* wires are needed to send 2^n unique messages with this scheme. For $n > 2$ *Dual-rail* coding has a marked improvement over *One-Hot* coding in terms of the number of wires required. *One-Hot* coding is able to

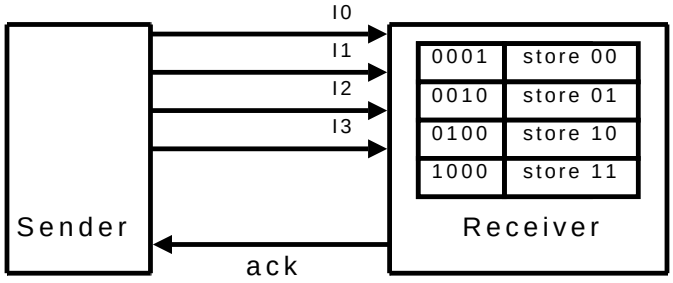


Figure 2.5: Data transfer using One-Hot code

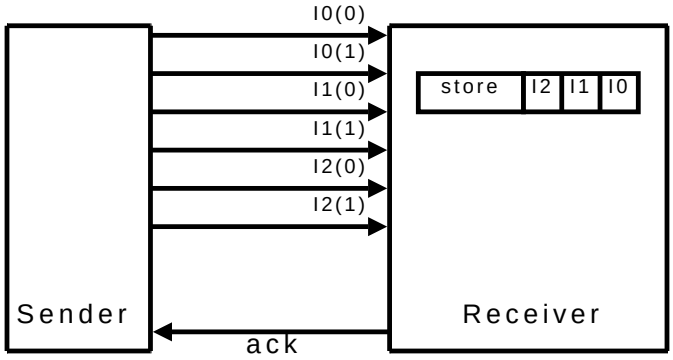


Figure 2.6: Data transfer using Dual-rail

send only n unique messages on n wires while *Dual-rail* coding is able to send $2^{\frac{n}{2}}$ unique messages on the same amount of wires.

For data transmission the *Dual-rail* coding can be used with either of the two handshaking protocols. With 4-phase handshaking the encoding presented in Table 2.1 would result. After each data transfer the inputs reset to '0' putting the communication back into the idle state ready for the next transfer.

State	Inputs	Encoding
1	00	Idle, data not ready
2	01	Valid 1
3	10	Valid 0
4	11	Illegal

Table 2.1: Dual-rail Encoding for 4-phase handshaking

With 2-phase handshaking the encoding is more complex. Whether a '0' or '1' is detected depends on the new input levels as well as on the previous input levels at the

moment the data is transferred. All the possible states are listed in Table 2.2 with the corresponding possible previous input combinations for each input combination. We can see that the input combination '00' can represent a '0' or '1' depending on whether it was '10' or '01' in the moment before '00' was detected.

For both protocols simultaneous signals on both inputs are prohibited.

State	Inputs	Previous Inputs	Encoding
1	00	01	Valid 1
2	00	10	Valid 0
3	01	00	Valid 1
4	01	11	Valid 0
5	10	00	Valid 0
6	10	11	Valid 1
7	11	01	Valid 0
8	11	10	Valid 1

Table 2.2: Dual-rail Encoding for 2-phase handshaking

There are also DI-codes that make much more efficient use of the amount of tracks available but use involved encoding schemes, such as *Sperner*, *Berger*, and *Knuth* codes. For example, with 20 lines *One-Hot* code can send 20 unique messages, *Dual-rail* code can send 1024, but *Sperner* can send 184,756. An in-depth analysis of DI-coding schemes can be found in [26].

2.2.2 Trace Theory

In contrast to synchronous circuit components, delay-insensitive circuit elements have to be described by defining the allowed sequences of input and output signals. This is necessary because, as mentioned before, the components function based on the arrival of signals rather than on the value of the signals themselves. The behavior of DIC components can be represented by using *trace theory*, which has been proposed by Ebergen in [8] and [9]. In *trace theory*, a component is described by means of the sequence which its inputs and outputs follow. An input or output is defined as a change of state or value of a line.

The following is a summary of the utilized *trace theory* notation:

- An input token is denoted with a lower case letter followed by a '?',
- An output token is denoted with a lower case letter followed by a '!',
- '**pref**' denotes prefix-closure, indicating that any prefix of a permitted behavior is also permitted,
- ';' denotes sequential composition,
- '||' denotes parallel composition,
- '*' denotes repetition,

- '|' denotes non-deterministic choice.

2.2.3 Universal Building Blocks

In [23] a selection of building blocks is presented from which delay-insensitive circuits can be constructed. In [15] and [16] a new class is proposed which allows for buffered and bidirectional lines, resulting in a reduced amount of required input and output lines.

Any schemes and protocols that are developed for delay-insensitive systems can be decomposed into these primitives and implemented. From these building blocks a *universal* set of primitives can be selected, which means that by combining such building blocks the same computations can be performed as in conventional computers.

We shall focus on delay-insensitive circuits with the no-buffering condition, which serves as a good starting point for further efforts to implement also the buffered and bidirectional variants.

A set of delay-insensitive circuit building blocks from which a universal set can be selected is as follows, along with the *trace theory* specification of the building blocks:

- *Wire*: $\mathbf{pref}[a?; b!]*$
- *Fork*: $\mathbf{pref}[a?; (b!||c!)]*$
- *Merge*: $\mathbf{pref}[(a?|b?); c!]*$
- *C-Element*: $\mathbf{pref}[(a?||b?); c!]*$
- *Sequencer*: $\mathbf{pref}[a?; ao!] ||\mathbf{pref}[b?; bo!] ||\mathbf{pref}[c?; (ao!|bo!)]]*$
- *Tria*: $\mathbf{pref}[((a?||b?); d!)|((b?||c?); e!)|(a?||c?); f!)]*$
- *Toggle*: $\mathbf{pref}[a?; b!; a?; c!]*$

2.2.4 Brownian Motion Theory

Brownian motion can be described as the random motion of suspended particles. In SET technology, as mentioned, at temperatures above 0 Kelvin, electrons have a non-zero probability of tunneling through a junction, even though the critical voltage condition is not met. The tunneling is caused by thermal fluctuations. This leads to a random movement of electrons through a SET circuit, which eventually distribute themselves evenly. This effect has been utilized in works such as [17] where the Brownian motion effect has been utilized to create Quantum Rachets.

Due to the difficulties at the nanometer scale to have stable circuitry without any signal fluctuation recent research has been conducted by F. Peper et al. to exploit the signal fluctuations in SET logic and arithmetic circuits [14, 24]. This approach finds its ancestor in an earlier proposal that employs signal fluctuations actively in a simulated annealing scheme for Boltzman machines [30]. Being focused on neural network implementations, however, this proposal has found no follow-up in the context of traditional arithmetic circuits.

In as yet unpublished work, two building blocks operating on a similar principle have been proposed in the context of the so-called *Brownian circuits* [24, 14]. The idea of Brownian circuits is to use fluctuations to guide signals through a circuit. That is, fluctuations drive a search process through a Brownian maze formed by the topology of the circuit [24]. The two proposed blocks, *Hub* and *Conservative Join* have been shown to form a computationally universal set from which any desired functionality can be constructed [14]. SET technology is especially suited for implementing the token-based nature of this architecture, with tokens represented by localized charges.

A detailed analysis of Brownian motion is outside the scope of this thesis and for an in depth exposition of Brownian motion the reader is referred to [7].

2.3 Conclusion

In this chapter we have presented the theory for SET technology and delay-insensitive circuits on which we base our implementations. In the next chapter we first utilize the DI theory to specify the desired functionality of the building blocks. We then use the SET theory presented in this chapter to arrive at SET topologies which can exhibit the required functionality.

SET Based Delay-Insensitive Circuits

3

"Nature is not a temple but a workshop in which man is the labourer."

- Ivan Sergeievich Turgenev

"Life is a partial, continuous, progressive, multiform and conditionally interactive, self-realisation of the potentialities of atomic electron states."

- John Desmond Bernal

In this chapter we present SET implementations of delay-insensitive circuits. In Section 3.1 we propose SET-based circuit implementations of building blocks for the DI paradigm. In Section 3.2 we present buffering techniques to connect the building blocks together to form larger networks. We then present examples of networks utilizing the building blocks and the buffering techniques. In Section 3.3 we evaluate and discuss the proposed implementations and networks. Finally, in Section 3.4 we provide an overview of the chapter.

3.1 Implementations of DI Building Blocks

In Chapter 2, seven components which serve as building blocks for the DI paradigm were introduced. The objective of this research is to arrive at SET implementations of these building blocks. In this section we present these SET implementations and how they were systematically found.

For the DI paradigm we chose to represent input and output tokens as *value transitions*. The value is represented by a voltage level resulting from the presence of a single electron charge at a certain node. Applying this representation of tokens, the implementations of the first two components, the *Wire*, and the *Fork* are rendered trivial. Thus we just focus on the implementations of the remaining five building blocks: *C-Element*, *Merge*, *Sequencer*, *Tria*, and *Toggle*.

In the trace theory section, the blocks are specified based on the arrival and the generation of tokens. If we represent tokens based on value *transitions* then the blocks are required to remember the value of the input lines and output lines to generate a correct output. For that reason, we first converted the token-based description into sequences of state transitions, based on the signal values, in the form of state graphs.

In state graphs each state remembers whether the input and output signals are low or high. If a certain signal is already high and the signal goes low a state transition may result. We invert the signal and utilize that value, which goes high when the original signal goes low, to determine whether the state graph should transition to the next state

or not. Each state knows what the value of all the outputs is, and if from that state a combination of inputs leads to an output, that output is made low if it was high and vice versa in the next state. The state graphs are constructed as follows:

- Each state graph consists of a set of nodes drawn as circles, which represent memory states of the component.
- Input signals are denoted by letters next to arrows, and can take on the values 'high' or 'low'. A letter with an appended '^ ' denotes the inverted value of an input signal.
- Output signals are denoted by letters placed on nodes. A letter placed on a node means the output is 'high' in that state, while the absence of an output signal on a node means the output is 'low' in that state.
- Arrows denote possible transitions between states, and are labeled with the signals which can cause transitions. An arrow labeled by two letters means that both the input signals have to be 'high' for a state transition to take place.
- Two types of arrows are used for transition between states.
  - The single-headed arrow denotes a non-reversible transition. If the relevant inputs go 'low' again the state doesn't transition back.
  - The double-headed arrow denotes a reversible transition. If the relevant inputs all go 'low' again the state will transition back to the previous state.
- The solid arrowhead denotes the transition direction caused by 'high' valued labeled signals and the lined arrowhead shows the direction of transition if the labeled signals go 'low'.
- A state node is labeled with the number of the corresponding node in the proposed circuit topology which follows.

From the state graph representation of the block we propose a SET circuit topology which had the potential to deliver the required behavior. We then arrive at a detailed analysis of the behavior of electrons in that circuit based on which junctions electrons must tunnel through and at what moments.

Next we derive the circuit parameters. The methodology followed was based on the one proposed in [20]. First, the characteristic equations for junctions in the circuit were set up. Subsequently, the operating modes for each combination of inputs and internal states were determined and the equations were solved with the aid of certain assumptions. Finally, the circuit designs and parameters were verified by means of simulation, which were performed using SIMON 2.0, a SET circuit simulation software [29]. The simulations were performed at $0K$ to focus on the functional behavior of SET circuitry.

In the following sections, the implementations of each of the selected building blocks for the DI paradigm are presented as follows:

1. The building block is introduced with its DI symbol and a trace theory specification of its function.

2. The functional behavior of the block is described in terms of the arriving input tokens and the resulting output tokens.

3. The derived state graph is presented.

4. A SET-based circuit topology is proposed for the building block, based on a mapping from the state graph.

5. The behavior that the SET-based circuit should exhibit is described in terms of electron tunneling events.

6. The derivation of the circuit parameters to achieve the desired circuit functionality is presented.

7. The results of the circuit simulations are presented and discussed.

Considerations and Notations

For the implementations the following considerations and notations apply:

- For the following delay-insensitive components the environment has to wait for a pending output transition to occur before allowing new input signals being applied to the component.

- In the following sections when reference is made to the 'voltage' over a junction, we actually mean 'absolute voltage'.

- When a tunnel junction appears in a figure it is designated with a 'J' and a number. When circuit parameters are described, the capacitance of the tunnel junction is referred to with a 'C' and the same number.

- An input signal designated with an appended '^ ' means that an inverted signal is used, for example 'V_a^ '. This inverted signal is produced with an inverting buffer being applied to the original signal, as proposed in [13] and graphically depicted in Figure 3.1.

- The source voltage V_s is set at $16mV$ and logic '1' is encoded as $16mV$ and logic '0' as $0mV$, for consistency with previous work [11] so those components and component parameters can be reused.

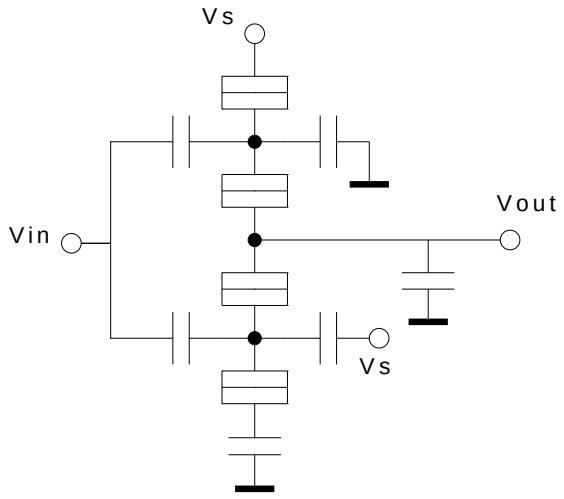
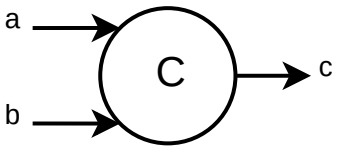


Figure 3.1: Inverter circuit diagram

3.1.1 C-Element

Symbol	Specification
	**pref**$[(a?\|\|b?);c!]*$

Functional Behavior

This is the most basic DI component. The component waits until two tokens have arrived, one on either input, before generating an output token at c. It is thus a state holding element. If a token is already waiting on one input, another token should be prohibited to arrive at that same input by the environment.

State graph

Based on the described behavior we can derive a state graph for the *C-Element*, presented in Figure 3.2. If we use value *transitions* to represent tokens then tokens arriving at a and b are represented by a and b going from 'low' $\rightarrow$ 'high' or from 'high' $\rightarrow$ 'low'. If both inputs and the output are initially low then the output, c, will go high when both inputs go high. In the state graph the initial state is denoted by node 1. The arrow is labeled with the two inputs a and b and when they both go high the new state denoted by node 2 results. This direction due to the inputs going high is shown by the filled arrowhead. Node 2 is labeled with the output c meaning that c is high at node 2. With both inputs

high, new tokens arriving at the inputs means that they go low. When both inputs go low then a token should be generated at c, meaning that c should also go low. In the state graph the direction due to the inputs going low is denoted by the lined arrowhead, showing the direction from node 2 to node 1. In node 1 there is no label, denoting that the output c is being low.

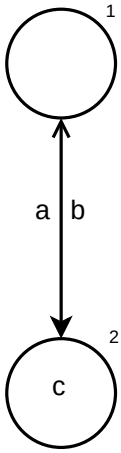


Figure 3.2: State graph for C-Element

Proposed SET-based Circuit Topology

In Figure 3.3 a SET circuit topology is proposed that is designed to provide the required behavior of a *C-Element*. The input signals are V_a and V_b and the output signal is V_c. The circuit comprises of 2 tunneling junctions and 4 capacitors.

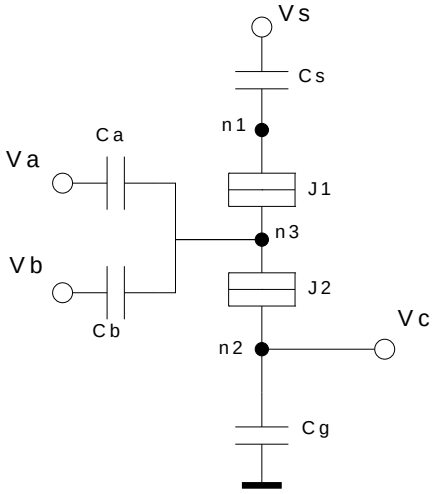


Figure 3.3: C-Element circuit diagram

Required Circuit Behavior

The circuit should operate as follows. If both input signals are low the circuit is in its initial neutral state and no charge transport should occur. If either one of the input signals undergoes a transition and becomes high the voltage over $J2$ increases but should not become critical yet. If the second signal also goes high then the voltage over $J2$ should become larger than its critical voltage so that an electron is transported from $n2$ to $n3$. This causes the voltage over $J1$ to exceed the critical voltage and the electron tunnels further from $n3$ to $n1$. The output is then high since its value is determined by the charge on $n2$. Subsequently, when either one of the inputs goes low the voltage over $J1$ decreases, while the voltage over $J2$ increases, but it should not go above the critical voltage. Only when the second signal also goes low should the voltage over $J2$ exceed the critical voltage causing an electron to be transported from $n3$ to $n2$ leaving a positive charge on $n3$. This charge on $n3$ should cause the voltage over $J1$ to exceed the critical voltage causing the electron residing on $n1$ to tunnel through into $n3$. This brings the circuit back to its initial charge neutral state.

Circuit Parameters Derivation

From the circuit behavior description we can observe that the circuit goes through certain states where the voltage conditions that must be met to achieve the required behavior depend on the input combinations as well as the amount of charge on each node in the circuit.

The possible input combinations, the amount of charge on each node, and the resulting voltage conditions that must be met for each junction are presented in tabular form in Table 3.1. For each circuit state, the amount of charge on each node is given in multiples of q_e.

Circuit state	V_a	V_b	n_1	n_3	n_2	$J1$	$J2$
1	0	0	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
2	1	0	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
3	0	1	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
4	1	1	0	0	0	$\|V_1\| < V_{c1}$	$V_{c2} < \|V_2\|$
5	1	1	0	-1	1	$V_{c1} < \|V_1\|$	$\|V_2\| < V_{c2}$
6	1	1	-1	0	1	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
7	1	0	-1	0	1	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
8	0	1	-1	0	1	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
9	0	0	-1	0	1	$\|V_1\| < V_{c1}$	$V_{c2} < \|V_2\|$
10	0	0	0	-1	1	$V_{c1} < \|V_1\|$	$\|V_2\| < V_{c2}$

Table 3.1: Critical voltage conditions for all circuit states of C-Element

To derive the circuit parameters from these conditions, first the voltages over the junctions and their critical voltages have to be determined.

