UM ESQUEMA DE CLOCK VIÁVEL (*USE*) E UMA BIBLIOTECA DE STANDARD CELLS (*QCA ONE*) PARA AUTÔMATOS CELULARES COM PONTOS QUÂNTICOS

CAIO ARAÚJO TEIXEIRA CAMPOS

UM ESQUEMA DE CLOCK VIÁVEL (*USE*) E UMA BIBLIOTECA DE STANDARD CELLS (*QCA ONE*) PARA AUTÔMATOS CELULARES COM PONTOS QUÂNTICOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Omar Paranaíba Vilela Neto
Coorientador: Frank Sill Torres

Belo Horizonte, Minas Gerais, Brazil

Março de 2015

CAIO ARAÚJO TEIXEIRA CAMPOS

A FEASIBLE CLOCKING SCHEME (*USE*) AND A STANDARD CELLS LIBRARY (*QCA ONE*) FOR FUTURE QUANTUM-DOT CELLULAR AUTOMATA

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Omar Paranaíba Vilela Neto
Co-Advisor: Frank Sill Torres

Belo Horizonte, Minas Gerais, Brazil

March 2015

© 2015, Caio Araújo Teixeira Campos.
Todos os direitos reservados

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Campos, Caio Araújo Teixeira.

C198f A feasible clocking scheme (USE) and a standard cells library (QCA ONE) for future quantum-dot cellular automata. / Caio Araújo Teixeira Campos. – Belo Horizonte, 2015.
xx, 81 f.: il.; 29 cm.

Dissertação (mestrado) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Omar Paranaíba Vilela Neto .
Coorientador: Frank Sill Torres.

1. Computação - Teses. 2. Autômatos celulares com pontos quânticos .3 Esquema de Clock. 4. Standard Cells I.Orientador. II. Coorientador. III. Título.

CDU 519.6*17(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A feasible clocking scheme (USE) and a standard cells library (QCA ONE) for future quantum-dot cellular automata

CAIO ARAÚJO TEIXEIRA CAMPOS

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. OMAR PARANAÍBA VILELA NETO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. FRANK SILL TORRES
Departamento de Engenharia Eletrônica - UFMG

PROF. JOSÉ AUGUSTO MIRANDA NACIF
Departamento de Informática - UFV

PROF. LUIZ FILIPE MENEZES VIEIRA
Departamento de Ciência da Computação - UFMG

PROF. RENATO PEREZ RIBAS
Departamento de Informática Aplicada - UFRGS

Belo Horizonte, 19 de março de 2015.

Resumo

A tecnologia CMOS baseada em silício está chegado ao seu limite físico enquanto os problemas relacionados ao consumo de energia e confiabilidade crescem de forma alarmante. Com o objetivo de superar esses problemas e permitir que a densidade de componentes colocados dentro de um chip continue crescendo, várias novas tecnologias têm sido propostas nos últimos anos, como por exemplo, *Single Electron Transistors*, *Molecular Electronics*, *Carbon Nanotube Transistors*. Autômatos Celulares com Pontos Quânticos, em inglês *Quantum-dot Cellular Automata* (QCA), é uma dessas tecnologias emergentes com grande capacidade de integração, alta frequência de *clock* e baixo consumo de energia. Em QCA, células biestáveis são conectadas localmente através de forças de efeito de campo e podem ser organizadas de forma que funções lógicas são realizadas, eliminando a necessidade de corrente elétrica. Os principais desafios para o avanço da tecnologia QCA estão relacionados com a automação do processo de desenvolvimento e a integração de fluxos já existentes. Um dos principais problemas desta tecnologia é o *clock* dos circuitos QCA. Ele é necessário para permitir o chaveamento adiabático, além da sincronização do fluxo de informação. A maioria dos circuitos QCA propostos ignoram o circuito gerador de *clock*, o que impossibilita que eles sejam fabricados. Alguns circuitos de *clock* e esquemas foram propostos, mas eles possuem limitações como caminhos de realimentação longos. Com o objetivo de superar esses problemas, um circuito e um esquema de *clock* universal, escalável, eficiente e de fácil fabricação é proposto neste trabalho. Este esquema de *clock*, chamado de *USE*, permite que circuitos QCA sejam criados com a maior liberdade possível e ainda evita problemas relacionados à termodinâmica. Além disso, o *USE* é flexível o suficiente para permitir caminhos de realimentação e um roteamento eficiente. Os resultados mostram uma redução de área de até 5 vezes e diminuição do atraso de até 3 vezes em comparação com um esquema de *clock* existente. A partir do *USE* uma biblioteca de *standard cells* (*QCA ONE*) pode ser proposta. As características do esquema de *clock* proposto, especialmente a possibilidade de criar fios retos e pequenos *loops*, permitem que diversas *standard cells* combinacionais ou sequenciais sejam criadas, além

de favorecer o posicionamento e o roteamento dessas células. Os resultados apresentados neste trabalho são passos importantes para o futuro da fabricação dos circuitos QCA. Além de permitir o desenvolvimento de circuitos QCA robustos e algoritmos de posicionamento e roteamento eficientes, o esquema de *clock* aqui proposto pode ser fabricado utilizando tecnologias de fabricação conhecidas e bem estabelecidas.

Abstract

CMOS technology based on silicon is reaching its physical limits while at the same time reliability and power issues are rising at alarming pace. In order to overcome these problems and to continue the trend of increasing integration densities, several new technologies have been proposed in recent years, such as Single Electron Transistors, Molecular Electronics, Carbon Nanotube Transistors. Quantum-dot cellular automata (QCA) is one of these emergent technologies with very high scale integration, high switching frequency and very low power characteristics. In QCA, bistable cells are locally connected through field effect forces that can be organized in such a way that logic functions are performed, eliminating the application of electric current. The main challenges towards the progress of QCA technology are related to the automation of the design process and integration into existing design flows. One of the main issues of this technology is the QCA clock. It is necessary in order to allow adiabatic switch, in addition to allow the correct and synchronized information flow. Most proposed QCA circuit designs ignore the clocking generation circuit, which prevents the future synthesis of these QCA circuits. Some clocking circuits and schemes have been proposed, but they introduce new difficulties such as long paths for feedbacks. In order to overcome these problems, in this work we propose a universal, scalable, efficient and easily manufacturable clocking circuit and scheme which allows QCA circuits to be created with the highest freedom possible, besides avoiding thermodynamics problems. This clocking scheme, called USE, is flexible enough to allow feedback paths and efficient routing. The results show an area reduction up to factor 5 and delay decrease by up to factor 3 in comparison with an existing advanced clocking scheme. Within the development of the new clocking scheme, we also propose a standard cells library (*QCA ONE*) based on it. The main features of *USE*, specially the design of straight wires as well as small feedback loops, allows the creation of different combinational and sequential cells, in addition to favor placement and routing. We do believe that the results presented in this work are important steps towards the future of QCA synthesis. First, it allows the development of robust circuits and efficient placement and routing

algorithms. Second, the proposed clocking scheme architecture can be fabricated using known and well-established fabrication technologies.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

More than one fourth of United States' GDP (gross domestic product) and 50% of its economy growth comes from information technology (IT) industry, or from companies that heavily depend on IT [Jorgenson, 2005]. This gives an idea of how important the development of integrated circuits is.

In 1965 Gordon Moore observed that the number of transistors in a chip could double annually for at least ten years [Moore, 1965]. Since then, CMOS technology has evolved and grown, but recently specialists have found that the physical size limit of the transistor is getting closer [Haensch et al., 2006], making it harder to maintain this trend.

With this in mind, in 2000, the semiconductors fabrication leaders (Europe, Japan, Korea, Taiwan and United States) organized themselves through the ITRS (International Technology Roadmap for Semiconductors) in order to guarantee the advances in the development of integrated circuits, keeping the cost-benefit ratio. Moreover, the Semiconductor Research Corporation (SRC), and the National Science Foundation (NSF), organized workshops involving industry, academy and government on the search for the new "switch" which could replace the CMOS transistor, a semiconductor device that can be used to amplify and switch electronic signals. At the same time, the Technology Strategy Committee of Semiconductor Industry Association (SIA) also organized several workshops with objective of identifying research initiatives for the advance of technologies related to integrated circuits towards the limits of CMOS [Bernstein et al., 2010].

Some technologies, alternative or complementary to CMOS, have been proposed and studied already. These technologies are based on diverse physical phenomena, such as electronic spin and magnetic momentum, instead of electric current, used by transistors. Besides the different physical phenomena, the way they have been used in

order to realize logic is also being studied. An example of device architecture which performs computation in different way than transistors, is the Quantum-Dot Cellular Automata (QCA) architecture.

QCA consists of bistable cells locally connected through field effect forces which can be organized to perform logic operations [Lent and Tougaw, 1997]. A QCA cell has nano size and circuits are expected to have ultra-low power consumption and promising high clock rate. QCA has been widely applied to the development of logic circuits and hardware components, glimpsing the development of future computers. This demonstrates the importance of research about this technology, as demonstrated in some recent studies [Cho and Swartzlander, 2007; Sardinha et al., 2013; Sen et al., 2014].

The development process of QCA circuits has also been a topic of studies which indicate that it might be similar to CMOS development process, but there are still some problems on this process that haven't been addressed yet. One example are standard cells, that haven't been developed for this technology. Standard cells are cells which perform some specific functions, have similar dimensions and that can be used as building blocks for more complex systems. Other example are the external clocking circuits, responsible for latching the QCA cells, and thus necessary to the correct operation of QCA circuits. Only a few studies discuss these clocking circuits, although they are critical for the development of QCA systems in many steps, including standard cells, placement and routing, which are close to the final steps of the development process.

1.1 Goals

Motivated by the limitations of current technologies and the possibilities of QCA, this study proposes a clocking scheme and a standard cell library for QCA, which are important requirements for the development of more complex circuits.

A QCA standard cell library is presented in this work. Standard cells allow tools to automatically synthesize a circuit optimizing area, power consumption and timing. Also, once a standard cell is developed it can be reused in many projects, avoiding rework. Furthermore, standard cells can comprise the layout of the physical implementation, allowing placement and routing tools to use this information to automatically generate the final circuit implementation.

In order to create a standard cell library for QCA, a clocking scheme is necessary. QCA cells completely depend on a clock signal to work correctly, and thus a clocking

scheme is essential. A clocking scheme must integrate with QCA technology and have its circuit well defined, such as the one proposed in this work.

1.2 Contributions

This work contributes in two different aspects for the development of QCA technology. The standard cell library and the clocking scheme presented here are important in many steps of the QCA circuit development flow.

USE, the proposed clocking scheme, has shown to be more efficient than all known clocking schemes on the literature in terms of area, delay and number of cells used. Moreover, it allows the development of a standard cell library.

The standard cell library presented in this work is, to the best of our knowledge, the most complete QCA standard cell library on the literature. With this standard cell library, a specification format for QCA standard cells based on *USE* is also presented, which can be used by synthesis, placement and routing tools, automating the QCA circuit development process.

1.3 Structure of this work

In chapter 2, a background discussion about what is Quantum-dot Cellular Automata (QCA) is presented, in order to help readers from different backgrounds to better understand this work.

Chapter 3 discusses previous works related to QCA development flow and clocking schemes.

The proposed clocking scheme, *USE*, is described in chapter 4, including all its features, implementation circuit and examples.

A standard cell library developed using *USE* is presented in chapter 5, in addition to a specification format for QCA standard cells based this clocking scheme.

The results of *USE* and the standard cells are shown on chapter 6, which demonstrates a circuit created with cells from the standard cell library developed in this work and also custom circuits, which were compared to other implementations on the literature.

Finally, chapter 7 concludes this thesis and discusses some possible future works.

Chapter 2

QCA Background

A possible alternative to the current CMOS/VLSI circuits is a computing paradigm known as Quantum-dot Cellular Automata - QCA [Lent et al., 1993]. QCA technology consists of a group of cells which, when combined and arranged in a particular way, are able to perform computational functions. These cells can be built using different device technologies such as semiconductor [Toth, 2000], metal-island [Tougaw and Lent, 1994; Toth, 2000], magnetic [Bernstein et al., 2005], molecular [Lent, 2000] and others. QCA technology transfers information by means of the polarization state of various cells in contrast to traditional computers, which use the flow of electrical current to transfer information [Lent et al., 1993; Tougaw and Lent, 1994; Lent and Tougaw, 1997]. This technology is expected to create high clock frequency in addition to low-power consumption in ultra small area, compared to current technologies such as CMOS.

The basic units of QCA circuits are cells made of quantum dots. A dot, in this context, is just a region where an electric charge can be located or not. QCA cells are squares containing four quantum dots close to its corners. Two extra electrons, which are able to tunnel between the quantum dots, are present in each cell. These electrons are not able to tunnel to the outside of the cell, due to a high potential barrier existent on the cell border. The configuration with the lowest energy (ground state) occurs when the electrons have the largest possible distance between each other. The Coulomb interaction between the electrons pushes them to stay at opposite corners, creating only two stable states (Fig. 2.1). These states are called cell polarizations P = -1 and P = +1, which means logic states 0 and 1, respectively.

Although electrons are not allowed to tunnel from one cell to another, the electrons of different cells interact with each other through Coulomb forces. It means that an electron from a cell tends to repel the other electron within the same cell and also the electrons in adjacent cells. That is, two adjacent QCA cells will interact based on

(a) (b)

Figure 2.1: QCA cells on the two possible states. (a) Polarization P = -1 represents a logic 0 and (b) P = +1 represents a logic 1. Squares' edges represent the cell boundaries where exists a high potential barrier and circles represent quantum dots. Black circles are quantum dots containing electrons

the two quantum dots on the edge closer to the other cell, in such a way that a corner of cell containing an electron forces the closest corner of its adjacent cells to not have an electron, and vice versa. Fig. 2.2 shows a sequence of QCA cells side by side, all with the same polarization, which forms in a QCA wire.

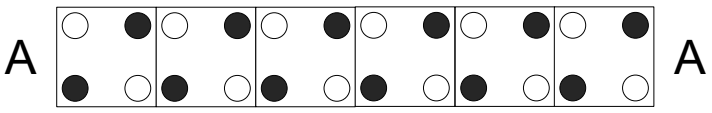


Figure 2.2: QCA wire composed by a sequence of QCA cells which propagates the information. Signal "A" on the left side is propagated without any change to the right side

The idea of combining QCA cells can be extended to create logic structures. A logic inverter, for example, is a structure which transforms a 0 in 1 and the other way around. In order to achieve this inversion, cells need to be placed diagonally, as presented in Fig. 2.3. The "Polarization inversion" region on the figure shows that the cell on the right has a different polarization than the cells on the left. Since these cells are not aligned, as the cells on the wire shown on Fig. 2.2, they tend to be in opposite polarization. That is, both cells on the left side have the same polarization but are not aligned with the right cell, forcing it to a inverted polarization.

Another example of logic structure that can be built with QCA is the majority voter. The output of a majority voter is equal to the value of the majority of the inputs, which is equivalent to the boolean function MV(A,B,C) = AB + AC + BC. A majority voter can be easily transformed into a logic AND or OR, setting one of its inputs to 0 or 1, respectively [Tougaw and Lent, 1994], thus combinations of the majority voter with the inverter are able to create any logic function, because AND, OR and INV form a functionally complete set. Three cells that can be polarized by

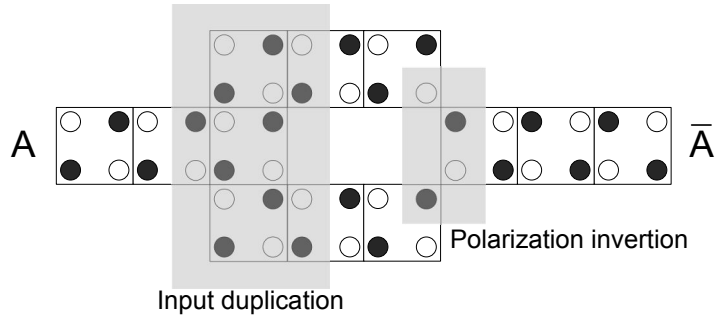


Figure 2.3: QCA Inverter

external signals, that is, the input cells, can be arranged in such a way that a fourth is equally influenced by the first three, therefore creating a majority voter (Fig. 2.4).

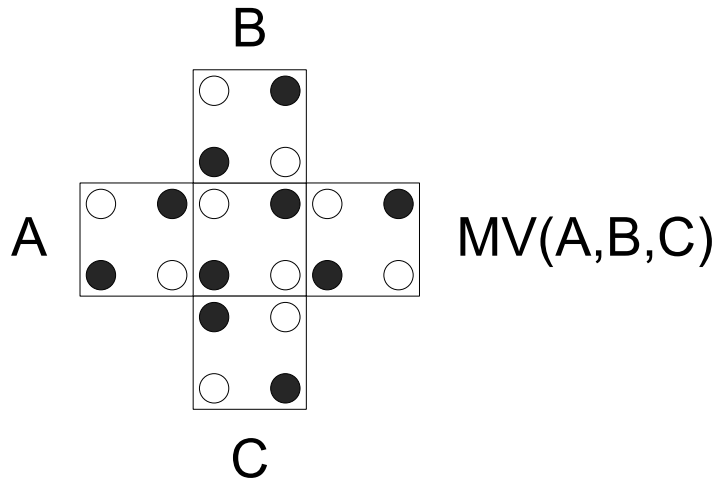


Figure 2.4: QCA Majority Gate

One of the main issues concerning QCA circuits is the switching of QCA arrays, that is, the change of an array of cells from one state to another. For example, cells on the wire of Fig. 2.2 are on polarization P = +1 and thus this array could be switched to P = -1. As we have discussed earlier, QCA takes advantage of the physical ground state. If the input to a QCA array (eg. a wire) is switched suddenly the array will be momentarily in some combination of exited states (eg. cells with polarizations P = -1 and P = +1 in the same wire at the same time). As has been pointed by Welland and Gimzewski [1995], the presence of metastable states on the cells could cause a significant delay in the system reaching its new ground state or even remain in a metastable state, not getting the necessary logic. The solution for this problem is to apply an adiabatic switch. In the first phase of the adiabatic switching the interdot barriers of the cells are lowered, gradually removing the old polarizations induced by the old input. At the end of this phase, the cells exhibit little or no polarization. In the second phase, the

interdots barriers of the the cells are raised while the new input is being applied. The increased interdot barriers cause the cells to repolarize into the well defined bistable states reaching the ground state corresponding to the new inputs. More details about the adiabatic switching can be found in Lent and Tougaw [1997].

The approach described earlier assumed that the interdot potential barrier was being modulated simultaneously for all cells in the array. However, if one subdivides an array into subarrays, one can partition the computational problem and gain the advantages of multi-phase clocking and pipelining. These subarrays (group of cells) are known as clock zones. The scheme of clock zones allows a cluster of QCA cells to make a certain calculation and then has its states frozen, and, finally, has its outputs used as inputs to the next clock zone. Also, this QCA clocking can be used to synchronize the information, avoiding having a signal reaching a logic gate and propagating before other inputs reach the gate. Moreover, the duplex nature (symmetric behavior) of QCA is avoided, ensuring that the signal does not go back to the input during its propagation along the wires and across the logic gates. Another advantage of a clock zone is that if a wire (or logic circuit) grows in length the probability that all cells will switch successfully decreases due to thermodynamic limitations [Lent and Tougaw, 1997]. Breaking the wire in zones can be viewed as pruning it in different small wires. These characteristics are extremely important in QCA circuits, guaranteeing their correct operation.