To keep the equations readable the following shorthand notations are introduced:

$$
\begin{aligned}
C_\alpha &= C_a + C_b + \frac{C_2 C_g}{C_2 + C_g} \\
C_\beta &= \frac{C_s C_1}{C_s + C_1} \\
C_{es} &= \frac{C_1 C_\alpha}{C_1 + C_\alpha} \\
C_{ea} &= C_b + \frac{C_2 C_g}{C_2 + C_g} + \frac{C_s C_1}{C_s + C_1} \\
C_{eb} &= C_a + \frac{C_2 C_g}{C_2 + C_g} + \frac{C_s C_1}{C_s + C_1} \\
C_{eg} &= \frac{C_2 (C_a + C_b + C_\beta)}{C_a + C_b + C_\beta + C_2}
\end{aligned}
$$

The circuit equivalent capacitances of the two junctions $J1$ and $J2$, C_{e1} and C_{e2} can be expressed as follows:

$$
\begin{aligned}
C_{e1} &= \frac{C_s C_\alpha}{C_s + C_\alpha} \\
C_{e2} &= \frac{C_g (C_a + C_b + C_\beta)}{C_g + C_a + C_b + C_\beta}.
\end{aligned}
$$

The voltage over the tunneling junction $J1$ is determined by the source voltage V_s, the two input voltages V_a and V_b, and by the charges residing on the three nodes q_{n1}, q_{n2}, and q_{n3}. V_1 can be expressed as:

$$
\begin{aligned}
V_1 = {} & V_s \cdot \frac{C_s C_\alpha}{(C_s + C_{es})(C_1 + C_\alpha)} - V_a \cdot \frac{C_a C_s}{(C_a + C_{ea})(C_1 + C_s)} - V_b \cdot \frac{C_b C_s}{(C_b + C_{eb})(C_1 + C_s)} \\
& + q_{n1} \cdot \frac{C_\alpha}{(C_s + C_{es})(C_1 + C_\alpha)} - q_{n2} \cdot \frac{C_s}{(C_\beta + C_\alpha)(C_1 + C_s)} \\
& - q_{n3} \cdot \frac{C_2 C_s}{(C_g + C_{eg})(C_2 + C_a + C_b + C_\beta)(C_1 + C_s)}.
\end{aligned}
\tag{3.1}
$$

The critical voltage over the tunneling junction $J1$ is determined by its capacitance C_1 and by the capacitance $J1$ sees from the rest of the circuit C_{e1}. V_{c1} can be expressed as:

$$
V_{c1} = \frac{q_e}{2(C_1 + C_{e1})}. \tag{3.2}
$$

The voltage over the tunneling junction $J2$ is determined by the source voltage V_s, the two input voltages V_a and V_b, and by the charges residing on the three nodes q_{n1},

q_{n2}, and q_{n3}. V_2 can be expressed as:

$$\begin{aligned} V_2 = V_s \cdot \frac{C_s C_\beta C_g}{(C_\beta + C_\alpha)(C_2 + C_g)} - V_a \cdot \frac{C_a C_g}{(C_a + C_{ea})(C_2 + C_g)} - V_b \cdot \frac{C_b C_g}{(C_b + C_{eb})(C_2 + C_g)} \\ + q_{n1} \cdot \frac{C_1 C_g}{(C_s + C_{es})(C_1 + C_\alpha)(C_2 + C_s)} - q_{n2} \cdot \frac{C_g}{(C_\beta + C_\alpha)(C_2 + C_g)} \\ - q_{n3} \cdot \frac{C_a + C_b + C_\beta}{(C_g + C_{eg})(C_2 + C_a + C_b + C_\beta)}. \end{aligned} \tag{3.3}$$

The critical voltage over the tunneling junction $J2$ is determined by its capacitance C_2 and by the capacitance $J1$ sees from the rest of the circuit C_{e2}. V_{c2} can be expressed as:

$$V_{c2} = \frac{q_e}{2(C_2 + C_{e2})}. \tag{3.4}$$

The capacitance of $J2$, C_2, must be small to limit dynamic cross-talk and so $C_2 << C_g$ [11]. To achieve input-output voltage-level consistency, C_g is chosen as $10aF$ since the presence of an electron charge would result in an output voltage:

$$V_o = \frac{q_e}{C_2} = \frac{1.6 \cdot 10^{-19}}{10 \cdot 10^{-18}} = 16mV.$$

We first concentrate on $J2$. Both inputs going high should result in a tunneling event, so the initial step is similar to an *AND* gate. The conditions for states 1 through 6 in Table 3.1 are thus satisfied through the *AND* functionality. The conditions for states 7 and 8 are identical since the V_a and V_b terms are the same in Equation (3.3). There are thus three conditions, from states 8, 9, and 10, that need to be satisfied for $J2$.

We use the values of the proposed *AND* gate from [12] as a starting point. So, $C_a = C_b = 0.5aF$, $C_2 = 0.1aF$ and as previously determined $C_g = 10aF$. The bias voltage in the *AND* gate is connected to a capacitor of value $9.5aF$ therefore it has to hold that:

$$\frac{C_s C_1}{C_s + C_1} = 9.5aF. \tag{3.5}$$

C_1 is much larger than C_2 so the voltage over C_2 due to V_s is approximately the same.

If we look at $J1$, we can see that if the condition in state 1 is satisfied, so must all the conditions in 2, 3, and 4, because the inputs always reduce the voltage over the junction. Likewise, satisfying the condition in state 9 automatically satisfies those in states 6, 7, and 8. There are thus four conditions, from states 1, 5, 9, and 10, that need to be satisfied for $J1$.

Substituting the known values, the input values, the node charge values and Equation (3.5) into Equation (3.3), for each of the above mentioned circuit states, results in the following condition for C_s:

$$9.5 < C_s < 10.27$$

We choose C_s as $10aF$ resulting in a value for C_1 of $200aF$.

Circuit Parameters

The circuit parameters are summarized in Table 3.2.

Capacitance(s)	Value (aF)
C_a, C_b	0.5
C_1	200
C_2	0.1
C_s, C_g	10

Table 3.2: Circuit parameters for C-Element

Simulation Results

The simulations results can be seen in Figure 3.4. As it can be observed that the proposed SET circuit produces the correct *C-Element* functionality. The output goes high only when both input signals have gone high, and subsequently goes low only when both input signals have gone low.

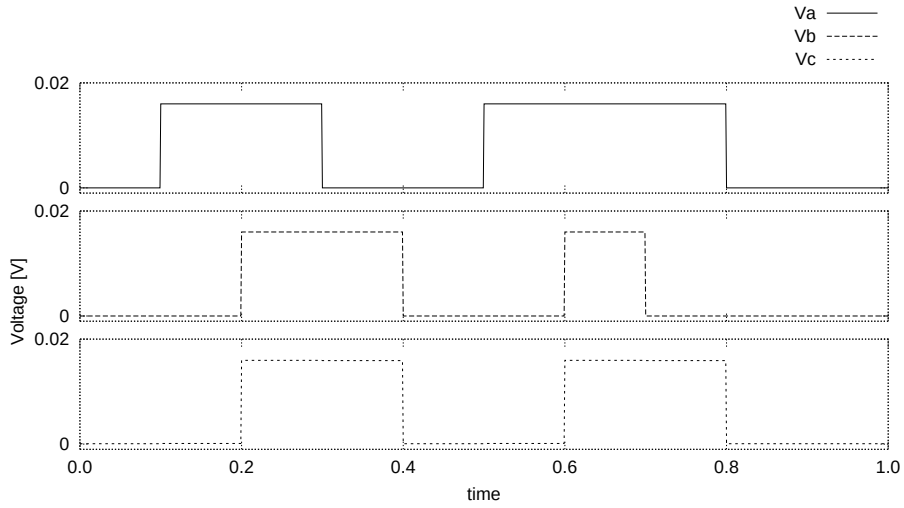
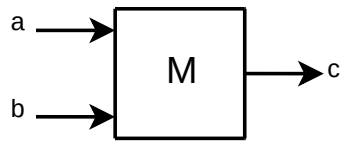


Figure 3.4: Simulation results for the C-Element

3.1.2 Merge

Symbol	Specification
	**pref**$[(a?\|b?);c!]*$

Functional Behavior

As the name suggests, tokens from either of the two inputs are routed to one output. If a token arrives at a or at b then a token is generated at c. The environment should prohibit simultaneous arrival of tokens at both inputs.

Two implementations are proposed for the *Merge*. The first one utilizes a mapping from the state graph to propose a SET circuit. The second one utilizes the generic design of the linear threshold gate described in [12]. The two alternative implementations are analyzed in the discussion at the end of this chapter.

3.1.2.1 State Graph based Implementation

State graph

Based on the described behavior we can derive a state graph for the *Merge*, depicted in Figure 3.5. Using value *transitions* to represent tokens results in the *Merge* functioning as a conventional *XOR* gate. The graph starts at state 1 with all inputs low and the output low. A token arriving at a or at b means that their value becomes high resulting in a token at c, meaning that c also goes high. In the state graph the new state due to a or b going high is shown with respectively labeled arrows, with a resulting in state 2 and b resulting in state 4. The output c becomes high due to a or b going high and the nodes 2 and 4 are labeled with a 'c' to denote c being high at those states. In state 2, an input token at a means the value of a going back low resulting in the graph going back to state 1. This is shown by the reversible arrow between nodes 1 and 2, with the direction of the lined arrowhead now valid. Alternatively in state 2, an input token at b, meaning b going high, results in the graph going to state 3, shown by the arrow between nodes 2 and 3. The output c is then low with no label in node 3. In state 3 both inputs are high and b going low results in state 2 and a going low results in state 4, both transitions shown by respective arrows with the lined arrowhead direction valid. In state 4 the same state transitions hold as for state 2 with the inputs a and b interchanged.

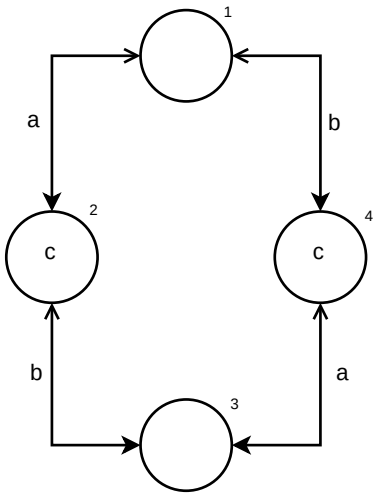


Figure 3.5: State state graph for Merge

Proposed SET-based Circuit Topology

We observe that the state graph for the *Merge* is constructed from 4 *C-Element* state graphs combined together, but with one input instead of two. We thus place 4 *C-Element* SET structures and connect them together based on the state graph, grounding each state node with a capacitor to store the charge. The two input capacitances of the *C-Element* are combined together into 1 capacitance.

In the state graph we can see that the output c is high in node 2 and 4. In the proposed circuit topology this is translated to charges on nodes $n2$ and $n4$ signaling the output V_c to be high. This is evaluated by an *OR* gate implemented using a linear threshold gate as introduced in [12] and taking $n2$ and $n4$ as inputs to produce the V_c output signal.

Figure 3.6 presents the resulting proposed SET circuit topology, which provides the required behavior for a *Merge*. The input signals are V_a and V_b and the output signal is V_c. The circuit comprises of 9 tunneling junctions and 12 capacitors.

Required Circuit Behavior

The circuit should operate as follows. With all inputs low the circuit is in a neutral state and no charge should be transported. If V_a goes high, an electron should tunnel through $J2$ and then through $J1$ and propagate from node $n2$ to $n1$, leaving a positive charge on $n2$ and a negative one on $n1$. From here, V_b going low should cause an electron to tunnel from $n3$ through $J4$ and $J3$ into $n2$, leaving a positive charge on $n3$ and no charge on $n2$. Subsequently, with both inputs V_a and V_b high, V_a going low should cause an electron to propagate from $n2$ into $n3$ through $J4$ and $J3$, and V_b going low should cause an electron to propagate from $n4$ into $n3$ through $J8$ and $J7$. With a positive charge residing on $n2$, V_a going low should cause the electron on $n1$ to tunnel through to $n2$ returning the circuit to its initial neutral state. Similarly, with a positive charge residing on $n4$, V_b going low should cause the electron on $n1$ to tunnel through to $n4$ returning

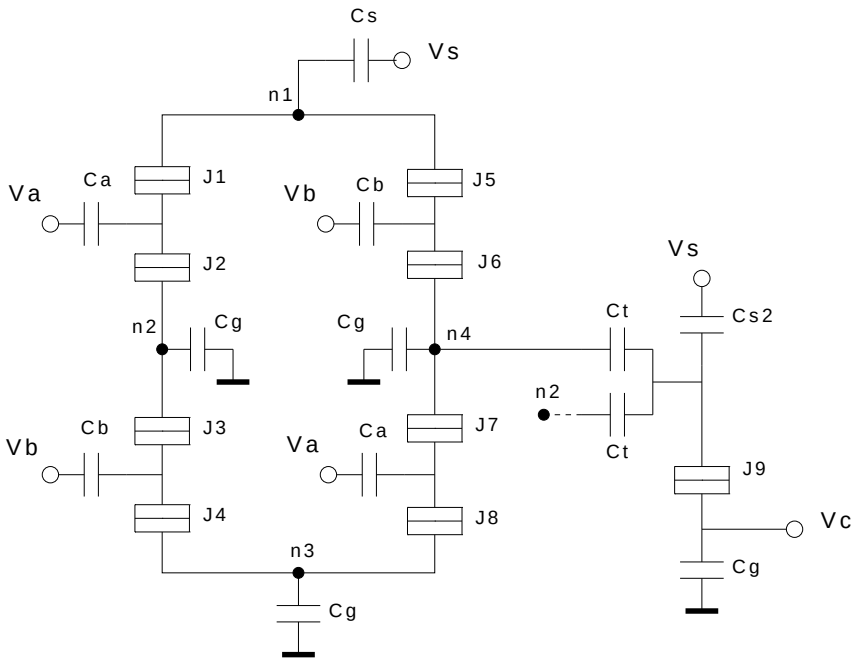


Figure 3.6: SET Circuit diagram for alternative Merge implementation

the circuit to its initial neutral state. The charges on node $n2$ and $n4$ drive the output through the *OR* gate.

Circuit Parameters Derivation

As mentioned above the *Merge* state graph is constructed from 4 *C-Element* structures with one input instead of two. For the *C-Element* structure the capacitances of both capacitors connected to the inputs were $0.5aF$. Since in the *Merge* functionality state transitions occur as though a *C-Element* had two inputs transitioning simultaneously all the time, we can simply take both inputs of the *C-Element* as from the same input. The capacitance values can be added to form a single capacitor resulting in a capacitance of $1aF$. The values for the threshold based *OR* gate are taken from [12]. The rest of the values can be taken from the *C-Element* implementation, with $10aF$ being used as the value for all the grounded capacitors so that a consistent output voltage can result.

Circuit Parameters

A summary of the circuit parameters for the *Merge* implementation based on the state graph is presented in Table 3.3.

Capacitance(s)	Value (aF)
C_a, C_b	1
C_g, C_s	10
C_1, C_3, C_5, C_7	200
C_2, C_4, C_6, C_8, C_9	0.1
C_t	0.5
C_{s2}	10.5

Table 3.3: Merge circuit parameters for state graph based implementation

3.1.2.2 Threshold Boolean Gate based Implementation

Proposed SET-based Circuit Topology

Figure 3.7 presents an alternative SET circuit topology, which provides the required behavior for a Merge. The input signals are V_a and V_b and the output signal is V_c. Since the *Merge* functionality is equivalent to that of an *XOR* gate an *AND-OR-INVERT* implementation is the most appropriate design. The *AND* gate and the *OR* gate are built by utilizing the generic design of the linear threshold gate described in [12]. The inverter is implemented using the inverting buffer from [12] depicted in Figure 3.1. Two instances of the inverting buffer are also used to produce the inverted signals of V_a and V_b, each requiring 4 tunneling junctions and 5 capacitors. In total the circuit comprises of 15 tunneling junctions and 27 capacitors.

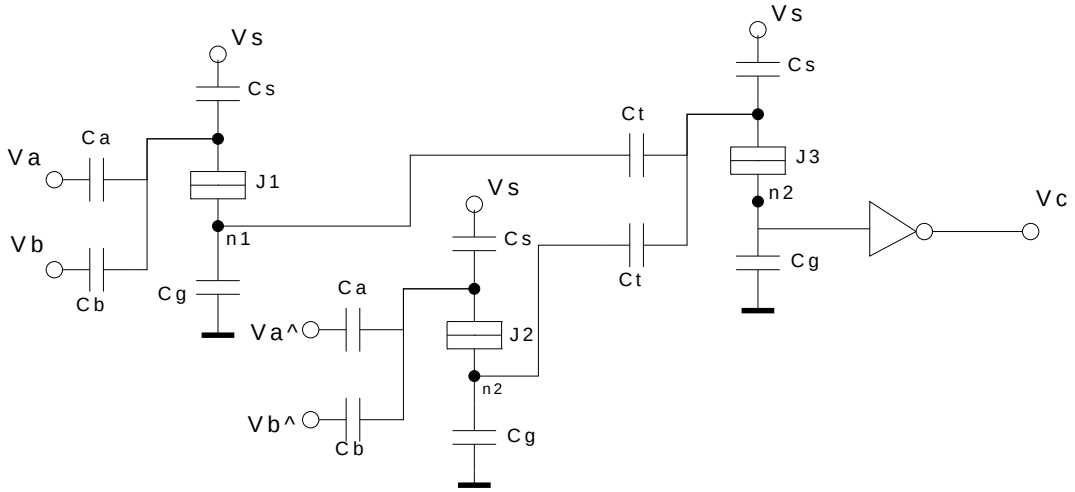


Figure 3.7: Merge circuit diagram

Required Circuit Behavior

The circuit operates as follows. When both inputs are low the inverted input signals are high causing an electron to tunnel through $J2$ leaving a positive charge on $n2$. This in turn causes an electron to tunnel through $J3$ leaving a positive charge on $n3$.

This is inverted and so the output becomes low, as it should be. If one of the input signals undergoes a transition then the voltage of $J2$ (and $J1$) becomes lower than the critical voltage causing the electron to tunnel back leaving no charge on $n2$ (and $n1$). This reduces the voltage over $J3$ to under the critical voltage and the electron tunnels back leaving no charge on $n3$. This value is complemented afterward by the output inverter thus the output becomes high, as it should. If the second signal also undergoes a transition then the voltage over $J1$ becomes higher than the critical voltage causing an electron to tunnel. This cause an electron to tunnel through $J3$ leaving a positive charge on $n3$ corresponding to a low output, as it should. If subsequently one of the signals transitions again, going low this time, the circuit goes into the previous state with no charges on $n1$, $n2$, and $n3$. If after that the other signal also transitions the circuit returns to the original charge neutral state.

Circuit Parameters

The circuit parameters correspond to those in [12] and are summarized in Table 3.4.

Capacitance(s)	Value (aF)
C_a, C_b, C_t	0.5
C_{s1}	9.5
C_{s2}	10.5
C_g	10
C_1, C_2, C_3	0.1

Table 3.4: Merge circuit parameters for threshold gate based implementation

Simulation Results

The simulation results for the circuit topology presented in Figure 3.6 are displayed in Figure 3.8 and the simulation results for the topology presented in Figure 3.7 are identical. One can observe that the two proposed circuits exhibit the expected *Merge* behavior. When V_a goes high V_c goes high. Subsequently V_b also goes high and V_c then goes low. When V_a then goes low V_c goes high again and when V_b goes low V_c also goes low. From time 0.6 two transitions of V_a result in two transitions of V_c.

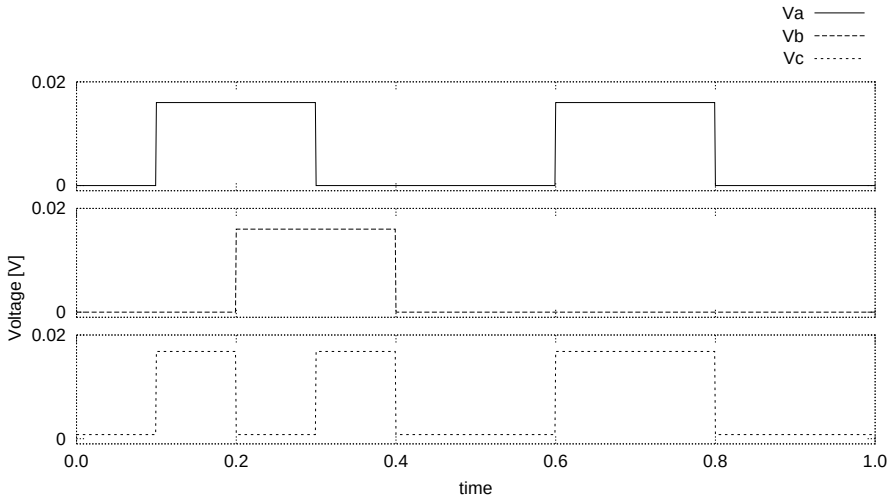
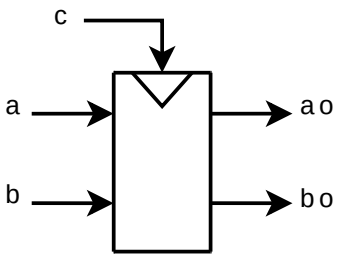


Figure 3.8: Simulation results for the Merge

3.1.3 Sequencer

Symbol	Specification
	**pref**$[a?; ao!]$ || **pref**$[b?; bo!]$ || **pref**$[c?; (ao!\|bo!)]]*$

Functional Behavior

The *Sequencer* has two inputs, a and b, each with a corresponding output, ao and bo. Another input, c, is the control input. A token arriving at a and at c, in any order, results in a token at ao and a token arriving at b and at c results in a token at bo. A token arriving at a and then one at b results in the component waiting for a token to arrive at c. Once a token arrives at c a token is generated at either ao or bo, arbitrarily chosen. Subsequently, another token arriving at c results in the other output generating a token. If an input token has arrived and the *Sequencer* is waiting for another input token, the environment should prohibit the arrival of a token at that same input.

State graph

Based on the described behavior we can derive a state graph for the *Sequencer*, shown in Figure 3.9. The graph starts at state 1 with all inputs low and the output low. Inputs a and c going high and resulting in ao going high is shown in the state graph as a state transition from state 1 to state 2, with an arrow labeled with 'a' and 'c', the filled arrowhead direction being valid in this case. The state 2 node is labeled with 'ao' to denote ao being high at that state. For inputs b and c going high and resulting in bo going high this similarly shown in the state graph as a state transition from state 1 to state 4. From state 2, two input combinations are possible. First, a and c both going back low, resulting in a state transition back to state 1, shown by the same arrow, but with the lined arrowhead direction now valid. Second, b going high and c going low, or $c\hat{}$ going high, resulting in a state transition to state 3. This is shown by the arrow labeled with 'b' and '$c\hat{}$ ' with the direction of the filled arrowhead being valid. In state 3 bo is also high and node 3 is labeled with 'ao' and 'bo'. From state 3, either b goes low and $c\hat{}$ goes low resulting in state 2, or a goes low and $c\hat{}$ goes low resulting in state 4. In state 4 ao is low and it is not labeled in node 4. Between states 1, 3, and 4 the same state transitions hold as for between states 1, 2, and 3, with the inputs a and b interchanged and the outputs ao and bo interchanged.

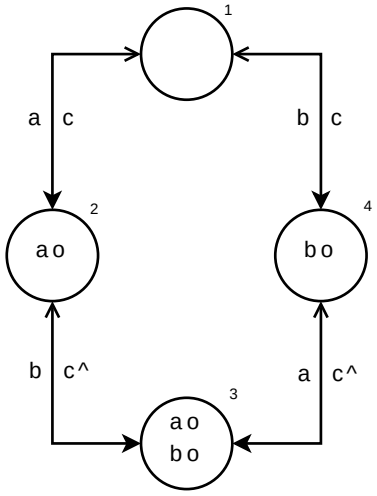


Figure 3.9: State graph for Sequencer

Proposed SET-based Circuit Topology

The state graph for the *Sequencer* is constructed from 4 *C-Element* structures combined together. The state graph can be mapped onto a SET circuit topology by means of substituting each arrow with a *C-Element* structure and grounding each state node with a capacitor to store the charge.

In the state graph we can see that the output ao is high on nodes 2 and 3, and that output bo is high on node 3 and 4. In the proposed circuit topology this is translated to

two *OR* gates, taking $n2$ and $n3$ as inputs to produce the V_{ao} output signal and taking $n3$ and $n4$ as inputs to produce V_{bo}.

Figure 3.10 presents the resulting SET circuit topology, which provides the required behavior for a Sequencer. The input signals are V_a, V_b, and V_c and the output signals are V_{ao} and V_{bo}. The signal V_c has to be inverted to generate a $V_c\hat{}$ signal and this requires an inverting buffer, which consists of 4 tunneling junctions and 5 capacitors. In total the circuit comprises of 14 tunneling junctions and 25 capacitors.

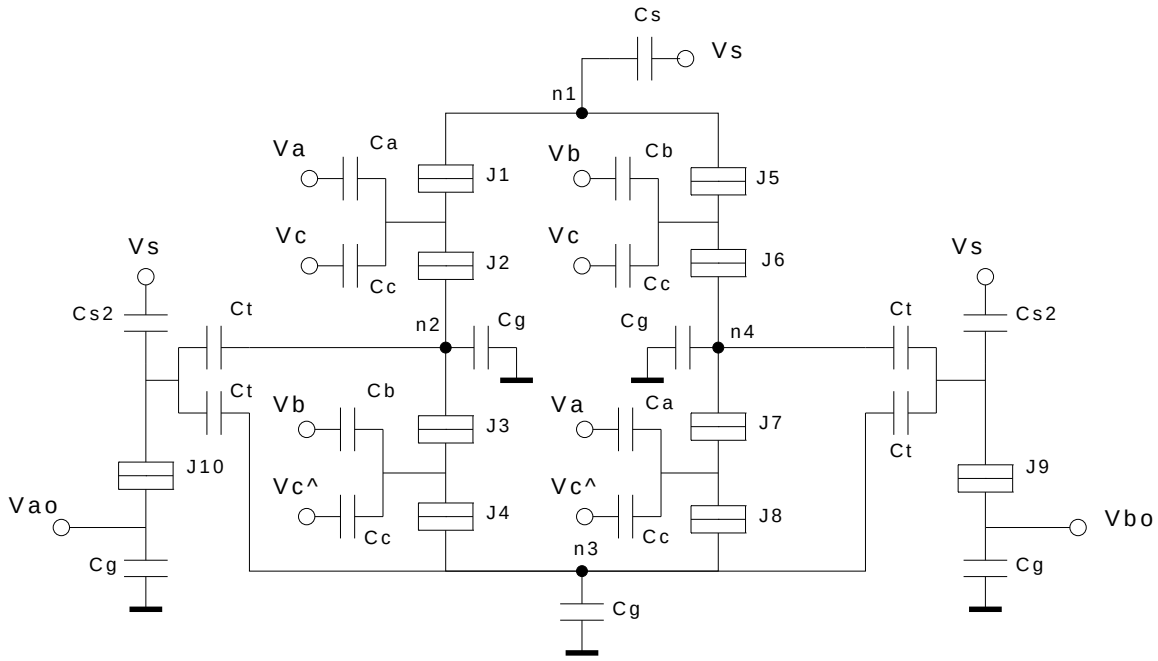


Figure 3.10: Sequencer circuit diagram

Required Circuit Behavior

The circuit should operate as follows. With all inputs low the circuit is in a neutral state and no charge should be transported. If the two inputs V_a and V_c go high an electron should tunnel through $J2$ and then through $J1$ and propagates from node $n2$ to $n1$ due to the transport mechanism built in such a *C-Element*, leaving a positive charge on $n2$ and a negative one on $n1$. From here, V_a and V_c going low should cause the electron to go back to $n2$ resulting in the neutral state, or V_b going high and V_c going low ($V_c\hat{}$ going high) should result in an electron tunneling through $J4$ and $J3$. This propagation through $J4$ and $J3$ is again similar to that of the *C-Element* mechanism. With $n2$ positive and the two corresponding inputs high the critical voltage of $J4$ should be exceeded resulting in an electron from $n3$ tunneling through $J4$. The voltage over $J3$ should then exceed the critical voltage due to the positive charge on $n2$ and the negative charge on the other side and the electron should tunnel to $n2$ leaving no charge on $n2$ and a positive one on $n3$. Subsequently, with inputs V_a and V_b high and V_c low, either V_a and $V_c\hat{}$ go low, or V_b and $V_c\hat{}$ go low, and this should result in the voltage over $J8$

and $J4$, respectively, increasing and exceeding the critical voltage. This should cause an electron to tunnel into $n3$ and increase the voltage over $J7$ and $J3$, respectively, above the critical voltage causing an electron to tunnel through $J7$ or $J3$ leaving a positive charge on $n4$ and $n3$, respectively. An electron should effectively propagate from $n4$ or $n2$ into $n3$. From there the aforementioned mechanisms would apply.

The net effect is thus that the positive charge moves around the circuit depending on input signal transitions, until it reaches node $n1$ where the circuits returns its initial neutral state. If, from the neutral state V_{ao} and V_{bo} both go high and then V_{co}, then one of the propagation routes is chosen arbitrarily and either node $n2$ or $n4$ becomes positively charged. Subsequently, the circuit should be in the state as described above after the first propagation and the extra signal which went high is still pending for the V_c to transition.

Circuit Parameters Derivation

To achieve the above mentioned behavior, the circuit parameters are taken from the *C-Element* implementation but have to be adjusted, since on node $n1$ there are two junctions attached and the other nodes are grounded rather than connected to the source via a capacitor.

The following input gate capacitances remain unchanged, $C_a = C_b = C_c = 0.5aF$, as do the capacitances of the lower junctions of the *C-Element* structures, $C_2 = C_4 = C_6 = C_8 = 0.1aF$. We also take $C_g = 10aF$, and even though the effective ground capacitance of the nodes $n1$ and $n2$ is $11aF$ due to C_a, C_b, and C_c through C_3 and C_7, this doesn't affect the functionality.

If we look at $J2$, for the initial *AND* functionality the capacitance must be adjust because the capacitance equivalent to the V_{bias} now has C_5 attached. We can neglect the capacitance of $J6$ and all attached capacitors as C_6 is small when compared to C_1. This results in:

$$\frac{C_1(C_s + \frac{C_5}{C_5+1})}{C_1 + C_s + \frac{C_5}{C_5+1}} = 9.5aF.$$

If we take $C_s = 10aF$ and $C_1 = C_5$ this results in $C_1 = C_5 \simeq 70aF$. The values of C_3 and C_7 can be taken as $200aF$ from the *C-Element* structure and the values of C_2, C_4, C_6, and C_8 can be taken as $0.1aF$. The values for the *OR* gate are taken from [12].

Circuit Parameters

The circuit parameters are summarized in Table 3.5.

Simulation Results

The simulation results are displayed in Figure 3.11. For the simulations, the signal $V_c\hat{}$ was separately defined with a small delay after the transition of V_c to simulate the delay that may occur in the formation of the inverted signal. As one can observe from the first segment of the simulation, which corresponds to the first V_a pulse, the basic *Sequencer* functionality is delivered. Input V_a going high or low followed by or preceded

Capacitance(s)	Value (aF)
C_a, C_b, C_c	1
C_1, C_5	70
C_3, C_7	200
C_g, C_s	10
C_{s1}, C_{s2}	10.5
C_2, C_4, C_6, C_8, C_9, C_{10}	0.1
C_t	0.5

Table 3.5: Sequencer circuit parameters

by input V_c going high or low produces a toggling of the corresponding output. The same holds for input V_b. In the second part of the simulation we can see the additional *Sequencer* functionality showing that if V_a goes high and then V_b goes high before V_c going high two subsequent transitions of V_c cause a toggling of V_{ao} and V_{bo} outputs in an arbitrary order.

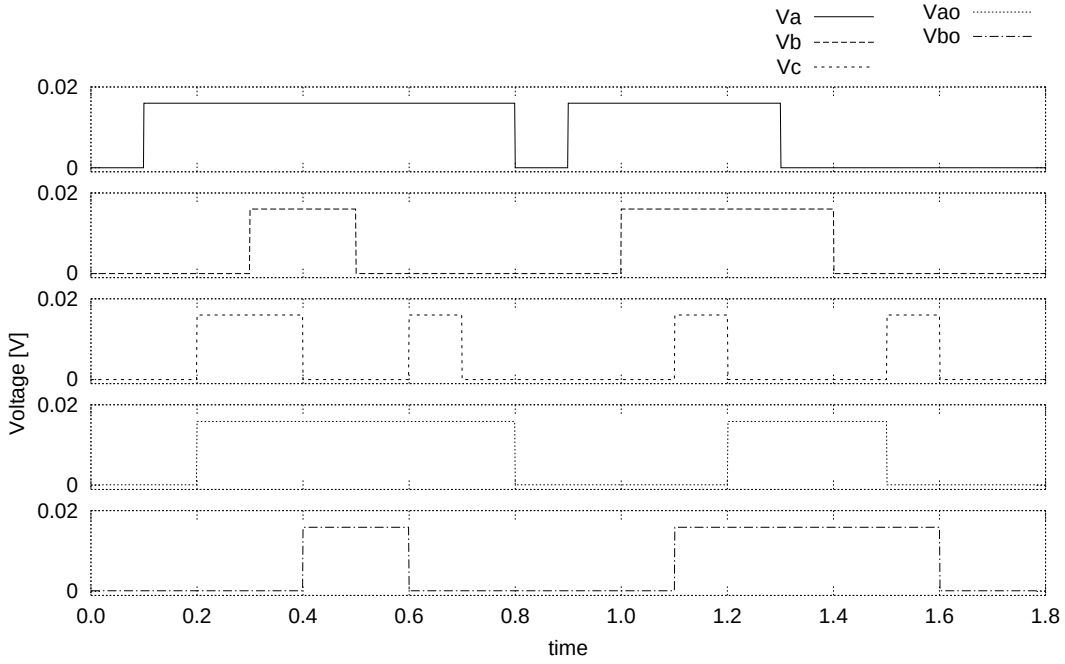
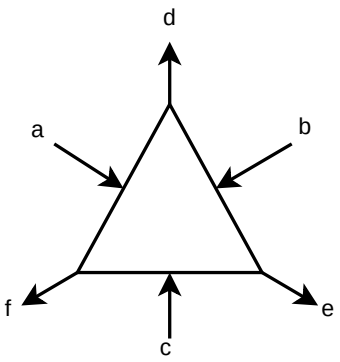


Figure 3.11: Simulation results for the Sequencer

3.1.4 Tria

Symbol	Specification
	**pref**$[((a?\|\|b?);d!)\|$ $((b?\|\|c?);e!)\|$ $((a?\|\|c?);f!)]*$

Functional Behavior

The *Tria* functions similar to the *Sequencer*. Tokens at a and b result in a token at output d, tokens at b and c result in a token at e, and tokens at a and c result in a token at f. If a token has arrived at a certain input and the *Tria* is waiting for another input token to arrive, then the environment should prohibit another token arriving at the same input again.

State graph

Based on the described behavior we can derive a state graph for the *Tria*, as presented in Figure 3.12. The graph starts at state 1 with all inputs low and the output low. Inputs a and b going high and resulting in d going high is shown in the state graph as a state transition from state 1 to state 2, with an arrow labeled with 'a' and 'b', the filled arrowhead direction being valid in this case. The state 2 node is labeled with a 'd' to denote d being high at that state. Similarly, b and c going high results in a transition to state 3 where e is high, and a and c going high results in a transition to state 4 where f is high. From state 2, three input combinations are possible. First, a and b both going back low, resulting in a state transition back to state 1, shown by the same arrow, but with the lined arrowhead direction now valid. Second, c going high and b going low, or $b\hat{}$ going high, resulting in a state transition to state 5, where e is then high in addition to d. This is shown by the arrow labeled with 'b$\hat{}$ ' and 'c' with the direction of the filled arrowhead being valid. In state 5 e is also high and node 3 is labeled with 'd' and 'e'. Third, c going high and a going low, or $a\hat{}$ going high, resulting in a state transition to state 6, where f is then high in addition to d. From state 5, both a and c going low, or $a\hat{}$ and $c\hat{}$ going high results in state 8, where all output signals are high. Thus, node 8 is labeled with the three output signals 'd', 'e', and 'f'.

From each state there are three paths that can be taken depending on which combination of inputs undergo value transitions, resulting in a state where the relevant output

is low if it was high in the previous state, and high if it was low in the previous state.

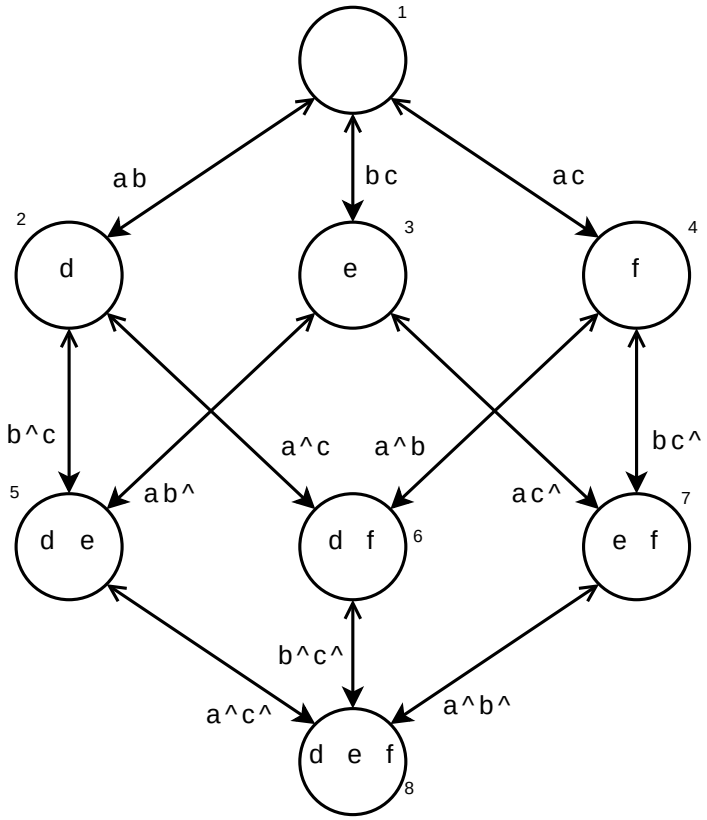


Figure 3.12: State graph for Tria

Proposed SET-based Circuit Topology

The state graph for the *Tria* is constructed from 12 *C-ELement* structures combined together. The state graph can be mapped onto a SET circuit topology by means of substituting each arrow with a *C-Element* structure and grounding each state node with a capacitor to store the charge.

In the state graph we can see that each output signal appears on 4 separate nodes. In the proposed circuit topology this is translated into three four-input *OR* gates. Nodes $n2$, $n5$, $n6$, and $n8$ form the inputs for the first *OR* to produce V_d. Similarly, output V_e is formed from $n3$, $n5$, $n7$, and $n8$ and output V_f is formed from $n4$, $n6$, $n7$, and $n8$.

Figure 3.13 presents the resulting proposed SET circuit topology, which provides the required behavior for a for a *Tria*. The input signals are V_a, V_b, and V_c and the output signals are V_d, V_e, and V_f. All input gate capacitances are C_i. Each input signal has to be inverted with an inverting buffer for use in the circuit, requiring 12 tunneling junctions and 15 capacitors. In total the circuit comprises of 39 tunneling junctions and 45 capacitors.