The QCA clock has four different phases [Lent and Tougaw, 1997]:

1. Switch: The QCA cells start depolarized with the tunneling potential barriers low. During this phase, the barriers between the dots are progressively increased and the cells start to polarize according to the state of their drivers (that is, their neighbor cells). The actual computation, that is the polarization switching according to the cells' drivers polarization and organization, is made exactly at this phase. At the end of the first phase, the barriers are high enough to avoid the tunneling of any electron, so the states of the cells are fixed

2. Hold: The cells have fixed states as the barriers are kept high. So they can be used as inputs to the next stage

3. Release: the barriers are lowered and the cells are allowed to relax to a depolarized state

4. Relax: The barriers are kept low and the cells remain depolarized

Fig. 2.5 shows an example of a wire with four clocking zones. In the first line, the cells in clock zone 1 is in Switch phase and the cells are allowed to polarize according to the polarization of the input cell (Black cell). At the same time, the cells in clock zone 4 are in Hold phase due to an older signal that was transmitted by the wire. In the next clock time, clock 1 is in Hold phase and the polarizations of its cells is used as input for the cells of clock zone 2, in Switch phase. Next, the cells in clock zone 1 are depolarized (clock phase Release) and the cells of clock zone 2 are used to polarize cells of clock zone 3 and so on. This is a nice example of the natural QCA clocked pipelining.

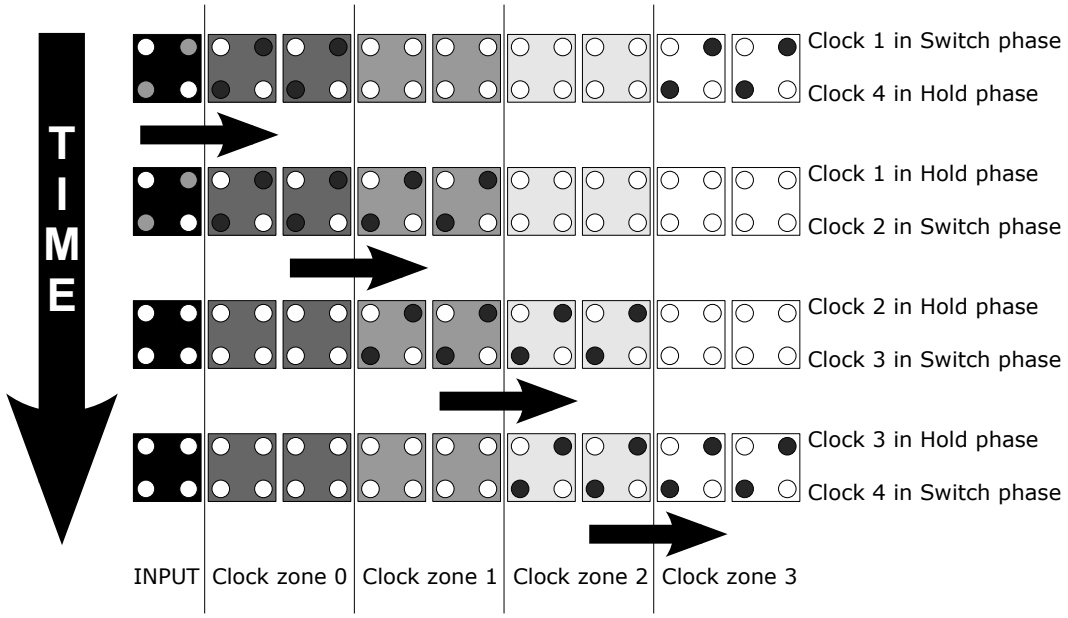


Figure 2.5: An example of clocking zones controlling a wire

This being said, a clocking circuit must be created in order to define regions of clock on the QCA layers, but keeping the flexibility necessary to build any QCA circuit.

Chapter 3

Related Works

In 1965, Gordon Moore observed that the number of transistors in a chip could double annually for at least ten years [Moore, 1965]. Later, David House, an executive from Intel, predicted that the time necessary to double the performance was 18 months, which was close to what have been seen until now. This was possible due to the big development on the fabrication of integrated circuits (IC), especially complementary metal-oxide-semiconductor (CMOS), which enabled the reduction of the transistor size and the increasing number of elements in a chip.

Although advances in CMOS technology were huge, some studies indicate that it is reaching its physical limits, [Haron and Hamdioui, 2008; Rairigh, 2005; Frank, 2002]. At the same time, reliability and power issues are rising at alarming pace. The International Technology Roadmap for Semiconductors (ITRS) [ITRS, 2000] projected the minimal physical gate length in high-performance logic FET (field effect transistor) to be in the range from 4.5nm(2007 ITRS) to 5.9nm(2011 ITRS).

In order to overcome these problems, alternative technologies have been proposed such as Spinwave, Nanomagnet logic (NML), FET based technologies and many others. One of these alternatives is the Quantum-dot cellular automata (QCA) which was proposed by Lent et al. [1993] and consists of bistable cells locally connected through field effect forces.

CMOS fabrication process has been enhanced and refined for years. It is already well established, robust and reliable. On the other hand, QCA development process still being investigated. Henderson et al. [2004] proposes a flow for the development of QCA circuits based on a high level development flow for CMOS circuits. A slightly modified version of the CMOS flow shown by Henderson et al. [2004] is depicted in Fig. 3.1.

In the first step of the flowchart, *Conceptual Design*, high-level models are de-

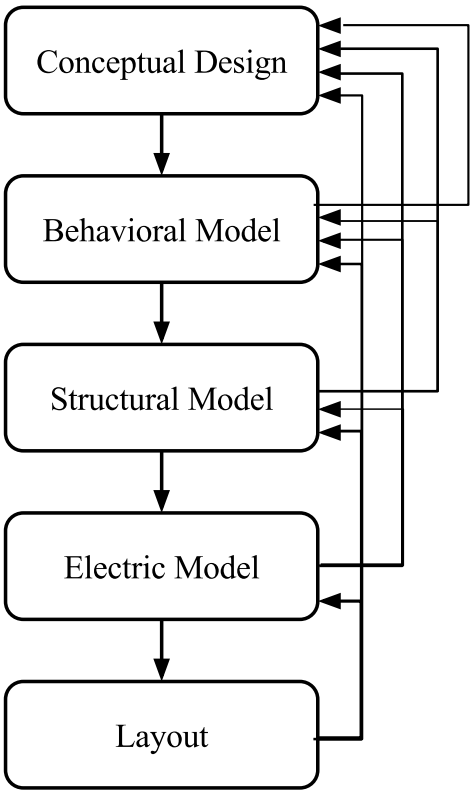


Figure 3.1: CMOS Flow shown by Henderson et al. [2004]

signed to define architectural details, interfaces and others using programming languages such as C, C++ and Java.

The *Behavioral Model* step consists in defining the main logic blocks and its connections using a hardware description language (HDL), such as VHDL and Verilog.

Structural Model is the step in which aspects of the technology start to be considered, such as area, delay, power consumption of the standard cells to be used during the fabrication. The process which transforms the *Behavioral Model* to the *Structural Model* is called synthesis and is usually done automatically by softwares.

The *Electric Model* is the model composed by transistors represented as discrete elements connected in a way that they can perform logic functions. Each standard cell used during the synthesis of the *Structural Model* already has it own electric model, and because of that the *Electric Model* can be seen as just a less abstracted representation of the *Structural Model*. At this stage the organization of these elements on the silicon wafer is ignored.

Finally, in the *Layout* step, all electrical components, such as the transistors, are represented in terms of metal, oxide and semiconductor layers. Prior to the conversion of the *Electric Model* to the *Layout*, the standard cells need to be placed and connected, according to the *Structural Model*, on the wafer. In order to be able to place and

connect the standard cells, the specification of these cells needs to have size and shape information, as well as port location information. This process is called placement and routing and is determinant to the final area of the circuit.

It is important to say that each of the steps is associated with a verification process. Problems identified in these verifications can come from any of the previous models, which makes necessary the review of all previous steps. For example, a problem identified on the *Structural Model* step might be caused by an issue on the *Conceptual Design* and thus not only the *Structural Model* will need to be changed but also the *Conceptual Design* and the *Behavioral Model*. This gives the idea of how important is to automate the development and verification through this flow.

The QCA development flow based on the CMOS flow proposed by Henderson et al. [2004] is shown in Fig. 3.2.

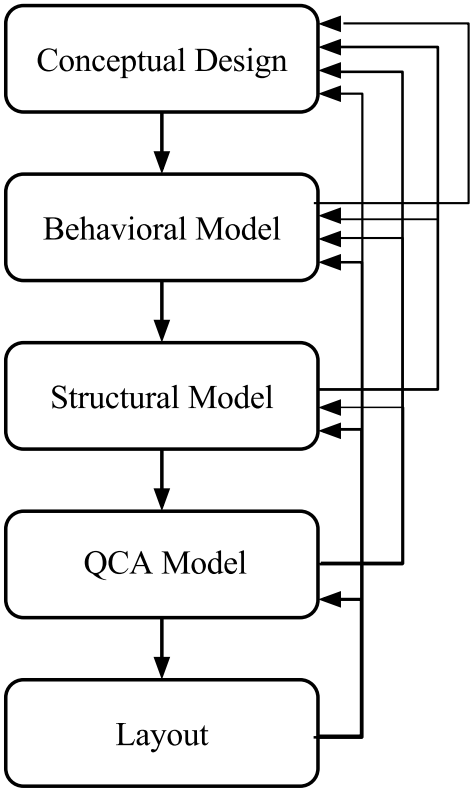


Figure 3.2: QCA Flow proposed by Henderson et al. [2004]

The first two steps of the QCA flow match the first two steps of the CMOS flow without any difference, but the third, *Structural Model*, already considers some aspects of the target technology. An example of this is the synthesis process that needs to be able to prioritize the use of different standard cells. Majority gates are a good example of primitives commonly used in QCA circuits which demand changes on the synthesis process [Akers, 1962; Huo et al., 2006].

QCA Model can be seen as the step analogue to the *Electric Model* step on the CMOS flow, which works with QCA cells to perform logic operations, instead of transistors. Although similar, in QCA the position of the cell should be considered already at this stage, because QCA cells interact through field effect forces, and thus their positions play an important role on the standard cell operation. Another reason to consider the placement of the cells at this step is the fact that wires in QCA are made out of QCA cells, that is, wires are on the same layer than logic gates and the delay introduced by these wires are more significant than the delay introduced by CMOS wires. That is, depending on the routing, the *QCA Model* will need to be changed in order to reflect final implemented circuit.

Lastly, the *Layout* step in a QCA flow is a little bit simpler than this step in a CMOS flow. It is simpler because, as explained on the last stage, the *QCA Model* is a less abstracted representation of the final implementation than the *Electric Model* on the CMOS flow. That is, most of the work in this step is related to the physical characteristics of the QCA cells in terms of layers of materials depending on the QCA technology type.

An important aspect to be considered during the development of QCA circuits is the clocking circuit. Although most QCA circuits presented in the literature don't consider it, the clocking signals and how clock zones are organized are essential to QCA circuits design. Currently, most QCA related works specify each cell clock zone without well defined rules [Cho and Swartzlander, 2007; Teja et al., 2008; Ahmad et al., 2014]. Those that follow some simple rules such as a minimum and maximum of cells on the same clock region, usually don't justify their decision. Moreover, the implementation of the clocking circuit used to generate the clock regions is ignored most of the time. A good clocking scheme is determinant for an efficient standard cell library, placement and routing.

The clocking circuits can be built with wires buried below the QCA surface as proposed for molecular QCA in Hennessy and Lent [2001](Fig. 3.3). These wires and their connections can be built using the same techniques that are used for current integrated circuits (IC) fabrication technology. Because of that, some constraints need to be respected in order to define clock regions in a clocking scheme.

Based on the idea proposed by Hennessy and Lent [2001], Janez et al. [2012] states that clocking zones should have uniform, regular and bounded shapes. Despite this, they did not formally propose a clocking scheme. Their work only defines that CMOS wires could be placed side by side, creating a unidimensional arrangement of clock zones. The consequence of having only one dimension is that feedback paths are not allowed, that is, there is no way to compute anything that depends on the result

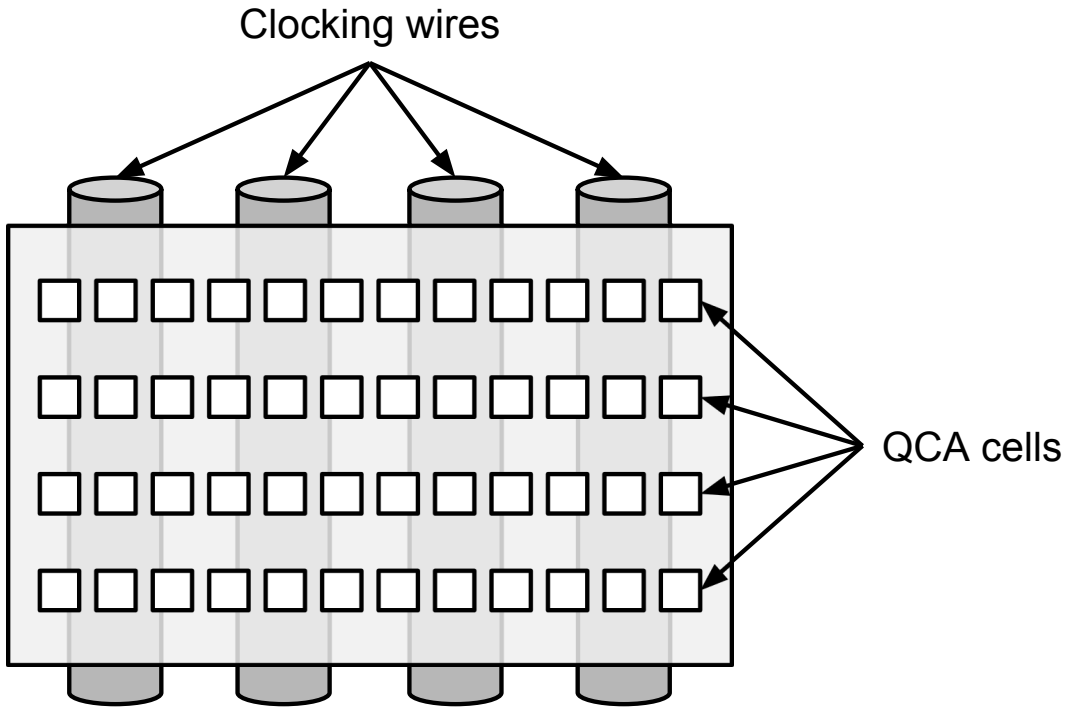


Figure 3.3: CMOS clocking wires below the QCA layer

of previous computations of that same structure. For example, a counter wouldn't be possible to implement since, in order to determine the next value of a counter its current value is needed.

A QCA ALU is used in Niemier [2004] to demonstrate 5 problems that should be addressed by a QCA clocking scheme such as, a huge difference on wire lengths, clock zones with non-uniform widths, big difference on the number of cells between zones, feedback paths, and unused area. Two clocking schemes are proposed in order to solve these problems: the Trapezoidal Clocking and the Universal Clocking Cell. However, their clocking circuitry is not revealed. On top of that, both models fall short since the clock zones don't have the same size and shape.

Two-dimensional QCA clocking schemes are proposed by Vankamamidi et al. [2008]. These clocking schemes take into account the zones size in order to avoid the thermodynamic effects on the QCA circuits. The most promising one, 2DDWave, consists in a grid which has all the zones as squares with the same size. A clocking circuitry is also shown for 2DDWave. Despite these good features, feedback paths remain a problem in this scheme. They are expensive in terms of area, which is a downside specially for sequential elements, and it is unclear whether the grid needs to be rotated, in order to create a floorplan for big circuits containing loops.

Recently, Giri et al. [2014] presented the first known, to the best of our knowledge,

QCA standard cell library. It is a magnetic QCA, also known as NML (NanoMagnet Logic), library containing only simple cells, such as wires, inverters, AND and OR gates. However, this work does not propose a way to specify other standard cells. There is no discussion about design rules or specification formats on it. Also, there is no simulation results shown for the NML circuits implemented, only a comparison of their area, power consumption and clock frequency.

The next chapter presents a clocking scheme which allows the creation of various standard cells and provides a great flexibility for the placement and routing of them. These are very important features for the QCA development flow previously described. A good set of standard cells is critical to the synthesis process, which generates the *Structural Model*. The standard cells are also part of the *QCA Model*, since it is a technology-dependent netlist which connects instances of standard cells, in this case QCA standard cells. Finally, the *Layout* is a result of the placement of the standard cells and the routing of connection wires through the floorplan defined by the clocking scheme.

Chapter 4

USE: A Universal, Scalable and Efficient clocking scheme for QCA

Given the discussion from previous chapter, no clocking scheme is known in the technical literature to satisfy all the following aspects. Clocking zones with uniform, regular and bounded shapes, flexibility to create feedback paths of any length and a well defined circuitry. In this chapter we propose a clocking scheme which complies with all these requirements and is scalable and flexible to allow efficient and easy placement and routing for any kind of circuit.

As stated in chapter 2, the QCA clock can be in one of four phases. Consequently, the complete QCA circuit can be created with four different types of clock zones, in the following numbered from 1 to 4, whereas the current phase of each clock zone is different from the others. Further, it is assumed that the shift between the phases of the clock zones is according to the order Switch, Hold, Release, Relaxed. For example, if clock zone 1 is in phase Hold then clock zone 2 is in Switch, and if clock zone 1 changes to Release then clock zone 2 changes to Hold.

Information passing between QCA cells located in two different clock zones only occurs if the clock zone of the cells that receive the information is in Switch phase, while the cells passing the information have to be in a clock zone in Hold phase (see also chapter 2). Consequently, it is mandatory that cells in clock zone 2 always are driven by cells in clock zone 1, which always are driven by cells in clock zone 4, and so on.

In order to create QCA circuits, QCA cells must be organized in such a way that their interactions through field effect forces perform the desired logic function. As previously mentioned, the circuit clock plays an important role in the QCA circuit operation and thus there must be a clocking scheme which allow the design of QCA

circuits. This clocking scheme needs to be flexible enough to allow the development of logic gates and also the routing of connection wires. Standard cells also need to be able to be developed and placed, with the highest freedom possible, on this clocking scheme. Also, in order to create a QCA clocking scheme which fulfills these requirements, constraints naturally imposed by QCA technology must be considered, such as the data flow direction through the clock zones and number of cells in a single clock zone.

The main idea of the proposed Universal, Scalable and Efficient clocking scheme (*USE*) is that clock zones with adjacent numbers are always located close to each other, while zones with non-adjacent numbers are distant. For example, clock zone 2 should always be close to clock zone 1, from where its inputs should come, and close to clock zone 3, to where its outputs should go. On the other hand, clock zone 4 can be distant to clock zone 2, as the cells of both never interact with each other. Although 1 and 4 are not adjacent numbers, they are considered adjacent zones because zone 4 is the last one and 1 is the first one, thus once data has reached zone 4 it must go to zone 1 and loop through the four clock zones again. Fig. 4.1 depicts the concept of the proposed clocking scheme. Here, each square is a clock zone that contains QCA cells, while the arrows indicate the flow of information between QCA cells located in adjacent clock zones.

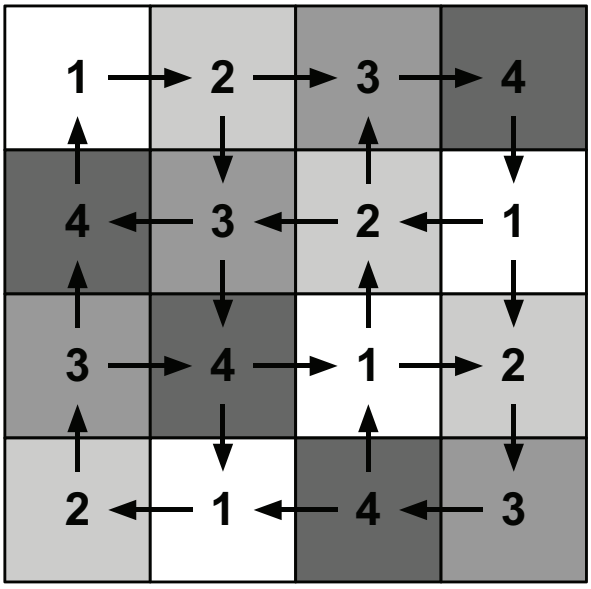


Figure 4.1: Structure of the proposed Universal, Scalable and Efficient clocking scheme (*USE*). Each square is a clock zone that contains QCA cells, while the arrows indicate the information flow. Numbers indicate each of the clock zones and which of them are on the same clock phase.

USE is scalable because it can be easily extended in order to create large floorplans allowing circuits of any size to be developed within it. Fig. 4.2 shows how instances of the proposed structure can be concatenated independently of the circuit being designed. Since expanding the floorplan is just a matter of placing multiple instances of the clocking scheme side-by-side, this clocking scheme is universal, that is, there are no different or special types of floorplans depending on the circuit structure. Other clocking schemes, such as the 2DDWave, proposed by Vankamamidi et al. [2008], are not universal in this sense, since it is unclear when the pattern needs to be rotated in order to allow feedback paths, for example.

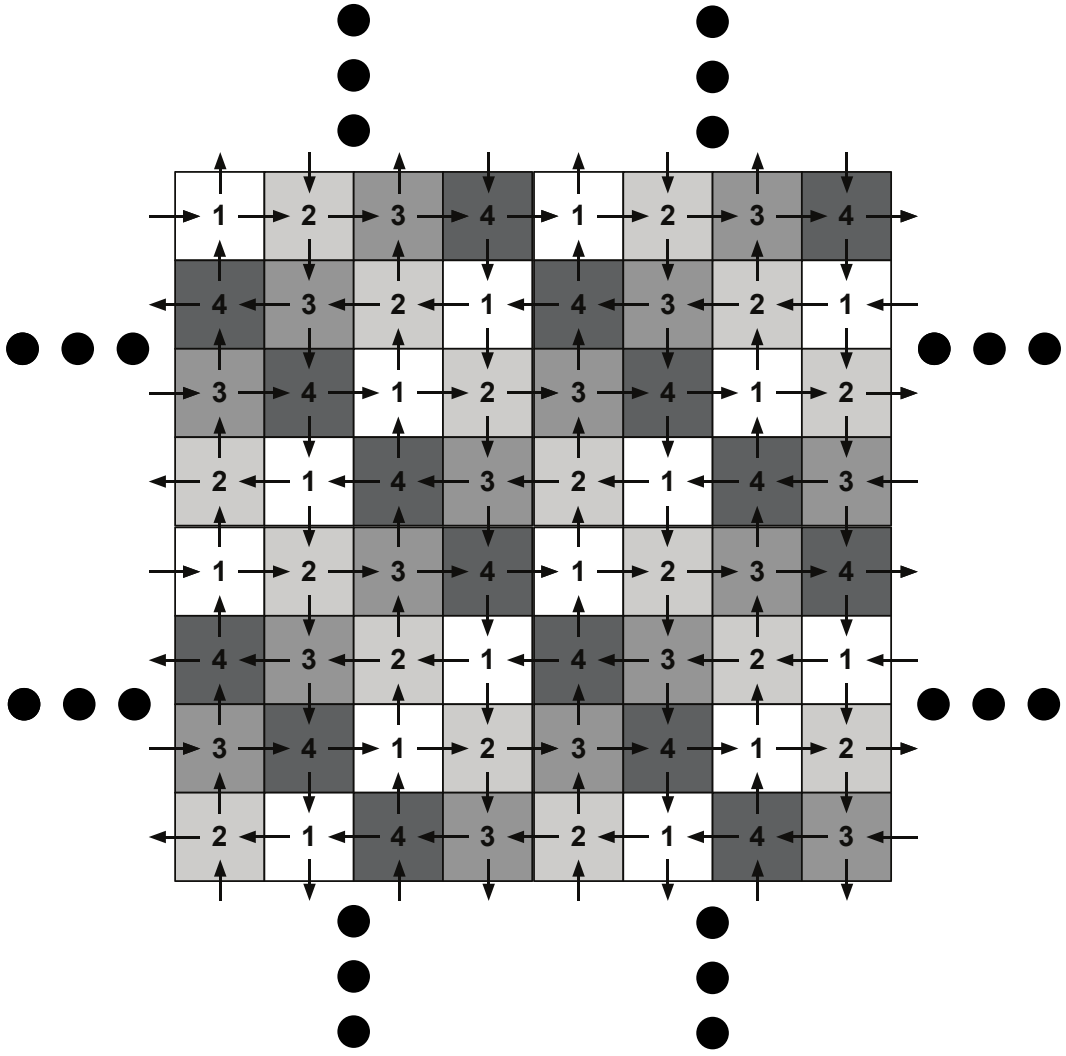


Figure 4.2: Extended version of the clock zone scheme showing its easy expandability

The clocking scheme proposed in this work is also universal because it permits the creation of wires of any type, straight or bent, long or short, to any direction, while it still allows logic gates to be developed respecting its constraints without the need

of any change. All these possibilities are shown on Fig. 4.3 which illustrates all these type of wires, simple logic gates.

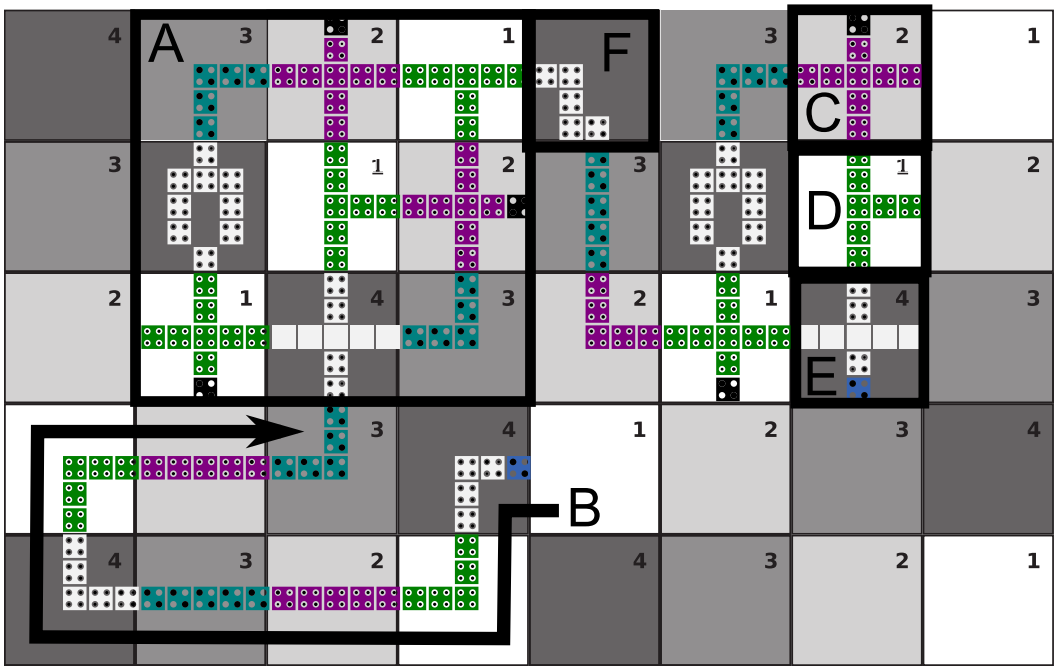


Figure 4.3: *USE* clocking scheme containing a variety of components. A standard cell of an XOR is shown on "A". "B" depicts a long wire with short segments connected by small bent wires. These segments illustrate all possible directions for a wire on the same layer. A simple AND gate is shown on "C" while a fanout is shown on "D". "E" illustrates a wire crossing and "F" a more complex bent wire. The QCA cell pattern used in this figure is the same used by QCADesigner [Walus et al., 2004], that is, each cell color is related to a clock zone, blue cells are inputs and black cells have their polarization fixed. Big black points on the squares indicate the presence of an electron while the circles without black points indicate a quantum dot without an electron. Squares without any points are QCA cells on the top layer and cells with four black dots are one the following phases, Switch, Release or Relax

Finally, *USE* is an efficient scheme due to its great flexibility. Since wires and logic components share the same space without any restriction, gates/standard cells can be placed virtually anywhere and the router would still have lots of places to pass the wires. That is, because of the regular pattern of *USE*, there is no need to define different places for logic components and wires, thus once a gate/standard cell is placed, any free space can still be used by wires.

Moreover, clock zones are always close to two input clock zones and two output clock zones, which enables a huge number of routing paths, and also loops and feedback paths. For example, a clock zone 1 is always adjacent to two clock zones 4 (from where its inputs come from) and to two clock zones 2 (to where its inputs should go).

A small loop, for example, can be created going through only four clock zones, which is excellent for the development of sequential circuits, such as memory elements. This is the minimum number of zones that needs to be used to create a loop in QCA. Since a loop must start and end on the same clock zone and it has to respect the QCA data flow, it has to go through all the four clock phases at least once. The clocking scheme proposed by Janez et al. [2012] does not allow feedback paths, while Vankamamidi et al. [2008] allows, but not as small as those possible with *USE*. This flexibility also results in circuits with small area and delay as shown in the results in chapter 6.

USE permits also the unproblematic implementation of multilayer wire crossing. That is, two wires going in different directions in such a way that one of them needs to pass through another QCA layer, different than the main cell layer, in order to avoid interference on the signals being transmitted by both wires. The ends of the wire on the higher layer are connected with the cells on the main cell layer through stacked QCA cells on via layers. That is, between the lower cell layer and the higher cell layer, QCA cells are stacked in intermediate layers called via. According to Janez et al. [2012], this is the most robust technique to cross QCA wires. This is also relevant for the development of standard cells and for the routing. Fig. 4.3 shows an example of wire crossing on letter "E".

Although *USE* pattern is a 2D floorplan, clock zones in different layers are aligned in such a way that stacked zones always have the same number. That is, given a zone 1, all zones aligned to it in above or below layers will also be a zone 1. This is a consequence of the circuit used to generate the clock zones, that will be explained later in this chapter. The multilayer wire crossing can be done respecting this constraint as explained by Janez et al. [2012].

Besides the aforementioned advantages of *USE*, the proposed circuitry necessary to generate the electric fields for the clock zones has a low complexity, too. As it can be seen on Fig. 4.2, clock zones of the same type are placed diagonally. For example, in this figure there are four parallel diagonals formed by clock zones 1. There are also four diagonals formed by clock zones 3 interspersed among diagonals of clock zones 1. Zones 2 and 4 follow the same pattern, but their diagonals are perpendicular to zones 1 and 3 diagonals.

Similarly to what was proposed by Vankamamidi et al. [2008], a metal wire could run under each of these diagonals in order to generate the electric field which controls the polarization of the cells within these zones. A high-level schematic of this clocking is depicted in Fig. 4.4. Each of the diagonal metal wires connects to the corresponding wire of the 4-Phase Clock Generator according to the clock zone it is controlling.

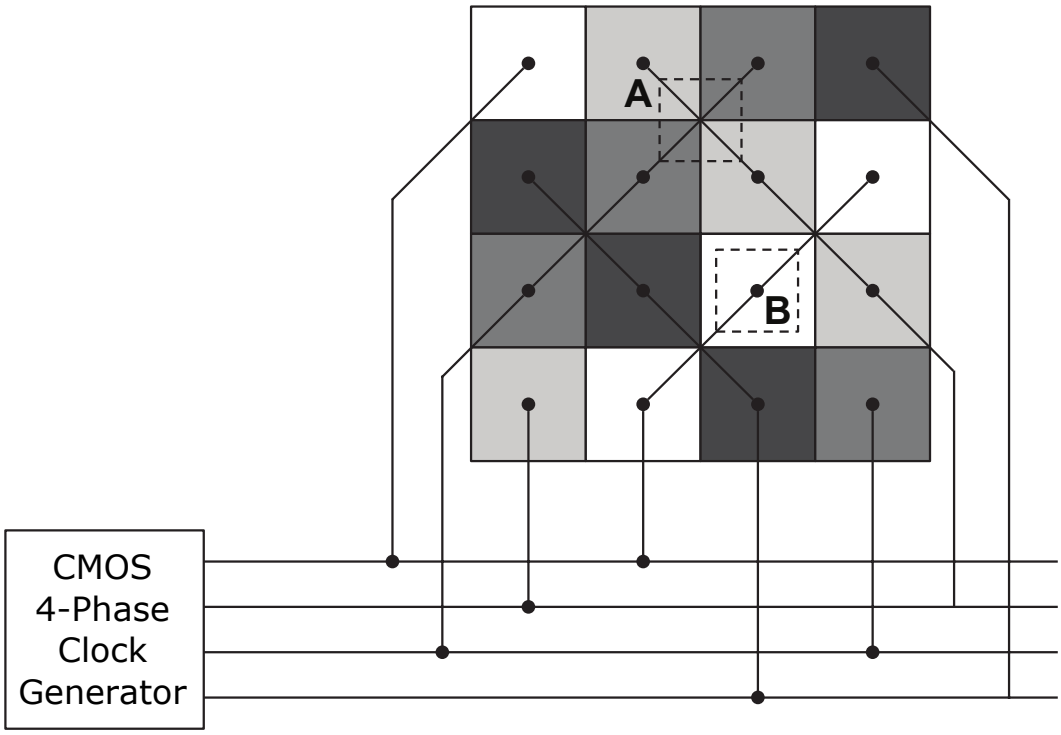


Figure 4.4: Proposed circuitry to generate the electric fields for the clock zones of *USE*

A difference between *USE* and the 2DDWave, one of the clocking schemes proposed by Vankamamidi et al. [2008], is the fact that all diagonal wires in the 2DDWare are parallel to each other. In *USE* there are two groups of parallel wires, that are perpendicular to each other. Since these groups of wires are on the same layer and are perpendicular, they will cross at some point (see "A" dashed square on Fig. 4.4) and thus the contact between these two wires must be avoided. One way to do this is shown on Fig. 4.5.

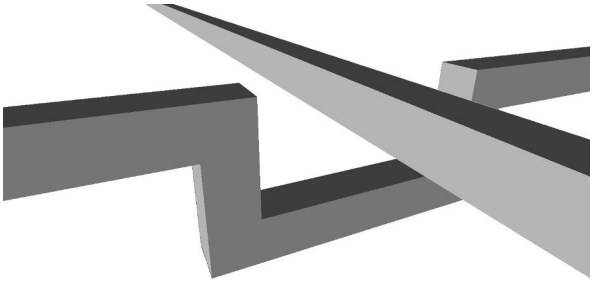


Figure 4.5: Example of metal wire crossing

Finally, a metal pad can be connected to the diagonal metal wires in order generate a more uniform electric field perpendicular to all QCA cell layers, which allows the clock zones to be more well defined. This was also proposed by Vankamamidi et al. [2008] and can be seen on Fig. 4.6. It is important to note that what defines the

dimensions of the clock zones are the dimensions of these wires, and thus they have to be planned according to the number of QCA cells desired for each zone.

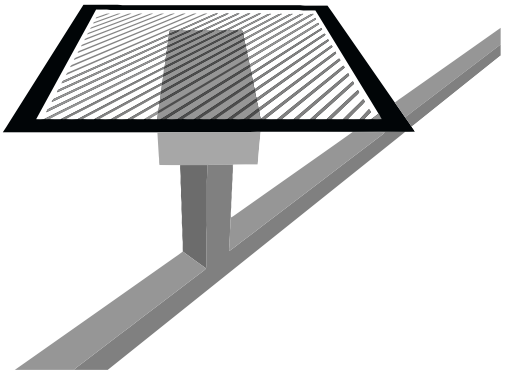


Figure 4.6: Uniform clock zone generated by metal pad, corresponding to the "B" dashed square on Fig. 4.4

With a clocking scheme like this standard cells could be developed and reused across multiple projects. Next chapter proposes a way to specify standard cells based on *USE* and also design rules for a standard cell library developed in this work.

Chapter 5

Standard cells and QCA

Standards cells are cells which perform some specific functions, such as INV, AND, OR, XOR, storage function, etc., have similar dimensions (usually one of the dimensions is the same for all cells allowing them to be lined up in rows) and are not customizable as plain cells, that could potentially have each transistor with specific characteristics. In CMOS these cells are built with interconnected transistors and are used during the processes of synthesis, placement and routing. These processes are usually done with the assistance of Computer Aided Design (CAD) or Electronic Design Automation (EDA) tools.

Once a standard cell is designed, it can be used across lots of projects. Also, high-level models, such as RTL (Register Transfer Level) or the *Behavioral Model*, introduced on chapter 3, can be developed independently of the target technology. Because of this, the RTL/*Behavioral Model* can be used along with a technology library (a complete set of standard cells) as an input for the synthesis tool which generates a technology-dependent netlist, similar to the *Structural Model*, explained on chapter 3.

The technology-dependent netlist contains instances of the standard cells corresponding to the logic gates necessary to generate the behavior specified by the RTL and also the port connection between these gates. This netlist is later used on the placement step, which defines where each standard cell should be placed. A 2-D floorplan is usually an input for the placement step and is used to guide the process.

Finally, the routing process uses the result of the placement and the specification of the standard cells in order to route the wires according to the connections of the netlist. At the end of the routing, a technology-dependent netlist with the information about the placement of each cell and a draw of the connection wires is obtained.

In order to use the standard cell methodology, a standard cell needs to have the necessary information for the processes of synthesis, placement and routing, allowing

the user to abstract the technology details while working on high-level models. The synthesis, for example, needs to know parameters such as the cell delay, area and power consumption to generate a netlist, optimizing these parameters according to the final circuit specification. On the other hand, the placement needs to know the cell size and shape, besides the connections between the cells, in order to determine where each cell should be placed. Finally, the routing needs the result of the placement, as well as the port locations for the cells, to add the connect lines and connections between the cells.