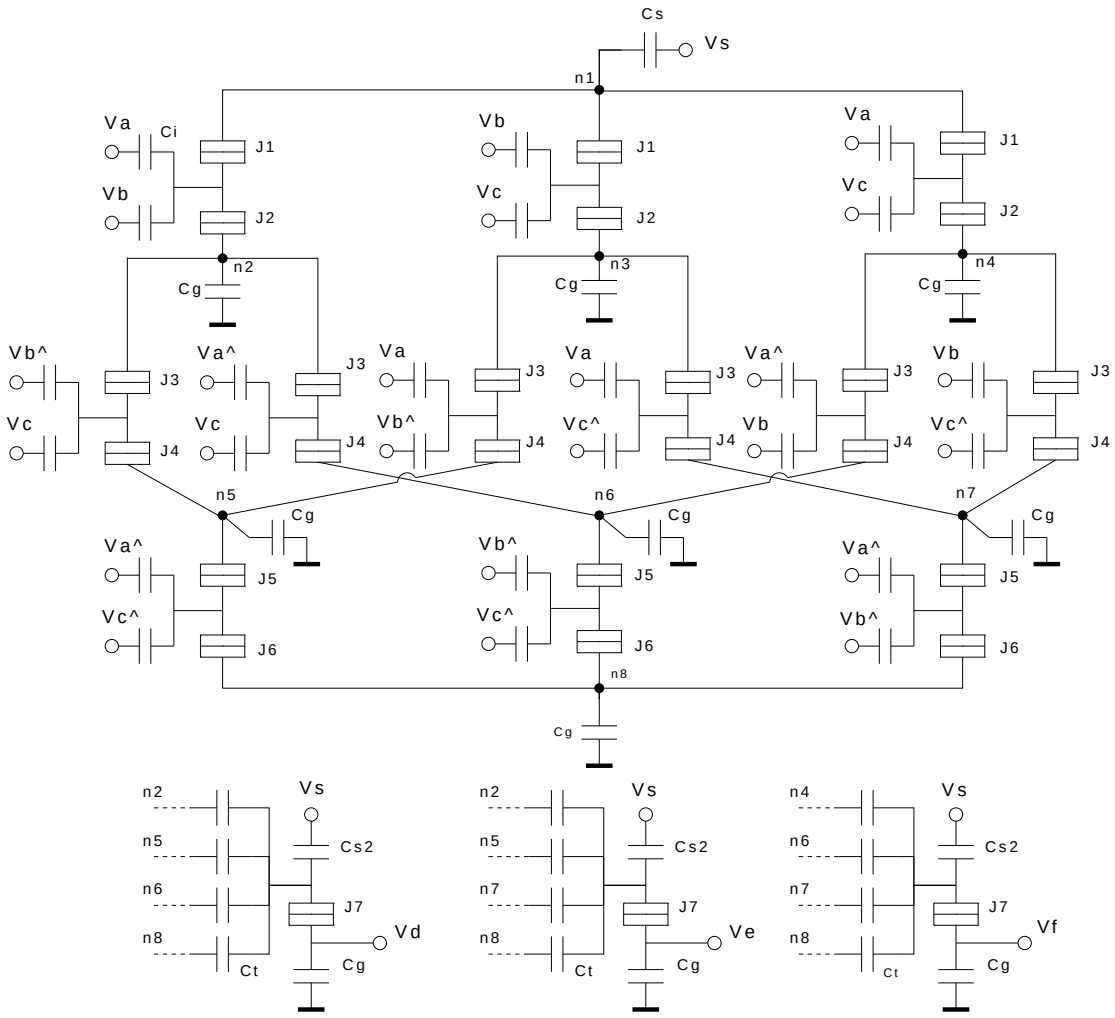


Figure 3.13: Tria circuit diagram

Required Circuit Behavior

The circuit should operate as follows. An electron should move to node $n1$ from $n2$, $n3$ or $n4$ depending on which input signals go high. Only if the same two signals then go low again should the electron propagate back, which follows the *C-Element* mechanism. Subsequently, the node which has the positive charge has three nodes from which an electron may potentially arrive, depending on which input signals undergo transitions. Thus, each node has a pathway to other 3 nodes opening up only due to transitions which are possible if that node has been reached. As with the *Sequencer*, the positive charge moves from node to node depending on the transitions made by the input signals.

Circuit Parameters Derivation

We can observe from the circuit topology and circuit behavior that there are a large number of input signals spread throughout the circuit. This results in increased feedback

effects which cannot be neglected. Furthermore, the *C-Element* structures not directly attached to node $n1$ do not follow the same circuit state behavior as the *C-Element* and have to thus be reevaluated. These structures transfer positive charges from one node to the other with the charges remaining on the nodes, while in the *C-Element* behavior an electron is generated on one node.

We first focus on the circuit segment between nodes $n6$ and $n8$. For this segment we can neglect any feedback effects from input signals elsewhere, as the nodes are directly connected to junctions with a value of $0.1aF$, which reduces the voltage transfer through the capacitors to low levels so that no interference occurs. The circuit state table that results based on the required circuit behavior is displayed in Table 3.6.

Circuit state	$V_b\hat{}$	$V_c\hat{}$	n_6	n_{6-8}	n_8	$J1$	$J2$
1	0	0	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
2	1	0	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
3	0	1	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
4	1	1	0	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
4	0	1	1	0	0	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
5	1	1	1	0	0	$\|V_1\| < V_{c1}$	$V_{c2} < \|V_2\|$
6	1	1	1	-1	1	$V_{c1} < \|V_1\|$	$\|V_2\| < V_{c2}$
7	1	0	0	0	1	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
8	0	1	0	0	1	$\|V_1\| < V_{c1}$	$\|V_2\| < V_{c2}$
9	0	0	0	0	1	$\|V_1\| < V_{c1}$	$V_{c2} < \|V_2\|$
10	0	0	0	1	0	$V_{c1} < \|V_1\|$	$\|V_2\| < V_{c2}$

Table 3.6: Critical voltage conditions for circuit states of Tria segment between nodes $n6$ and $n8$

We keep the input gate capacitances at $0.5aF$. For the calculation we take C_g to be $10.3aF$ and C_s to be $10.1aF$. C_2 in the *C-Element*, C_6 in this case is taken as $0.1aF$. Evaluating the voltage conditions for each circuit state in Table 3.6 using the respective input values and node charges by using Equations (3.1), (3.2), (3.3), and (3.4), results in the following condition for C_5:

$$114 < C_5 < 548$$

We use the value of $120aF$ for C_5 to allow for estimation errors, but keeping C_5 to a minimum to keep crosstalk interference as low as possible for the rest of the circuit. These values can be used for the segments between $n5$-$n8$ and $n7$-$n8$.

Next, we focus on the middle segments of the circuit. We take the segment between $n3$ and $n7$ for illustrating the calculation. The state circuit table for this segment is identical to the one in Table 3.6. We take C_g and C_s to be $10.1aF$. C_2 in the *C-Element*, C_4 in this case is taken as $0.1aF$. The effect of the voltage over $J4$ due to $V_a\hat{}$ and $V_b\hat{}$ in the lower segment cannot be ignored. Assuming $C_5 = C_3$, a new term is added

into Equation (3.4):

$$-V_s \cdot \frac{\frac{C_a C_3}{C_a+C_3}}{\frac{C_a C_3}{C_a+C_3} + \frac{C_g C_4}{C_g+C_4}} \cdot \frac{C_a + C_z + C_\beta}{C_4 + C_a + C_z + C_\beta}.$$

Also, the effect of the voltage over $J4$ due to the inputs in the other segment connected to node $n3$ cannot be ignored. Another term has to be added into Equation (3.4):

$$+V_s \cdot \frac{\frac{C_a C_3}{C_a+C_3}}{2\frac{C_a C_3}{C_a+C_3} + C_g} \cdot \frac{C_1 C_g}{(C_s + C_{es})(C_1 + C_\alpha)(C_2 + C_s)}.$$

Following the same method as for the first segment results in the following condition for C_3:

$$92 < C_5 < 10190$$

We use the value of $95aF$ for C_3 to limit crosstalk and allow for estimation errors. This applies to all the segments between nodes $n2$, $n3$, $n4$ and $n5$, $n6$, $n7$.

For the top segment the state circuit table of the *C-Element*, Table 3.1, applies. We thus take the parameter for C_1 in the *Tria* as $200aF$.

For the three *OR* gates we adjust the parameters so that there is reduced feedback into the circuit. If we reduce the input gate capacitances C_t to $0.1aF$ then C_{s2} has to be 10.3 to retain the *OR* functionality.

Circuit Parameters

The circuit parameters are summarized in Table 3.7.

Capacitance(s)	Value (aF)
C_g, C_s	10
C_1	200
C_2, C_4, C_6	0.1
C_3	95
C_5	120
C_t	0.1
C_{s2}	10.3
C_i	0.5

Table 3.7: Tria circuit parameters

Simulation Results

The simulation results are displayed in Figure 3.14. As with the *Sequencer*, the signals $V_a\hat{}$, $V_b\hat{}$, and $V_c\hat{}$ were separately defined with a small delay after the transition of their respective signals to simulate the delay that may occur in the formation of the inverted signal. The first two transitions are of V_a and V_b going high. An electron thus propagates from $n2$ to $n1$, and the charge on $n2$ signals the output V_d. Subsequently, V_b

goes low and V_c goes high causing the electron to propagate from $n5$ to $n2$. The charge on $n5$ thus signals the output V_e, however, the charge on $n2$ is now zero but the output hasn't changed, it has been moved to node $n5$. Node $n5$ thus signals both output V_d and V_e. The rest of the simulation proceeds in a similar fashion and clearly indicates that the topology in Figure 3.13 is able to deliver the *Tria* functionality.

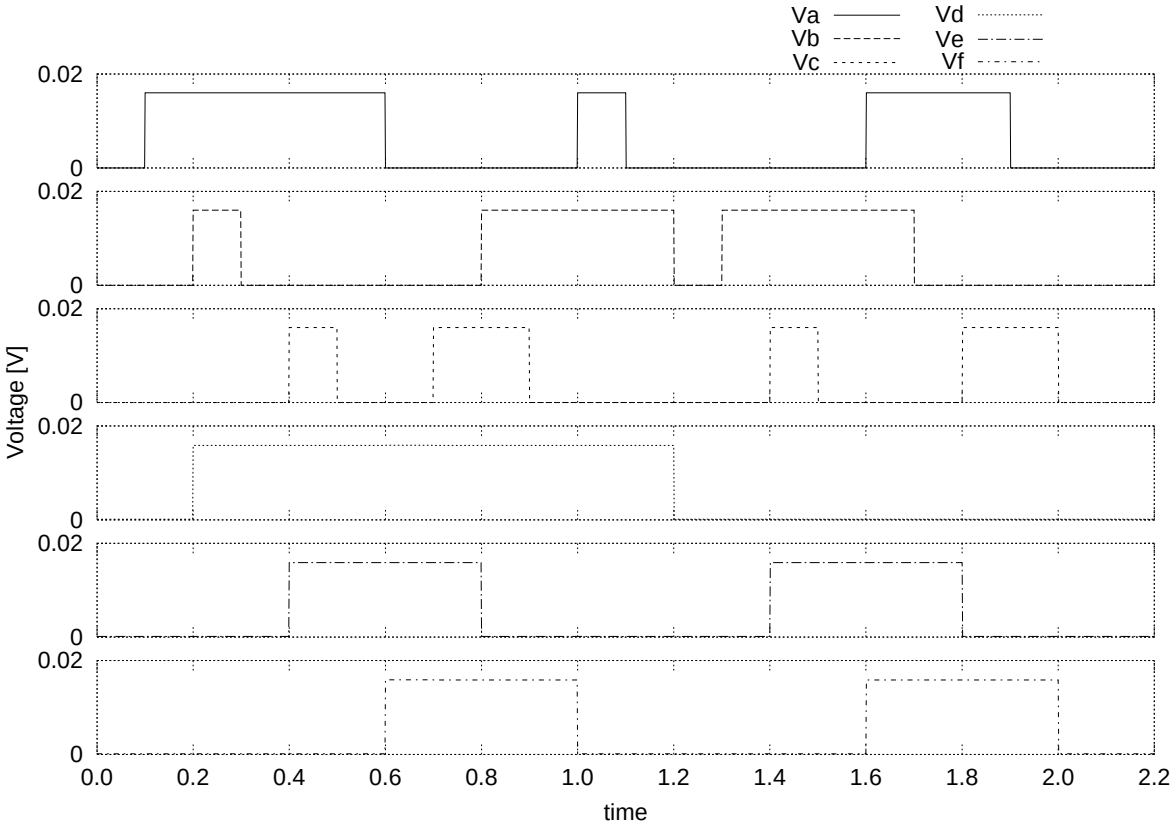
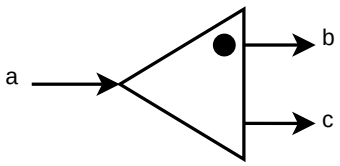


Figure 3.14: Simulation results for the Tria

3.1.5 Toggle

Symbol	Specification
	**pref**$[a?; b!; a?; c!]*$

Functional Behavior

The *Toggle* has only one input, a, and two outputs, b and c. A token arriving at input a results in a token generated at output b and a subsequent token at a results in a token generated at output c. The process then repeats itself. The *Toggle* functions like the conventional toggle flip-flop when using value transitions to represent tokens.

State graph

Based on the described behavior we can derive a state graph for the *Toggle*, displayed in Figure 3.15. The graph starts at state 1 with all inputs low and the output low. A token arriving at a, meaning a goes high, results in b going high. This is shown in the state graph as the transition to state 2. The output b is high in state 2 so node 2 is labeled with 'b'. Subsequently, when another token arrives at a, meaning a goes low and $a\hat{}$ goes high, c goes high. This is shown with the transition from state 2 to state 3. In state 3 b and c are both high and node 3 is labeled with both outputs. A third token arriving at a, meaning a goes high, results in a transition to state 4, with the value of b low in state 4. A fourth token arriving at a, meaning $a\hat{}$ going high, returns the graph to the initial state 1 and c goes low. The cycle then repeats.

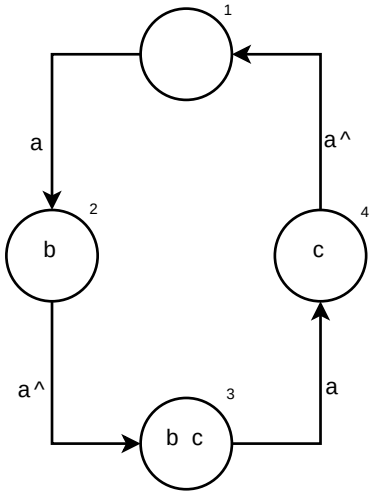


Figure 3.15: State graph for Toggle

Proposed SET-based Circuit Topology

The state graph for the *Toggle* is constructed from 4 *C-Element* structures combined together. Unlike in the *C-Element* however, the arrows in the state graph are not reversible, meaning that the labeled input signal going low doesn't result in a state transition to the previous state. To achieve this functionality we hard-wire one of the inputs of *C-Element* SET structures to the source and connect the structures together based on the state graph.

In the state graph we can see that the output b is high on nodes 2 and 3, and that output c is high on node 3 and 4. In the proposed circuit topology this is translated to two *OR* gates, taking $n2$ and $n3$ as inputs to produce the V_b output signal and taking $n3$ and $n4$ as inputs to produce V_c.

Figure 3.16 presents the resulting proposed SET circuit topology, which provides the required behavior for a *Toggle*. The input signal is V_a and the output signals are V_b and V_c. The input signal V_a has to be inverted using the inverting buffer, requiring 4 tunneling junctions and 5 capacitors. In total the circuit comprises of 14 tunneling junctions and 22 capacitors.

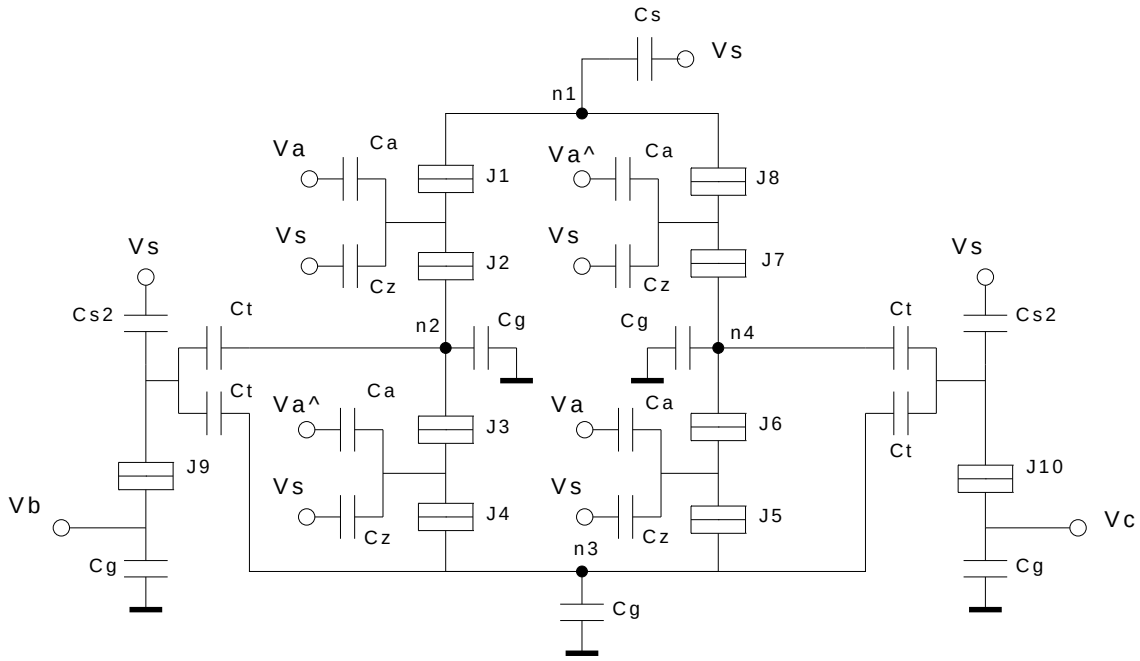


Figure 3.16: Toggle circuit diagram

Required Circuit Behavior

The circuit should operate as follows. The circuit starts in a neutral state with the input low and with no charge transported. If V_a goes high an electron should tunnel through $J2$ from node $n2$ to node $n5$ and raises the voltage over $J1$ above its critical voltage. The electron should then tunnel further to $n1$ through $J1$, leaving a positive charge on $n2$. The electron shouldn't return if V_a goes low, so the critical voltage of either $J1$ or $J2$ should not be exceeded. When V_a goes low, $V_a\hat{}$ goes high an electron should be transferred from node $n3$ to $n2$ by tunneling through $J4$ and then $J3$. With a positive charge on $n3$ if V_a goes high again an electron should tunnel through $J6$ and then $J5$ from $n4$ to $n3$. Finally, if V_a goes low again, $V_a\hat{}$ going high should cause the voltage over $J8$ to exceed its critical voltage causing the electron residing on $n1$ to tunnel through $J8$ and then through $J7$ to neutralize the positive charge on $n4$ and bring the circuit back to its initial neutral state.

Circuit Parameters Derivation

We can observe from the circuit behavior description that the propagation of positive charge is one way. To achieve such a behavior one of the inputs for each *C-Element* structure is hard-wired to the source. The parameters from the *C-Element* have to be adjusted for the implementation of the *Toggle* to arrive at the required functionality. We take C_a to be $1aF$ from the *Tria* implementations.