Henderson et al. [2004] proposed that QCA development flow is likely to be very close to CMOS development flow, that is, standard cells are very important for the advance of QCA technology. The sections of this chapter present a general specification for QCA standard cells and *QCA ONE*, a QCA standard cell library.

5.1 QCA standard cells

While CMOS standard cells are built with interconnected transistors, QCA standard cells are built with QCA cells. As previously explained in chapter 2, QCA cells interact through field effect forces, and thus a QCA standard cell consists of a set of cells organized in such a way that the desired logic function is achieved. Besides the cells placement, the clocking zones organization is essential to the correct operation of the standard cell.

Clocking zones are generated by a clocking circuit separated from the QCA circuit and cannot be arbitrarily assigned to QCA cells. That is, once a QCA cell is placed, its clocking zone has been determined. In this sense, a QCA standard cell is completely dependent on the clocking scheme. The standard cells proposed in this work are based on the *USE* clocking scheme, also proposed in this work on chapter 4. Moreover, the specification format proposed in this chapter is meant to be used with *USE*.

An example of this specification format is shown in Fig. 5.1. This example is the specification of the standard cell of a QCA inverter shown in Fig. 5.2, which has all fields that could be used to specify a QCA standard cell based on *USE*.

The first field is *name* which is used to identify the cell. In the technology dependent netlist this name can be used to identify that a given instance of a standard cell is of a given type with the parameter *name*.

The two next fields are *n_ input* and *n_ output* and are used to specify the number of inputs and outputs of the standard cells. It is important to note that this information could potentially be extracted from the list of inputs and outputs, which are also on the specification, but having these fields explicitly makes the reading of the specification,

name: inv #cell name

n_input: 1 #number of inputs
n_output: 1 #number of outputs
input: in #list of inputs
output: out #list of outputs

expression: #expression corresponding to the function performed
$out = -in$

zone_dimension: 5 #dimension of *USE* zone in terms of QCA cells

width: 5 #width in number of QCA cells
height: 5 #height in number of QCA cells
layers: 1 #number of layers used
l_reference_zone: 1 #left reference zone
r_reference_zone: 2 #right reference zone

delay_table: #table of delays between each pair input/output
/ in
out 0

port_location: #coordinates for input and output ports
in 0 2 0
out 4 2 0

free:

Figure 5.1: Specification of inverter standard cell

by a person or a parser, easier.

Fields *input* and *output* , which follow *n_input* and *n_output*, are the lists of port names of the standard cell separated by comma. They could also be extracted from the *expression* field, but again are used to ease the reading.

The *expression* field contains the logic expression corresponding to the function which the standard cell performs and can be used by the synthesis tool during the technology mapping process. Each line starts with the name of an output followed by an "=" and then a expression that can have signal names and operators "&" (AND), "+" (OR) and "−" (INV). If the expression describes a sequential component, then "[]" are used to define the cycle that should be considered to for the computation. For example, $out[t] = (in0 \ \& \ in1) + (out[t-1] \ \& \ in1)$ indicates that "*out*", at a given cycle

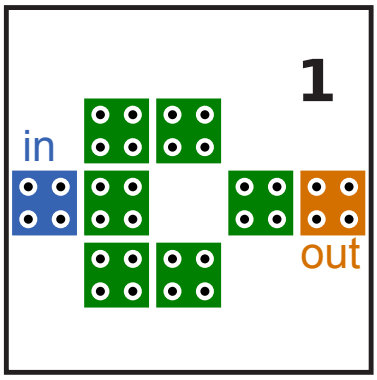


Figure 5.2: Standard cell of an inverter inside *USE* grid

"t" will have the value of "*out*" on cycle "$t-1$" if, at cycle "t", if "$in1$" is true, where a cycle indicates a cycle of the system clock generated by the external clocking circuit.

In *USE* all the clock zones are squares with the same dimension. This dimension is specified by *zone_dimension*, which is an integer which represents the length of the square side in terms of numbers of QCA cells. One important thing to notice about this parameter is that it has to be equal for all standard cells on the same library because it defines how is the *USE* grid in which that standard cell will be placed.

In order to facilitate the placement of the cells and also keep the specification simple, the bounding rectangle containing the standard cell is specified by the fields *width* and *height*. These fields are integers which correspond to the width and height of the bounding rectangle in terms of number of cells. Although this can lead to a rectangle with a lot of empty space, the field *free*, explained later in this section, allows the specification of free space within the standard cell which can be used by routing wires or even other standard cells.

Some QCA standard cells need wire crossing within the cell in order to be able to perform a specific logic function. The best wire crossing method according to Janez et al. [2012] is the multilayer wire crossing. This being said, there must be a way to specify the number of layers needed by a QCA standard cell. The *layers* parameter is used for this, in the same way that *width* and *height* fields.

USE has a well defined clock zone organization pattern which defines the data flow directions along the circuit floorplan. Because of this, even a cell which is totally within a single *USE* zone cannot be placed in any zone. The inverter, for example, has its data flow from the left to the right, because its input port is on the left side of the standard cell while the output port is on the right side. That is, if it is placed on a zone 1, for it to operate correctly, this zone 1 has to be on the left side of a zone 2. Fields *l_reference_zone* and *r_reference_zone* are used to guide the placement tool

specifying where the standard cell can be placed according to the data flow constraint. As the name of these fields indicates, they are references, and thus any standard cell can be shifted by two clock zones on any direction without any harm to the data flow direction. These reference zones could be used as a reference for cell rotations also, but rules necessary to keep the data flow correct would be more complex than the shifting rules and thus were not considered in this work. If cell rotations were possible, the processes of placement and routing could take advantage of this and possibly circuits with reduced area and delay could be generated.

Since QCA standard cells go through various clock zones, delays are introduced. These delays can be expressed in terms of clock phases, that is, the number of clock zones through which the data went through from one point to another. In order to express the delay between each pair of input and output of a standard cell, the field *delay_table* is used. It consists of a table which starts with a slash (/) followed by a list of the inputs. The other lines of the table have the name of each output followed by integer numbers representing the delay to each of the inputs in the same order that the inputs were reported on the first line. In the inverter example the delay between input *in* and the output *out* is 0, because both are connected by cells that are on the same clock zone.

Delays between input/output pairs might not be the same for all pairs of a single standard cell. This is done purposely because delays are naturally introduced by wires and thus the circuit synchronization can be done during the routing. This allows the cells to be smaller, because there is no need to extend cells port to compensate the delays of other ports. For example, in order to fit the Mux 2-1 presented on appendix A.8 in a 3x3 USE grid, its inputs have delays of 1, 2 and 4 to the output that can be synchronized later. Although it seems to deliver extra effort to the routing process this is not true, because wires introduce these delays and the router needs to compensate wire lengths anyway.

Another important thing that needs to be observed is that an input/output pair can be connected through more than one path. For example, an XOR of the form $C = (A+B)+-(A \,\&\, B)$ has both inputs, "A" and "B", connected to "C" through two different paths. The first path, which corresponds to the first part of the expression, $A+B$, goes through an OR gate only and then is connected to the second path, which corresponds to the second part of the expression, $-(A \,\&\, B)$, and goes through an AND and an INV gates (see Fig. 5.3). The delays along multiple paths connecting a pair input/output might or might not be synchronized within a standard cell. The synchronization might lead to bigger standard cells, but, on the other hand, synchronized standard cells can be used to create circuits which take advantage of the device level pipeline inherent

to QCA. That is, naturally pipelined computations can use this feature [Walus et al., 2003] and circuits can be created without the need of extra sequential elements with the meaning of synchronization only.

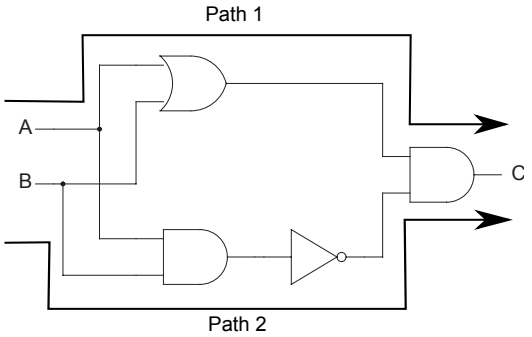


Figure 5.3: XOR function implemented with AND, OR and INV gates. Inputs "A" and "B" are connected to output "C" via two different paths. In "Path 1" inputs are connected to the output going through two gates while in "Path 2" inputs are connected to the output going through three gates

The next field on the QCA inverter cell specification example is *port_location*. This field consists in a list of ports of the standard cell followed by its coordinates. These coordinates are integer numbers and are given in terms of QCA cells considering the three dimensions of the cell. The first number of the coordinates represents the horizontal position of the cell, while the second represents the vertical position and the last one, the layer. All the coordinates start with 0 and the coordinate 0 0 0 represents the top-left place of the bounding box of the standard cell on the main QCA layer, as illustrated in Fig. 5.4.

Finally, *free* is another field that can be used to specify the standard cell. It determines places within the bounding box of the standard cell which don't have any QCA cell. As previously mentioned, this information can be used by the router in order to determine how to pass wires through the standard cells without affecting its operation. This specification only guarantees that there is no cell at the specified places, but it is up to the router to respect the design rules and add the necessary spaces between the QCA cells of the standard cells and the ones added by it. Nevertheless, the standard cell can be specified in a more conservative way and restrict more than just the places containing QCA cells. When this field is omitted, it is assumed that there is no place that could be used by the router within its bounding box. It is important to note that the *free* information must consider the layers used by the standard cell, that is, if a standard cell has cells only in a single layer, then the layers below and above it can be considered free.

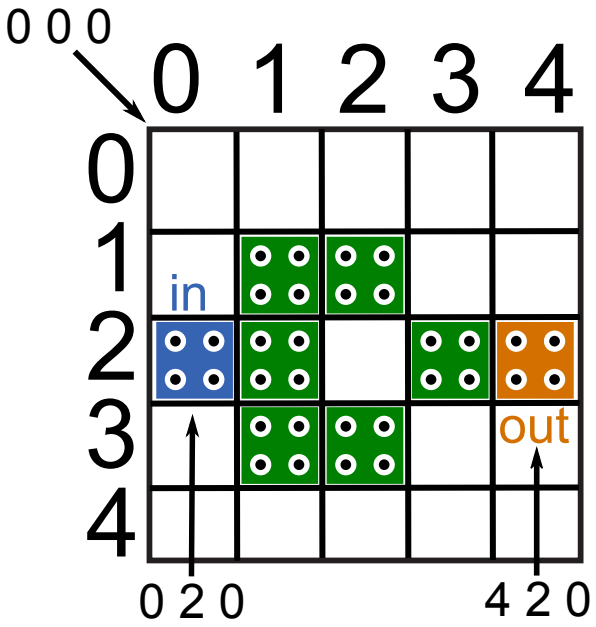


Figure 5.4: Standard cell of an inverter inside *USE* grid showing coordinates for its input and output. All cells in this example are in the main cell layer

The specification of these free spaces can be made with ranges of coordinates following the same guidelines used by *port_location*. For example, something like 0,14 0,4 0,3 indicates that there is some free space on the top-left of the bounding box of the first layer, which has a width of 15 cells and a height of 5 cells through 4 layers. The standard cell for an XOR, shown on Fig. 5.5, has some free spaces that are specified by the *free* field on its specification shown on Fig. 5.6.

This XOR standard cell is more complex than the inverter and thus it is a good example to illustrate how the *free* field could be used. Also, it has two wires crossing, which help with the understanding of cells with more than one layer. The first thing to notice on the specification of the free spaces is that there are many ways to describe them, and there is no problem to specify the same space as free more than once. For example, 0,19 0,9 1,3 means that there is a rectangle along the top of the bounding box going with the height of 9 cells on the three top layers, that is, 0 0 1, and many other spaces, are free. Line 0,16 0,4 0,3 means that there is a rectangle starting on the top-left of the bounding box going from column 0 to column 16, from row 0 to row 4 in all layers, that is, 0 0 1, and many other spaces, are also specified as free by this line. Something that worth mentioning is that this specification format was meant to be compact, while flexible enough to allow the indication of any free space within the standard cell. Moreover, it is up to the standard cell designer to decide whether he wants to specify all the free spaces or not.

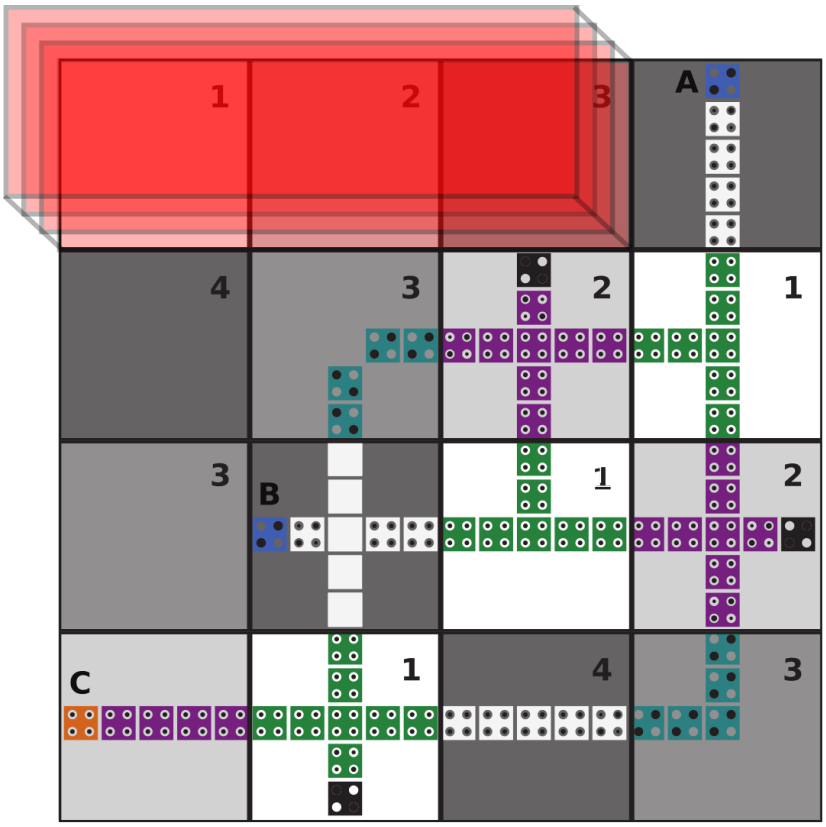


Figure 5.5: Standard cell of an exclusive or inside *USE* grid with some free space highlighted in red. This free space can be described as 0,14 0,4 0,3, that is, a rectangle with vertices on coordinates 0 0, 14 0, 0 4 and 14 4 on the four layers used by this standard cell

Finally, it is important to say that there is no field for the specification of power consumption on the proposed specification format, but the necessary fields could be simply appended to the current specification. Also, the character # indicates the start of a comment that ends at the end of the line. Moreover, the order of the fields presented here is not mandatory, but the order used in this work is meant to be easily readable either by a person or a parser.

5.2 *QCA ONE*, a standard cell library

In this section, the rules for *QCA ONE*, a QCA standard cell library, are presented. This library has 10 standard cells which are shown with their specification and simulation results in Appendix A. These cells are INV, AND, OR, NAND, NOR, XOR, MAJ, MUX 2-1, SR-Latch and D-Latch.

The first characteristic of *QCA ONE* is the zone dimension, which is 5. That

is, each *USE* clock zone has the dimension of 5x5 QCA cells. This makes the clock zone compact, which is good to avoid thermodynamic effects, while still big enough to contain small gates such as, INV, AND and OR, within only one clock zone. Since this dimension is used along all the circuit, it is important to notice that a 5x5 zone can handle up to two parallel wires without signal interference, bent wires and wire crossing with multilayer without any problem.

In *QCA ONE* the standard cells do not have the delays between each input/output pair compensated, which means that there might be different delays between two inputs to the same output. As previously explained, this is not a problem because wires added by the routing process are going to introduce new delays that will need to be compensated anyway during the process. The only synchronization that must exist within the standard cell is when there is a reconvergent path, that is, an input influencing the same output through more than one path. This is what allows circuits built with this library to take advantage of the device level pipeline inherent to QCA.

Input and output ports can be located anywhere in the standard cell given that there is at least one way to a wire to connect to the port, even if this wire needs to cross parts of the standard cell. Although it seems that having a port not on the boundary of the standard cell is worse than extending it until the boundary, it is not. The extension of the ports to the boundary will need to be done anyway, but it can be done by the router. The advantage of this is that this extension will be done accordingly to what is more convenient to each instance of that cell, taking into consideration its position and where are the cell to which it is going to connect to. Every *USE* clock zone always have two input zones and two output zones, thus each port not extended to one of the boundaries of this clock zone can be extended to two different zones. For example, if a input cell is in a clock zone 1, there will be two adjacent clock zones 4 that might have QCA cells to be connected to the input cell, but if this input is already extended to one of the boundaries of the clock zone, there will be only one clock zone 4 that can have QCA cells to connect to the input cell. That is, if this extension is done by the designer during the development of the cell without any information of circuit using that standard cell, then he would need to choose where to extend, which might not be the best option for all circuits. Fig. 5.7 shows this for two different XOR implementations. In Fig. 5.7a, input B was extend to a zone 4 on the left of a zone 1, but it could have been extended to the zone 4 for below the zone 1 instead. If during the routing the connection to B is found to be better going through the zone 4 below the zone 1 nothing could be done. On the other hand, Fig. 5.7b shows an input in1 in a zone 1 that can be extended to either of the zones 4, on the left or below, according

to what is found to be better by the router.

Lastly, the minimum size of standard cell of *QCA ONE* is one *USE* clock zone, although there is no maximum size, because any free space can be specified to be used by the router.

Two standard cells were presented in this chapter, the inverter and the exclusive or, but the possibilities are endless. Some examples are shown on Appendix A, but more cells could be easily developed.

Next chapter shows the result of this work including custom circuits but also a circuit developed using standard cells from *QCA ONE* technology library.

name: xor

n_input: 2
n_output: 1
input: A B
output: C

expression:
$C = -(A \ \& \ B) + (A + B)$

zone_dimension: 5

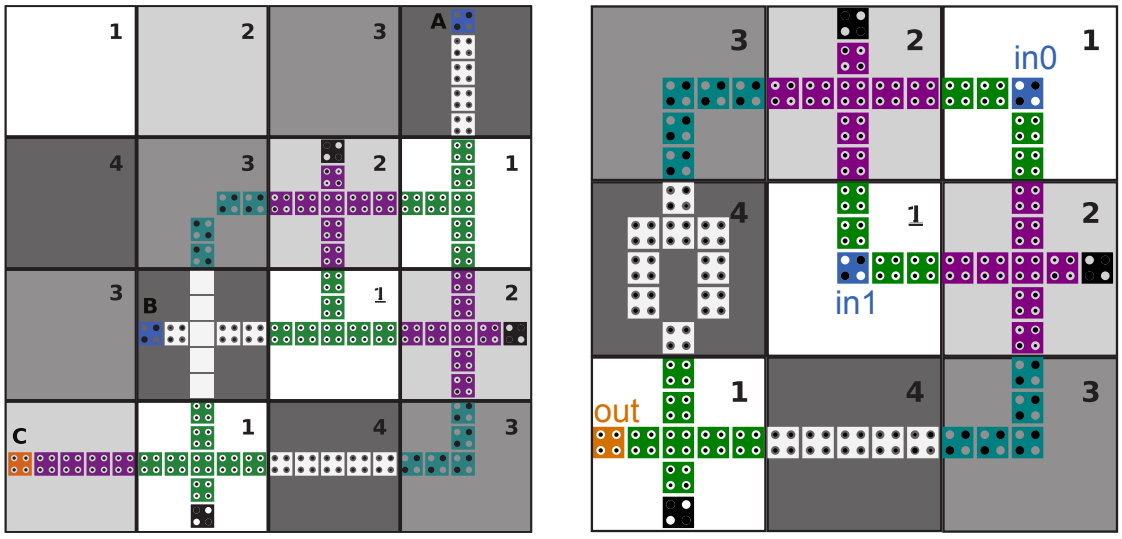
width: 20
height: 20
layers: 4
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ A $\overline{\text{B}}$
out 6 6

port_location:
in0 17 0 0
in1 5 12 0
out 0 17 0

free:
0,14 0,4 0,3
15,19 0,4 1,3
0,4 5,9 0,0
0,19 5,9 1,3
0,4 10,14 0,3
10,19 10,14 1,3
0,19 15,19 1,3

Figure 5.6: Specification of exclusive or standard cell



(a) (b)

Figure 5.7: Two XOR standard cells with different port locations. (a) Ports extended to the boundary of the standard cell and (b) with input in1 inside the standard cell

Chapter 6

Results

In this chapter the results of this work are presented. First section presents custom circuits created in order to show how QCA circuits can be built based on *USE*. These circuits were created manually and compared to other circuits based on a different clocking scheme proposed in the literature. The second section presents a full adder created using the developed standard cells. These standard cells were placed and routed manually in *USE*.

All the simulation results presented here were simulated using QCADesigner through the coherence vector simulation engine [Walus et al., 2004]. QCADesigner is a design and simulation tool for quantum-dot cellular automata (QCA) which allows the designer to quickly layout a QCA design by providing an extensive set of CAD tools. As well, several simulation engines facilitate rapid and accurate simulation. This tool has already been used to design full-adders, barrel shifters, random-access memories, etc. Table 6.1 shows the simulation parameters values used in QCADesigner.

6.1 Custom circuits

In order to exemplify the application of the proposed clocking scheme, three circuits are presented in the following subsections. An SR-Latch, an XOR and an One-bit full adder. The schematic, the QCA circuit and the simulation results are shown for all of them. The SR-Latch and the One-bit full adder are also compared to the equivalent circuits implemented using the 2DDWave clocking scheme proposed by Vankamamidi et al. [2008].

Table 6.1: QCA Design Simulation Settings

Parameter	Description	Value
Cell Width	Width of each QCA square	18 nm
Cell Height	Height of each QCA square	18 nm
Dot Diameter	Diameter of each dot in a QCA cell	5nm
Number of Samples	Number of tested data during the simulation. Accuracy depends on this parameter	12.800 and 50.000
Convergence Tolerance	Simulation of each sample iterates until the new value of polarization deviates from the old value by more than this predefined error limit	0.001
Radius of effect	Radius of effect of a cell is the radius at which it will interact with other cells	80 nm
Relative permittivity	Relation of the permittivity of fabrication material (GaAs/AlGaAs) to the vacuum permittivity	12.9
Clock high	Saturation of energy of clock signal when it is high	9.8E-22J
Clock low	Saturation of energy of clock signal when it is low	3.8E-23J
Clock amplitude factor	To make and effective clock, top 25% and bottom 25% of a single signal is dismissed	2
Layer Separation	Distance between two layers	11.5 nm
Maximum iterations per sample	When the simulation of each state is not convergence based on this parameter, it automatically goes to the next state	100

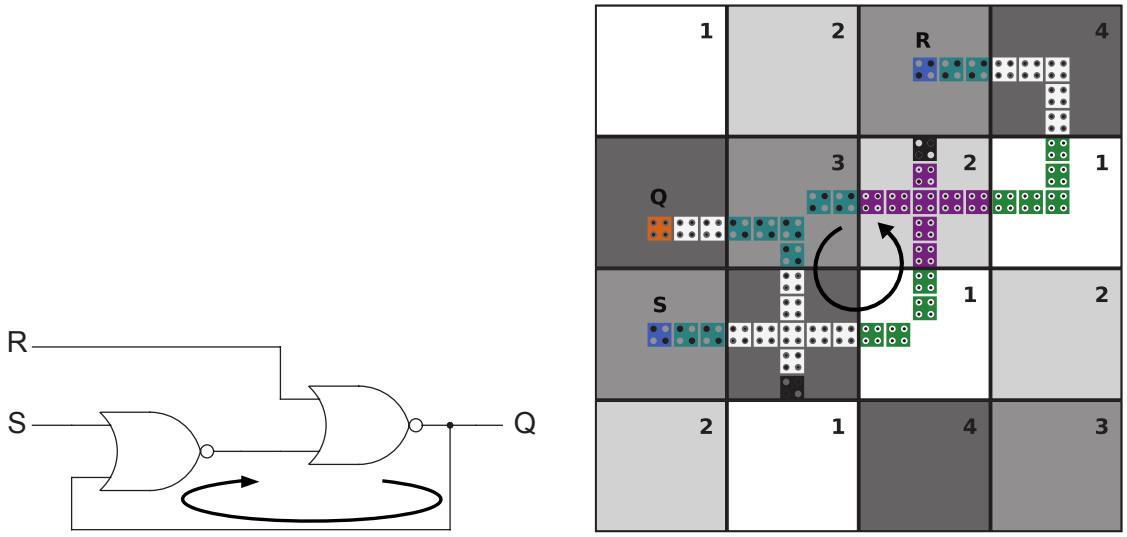
6.1.1 SR-Latch

The SR-Latch was implemented following *USE* rules in order to demonstrate its feedback features (see Fig. 6.1). Fig. 6.1a shows the schematic of the implemented SR-Latch and Fig. 6.1b depicts the QCA implementation with the clock zones highlighted. In both figures, the arrows indicate the direction of the data through the feedback loop which retains previous data when both inputs are 0.

The simulation results for the SR-Latch can be seen in Fig. 6.2. It shows the latch in the following sequence: set, reset, hold, set, hold and reset. This indicates the correct operation of the latch for all allowed input values.

This circuit was compared to the one implemented by Vankamamidi et al. [2008] using the 2DDWave clocking scheme, with respect to circuit area (area of the bounding rectangle formed by the clock zones containing cells), number of QCA cells and delays, as shown in Table 6.2. The implementation using *USE* has shown better results for all metrics, specially due to high number of possible feedback ways.

Lastly, an important thing to note is that the proposed feedback loop, which retains the data inside the implemented latch, has exactly four clock zones. Thus, the



(a) (b)

Figure 6.1: (a) SR-Latch schematic and (b) its implementation in QCA with the proposed clocking scheme with a small feedback loop

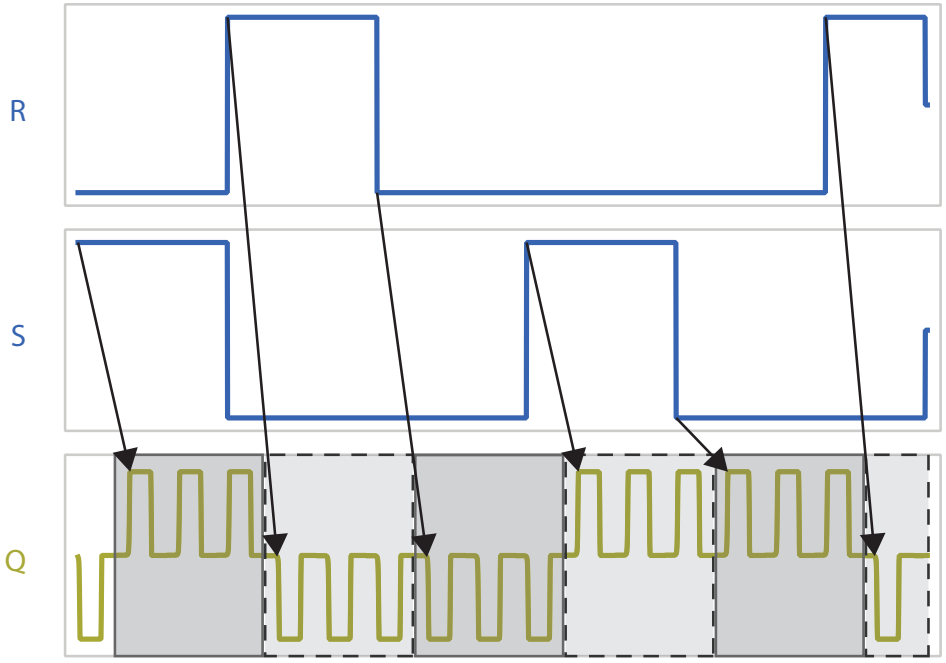


Figure 6.2: Simulation results for the SR-Latch

data is held for only one clock cycle, being as fast as the QCA clocking system and being able to take advantage of the pipeline inherent to QCA. That is, the minimum time that some data could be latched for on this gate is the same as the minimum time data is latched on a QCA wire due to clock phases that QCA cells have to got through because of the QCA system clock.

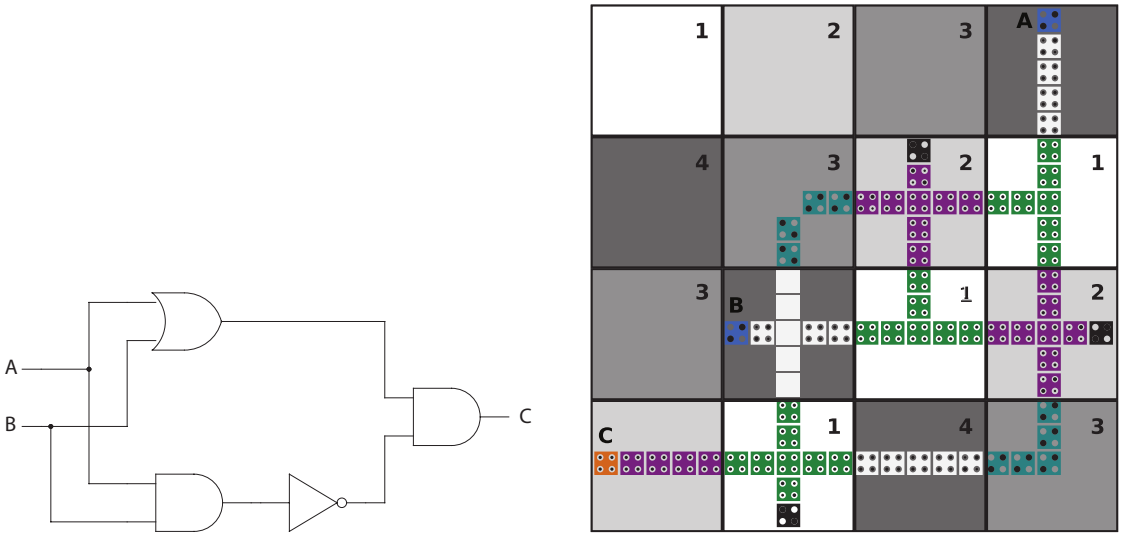
Table 6.2: SR-Latch on *USE* clocking scheme compared to the SR-Latch on the 2DDWave clocking scheme [Vankamamidi et al., 2008]

	USE	2DDWave	Gain
Clock zones from inputs to outputs	6	8	1.33
Clock zones on latch feedback	4	12	3
Number of cells	47	168	3.57
Area (nm^2)	120000	627200	5.23

6.1.2 XOR

The second example is an XOR gate (see Fig. 6.3a), which has been realized based on the proposed clocking scheme (see Fig. 6.3b). In this case, multilayer wire crossing was applied in order to show another feature of *USE*.

As it can be seen in Fig. 6.3b, input "B" connects to a QCA wire which goes from zone 4 to zone 1 on the main cell layer, while the output of the inverter, in zone 3 on the second column of the *USE* grid shown, pass through the same zone 4, but on the top layer, to another zone 1. QCA cells are stacked on two other via layers, between the aforementioned layers, in order to allow signals to propagate from the main to the top layers without any interference on the wire which starts on input "B".



(a) (b)

Figure 6.3: (a) XOR operation schematic and (b) its implementation in QCA with the proposed clocking scheme with a multilayer wire crossing close to input "B"

The simulation results for the XOR can be seen in Fig. 6.4. In this simulation all possible values for the XOR inputs were tested and the correct values were verified at the output.

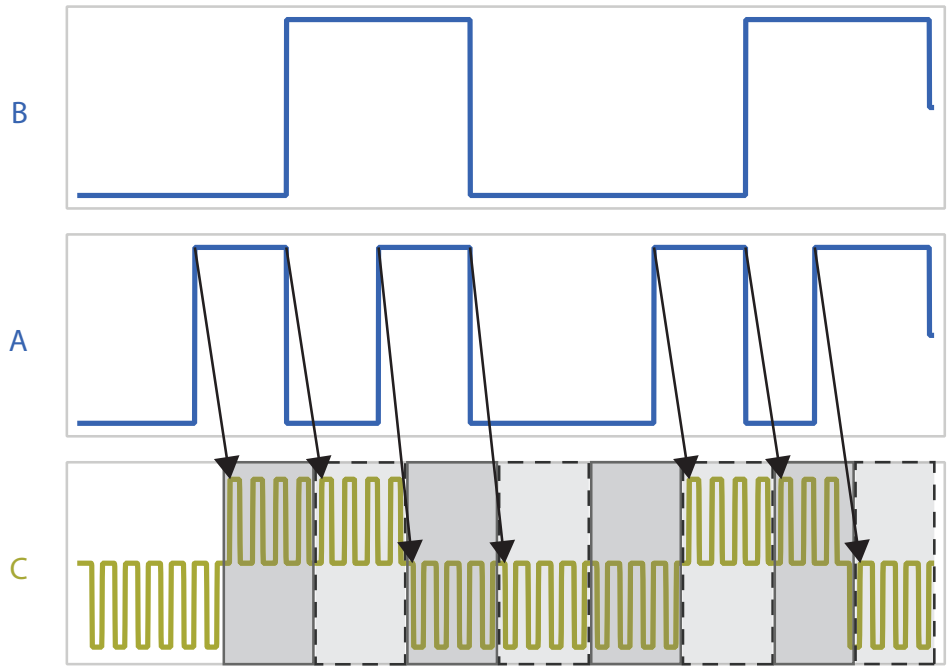


Figure 6.4: Simulation results for the XOR

6.1.3 One-bit full adder

In order to demonstrate that *USE* can be expanded to be applied on bigger circuits, an One-bit full adder was implemented. The schematic and the QCA circuit for this adder can be seen in Fig. 6.5 and Fig. 6.6 respectively.

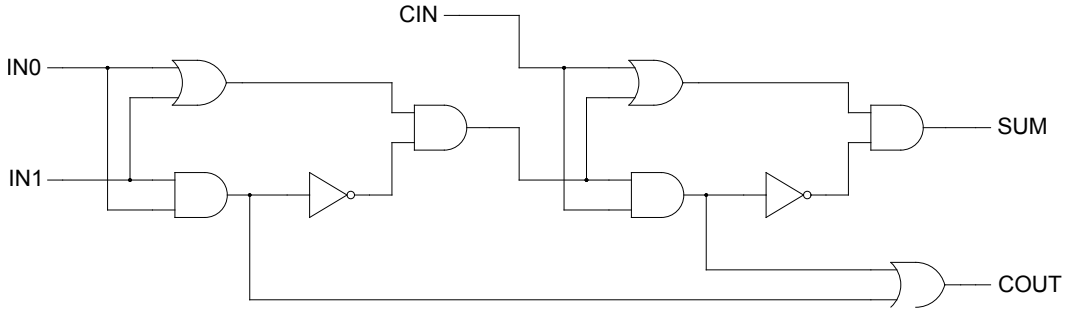


Figure 6.5: One-bit full adder schematic

Since this circuit is bigger than the others, it was necessary to extend one of the wires which connects to the "cout" logic, in order to synchronize the information within the circuit, keeping the number of clock zones between all inputs and outputs the same. This allows the QCA pipeline to be used without the need of sequential elements for synchronization. These wires are a good example of how flexible *USE* is allowing long wires to be created and delays to be compensated along the clock zones. Although the

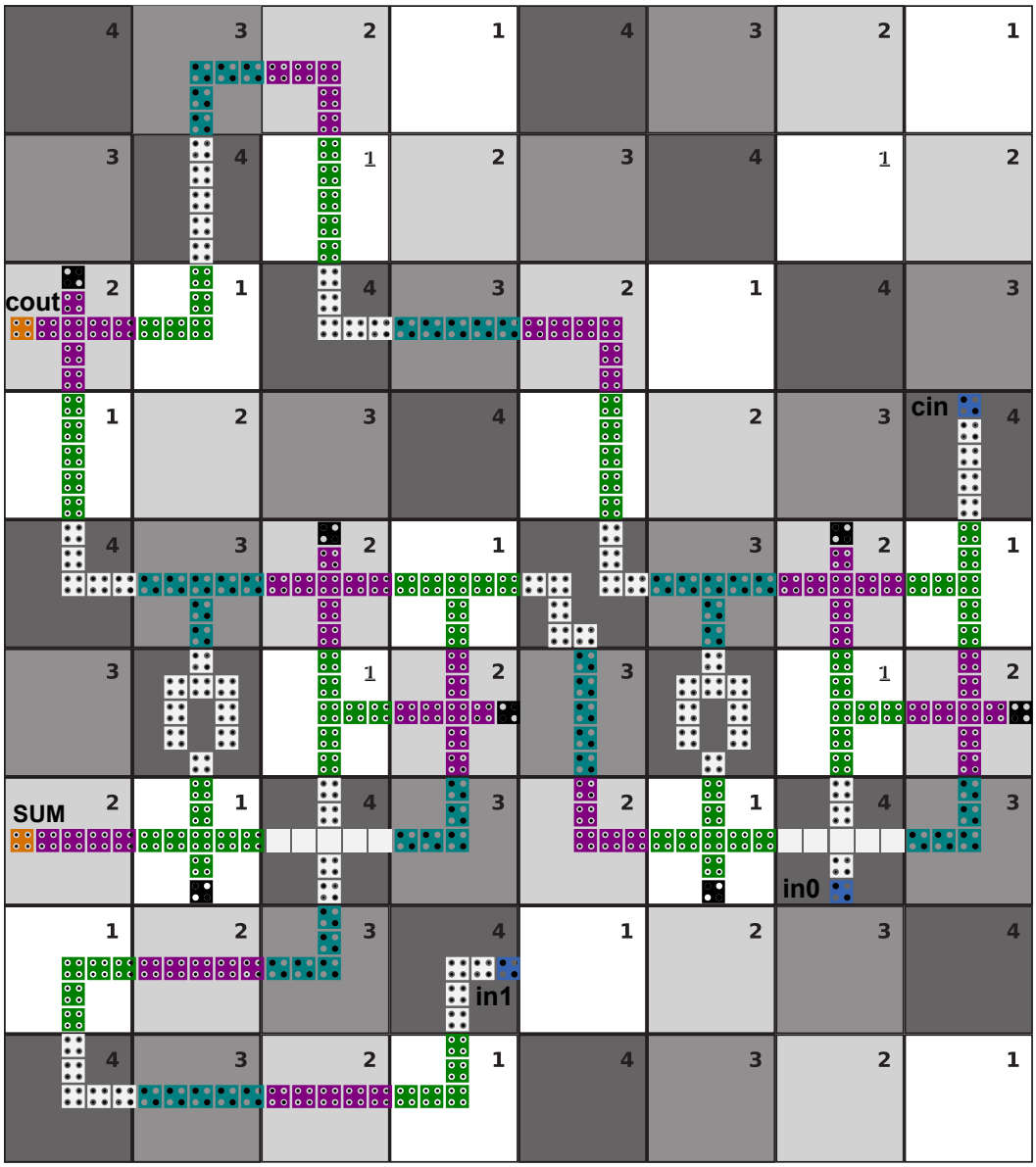


Figure 6.6: One-bit full adder implementation in QCA using more than one instance of the proposed clocking scheme

wires increase the area of the bounding box of the circuit, a routing process could use the empty spaces around the adder, and even the free space in other layers over the wires, to connect other circuits.