If we take the segment between $n2$ and $n3$ the circuit states that result are presented in Table 3.8:

Circuit state	$V_a\hat{}$	n_2	n_{2-3}	n_3	$J3$	$J4$
1	0	0	0	0	$\|V_3\| < V_{c3}$	$\|V_4\| < V_{c4}$
2	0	1	0	0	$\|V_3\| < V_{c3}$	$\|V_4\| < V_{c4}$
3	1	1	0	0	$\|V_3\| < V_{c3}$	$V_{c4} < \|V_4\|$
4	1	1	-1	1	$V_{c3} < \|V_3\|$	$\|V_4\| < V_{c4}$
5	1	0	0	1	$\|V_3\| < V_{c3}$	$\|V_4\| < V_{c4}$

Table 3.8: Critical voltage conditions for circuit states of the Toggle segment

We take C_a to be $1aF$ and C_4 to be $0.1aF$ from the *Tria* implementations we can take the effect of the voltage over $J3$ and $J4$ due to V_s through $J2$ to be negligible. Likewise, we neglect the effect of the voltage over $J3$ due to V_s through $J5$. However, the effect of the voltage over $J4$ due to V_s through $J5$ cannot be ignored. Assuming $C_5 = C_3$, a new term is added into Equation (3.4):

$$-V_s \cdot \frac{\frac{C_a C_3}{C_a+C_3}}{\frac{C_a C_3}{C_a+C_3} + \frac{C_g C_4}{C_g+C_4}} \cdot \frac{C_a + C_z + C_\beta}{C_4 + C_a + C_z + C_\beta}.$$

If we evaluate Equations (3.2) and (3.4) for the circuit states 1 to 5 in Table 3.8, substituting the capacitors that are in the segment between $n2$ and $n3$ the following condition results for C_3:

$$C_3 > 48.5$$

We choose the value of $50aF$ for C_3 so that the voltage crosstalk is minimized. If we utilize the same parameters for the rest of the circuit the correct functionality results. For the *OR* gate we adjust the parameters so that there is reduced feedback into the circuit. If we reduce the input gate capacitances C_t to $0.1aF$ then C_{s2} has to be $10.1aF$ to retain the *OR* functionality.

Circuit Parameters

The circuit parameters which result in the required behavior of a *Toggle* are summarized in Table 3.9.

Simulation Results

The simulation results are displayed in Figure 3.17. For the simulations, the signal $V_a\hat{}$ was separately defined with a small delay after the transition of V_a to simulate the

Capacitance(s)	Value (aF)
C_a, C_z	1
C_s, C_g	10
C_1,C_3,C_5,C_7	50
C_{s2}	10.1
C_t, C_2, C_4, C_6, C_8, C_9,C_{10}	0.1

Table 3.9: Toggle circuit parameters

delay that may occur in the formation of the inverted signal. As is expected, output V_b toggles for each positive edge of the input and output V_c toggles for each negative edge of the input. During the first pulse only V_b is high, which corresponds to node $n2$ and after the end of the first pulse both V_b and V_c are high corresponding to node $n3$. This is as expected as $n3$ supplies the output for both V_b and V_c. During the second pulse only V_c is high, corresponding to node $n4$. After the second pulse the circuit is in its initial neutral state again and none of the outputs are high. The cycle then repeats itself.

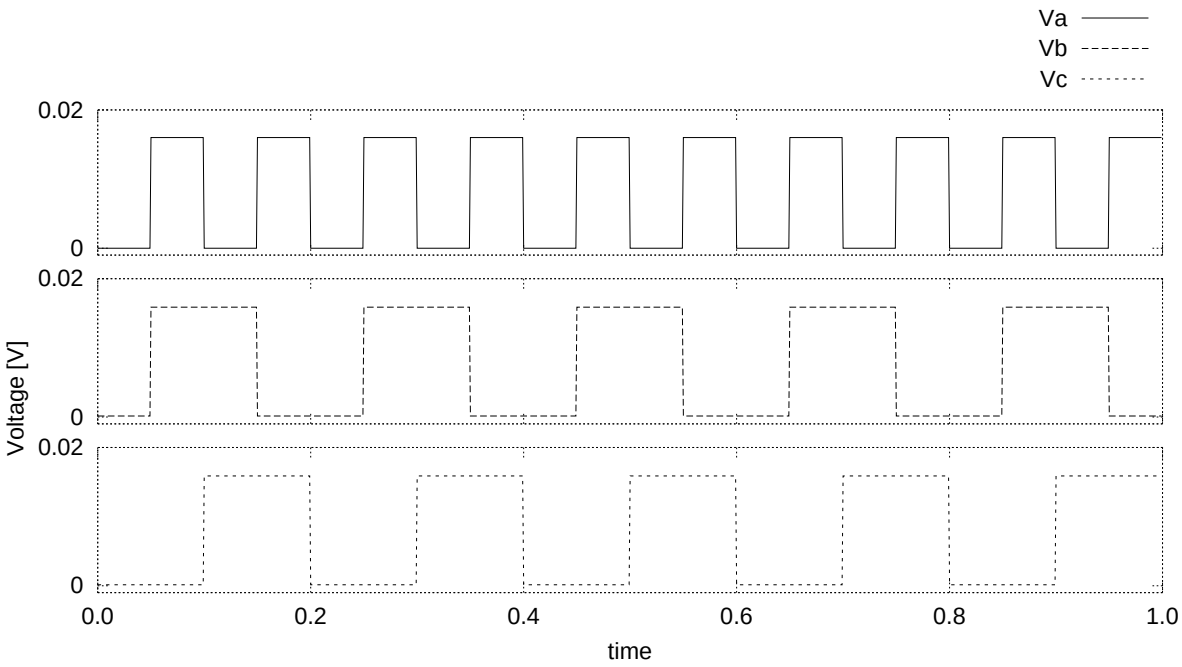


Figure 3.17: Simulation results for the Toggle

3.2 Networks of SET Based DI Building Blocks

"Computers are composed of nothing more than logic gates stretched out to the horizon in a vast numerical irrigation system."

- Stan Augarten

In the previous section we presented correctly functioning implementations of SET-based DI building blocks. In this chapter we investigate the possibility of combining the proposed circuits. Theoretically speaking, we can combine these building blocks together to achieve any desired higher level functionality. However, given that we have proposed passive networks, feedback and feed-forward effects may occur between the network elements. Thus we require buffering techniques to be able to connect the SET circuits together while having them maintain their correct behaviors. In [11] the feedback problem was analyzed and a solution was proposed, namely a buffering technique. In this chapter we first extend this technique such that it can be applied for the proposed DI basic building block. We then build two example networks using the introduced extended buffering technique and verify them by means of simulation.

3.2.1 Buffering Between Passive Networks

Due to the passive nature of the proposed building blocks feedback problems arise when attempting to connect the blocks to form circuits with a particular desired functionality. This problem was previously discussed in the context of Single Electron Encoded Logic (SEEL) circuits in [13] and a non-inverting buffer was proposed to be utilized as a front-end to SEEL gates with an input gate capacitance of $0.5aF$. Furthermore, the inverting buffer that is utilized in the implementations works only with input gate capacitances of $0.5aF$.

In the case of the proposed SET based DIC implementations the following buffering is required for the building blocks:

- The *C-Element* requires a non-inverting buffer for each input.
- The *Merge* (Boolean gate-based) block requires a buffer for each input. Additionally, an extra buffer is required to drive the inverting buffer to invert both signals.
- The *Merge* (state graph-based) has input gate capacitance of $1aF$. The input gate capacitances can be split in two by capacitive division. This results in a requirement of 4 buffers for a and 4 for b.
- The *Sequencer* requires two buffers for each of the three inputs a, b, and c. Additionally, an extra buffer for c is required to drive the inverted signal and a further two buffers for the two inverted c signals.
- The *Tria* requires 4 separate signals for a in the implementation as well as a further 4 for the inverted signal of a. This results in a requirement of 4 buffers for the a signal. Additionally, an extra buffer is required to drive the inverting buffer, and a further 4 buffers are required for the inverted signal $a\hat{}$. The same requirements hold for inputs b and c.

- The *Toggle*, has input capacitances of $1aF$ and uses two signals for a and two for $a\hat{}$. This results in a requirement of 4 buffers for a and 4 for $a\hat{}$ and an extra buffer to drive the inverting buffer.

Based on this analysis one can see that it is possible to use multiple buffers from [13] and combine them into an input signal. However, this will require a large amount of junctions. To reduce the buffer requirements we propose an extension for the non-inverting buffer in [13] such that the high demand for buffering can be reduced.

The design is essentially the same except that there are multiple islands serving as outputs instead of just 1, with each island connected to the input of the next block. Each island can drive a $0.5aF$ input gate capacitance as the original buffer was designed to do. Additionally, an island can be used as an input for an inverting buffer. This proposal is scalable and by varying the number of islands connected to the input various capacitances can be accommodated. To determine how many islands are required for a particular input signal in a particular building block the required total input gate capacitance can simply be divided by $0.5aF$. For input gate capacitances which are larger than $0.5aF$ the input gate capacitance can be split into multiple capacitors with a capacitance of $0.5aF$. The capacitors can each be connected to a separate island in the same buffer structure, and all the capacitors can be connected to the node serving as the input. Each extra buffer requires only 2 extra tunneling junctions and 1 capacitor, instead of an extra 4 tunneling junctions and 3 capacitors.

An example of such a buffer is depicted in Figure 3.18. A total input gate capacitance of $4aF$ can be driven with the first 4 islands. The fifth island is used to drive an inverting buffer. From the output of this inverting buffer the same buffer structure can be attached to drive a further set of capacitances where multiple instances of the inverted signal are required.

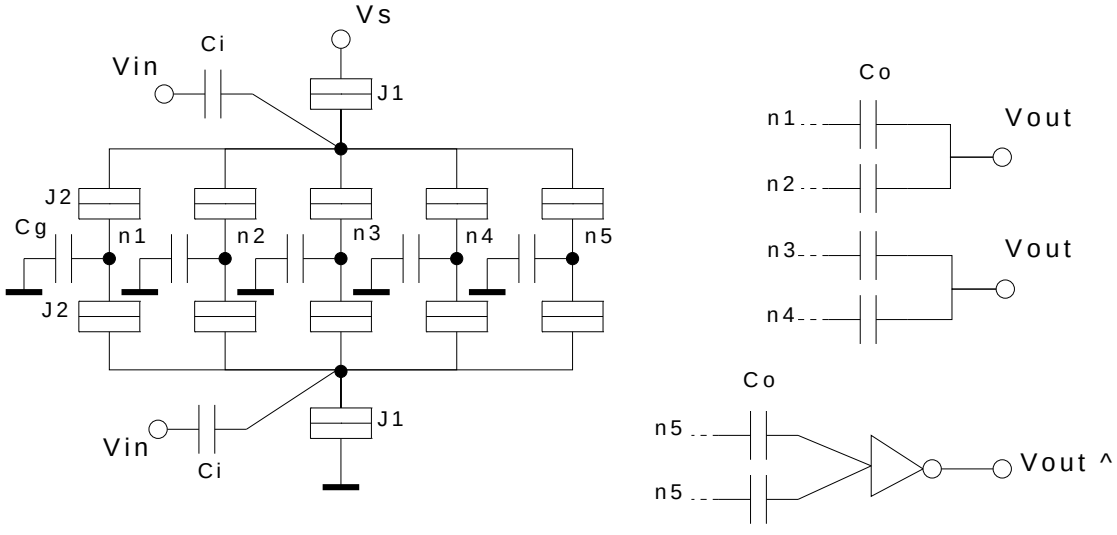


Figure 3.18: Extended buffer circuit diagram

The circuit parameters are presented in Table 3.10:

Capacitance(s)	Value (aF)
C_1, C_g	10
C_2	0.1
C_i	1
C_o	0.5

Table 3.10: Buffer circuit parameters

The example buffer circuit was verified by means of simulation. The results of the simulation are presented in Figure 3.19. The voltage on nodes $n1$, $n2$, $n3$, $n4$, and $n5$ are identical. We can observe that the voltage from V_{in} is duplicated on all the nodes. Furthermore, we can see that $V_{out}\hat{}$ is high when V_{in} is low and low when V_{in} is high, which is the desired inverter behavior.

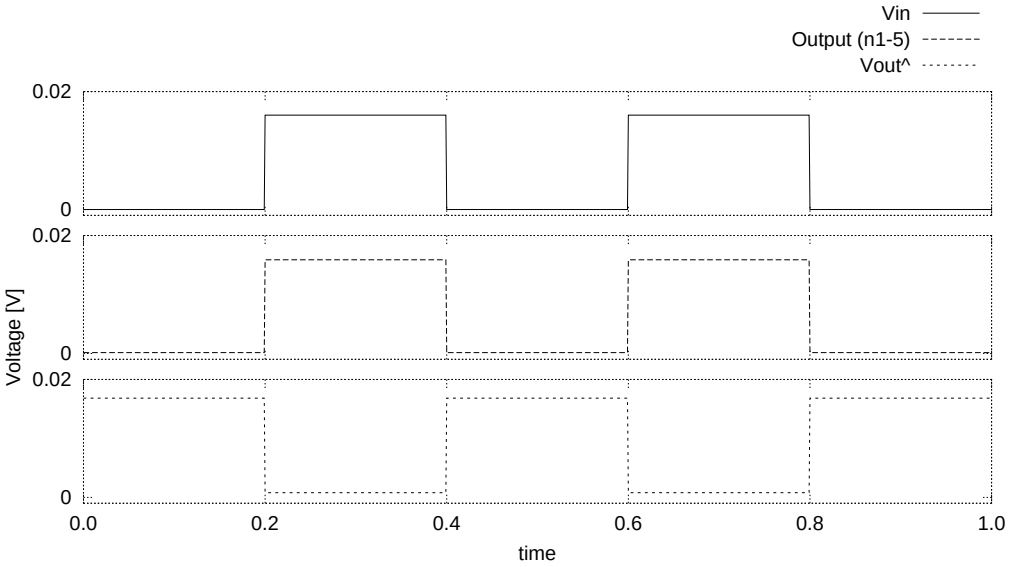


Figure 3.19: Simulation results for example buffer circuit

By augmenting the proposed DI building block with buffers we can make them function in larger networks. The buffering results in additional circuit requirements for each SET implementation, as summarized in Table 3.11.

In the following sections we utilize these proposed building blocks to construct networks using the proposed SET implementations presented in the previous chapter.

3.2.2 Quick Return Conversion Unit

The *Quick Return Conversion Unit*, or *QR42*, repeatedly converts a four-phase handshake from its left side environment to a two-phase handshake with its right side environment [18]. The construction is depicted in Figure 3.20. The trace theory specification is as follows:

DI Building Block	Buffering Required	Tunneling Junctions	Capacitors
C-Element	2 buffers	8	6
Merge (state graph-based)	2 four-island buffers	20	12
Merge (Boolean gate-based)	2 two-island buffers	12	8
Sequencer	2 two-island buffers	26	17
Tria	3 five-island buffers and 3 four-island buffers	66	39
Toggle	1 five-island buffer and 1 four-island buffer	22	13

Table 3.11: Circuit element requirements for buffering each SET implementation

$\mathbf{pref}[a?; b!; c!; a?; d?; b!]*$

The functional behavior of the *QR42* is as follows. A token arriving at a results in tokens being generated at b and c. Subsequently, another token arriving at a and one at d results in a token being generated at b only. This process then repeats itself.
This behavior can be exhibited by a network of three proposed building blocks: a *C-Element*, a *Merge*, and a *Toggle*. The construction of this network is presented in Figure 3.20. The blocks labeled 'B' are the buffers placed between the blocks. In total the construction requires 75 tunneling junctions and 69 capacitors including all the required buffers. The network construction including the buffers was verified by means of sim-

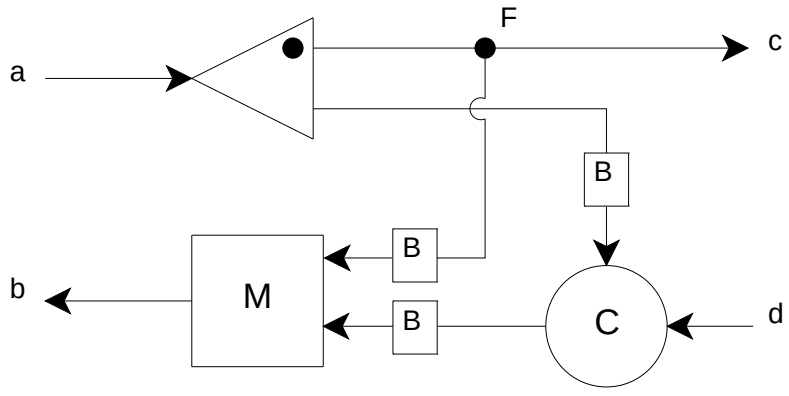


Figure 3.20: QR42 block diagram

ulations. The simulation results are depicted in Figure 3.21. We can observe that the correct *QR42* functionality is achieved by the combination of SET based building blocks. When a goes high b and c also go high. When a then goes low the *QR42* wait until d goes high after which b goes back low. When a goes high once more, outputs b and c

transition, with b going high and c going low. Subsequently, d goes low before a, and the *QR42* waits for a to go low this time, after which b goes low. With all inputs and outputs low the process then repeats itself.

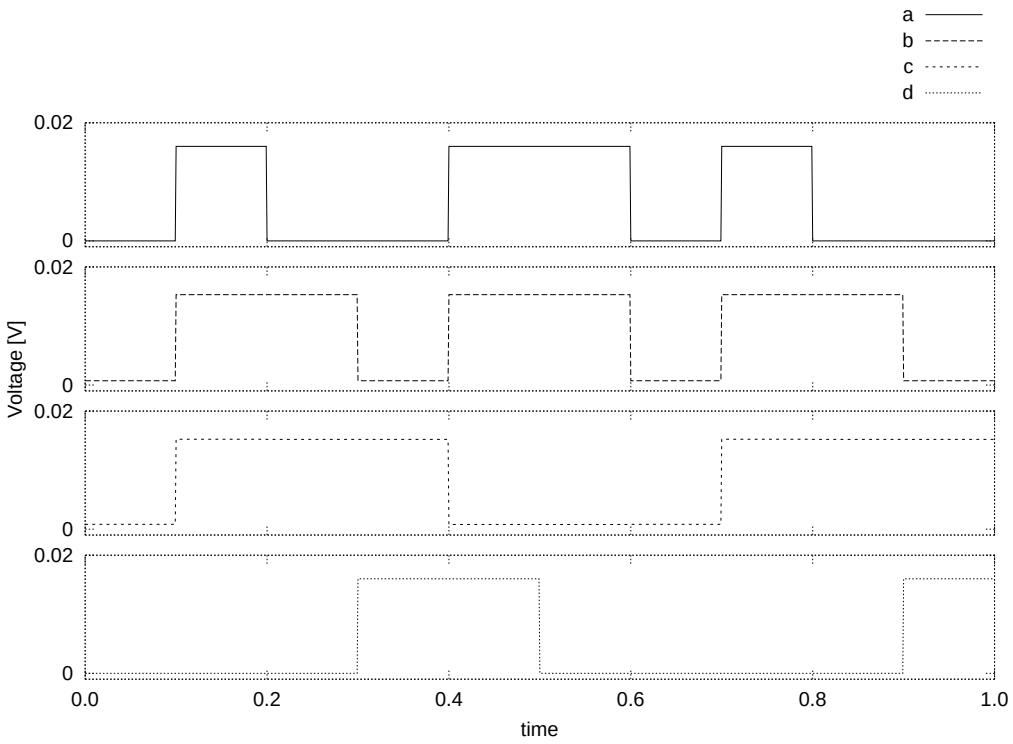


Figure 3.21: Simulation results for the QR42

Direct State Graph-based Realization

Alternatively, we can apply the state graph mapping to the *QR42*, as we did for the DI building blocks. Based on the described behavior we can derive a state graph for the *QR42*, as presented in Figure 3.22. We can see that the state graph is very similar to that of the *Toggle*. A SET circuit topology can thus be proposed, which is the same as that of the *Toggle* with a few changes. A new output has to be added with an *OR* gate evaluating the new output. Further, a buffer and a non-inverting buffer have to be added for the new input. Finally, the input capacitances have to be adjusted such that we obtain a non-reversible two input *C-Element*. The resulting topology is depicted in Figure 3.23. If we construct the SET topology in this manner we require 51 tunneling junctions and 51 capacitors.