The simulation results for the One-bit full adder can be seen in Fig. 6.7. In this simulation all possible values for the three inputs were tested and the correct values were verified at the outputs.

This circuit was compared to the one implemented by Vankamamidi et al. [2008] using the 2DDWave clocking scheme, with respect to circuit area (area of the bounding

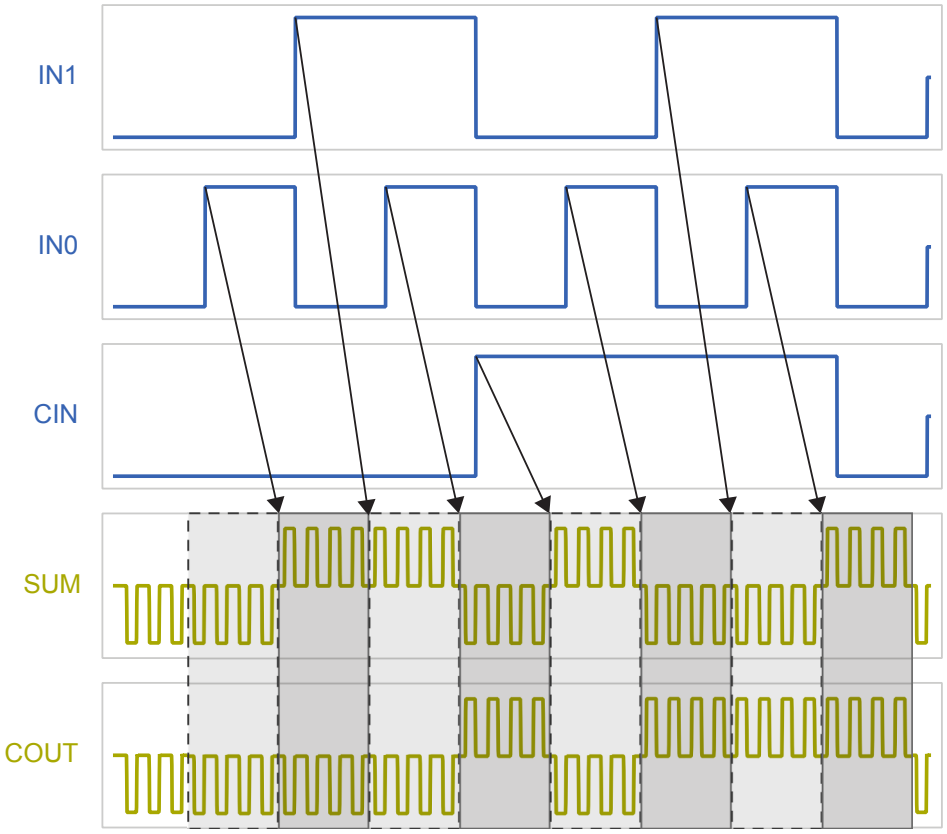


Figure 6.7: Simulation results for the One-bit full adder

rectangle formed by the clock zones containing cells), number of QCA cells and delay, as shown in Table 6.3. The implementation using *USE* has shown better results for all metrics again, probably due to the big number of routing paths, which allowed logic components to be placed very close without restrictions to the connection wires.

Table 6.3: One-bit full adder on *USE* clocking scheme compared to the One-bit full adder on the 2DDWave clocking scheme [Vankamamidi et al., 2008]

	USE	2DDWave	Gain
Clock zones from inputs to outputs	15	18	1.2
Number of cells	278	353	1.27
Area (nm^2)	560000	1764000	3.15

In addition to the gains shown in Table 6.3, it is important to highlight that some aspects that were used by the circuit developed using the 2DDWave clocking scheme in Vankamamidi et al. [2008] might have resulted in a smaller area and number of cells in their implementation. The carry out circuit was not implemented on the circuit, what would add more logic to it. Also, the inputs are assumed to have its negation readily available at convenient places, avoiding the need of inverters and extra QCA

cells to connect the inputs to the inverters. Finally, the number of clock zones between the inputs and the outputs is not the same for all pairs of input and outputs, which not only reduces the area and number of cells, but also prevents the usage of the QCA pipeline.

6.2 Circuit using standard cells

Chapter 5 described a way to specify standard cells and design rules created for a QCA standard cell library developed in this work, the *QCA ONE*. The QCA circuit, specification and simulation for all standard cells of this library can be seen in Appendix A.

In order to show that the standard cells of *QCA ONE* can be used to create bigger circuits, an One-bit full adder was implemented only with the standard cells of this library. From the synthesis to the placement and routing, everything was done manually. Nothing was changed on the standard cells and the placement and routing respected the specifications, as it would be done by an automatic tool. Since rotations are not discussed in this work, they were not done on any of the cells used, although shifts of two clock zones, explained in chapter 5, were used.

An One-bit full adder can be synthesized in many different ways, depending on the available standard cells. One possible synthesis was already shown in Fig. 6.5. In this cases only AND, OR and INV standard cells would be used. Another possible synthesis could use AND, OR and XOR gates, as shown in Fig. 6.8. This second synthesis option was chosen for the One-bit full adder implemented with standard cells.

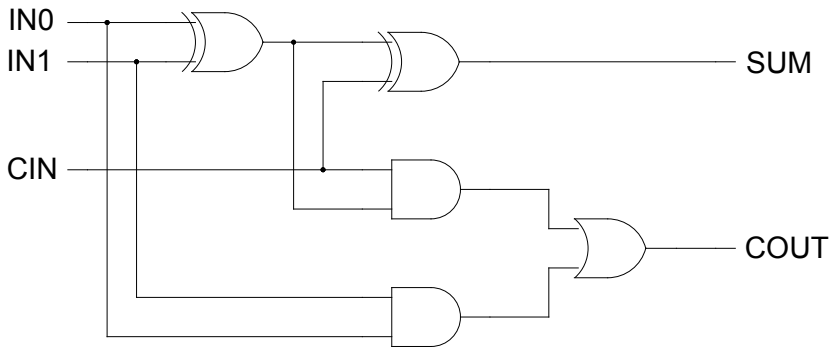


Figure 6.8: One-bit full adder schematic composed by AND, OR and XOR gates

Taking a closer look at the implementation of the XOR standard cell (Fig. 6.9), it is possible to see that it is composed by AND, OR and INV circuits. The advantage of using an XOR, instead of the AND, OR and INV standard cells, comes from the fact that the small circuits composing the XOR are not standard cells, and thus don't need to follow the rules for the standard cell for that circuit. For example, the INV standard

cell has 1 and 2 as reference zones and a bounding box of 5x5 cells (see Appendix A.1 for more details). This means that this cell is completely within a zone 1 which is located on the left side of a zone 2. Since cells cannot be rotated and shifts can be only of two clock zones, it would never be possible to put the INV standard cell in the same way it was done with the INV composing the XOR. On the other hand, the disadvantage of using bigger standard cells is that some intermediate results generated along the cell cannot be reused. In the XOR example, $in0$ & $in1$ and $in0 + in1$ are computed within the cell but cannot be reused because this information is not known by the synthesis tool.

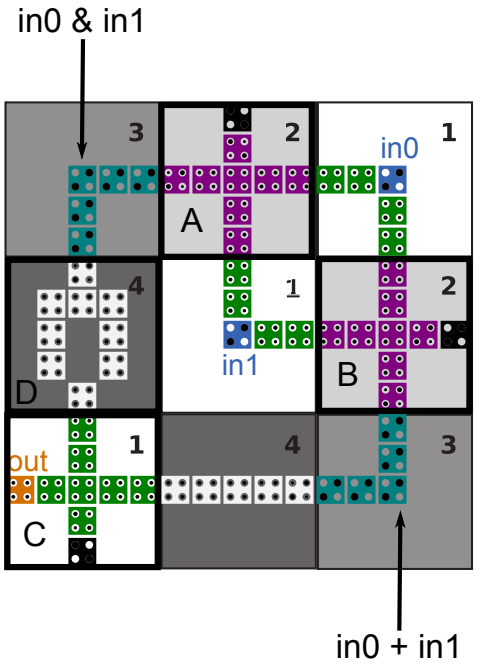


Figure 6.9: XOR standard cell with intermediate results and gates composing it highlighted. “A” indicates an AND, “B” indicates an OR, another AND is indicated by “C” and an INV is indicated by “D”

The QCA implementation of the One-bit full adder with standard cells is shown in Fig. 6.10. The two XOR, two OR and one AND standard cells were placed manually as close as possible, but in positions that the connection wires could be later placed including the delay compensation.

Fig. 6.10 highlights some capabilities of *USE*. The first thing that should be noticed in this figure is the fact that the floorplan was not changed and standard cells of different types could be placed on it along with wires. The wires placed manually were routed in a way that the delay introduced by QCA standard cells could be compensated on them, keeping the QCA pipeline. It is also possible to see wires running in parallel

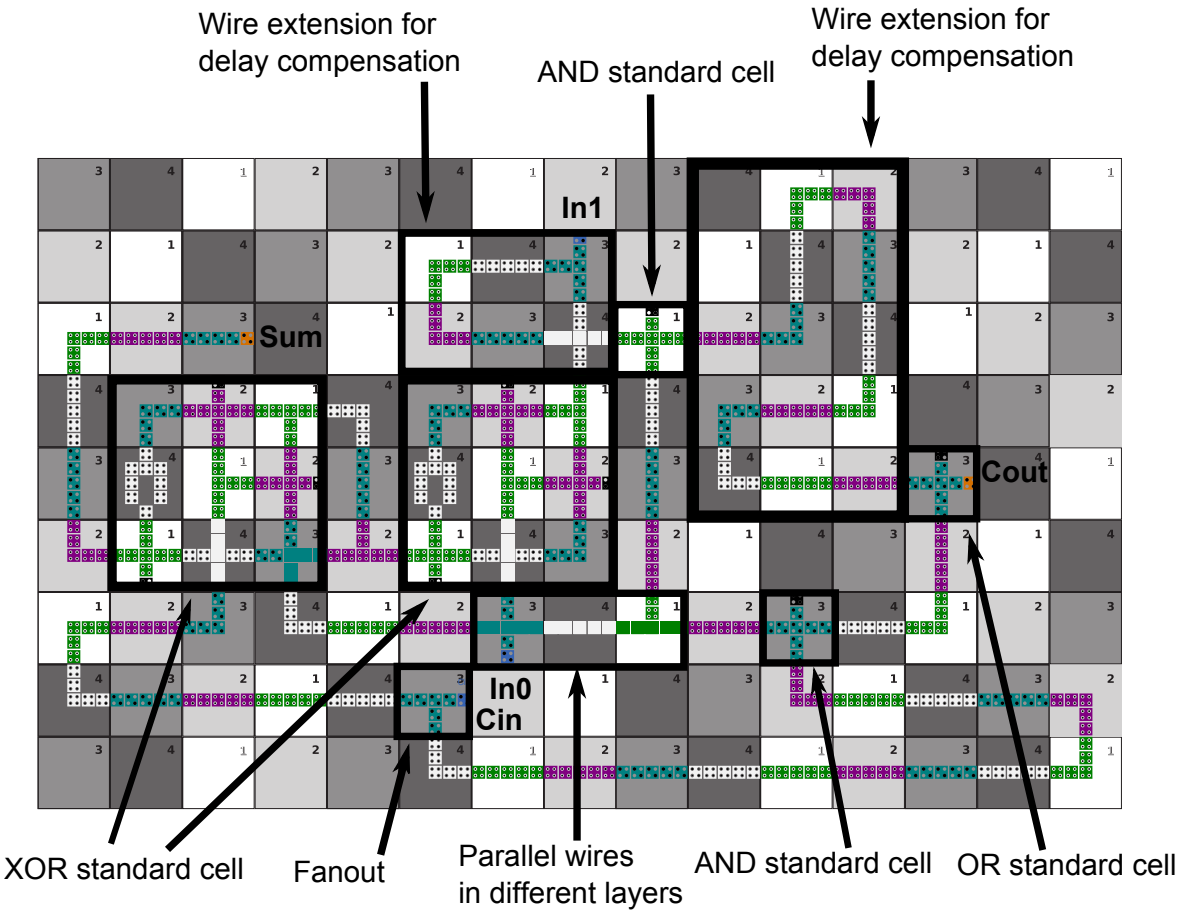


Figure 6.10: One-bit full adder created with *QCA ONE* standard cells with some *USE* capabilities highlighted

in two different layers as well as multilayer wire crossings. An example of fanout is also shown, demonstrating that a wire can be splited taking a signal to different directions.

The simulation result for this adder can be seen in Fig. 6.11 for all possible inputs.

Finally, Table 6.4 shows a comparison between the custom adder shown in section 6.1 and the adder created with standard cells. As it can be seen, the standard cell version is bigger and introduces more delay. The adder based on standard cells was not meant to be the most efficient and thus it was expected that it could be less efficient than the custom version [Keutzer and Chinnery, 2000]. On the other hand, the standard cell version could have been created by an automatic tool.

In order to show how these QCA standard cells could be automatically arranged in a *USE* grid, Fig. 6.12 is shown. This figure also shows how these standard cells could be connected by wires placed by a routing tool. The idea behind this high-level model of placement and routing for *USE* is similar to the organization of a FPGA (field-programmable gate array). FPGAs are integrated circuits that have some generic logic

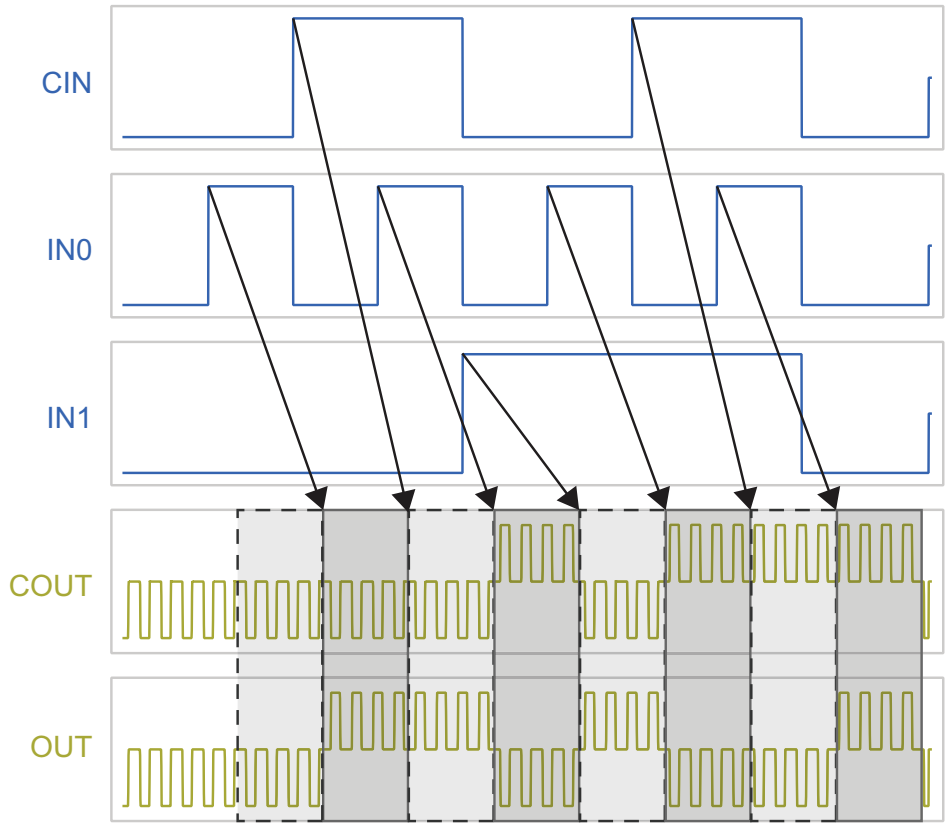


Figure 6.11: Simulation result for the one-bit full adder based on *QCA ONE* standard cells

Table 6.4: Custom one-bit full adder compared to the *QCA ONE* standard cell based one-bit full adder

	Custom	Standard cell	Gain
Clock zones from inputs to outputs	15	21	1.4
Number of cells	278	595	2.14
Area (nm^2)	560000	1350000	2.41

blocks and that can be configured after manufacturing. In the figure, each of the spots could have a standard cell as if it was a logic block of the FPGA, that is, many different standard cells could be placed there as if a logic block had been assigned to perform a specific logic operation. Meanwhile, the *USE* zones between these spots could be used by the router to pass the connection wires just like the interconnect wires a FPGA. Once more, it reinforces how *USE* can be used to create bigger circuits for any purpose in a more automated way.