3.2.3 Modulo-3 Counter

The *Modulo-3 Counter*, signals an output every 3rd transition, while signaling another output every 1st and 2nd transition [5]. The construction is presented in Figure 3.24. The trace theory specification is as follows:

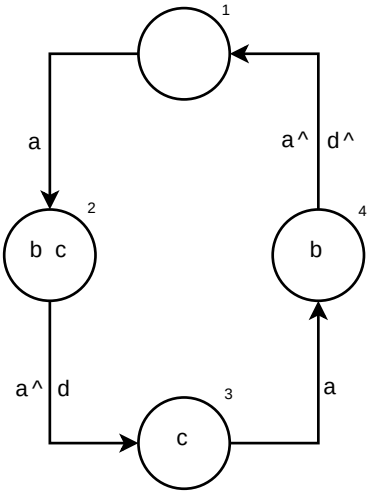


Figure 3.22: QR42 state graph

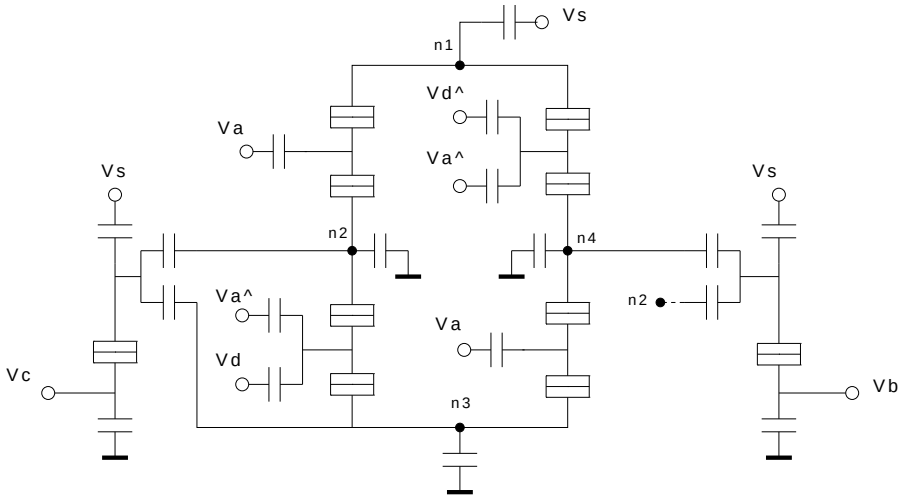


Figure 3.23: QR42 state graph based implementation

pref$[a?; q!; a?; q!; a?; p!]*$

The functional behavior of the *Modulo-3 Counter* is as follows. A token arriving at a results in a token generated at q and another token at a results in another token at q. Subsequently, a 3rd token at a results in a token at p after which the process repeats itself.

This behavior can be exhibited by a network of two proposed building blocks: a *Merge*, and two instances of the *Toggle*. The construction of this network is depicted in Figure 3.24. The blocks labeled 'B' are the buffers placed between the blocks. In total the construction requires 101 tunneling junctions and 94 capacitors. The network construction including the buffers was verified by means of simulations. The simulation results are

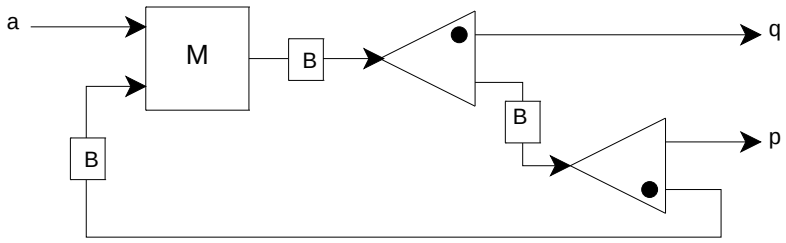


Figure 3.24: Modulo-3 counter block diagram

depicted in Figure 3.25. We can observe that the correct *Modulo-3 Counter* functionality is achieved by the combination of SET based building blocks.

The first pulse of a produces a pulse at q. When a then goes high, and this results in p going high. Subsequently, a going low and then high again results in another pulse at q. When a goes low again p goes low, and with all inputs and outputs low the process repeats itself.

With this example we can see that feedback wires in DI networks are dealt with in a correct manner.

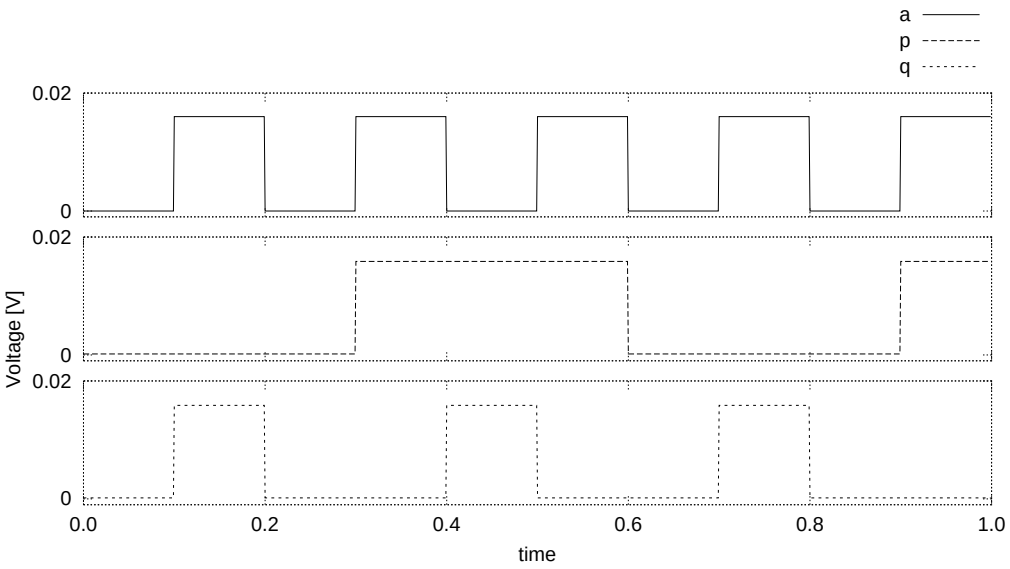


Figure 3.25: Simulation results for the Modulo 3-Counter

Direct State Graph-based Realization

Alternatively, we can apply the state graph mapping to the *Modulo-3 Counter*, as we did with the DI building blocks. Based on the described behavior we can derive a state graph for the *Modulo-3 Counter*, presented in Figure 3.26. We can see that the state graph is an extended version of the *Toggle* state graph with 2 new states and one new output. The resulting topology is depicted in Figure 3.27. If we construct the SET topology in this manner we require 45 tunneling junctions and 48 capacitors.

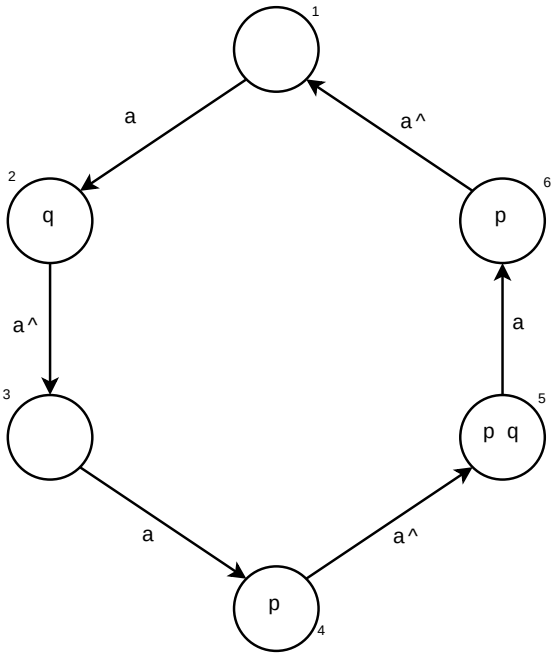


Figure 3.26: Modulo-3 counter state graph

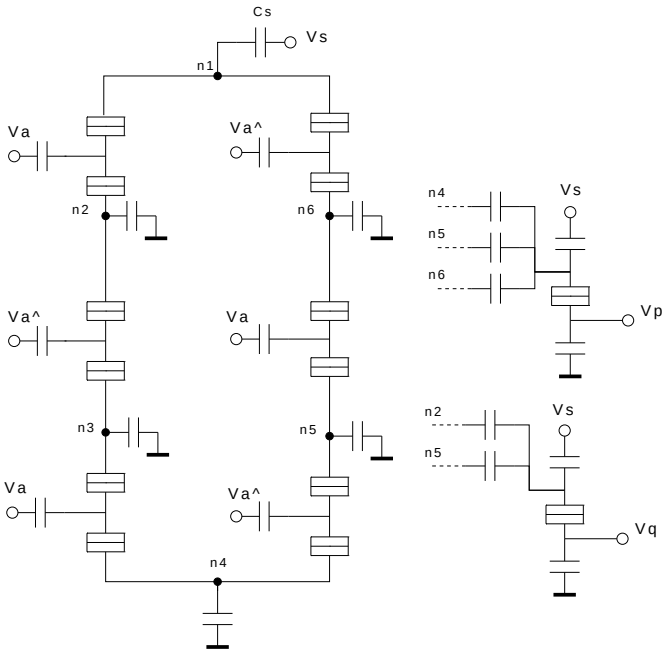


Figure 3.27: Modulo-3 counter state graph based implementation

3.3 Performance Evaluation and Comparison

A summary of the amount of required circuit elements for each DI basic building block SET implementation is reported in Table 3.12. The area is presented in terms of the

number of required circuit elements, composed of tunneling juctions and capacitors. As one can observe, there is a significant amount of area overhead associated to the element buffering. The requirement for buffering ranges from approximately 97% to 125% of the area of the block itself.

There is room for improvements of the implementations, specially where the buffering is concerned. Devising methods to reduce the required buffering would significantly reduce the required area.

DI Component	Area for component	Area for buffering	Total Area
C-Element	6	14	20
Merge (state graph-based)	19	32	53
Merge (Boolean gate-based)	42	20	62
Sequencer	39	43	82
Tria	84	105	189
Toggle	36	35	71

Table 3.12: Circuit element requirements for buffering each SET implementation

Energy and delay

We are dealing with asynchronous circuits and so the delay of the gates does not matter for the circuit functionality. However, we can evaluate the switching behaviour of the gates to gauge the performance of asynchronous circuits implemented in SET technology. The delay depends on the error rate and the number of tunneling events that occur. For every DI element there can be a number of different circuit scenarios leading to a wide range of input output switching events. We thus give the delay in terms of the delay due to tunneling when there is a switching of the output. We note here that the circuit designs are not optimized and the following calculations are for illustrative purposes only.

If we take the *C-Element* as an example case, we can see that two tunneling events occur once the output changes. The delay of the first tunneling event, through $J2$, is similar to that of an *AND* gate from [12] and for the assumed error rate it is $0.77ns$. For the second tunneling event, through $J1$, we can utilize Equation (2.4) to calculate a minimum delay. This results in an extra delay of $0.38ns$, meaning that the total delay for a *C-Element* is $1.15ns$. For the energy consumption a similar calculation is performed using Equation (2.1). The *AND*-gate consumes $0.76meV$ and the second tunneling event consumes $1.25meV$. Thus, the total energy required for the output to switch is $2.01meV$.

The state graph-based *Merge* exhibits similar behaviour to the *C-Element* and the calculations for delay and energy are also similar. The delay and energy consumption of the *OR* gate have to be added as well, taken from [12]. The total delay for the *Merge* can be estimated to be $2.00ns$, and the total energy consumption per switching event is $2.71meV$.

The *Sequencer*, *Tria* and *Toggle* have similar tunneling events for each output switching event and can be assumed to have delays and energy consumption of the same order as the *Merge* for each output switching event.

We also calculate the delay and energy consumption of the Boolean gate-based *Merge*. We assume that tunneling events occur in one *AND* gate, the *OR* gate and the inverter for the switching of an output. Adding the delays results in $2.02ns$ and adding the energy consumption results in a value of $12.36meV$. By following this computation approach one can determine the delay and the energy consumption for any DI SET based circuit.

Alternative Merge Implementations

In Section 3.1.2 we presented two implementations for the *Merge* block. The first was based on a state graph realization and the second utilized Boolean gates to arrive at an *XOR* functionality. We can see in Table 3.12 that the state graph-based implementation requires 9 less elements and we can conclude that the state graph-based method is a more efficient method to arrive at DI block implementations. The delays of the two implementations are almost the same, while the energy consumption is much higher for the Boolean gate-based implementation. We have not taken into account the energy consumption of the buffering however. If we assume a similar energy consumption for the state graph-based implementation buffer the energy consumption values are almost equal. For the rest of this thesis when we refer to the '*Merge*' we refer to the state graph-based realization of the *Merge*.

Direct state graph realization versus block decomposition

In Sections 3.2.2 and 3.2.3 two example networks were introduced and verified.

We found that implementing the network based on the desired functionality, by means of a direct realization from the state graph, was also possible. This resulted in a more compact circuit and the area comparisons for the *QR42* and *Modulo 3-Counter* are presented in Table 3.13. We can observe that a significant percentage of the area is saved if we use the state graph method of realizing SET topologies for DI networks.

When constructing higher level DI circuits it is thus advantageous to identify higher level building blocks and implement SET circuit designs for those blocks utilizing the state graph realization method. This optimization method can save significant amounts of chip area, with 30% and 52% being saved in the example cases. However, when implementing complex DI circuits the construction of the state graph may not be such an easy task.

	QR42	Modulo 3-Counter
Area with Building Block realization	144	195
Area with State Graph realization	102	93
Percentage area saved with State Graph realization (%)	30	52

Table 3.13: Circuit element requirements for buffering each SET implementation

3.4 Conclusion

In this chapter we have investigated the design of SET based DI building blocks and the construction of SET networks utilizing these building blocks. We proposed SET topologies for 5 DI building blocks and verified them by means of simulations. We then presented an extension of the buffering technique in [13] to enable the construction of networks made from the proposed building blocks in SET technology. We also verified these networks with simulations.

In order to demonstrate that these blocks can function correctly in a network structure, we implemented 2 examples of DI compositions, which lead to functionalities that are utilized in the DI paradigm. The second example contained a feedback wire in the network thus it was proving that theoretically any combination of the building blocks can be constructed.

In the next chapter we follow the same procedure for the second delay-insensitive based paradigm based on Brownian circuits.

4 SET Based Brownian Circuits

"Creativity is the ability to introduce order into the randomness of nature."

- Eric Hoffer

In this chapter we present SET implementations of Brownian circuits. In Section 4.1 we propose SET-based circuit implementations of the building blocks for the Brownian circuit paradigm. In Section 4.2 we present a method to connect the building blocks together to form larger networks. We then present an example of a network that utilizes the building blocks and the described method of connecting them. In Section 4.3 we evaluate and discuss the proposed implementations and networks. We also make a comparison between the SET implementations of DI and Brownian circuit paradigms based on NAND-gate circuits. Finally, in Section 4.4 we provide an overview of the chapter.

4.1 Implementations of Brownian Circuit Building Blocks

In the Chapter 2, two components which serve as building blocks for the Brownian circuits paradigm were introduced. The objective of this research is to arrive at SET implementations of these building blocks. In this section we present the SET implementations and how they were found. In particular, we present the implementations of two building blocks, the *Hub* and the *Conservative Join*, which form a universal set of primitives for Brownian circuits [24].

For the Brownian circuits paradigm we chose to represent input and output tokens as *physical ballistic tokens*. We used a positive charge of value q_e as a physical representation of a token. The SET circuit topologies are constructed based on the paths the input charges must take. The circuit parameters are derived such that we obtain the desired charge transport and afterwards the circuits are verified by means of simulation, which were performed using the SIMON 2.0 SET circuit simulation software [29].

The same considerations and notations apply that were introduced in Section 3.1. In the following sections, the implementations of each of the selected building blocks for the DI paradigm are presented as follows:

1. The building block is introduced with the symbolic representation and the functional behavior of the block described in terms arriving input tokens and the resulting sequence of tokens through the component.

2. A SET-based circuit topology is proposed for the building block, which has the potential to deliver the required behavior.

3. The behavior that the SET-based circuit should exhibit is described in terms of electron tunneling events.

4. The derivation of the circuit parameters to achieve the desired circuit functionality is presented.

5. The results of the circuit simulations are presented and discussed.

4.1.1 Hub

Functional Behavior

The *Hub* is essentially a routing mechanism enabling the transport of tokens from one block to another. Based on a random scheme of signaling, the *Hub* requires fluctuations to drive its operation.

The *Hub* consists of three outputs. Any one of these can serve as a momentary input as well. If a token arrives at any output wire the token enters the *Hub*. The *Hub* repeatedly offers this token at its output terminals, and takes it back when it cannot be delivered or when it is not consumed by other building blocks. There will be at most one token at a time on any of the *Hub*'s wires, and this token can move to any of the wires due to fluctuations. In Figure 4.1 the circuit symbol of the *Hub* is depicted. All three states are shown with the possible transitions and the resulting location of the token.

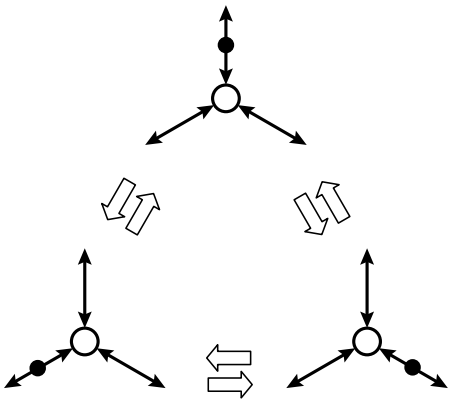


Figure 4.1: Hub transitions. Fluctuations cause a token to move between any of the Hubs three wires in any order.

Proposed SET-based Circuit Topology

Figure 4.2 presents a SET circuit topology that can exhibit the *Hub* functionality. For simulation purposes, we need to generate a token to inject into the *Hub* and this is achieved by a *C-Element* structure with one input. The *Hub* consists of three *C-Element* structures all attached to a common node, $n4$, through which the positive charge can be transferred to other nodes. In this case the token is supplied directly into the center of the *Hub*. Alternatively, the token supply could be connected to any of the nodes $n1$, $n2$, and $n3$. In total the circuit comprises of 6 tunneling junctions and 7 capacitors.

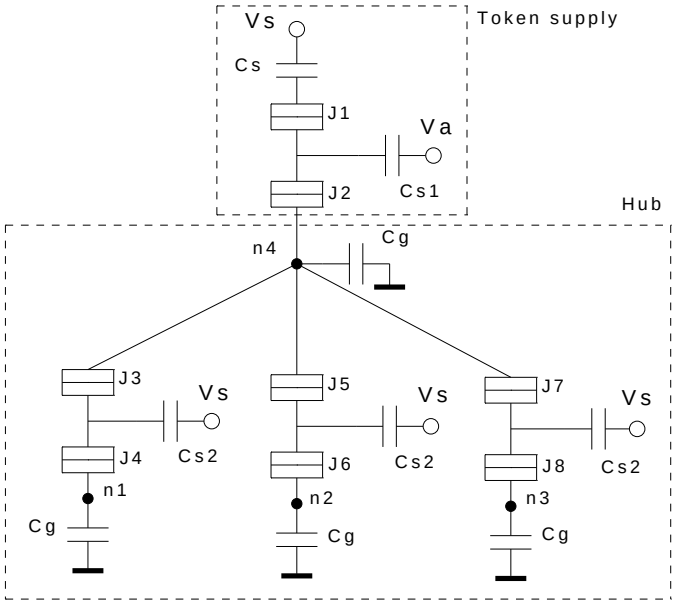


Figure 4.2: Hub circuit topology, with topology to supply a token

Required Circuit Behavior

The circuit should operate as follows. When V_a is high an electron travels away from $n4$ through $J2$ and $J1$. Once there is a vacancy on $n4$, one of the two SET structures, $J3-J4$, $J5-J6$ or $J7-J8$ supplies an electron to node $n1$, from $n2$ and $n3$, respectively. However, $Cs2$ is chosen with a low value such that the thermal energy of the electron is sometimes enough to overcome the voltage differences and the electron tunnels back into $n1$, $n2$ or $n3$. The charge thus jumps from $n4$ to $n1$, $n2$ or $n3$ and then back into $n4$ and then again randomly into $n1$, $n2$ or $n3$ as it should according to the *Hub* functional specification.

Circuit Parameters Derivation

As was previously mentioned in Chapter 2, the thermal energy should not dominate the circuit elements. However, to achieve the desired *Hub* functionality, there should be enough thermal energy fluctuations to enable electrons to tunnel from one island to another. In other words, the circuit has to be sensitive to thermal energy.

This was achieved by bringing the voltages over certain tunneling junctions close enough to their critical voltages such that an increase in the energy of an electron at the junction due to the extra thermal energy would at random moments cause the electron to tunnel forward. When the thermal energy decreases the electrons then tunnel back again. The thermal energy is thus effectively used as a random control voltage.

Circuit Parameters

The circuit parameters are summarized in Table 4.1.

Capacitance(s)	Value (aF)
C_1, C_3, C_5, C_7	10
C_2, C_4, C_6, C_8	0.01
C_s,C_g	10
C_{s1}	0.5
C_{s2}	0.2

Table 4.1: Hub circuit parameters

Simulation Results

The results of the simulation can be seen in Figure 4.3. The circuit is stable when there is no supply of a charge. After time step 0.1, when the charge is supplied, we can see a random traveling of the charge into and out of nodes $n1$, $n2$ and $n3$.

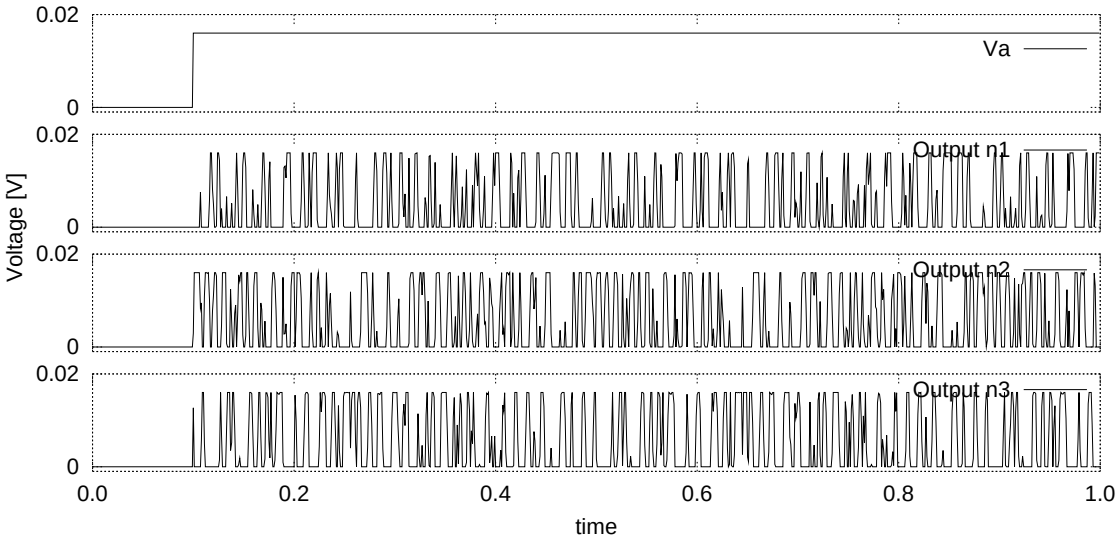


Figure 4.3: Hub circuit simulation results

4.1.2 Conservative Join

Functional Behavior

The *Conservative Join*, or *CJoin*, can be interpreted as a synchronizer of two signals passing through it. It has two input wires and two output wires (Fig. 4.4). When only one token is present at the input wires, the *CJoin* remains inactive. When tokens are supplied to both the input wires simultaneously, the *CJoin* fires and takes the two tokens and places them at the output while not allowing them to return back.

Tokens may fluctuate on the input wires of a CJoin, leaving and arriving randomly, but once processed by the *CJoin*, they will be placed on the output wires with no way back, while fluctuations on the output wires are allowed. We would like to mention that when connecting *CJoins* to each other, we should make sure that input terminals face

output terminals. *Hubs*, having bi-directional wires, may be connected in any way to *CJoins* or other *Hubs*.

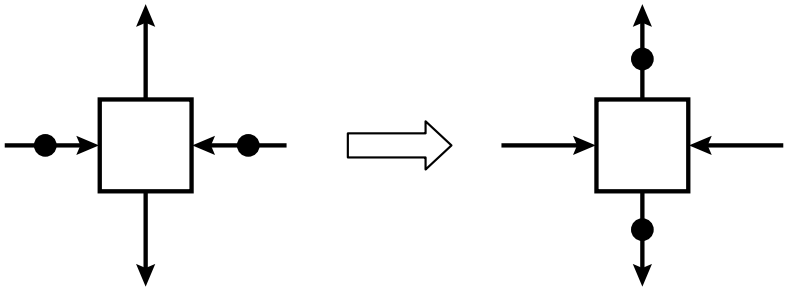


Figure 4.4: CJoin symbol

Proposed SET-based Circuit Topology

Figure 4.5 presents a SET circuit topology that implements the *CJoin*. The design of the *CJoin* is based on the non-inverting buffer design introduced in [13]. When tokens simultaneously arrive at the inputs the circuit draws a positive charge from the source placing it on the output of a buffer structure. This output drives the tokens residing at the input to the output and when this occurs the positive charge is grounded. The circuit comprises in total of 10 tunneling junctions and 13 capacitors.

Required Circuit Behavior

The circuit operates as follows. When inputs V_a and V_b go high, electrons tunnel through the junctions $J1 - J2$ and $J5 - J6$ leaving positive charges on nodes $n1$ and $n3$, respectively. When $n1$ and $n3$ simultaneously have a charge, an electron tunnels from node $n5$ into the source through junctions $J9$ and $J10$. Subsequently, an electron tunnels from $n6$ to $n5$ due to the SET transistor $J11 - J12$, which acts as a buffer and separates the influences of the input $Cn1$ gate capacitors and the $Cn2$ driving gate capacitors. Now the charge on node $n6$ causes the charge on $n1$ to be transferred into $n2$ through $J3$ and $J4$ and the charge on $n3$ to be transferred into $n4$ through $J7$ and $J8$. Nodes $n2$ and $n4$ are the output nodes and the input tokens, represented as positive charges, are thus transferred to the output. To bring the circuit back to a reusable state, the charge remaining on $n6$ has to be removed. This is achieved by connecting $n6$ to ground via a reversed transistor structure, $J14 - J13$. If there is a positive charge residing on $n6$ and the charges on $n1$ and $n3$ become zero an electron tunnels from ground into $n6$, resetting the circuit. Once the output charges on $n2$ and $n4$ have been consumed the circuit is then ready to accept new input tokens.

Circuit Parameters Derivation

The *CJoin* has to have a deterministic response thus it has to be designed in such a way that the thermal energy has little or no effect on its behavior. The parameters were taken from the non-inverting buffer in [13] and adjusted to deal with the thermal effects.

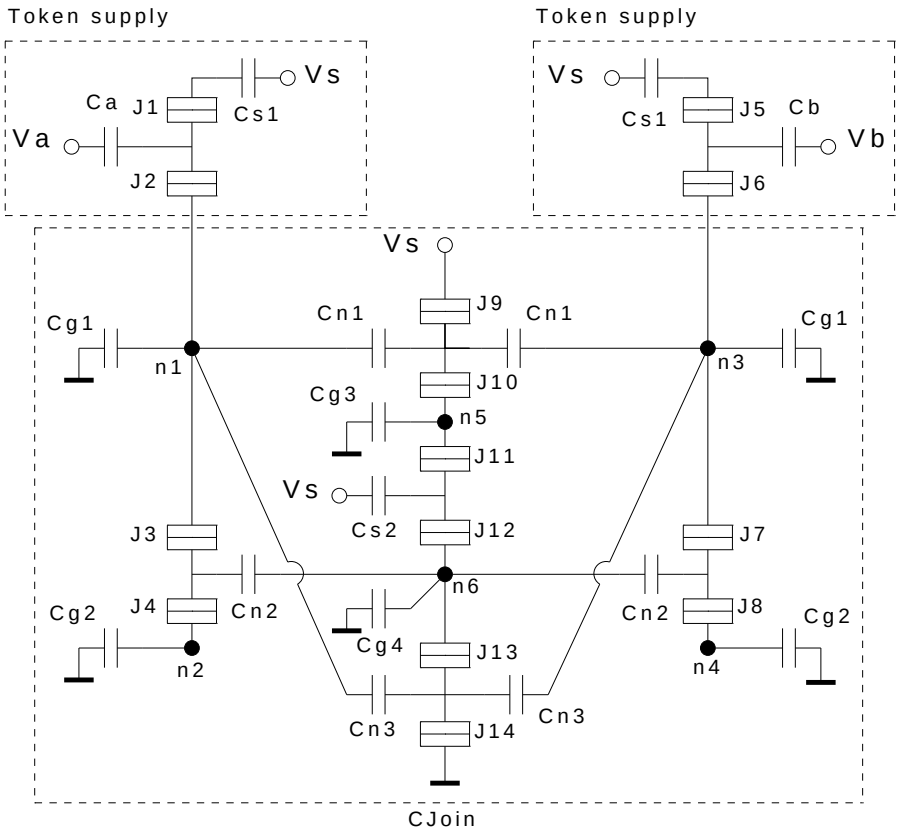


Figure 4.5: Conservative Join circuit implementation

Circuit Parameters

The circuit parameters are summarized in Table 4.2.

Capacitance(s)	Value (aF)
C_a, C_b	1
C_{s2}	0.5
C_{n1},C_{n3}	0.25
C_{n2}	0.3
C_{g2}	11.5
C_{g3}	10.5
C_1, C_3, C_5, C_7, C_{11}, C_{s1}, C_{g1},C_{g4}	10
C_9, C_{14}	5
C_2, C_4, C_6, C_8, C_{10}, C_{12}, C_{13}	0.01

Table 4.2: CJoin circuit parameters

Simulation Results

The results of the simulation can be seen in Figure 4.6. Charges appear on the output nodes $n2$ and $n4$ only when both Va and Vb go high, but not when only one of them goes high.

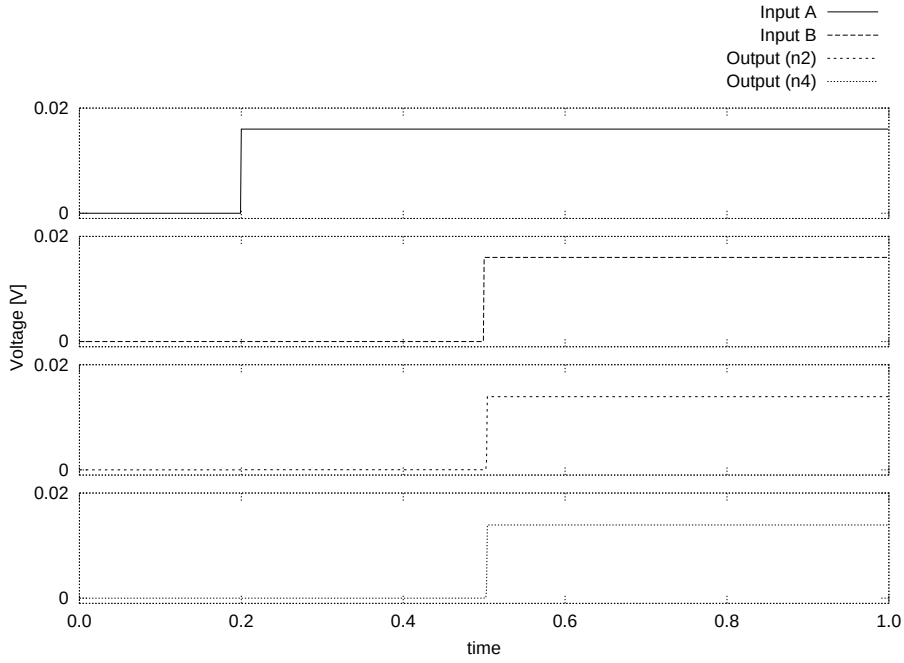


Figure 4.6: Conservative Join circuit simulation results

4.2 Networks of SET Based Brownian Circuit Building Blocks

4.2.1 Example: CJoin with Hubs

Even though the two circuit topologies function as required at the same temperature, simply connecting them results in crosstalk, which affects the functionality. To retain the random fluctuations in the *Hub* an adjustment need to be made in the circuit parameters. The circuit parameters of the *Hub* that have to be adjusted are C_{s1}, C_{s2}. C_{s1} has to be changed to $1aF$ and C_{s2} has to be changed to $0.7aF$. Also, the grounded capacitance C_g, corresponding to the unconnected node of a *Hub* ($n1$, $n2$ or $n3$) have to be changed to $10.4aF$. This is required to balance out the extra capacitance $n1$ has added on due to the extra capacitance from being connected to the *CJoin* circuit. The circuit parameters of the *CJoin* remain unchanged.

To demonstrate that these building blocks can be combined into one functional circuit we use an example network of two *CJoins* and three *Hubs* (Figure 4.7), with one *Hub* connected to both *CJoins*. With signals present on all Hubs it is a matter of chance as to whether it is the first *CJoin* that fires or the second one. The common token (at

the center) is consumed by the firing *CJoin*, leaving the other (non-firing) *CJoin* with only one token. A number of simulations were done using different random generator seeds and the results of two of these simulations are presented in Figures 4.8 and 4.9. In both simulations outputs which remain '0' are left out. Figure 4.8 presents the charge distributions on A and B (corresponding to the left Hub) and C and D (right Hub); we see fluctuations on both Hubs. The central *Hub* is not supplied with a token until time 0.5 and at this point the *CJoin* with outputs E and F fires and the charges are trapped at its output. Figure 4.9 depicts a similar behavior in the second simulation, except that now the *CJoin* with output G and H fires, trapping the charges at its respective outputs.

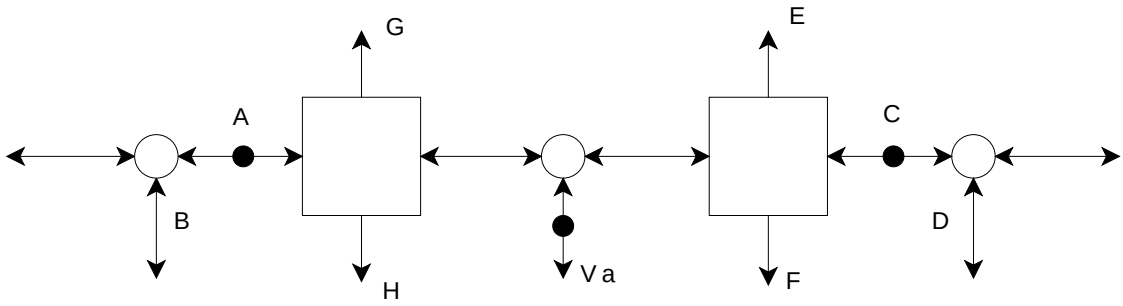


Figure 4.7: Conservative Joins with Hubs Network Example

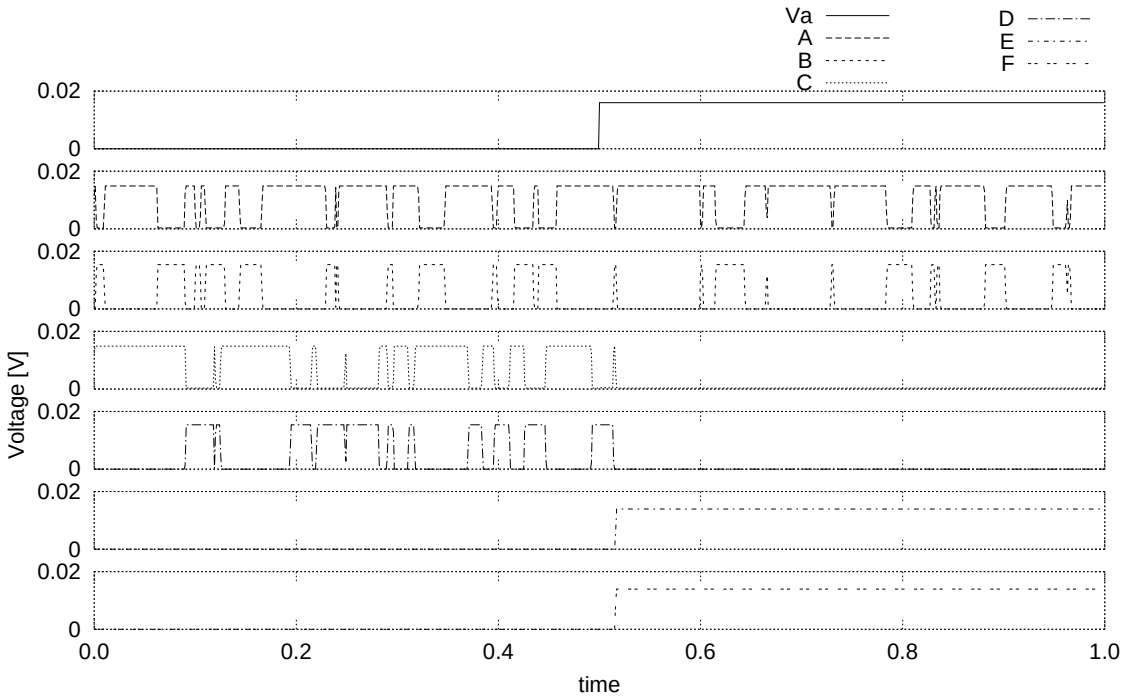


Figure 4.8: Conservative Joins with Hubs simulation 1

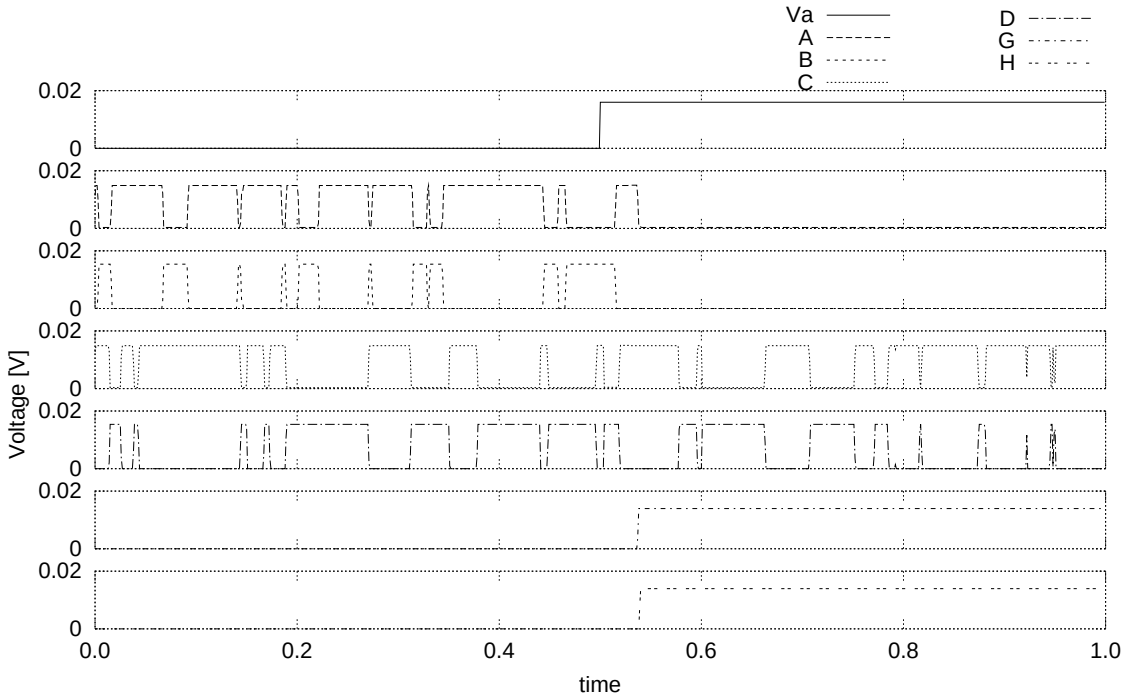


Figure 4.9: Conservative Joins with Hubs simulation 2

4.3 Discussion

We have demonstrated that the *CJoin* and the Hub can be implemented in SET technology and that they can coexist at the same temperature (1 Kelvin in this case) while maintaining their desired behaviors. These two blocks constitute the foundation of building Brownian SET based circuits, which is a computation paradigm able to turn the unpredictable SET behavior into an useful feature. Due to the nature of the thermal fluctuations there is a robustness issue, in that a thermal spike may cause the *CJoin* to malfunction. This could be dealt with by adjusting the parameters of the *CJoin* in such a manner that it becomes less sensitive to random thermal spikes. This, however, might require the *Hub* part to contend with a lower relative thermal control voltage, and would lead to a slower switching of electron charges. Thus a trade-off may have to be made between robustness and speed, though the charge fluctuations rate is high enough not to degrade the overall performance.

In this chapter we have demonstrated that SET technology is very suitable for implementing fluctuation-based calculations. The proposed architecture has the potential to reduce the strain on the fabrication technology by relaxing the constraints on the allowed thermal energy, and it allows implementations that are efficient and straightforward. Moreover, the delay-insensitive nature of the operations and the robustness to errors reduce the demands on the technology, underlining the promises of the proposed avenue.

4.3.1 Comparing the DI and Brownian Circuit Paradigms

To compare SET-based implementations within the two paradigms, we utilize a dual-rail encoded *NAND*-gate as the vehicle of analysis.

For the DI paradigm an optimized circuit design has been proposed for such a *NAND*-gate in [25] and it is depicted in Figure 4.10. In the figure a triangle represents a *Tria*, a diamond represents a *Merge* and a circle represents a *Fork*. For this circuit design two *Trias*, five *Merges*, and five *Forks* are required.

For the Brownian circuits paradigm an optimized circuit design has been proposed in [14] and it is depicted in Figure 4.11. For this circuit design four *CJoins* and six *Hubs* are required. Implementing both designs in SET technology using the implementations presented in Sections 3.1 and 4.1 results in the following area requirements: The DI *NAND*-gate requires 643 circuit elements and the Brownian *NAND*-gate requires 164 circuit elements.

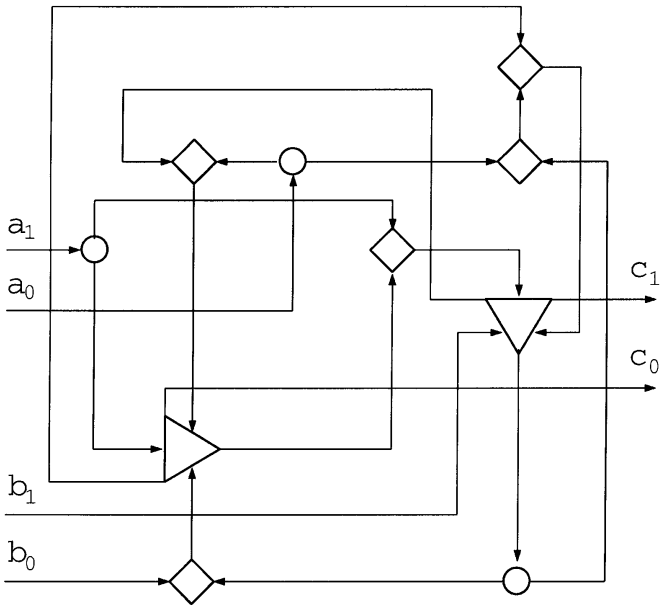


Figure 4.10: Delay-insensitive dual-rail encoded NAND-gate

We can see that the Brownian circuits implement a *NAND*-gate much more efficiently area-wise. However, the potential to build increasingly larger circuits is much higher with the DI paradigm implementations. With the proposed Brownian circuit SET implementations the effects of cross-talk could be ultimately limiting. Furthermore, we observed in Sections 3.2.2 and 3.2.3 that more efficient SET circuit implementations of the larger networks could be achieved. This was done by directly implementing the functionality of the network in a SET topology. The *NAND*-gate implementation for the DI paradigm could thus be optimized by utilizing higher level building blocks or by directly proposing a SET topology for the *NAND*-gate functionality.

To compare the delays of the two *NAND*-gates, we analyze the behaviour of the circuit with the arrival of two input tokens. We assume the circuit to be in the idle

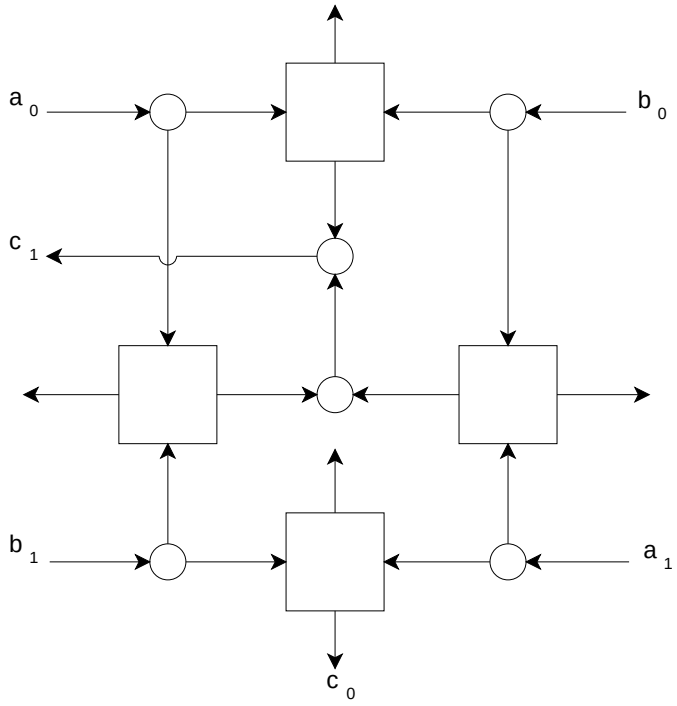


Figure 4.11: Brownian dual-rail encoded NAND-gate

state with all inputs and outputs low and no tokens residing within the construct. This means that no charge transport has occured in the implementations. We then analyze the behaviour of the internal blocks, and the tunneling events that occur within each block, with the arrival of a token at a_0 and b_0 for both *NAND*-gates. The *Hubs* actually operate in a random manner and so there is an extra delay before the *Hub* outputs 'find' each other. However, we simplify the analysis by assuming that the *Hubs* operate in the best case scenario and directly 'find' each other with no delay due to the random fluctuations.

For the DI *NAND*-gate this results in a total of five output switching events for the *Merge* and two for the *Tria*. Based on the SET implementations for the building blocks we can observe that 3 tunneling events result for an output switching event for each building block, resulting in 21 tunneling events. If we include the tunneling events that occur in the buffer constructions, this leads to an additional 8 tunneling events for the *Merge* blocks and an additional 12 for the *Tria* blocks. The total amount of tunneling events that occurs with the arrival of a token at a_0 and b_0 is thus 41.

For the Brownian *NAND*-gate there are three output switching events for the *Hub* and one for the *CJoin*. Based on the SET implementations for the building blocks we can observe that 2 tunneling events are required for one *Hub* switching event and 10 for one *CJoin* switching event. Thus, the total amount of tunneling events that occurs after the arrival of a token at a_0 and b_0 is 16.

Assuming the tunneling delays for all the tunneling junctions are of the same order, we can see that delay-wise the SET implementation of the Brownian circuit paradigm

seems to be faster. We did not include the delay due to the fluctuations of the *Hub* and this may slow down the *Hub* significantly, given that a minimum of two tunneling events would be required for the *Hub* signals to 'find' each other.

4.4 Conclusion

In this chapter we have investigated the design of SET based Brownian circuit building blocks and the construction of SET networks utilizing these building blocks. We proposed SET topologies for two Brownian circuit building blocks, which form a universal set, and verified them by means of simulations. We then presented a method to enable the construction of networks made of the proposed building blocks in SET technology. We also verified these networks via simulations.

In order to demonstrate that these blocks can function correctly in a network structure, we implemented an example of Brownian circuit. We discussed the implementations and the network construction, and evaluated the characteristics of the circuits. Finally, a comparison was made between the DI and Brownian circuit paradigms by evaluating a *NAND*-gate circuit constructed using building blocks of the two paradigms.

5 Conclusions

"Absolutum obsoletum."
(If it works it's out of date.)
- Stafford Beer

5.1 Summary

In this thesis Single Electron Tunneling (SET) implementations of two delay-insensitive circuit paradigms were investigated. A summary of the investigations and the results is as follows.

In Chapter 2 we presented some background theory on SET technology and Delay-Insensitive Circuits.

In Chapter 3 we proposed implementations of the following building blocks for the Delay-Insensitive (DI) paradigm in SET technology: the *C-Element*, the *Merge*, the *Sequencer*, the *Tria*, and the *Toggle*. We then proposed buffering techniques to enable the combination of these SET-based building blocks to form larger networks. We presented implementations of two example networks utilizing the proposed buffering techniques. The circuit topologies and networks were verified by means of simulations. We found that we can implement higher level DI functions by decomposition into basic building blocks implemented in SET technology. We also found that if we directly implement higher level DI functions in SET technology we can achieve more efficient realizations in SET technology.

In Chapter 4 we proposed the implementation of the following building blocks for the Brownian circuit paradigm in SET technology: the *Hub* and the *Conservative Join*. We demonstrated that blocks driven by signal fluctuations can coexist with blocks exhibiting deterministic behavior. We presented a method to form network contructions using SET-based Brownian circuit building blocks. We presented an example of a network using the proposed method to connect the SET-based Brownian circuit buildings blocks together. The proposed circuits and networks were all verified by means of simulations and proved that the proposed SET topologies can deliver the required behavior. Further in this chapter we made a preliminary comparison between the two paradigms based on a *NAND*-gate implementation. Comparing the implementations of the two paradigms we observed that the implementations of Brownian circuits in SET potentially require significantly less area when compared with equivalent DI implementations. Furthermore, due to the sensitivity of SET technology to environmental variations such as temperature the Brownian circuit paradigm appears to be promising, and an implementation in SET technology could potentially be more robust against effects which cause electrons to behave in a random manner. However, the implementations for the DI paradigm, though

requiring more area, can potentially be connected together to form larger networks than the Brownian circuit implementations.

5.2 Future Research

We have shown that there is definite potential in realizing delay-insensitive circuits in SET technology and as a continuation of the research presented in this thesis we suggest the follow directions for future research:

- All designs have been presented based on a simplified model of SET technology, without taking into account effects such as random background charge. It has been shown that theoretically DI circuits can function correctly using SET technology, but it would be of interest to research the practical applicability of the proposed designs when taking into consideration variations that could occur in a physical structure.
- Since the mapping of DI blocks onto SET circuits is fairly straightforward, design software could be developed that synthesizes SET circuit based on a construction of DI building blocks. The scope for optimization would be large, as functionalities could be directly mapped, or mapped by decompositions into basic blocks.
- In this research the building blocks for the DI paradigm were proposed to function with signal *transitions*. These could alternatively also be implemented based on a direct token representation, as it was done with the Brownian circuits.
- DI circuits can be further simplified in terms of numbers of inputs and outputs required for basic buildings blocks, by including *Bi-directional* and *Buffering* lines. The set of building blocks utilizing these concepts, [15, 16], can be developed in SET. It would be interesting to see if the feedback problem apparent in most of the currently proposed SET topologies, could be made into an advantage when implementing *Bi-directional* lines.
- The SET topologies proposed in this thesis made use of SET transistor-like structures. This results in twice as many capacitive islands as there are required states for DI blocks. It would be worthwhile to investigate whether a more compact topology can be utilized to create DI blocks, with electrons tunneling through only one junction to represent state transitions.
- Another method to reduce the amount of required islands could be to utilize multiple electrons to remember the states of the circuit. This could possibly be done by using electron counting, or by having two or more electrons moving around a circuit, or a combination of both. This could result in more efficient topologies specially for the large building blocks.
- Even though DI circuits function correctly with variations which lead to changes in signal and gate delays, at the nanometer scale defects and faults are almost unavoidable, and thus fault tolerance algorithms will have to be investigated for DI systems, with the focus on the non-arrival of tokens.

Bibliography

[1] *International technology roadmap for semiconductors, 2007 edition, emerging research devices*, 2007, Downloadable from website http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[2] *International technology roadmap for semiconductors, 2007 edition, executive summary*, 2007, Downloadable from website http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[3] N. Asahi, M. Akazawa, and Y. Amemiya, *Single-Electron Logic Systems Based on the Binary Decision Diagram*, IEICE Transactions on Electronics **E81-C** (1998), no. 1, 49–56.

[4] S. Cotofana, C. Lageweg, and S. Vassiliadis, *Addition Related Arithmetic Operations via Controlled Transport of Charge*, IEEE Transactions of Computers **54** (2005), no. 3, 243–256.

[5] A. Davis and S. M. Nowick, *An introduction to asynchronous circuit design*, Tech. report, Computer Science Department, University of Utah, 1997.

[6] S. Diner, D. Fargue, G. Lochak, and F. Selleri, *The wave-particle dualism - a tribute to louis de broglie on his 90th birthday*, D. Reidel Publishing Company, 1984.

[7] Bertrand Duplantier, *Brownian motion, "diverse and undulating"*, 2005, p. 201.

[8] J. C. Ebergen, *Translating programs into delay-insensitive circuits*, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, The Netherlands, 1989.

[9] Jo C. Ebergen, *A formal approach to designing delay-insensitive circuits*, Distrib. Comput. **5** (1991), no. 3, 107–119.

[10] S. Hauck, *Asynchronous design methodologies: An overview*, Tech. Report TR-93-05-07, 1993.

[11] C. Lageweg, *Single Electron Tunneling Based Arithmetic Computation*, Ph.D. thesis, Delft University of Technology, 2004.

[12] C. Lageweg, S. Cotofana, and S. Vassiliadis, *A Linear Threshold Gate Implementation in Single Electron Technology*, Proceedings of the IEEE Computer Society Workshop on VLSI (Orlando, USA), 2001, pp. 93–98.

[13] ———, *Static Buffered set Based Logic Gates*, Proceedings of the 2nd IEEE International Conference on Nanotechnology (IEEE Nano) (Arlington, USA), 2002, pp. 491–494.

[14] J. Lee and et. al., *Brownian Circuits — Part II: Efficient Designs and Brownian Cellular Automata*, In preparation (2008).

[15] J. Lee, F. Peper, S. Adachi, and S. Mashiko, *Universal Delay-Insensitive Systems With Buffering Lines*, IEEE Transactions on Circuits and Systems **52** (2005), no. 4, 742–754.

[16] J. Lee, F. Peper, S. Adachi, and K. Morita, *Universal Delay-Insensitive Circuits With Bidirectional And Buffering Lines*, IEEE Transactions on Computers **53** (2004), no. 8, 1034–1046.

[17] H. Linke and et. al., *Quantum ratchets and quantum heat pumps*, Applied Physics A: Materials Science and Processing **75** (2002), no. 2, 237–246.

[18] Paul G. Lucassen, *A denotational model and composition theorems for a calculus of delay-insensitive specifications*, Ph.D. thesis, Dept. of C.S., Univ. of Groningen, The Netherlands, May 1994.

[19] C. Meenderinck and S. Cotofana, *Computing Division Using Single-Electron Tunneling Technology*, IEEE TRANSACTIONS ON NANOTECHNOLOGY **6** (2007), no. 4, 451–457.

[20] C.H. Meenderinck, C. R. Lageweg, and S. D. Cotofana, *Design methodology for single electron based building blocks*, Proceedings of 2005 5th IEEE Conference on Nanotechnology, July 2005, p. (CD proceedings).

[21] G. E. Moore, *Cramming more components onto integrated circuits*, Electronics **38** (1965), no. 8, 114–117.

[22] Gordon E. Moore, *Progress in Digital Integrated Electronics*, Digest of the 1975 International Electron Devices Meeting, IEEE (1975), 1113.

[23] P. Patra and D. S. Fussell, *Building-blocks for Designing DI Circuits*, Tech. report, Computer Science Department, University of Texas, 1993.

[24] F. Peper and et. al., *Brownian Circuits — Part I: Concept and Basic Designs*, In preparation (2008).

[25] F. Peper, J. Lee, F. Abo, T. Isokawa, S. Adachi, N. Matsui, and S. Mashiko, *Fault-Tolerance in Nanocomputers: A Cellular Array Approach*, IEEE Transaction on Nanotechnology **3** (2004), no. 1, 187–201.

[26] Tom Verhoeff, *Delay-insensitve codes - an overview*, Distributed Computing **3** (1988), no. 1, 1–8.

[27] ———, *A Theory of Delay-Insensitive Systems*, Ph.D. thesis, Eindhoven University of Technology, 1994.

[28] C. Wasshuber, *About Single-Electron Devices and Circuits*, Ph.D. thesis, TU Wien, 1998.

[29] C. Wasshuber, H. Kosina, and S. Selberherr, *SIMON - A Simulator for Single-Electron Tunnel Devices and Circuits*, IEEE Transactions on Computer-Aided Design **16** (1997), no. 9, 937–944.

[30] T. Yamada, M. Akazawa, T. Asai, and Y. Amemiya, *Boltzmann Machine Neural Network Devices using Single-Electron Tunneling*, Nanotechnology **12** (2001), no. 1, 60–67.

List of Publications

"Connaître, découvrir, communiquer - telle est la destinée d'un savant."
(To get to know, to discover, to publish - this is the destiny of a scientist.)

- François Arago

1. S. Safiruddin, S. D. Cotofana, "Building Blocks for Delay-Insensitive Circuits using Single Electron Tunneling Devices", in *Proceedings of the 7th IEEE International Conference on Nanotechnology*, Hong Kong, August 2007

2. S. Safiruddin, S. D. Cotofana, F. Peper, "Single Electron Tunneling Delay Insensitive and Fluctuation Based Computation Paradigms and Circuits", *IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH'08)*, Anaheim, California, June 2008

3. S. Safiruddin, S. D. Cotofana, F. Peper, J. Lee, "Building Blocks for Fluctuation Based Calculation in Single Electron Tunneling Technology", in *Proceedings of the 8th IEEE International Conference on Nanotechnology*, Arlington, Texas, August 2008

Curriculum Vitae

Saleh Safiruddin was born in Karachi, Pakistan, on the 3rd of May 1982. He followed his secondary education in Nairobi, Kenya at Braeburn School where he took his IGCSE (CIE) exams in 1998 and A'Level (CIE) exams in 2000. In 2000 he enrolled as a student at the Faculty of Electronic and Electrical Engineering, Delft University of Technology, the Netherlands. After receiving his Bachelor of Science degree, he joined the Computer Engineering laboratory, and started his MSc graduation project under the supervision of associate professor Sorin Coţofană. During the course of his MSc graduation project he published three research papers. His research interests include nanocomputing, nanoelectronics, embedded systems, computer arithmetic and computer architecture.