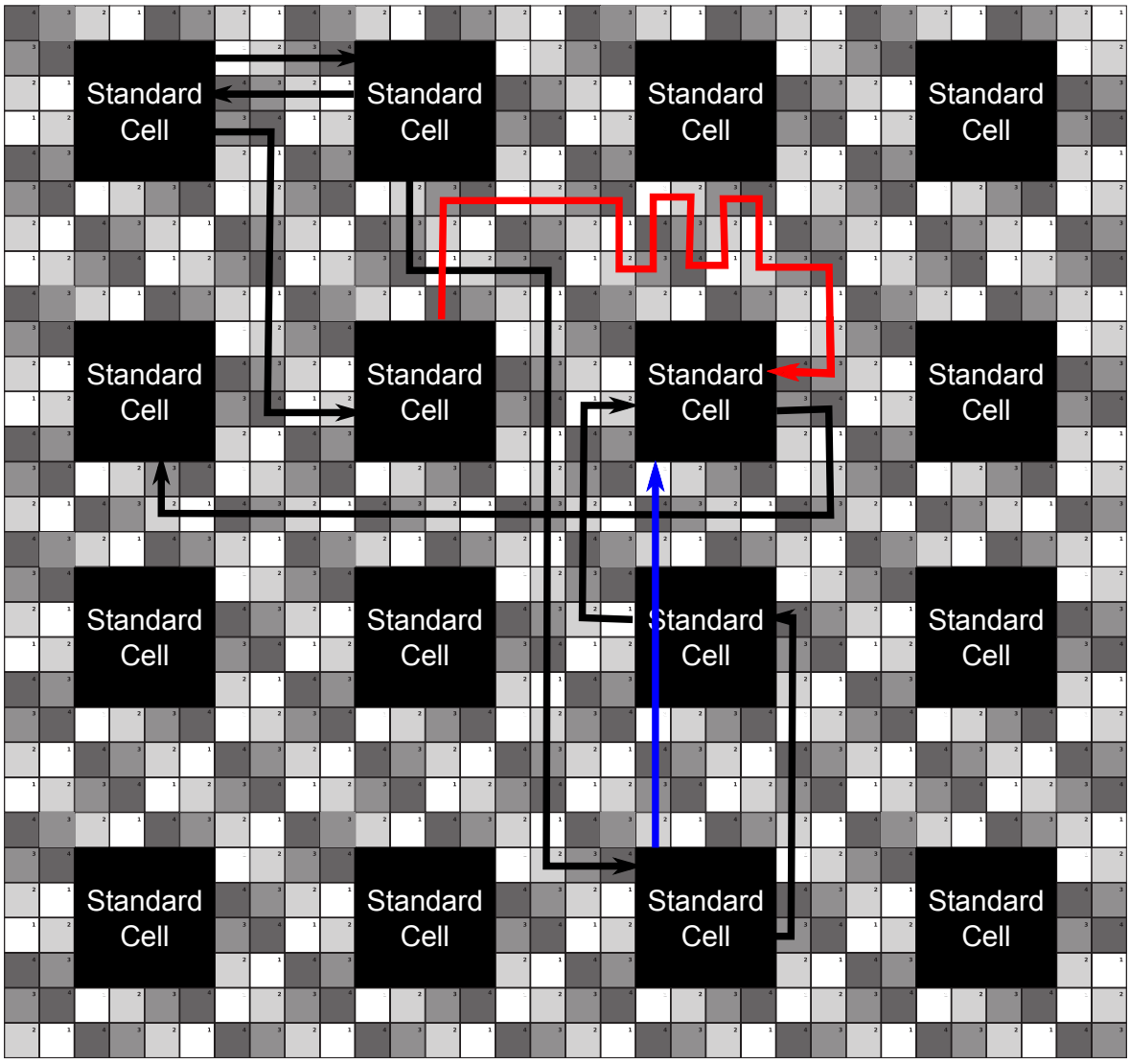


Figure 6.12: Big *USE* grid with standard cells placed as if they were logic blocks of a FPGA. Arrows indicate how wires could be used to connect these standard cells. The red arrow is an example of how a wire could be extended to introduce a delay necessary to synchronize the data propagation along the circuit. Wires crossing standard cells are also possible depending on standard cell specification and an example is show by the blue arrow

Chapter 7

Conclusion and Future works

QCA is one candidate technology to complement or replace current CMOS technology. While CMOS is reaching its physical limits QCA still being researched and is in early development stage with a promising future. In order to make QCA a viable technology its development process needs to be more well defined and each step of this process needs to be established. The standard cell methodology is something that can be integrated on the QCA development process and has been shown to be successful for CMOS process.

In this work, QCA standard cells were developed in order to contribute to one of the steps of the QCA development flow. These standard cells can be reused by many projects since they correspond to basic logic functions. Moreover, a specification format was proposed, which can be used to specify other standard cells. Also, this specification format can be used in other steps of the QCA process, such as the technology mapping, placement and routing.

An important aspect of QCA technology is that it needs an external clocking circuit to control the correct operation of the QCA circuit. This clocking circuit defines a floorplan which needs to be considered while developing the standard cells. A clocking scheme called *USE* was proposed in this work and it was used as the base for the development of the standard cells and their specification format. Besides the application of *USE* in this work, it also contributes to the development flow in the sense that it creates a universal, scalable and efficient floorplan that can help the placement and routing steps. *USE* has a great flexibility which allows a huge number of paths to be created by the router.

In order to show that *USE* can be used to create efficient QCA circuits, custom circuits were developed and compared to circuits shown n the literature which were created within clocking schemes. All of them have shown be more efficient in terms

of area, delay and number of cells when developed within *USE*. A set of standard cells, along with their specifications, was developed in order to show that standard cells based on *USE* can also be developed. These standard cells were used to create a bigger circuit that showed how a circuit could be developed with standard cells if an automatic tools had done its placement and routing.

7.1 Future works

This work can be used as the foundation for many others. Some examples are:

1. Placement and routing algorithms could be more easily developed on top of *USE*. Since *USE* is a uniform pattern, with clock zones of the same size, possible paths are well defined on it, and the delays introduced by wires can be easily compensated within the floorplan
2. The standard cells could be expanded in many different ways
    a) Power information could be introduced on the specification
    b) Standard cell rotation rules could be defined
    c) Standard cell shifting could be enhanced
    d) Intermediary results of standard cells function could be added to the specification in order to allow them to be reused
    e) More standard cells could be developed and current standard cells could be optimized
3. A synthesis tool could be created using the standard cell specification format as one of the inputs
4. Other QCA implementations, such as magnetic, could be investigated in order to have clocking schemes developed to them
5. A plugin for QCADesigner, including *USE* and a standard cell library, could be developed to make the development of QCA circuits easier
6. Reversible circuits could be based on *USE* since only the external clock generator would need to be changed in order to completely reverse the data flow direction of the circuits

Appendix A

QCA ONE reference manual

A.1 INV

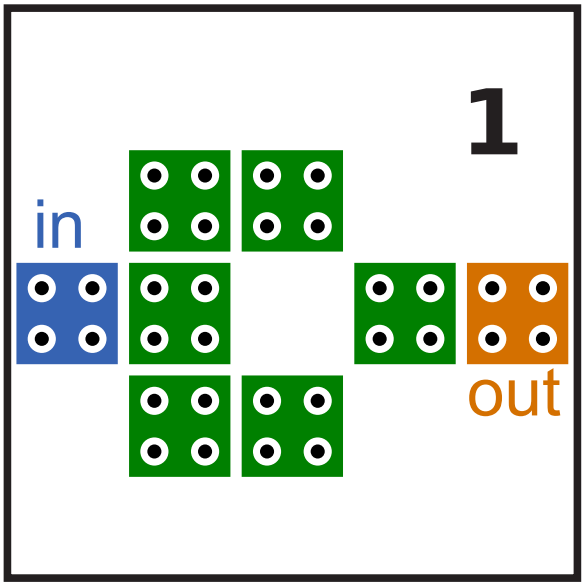


Figure A.1: Standard cell of an inverter inside *USE* grid

name: inv

n_input: 1
n_output: 1
input: in
output: out

expression:

$out = -in$

zone_dimension: 5

width: 5
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in
out 0

port_location:
in 0 2 0
out 4 2 0

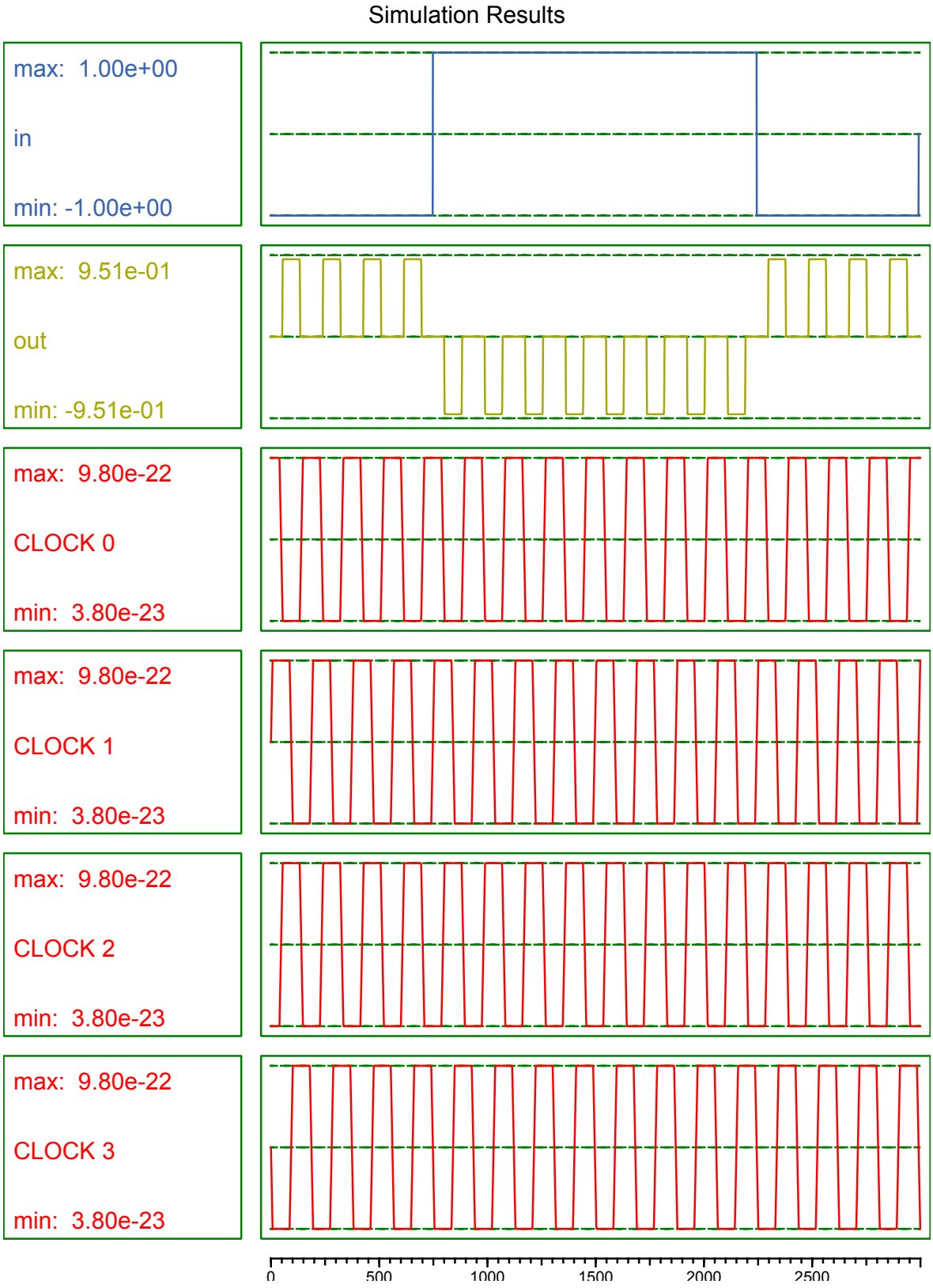


Figure A.2: Simulation results for the inverter standard cell

A.2 AND

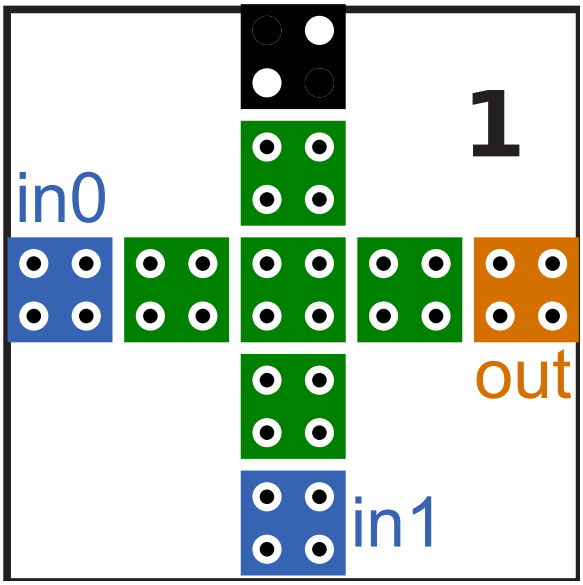


Figure A.3: Standard cell of an AND inside *USE* grid

name: and

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out = in0 \ \& \ in1$

zone_dimension: 5

width: 5
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1

out 0 0

port_location:
in0 0 2 0
in1 2 4 0
out 4 2 0

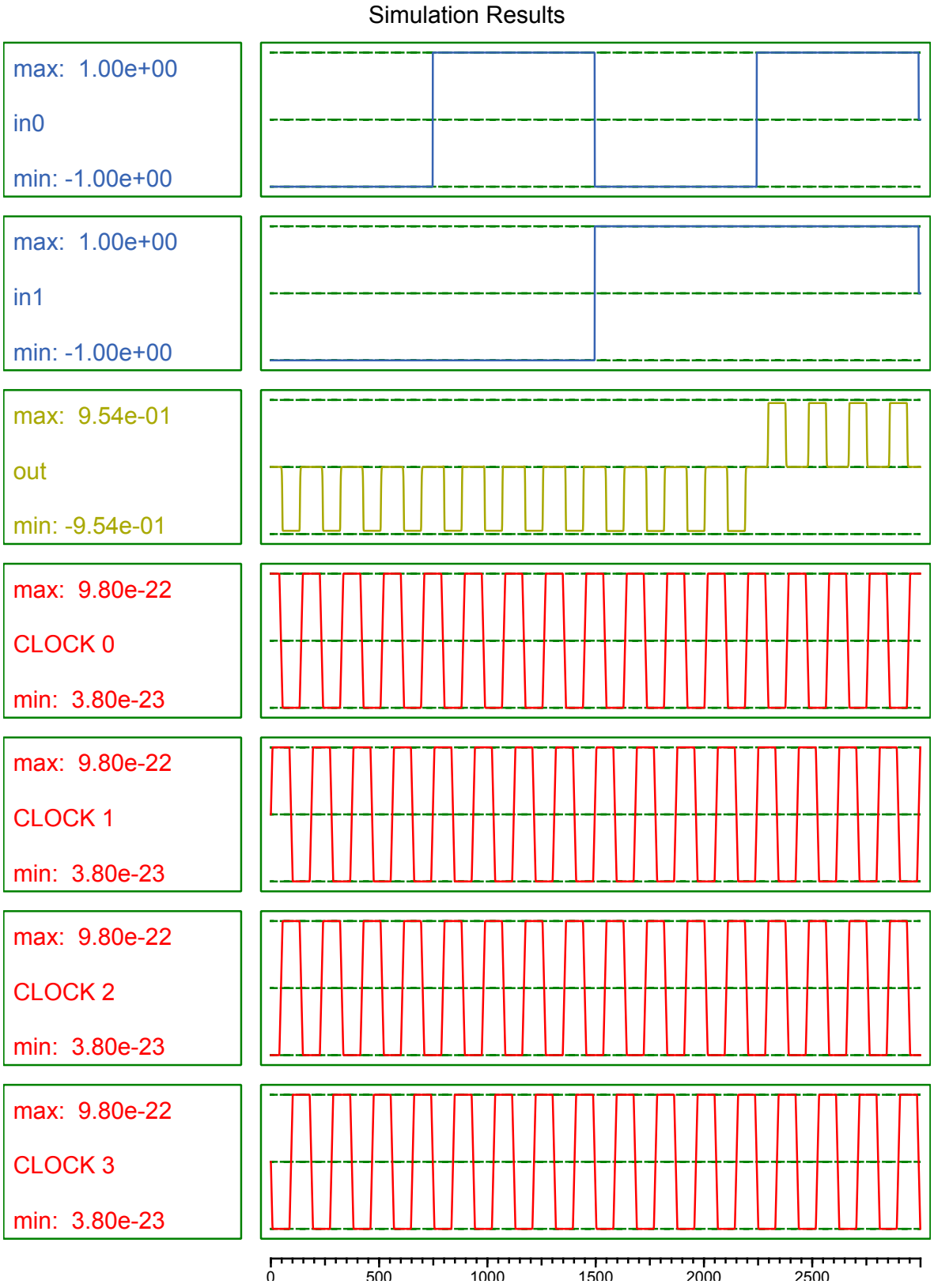


Figure A.4: Simulation results for the AND standard cell

A.3 OR

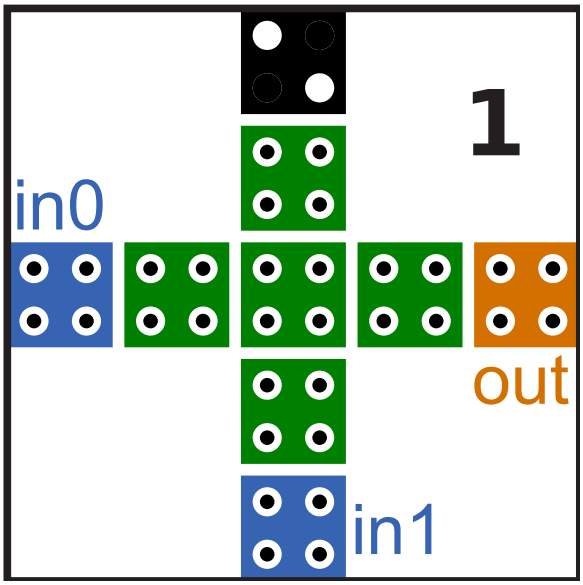


Figure A.5: Standard cell of an OR inside *USE* grid

name: or

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out = in0 + in1$

zone_dimension: 5

width: 5
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1

out 0 0

port_location:
in0 0 2 0
in1 2 4 0
out 4 2 0

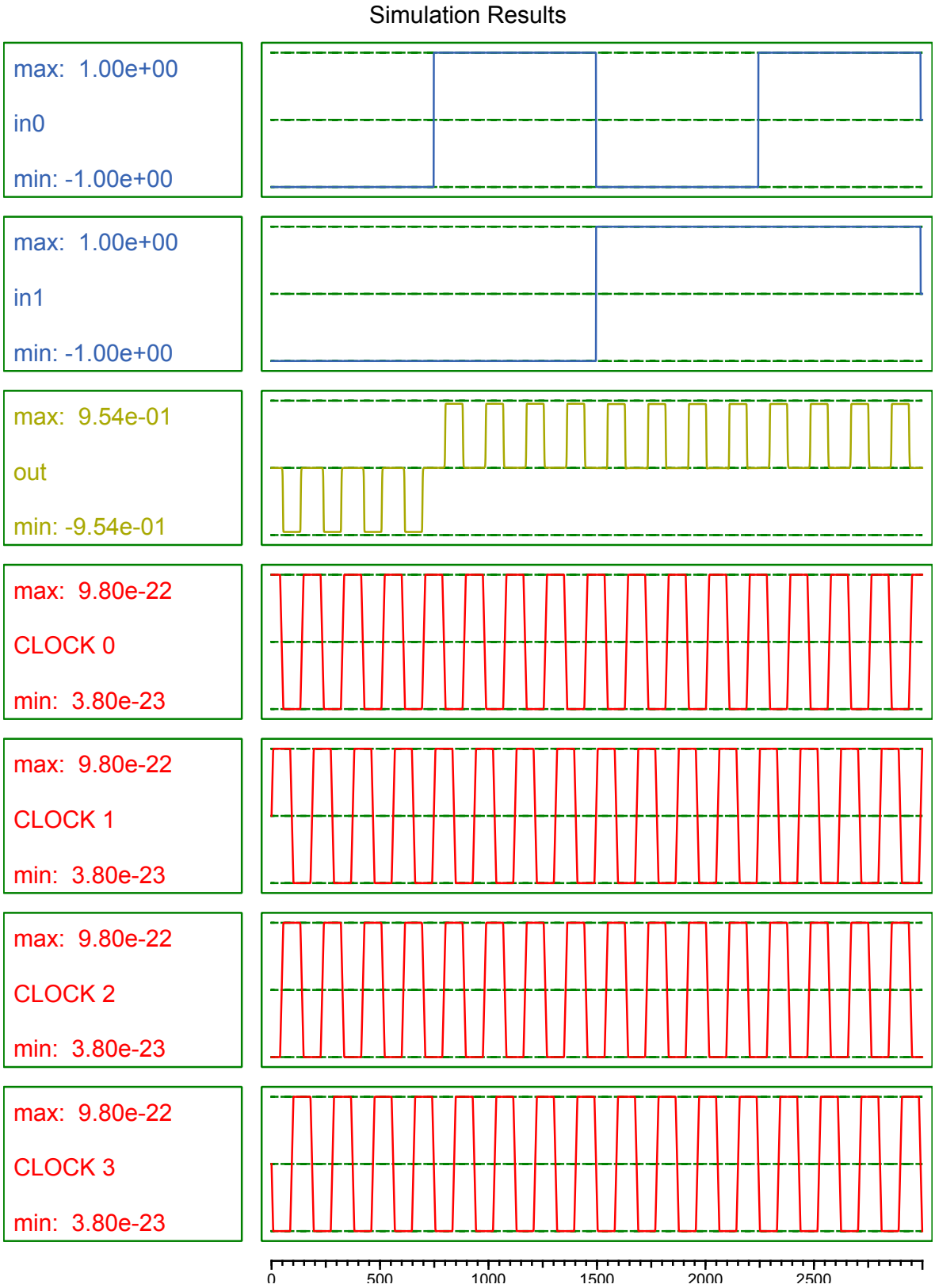


Figure A.6: Simulation results for the OR standard cell

A.4 NAND

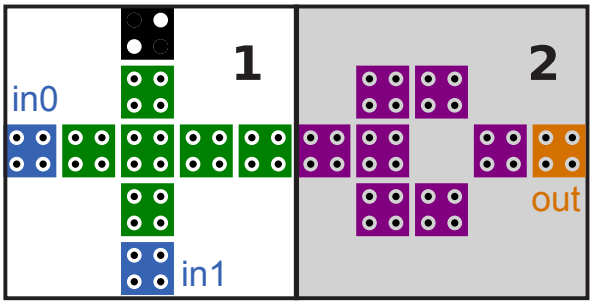


Figure A.7: Standard cell of a NAND inside *USE* grid

name: nand

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out = -(in0 \ \& \ in1)$

zone_dimension: 5

width: 10
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1
out 1 1

port_location:
in0 0 2 0
in1 2 4 0
out 9 2 0

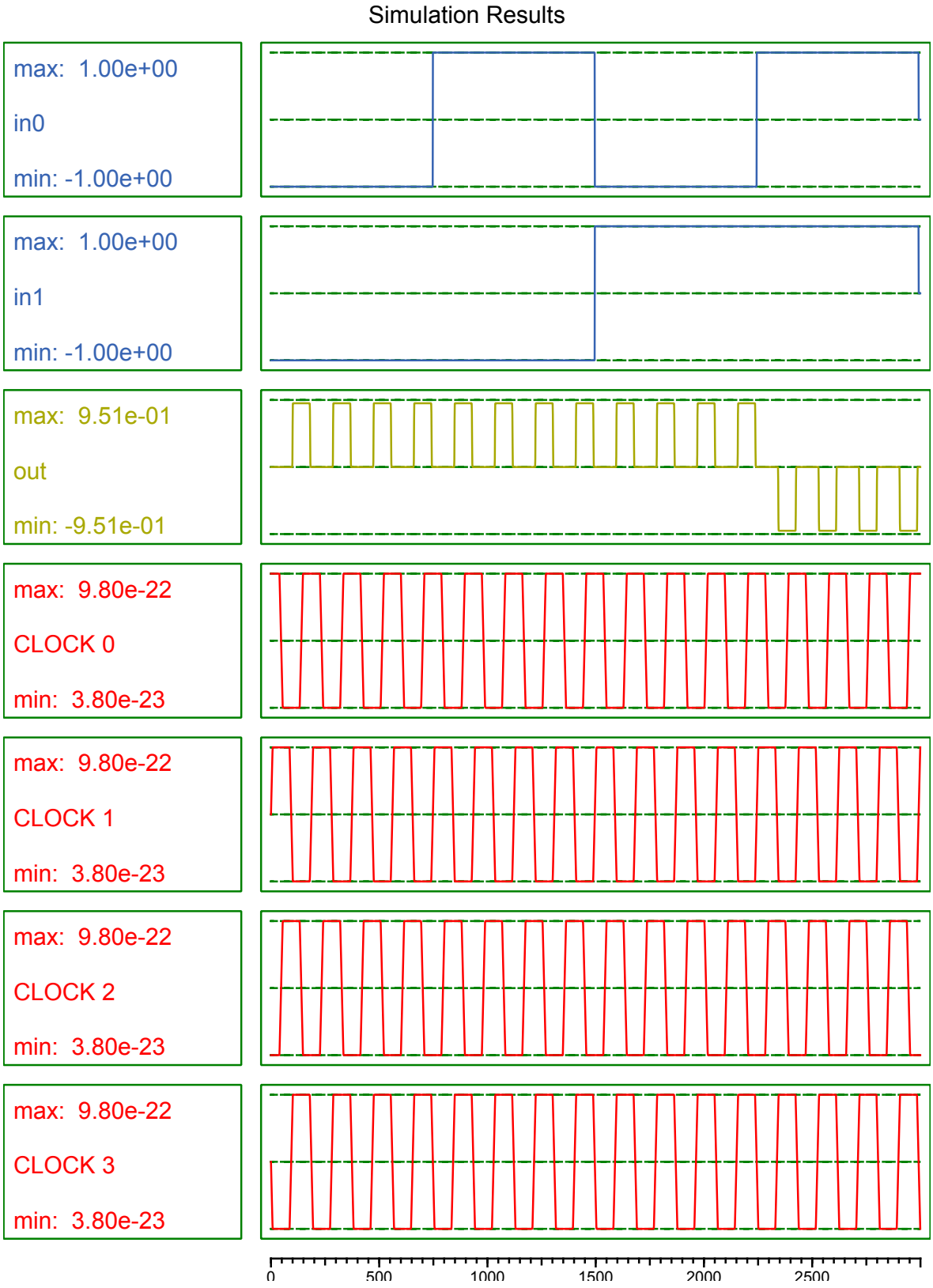


Figure A.8: Simulation results for the NAND standard cell

A.5 NOR

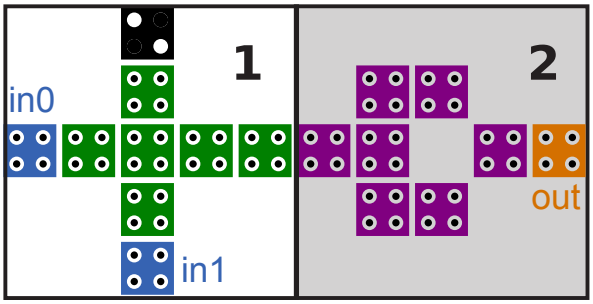


Figure A.9: Standard cell of a NOR inside *USE* grid

name: nor

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out = -(in0 + in1)$

zone_dimension: 5

width: 10
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1
out 1 1

port_location:
in0 0 2 0
in1 2 4 0
out 9 2 0

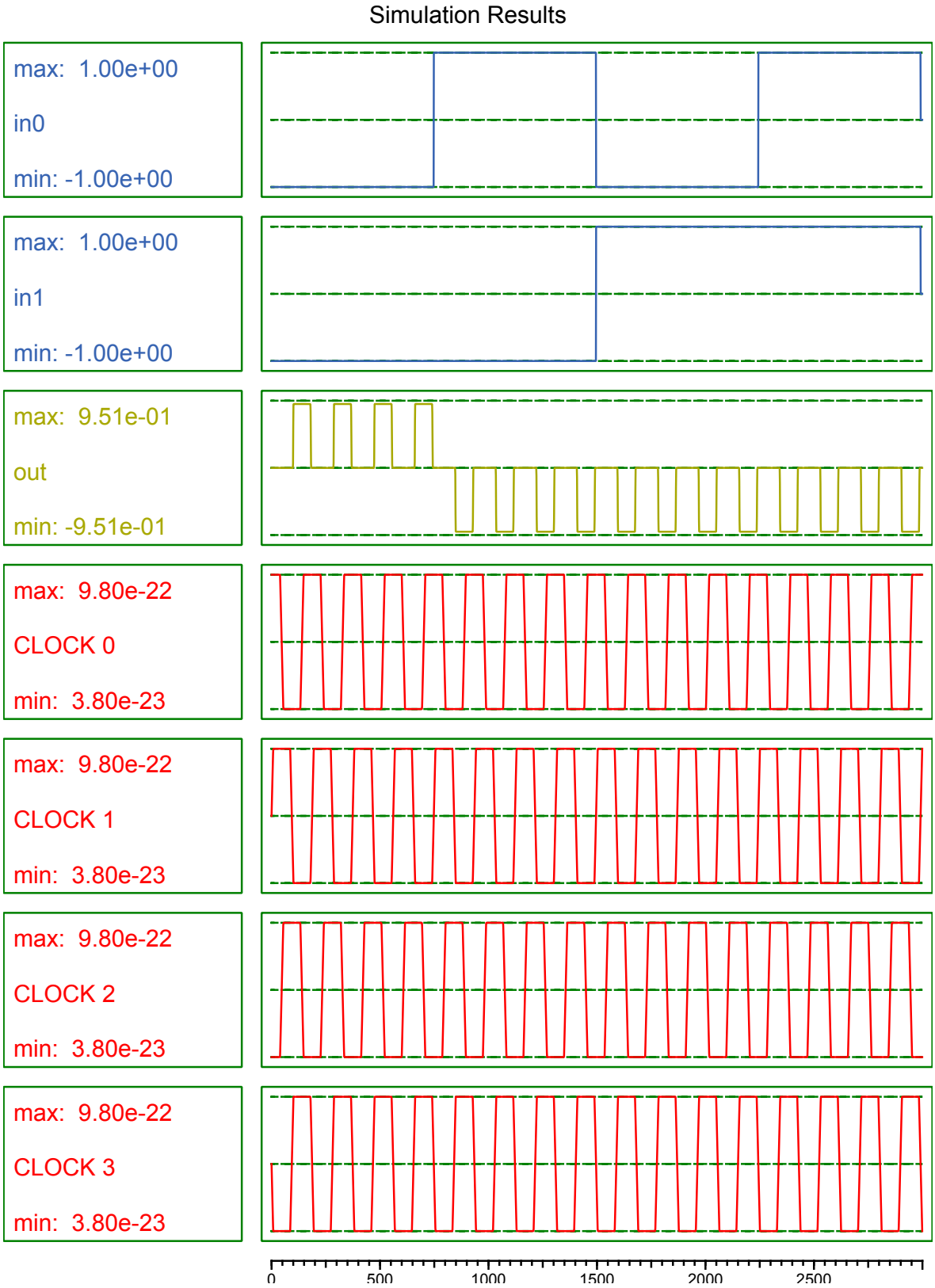


Figure A.10: Simulation results for the NOR standard cell

A.6 XOR

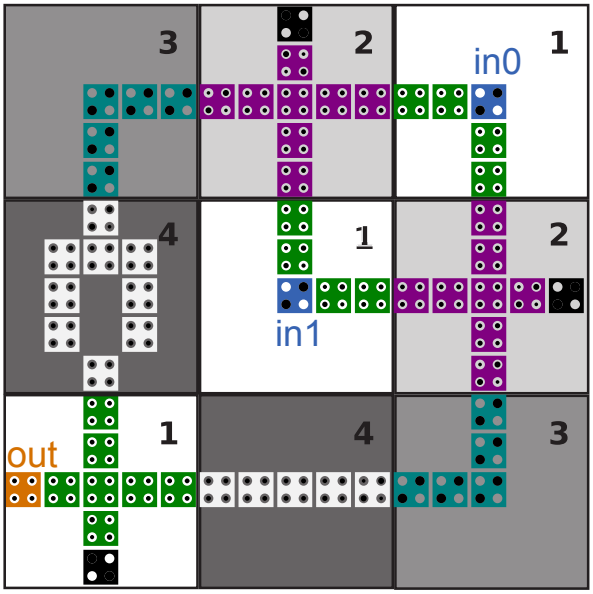


Figure A.11: Standard cell of an exclusive OR inside *USE* grid

name: xor

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out = -(in0 \ \& \ in1) + (in0 + in1)$

zone_dimension: 5

width: 15
height: 15
layers: 1
l_reference_zone: 3
r_reference_zone: 2

delay_table:
/ in0 in1

out 4 4

port_location:
in0 12 2 0
in1 7 7 0
out 0 12 0

free:
5,6 5,9 0,0
7,9 8,9 0,0

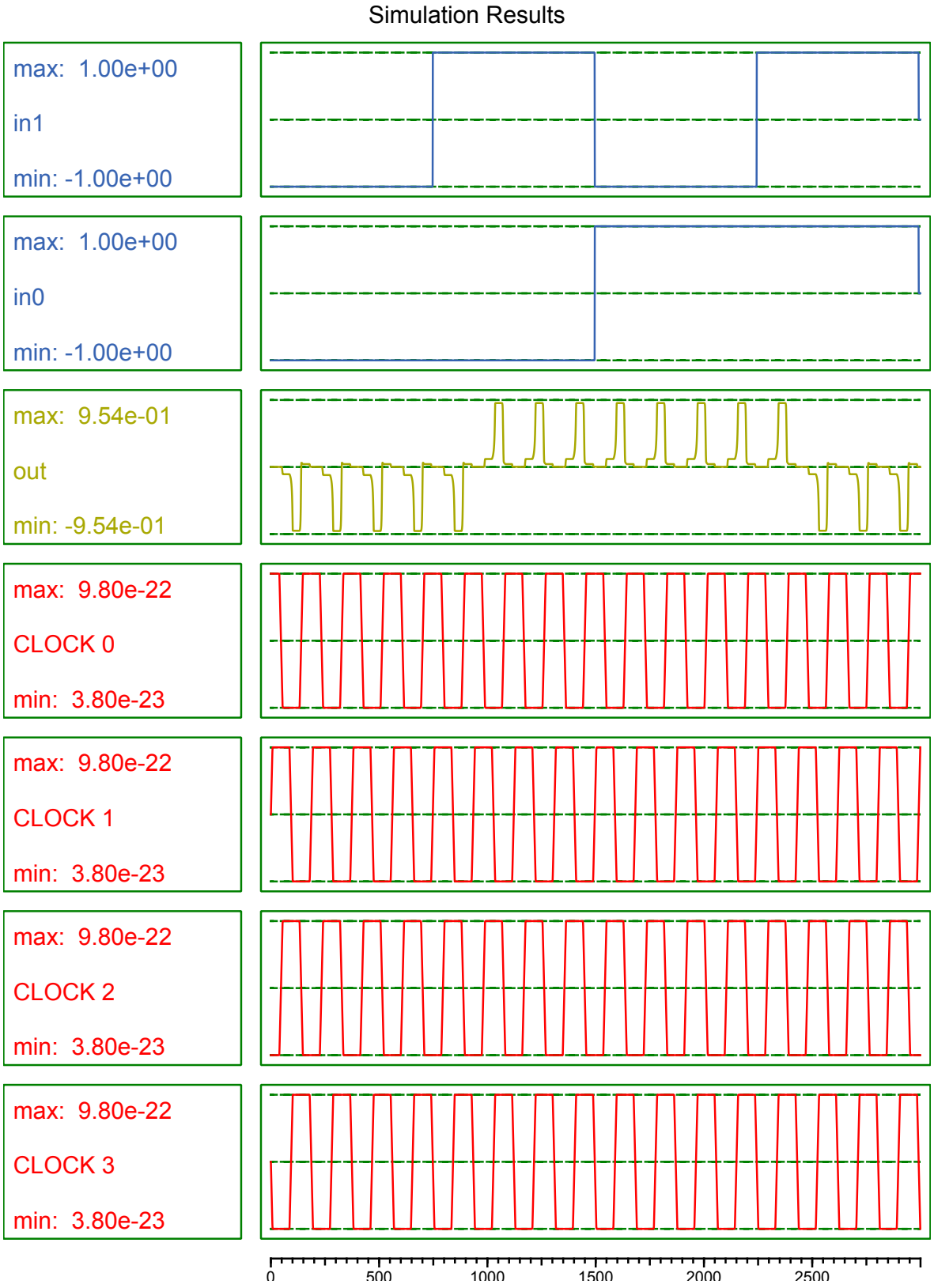


Figure A.12: Simulation results for the exclusive OR standard cell

A.7 MAJ

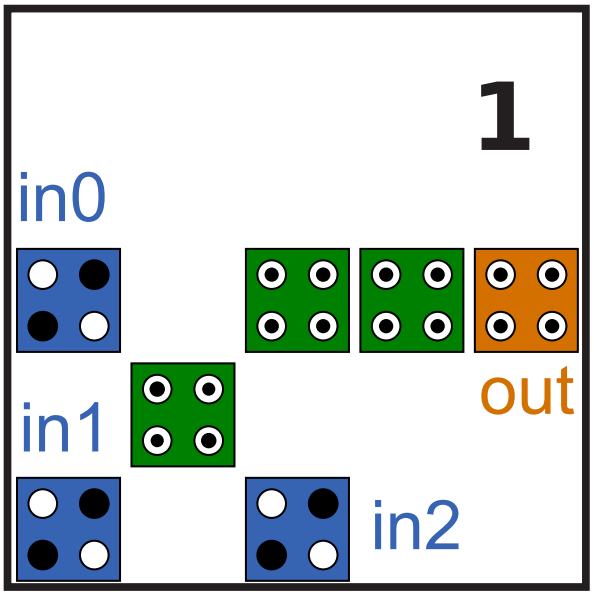


Figure A.13: Standard cell of a majority gate inside *USE* grid

name: maj

n_input: 3
n_output: 1
input: in0 in1 in2
output: out

expression:
$out = (in0 \ \& \ in1) + (in0 \ \& \ in2) + (in1 \ \& \ in2)$

zone_dimension: 5

width: 5
height: 5
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1 in2

out 0 0 0

port_location:
in0 0 2 0
in1 0 4 0
in2 2 4 0
out 4 2 0

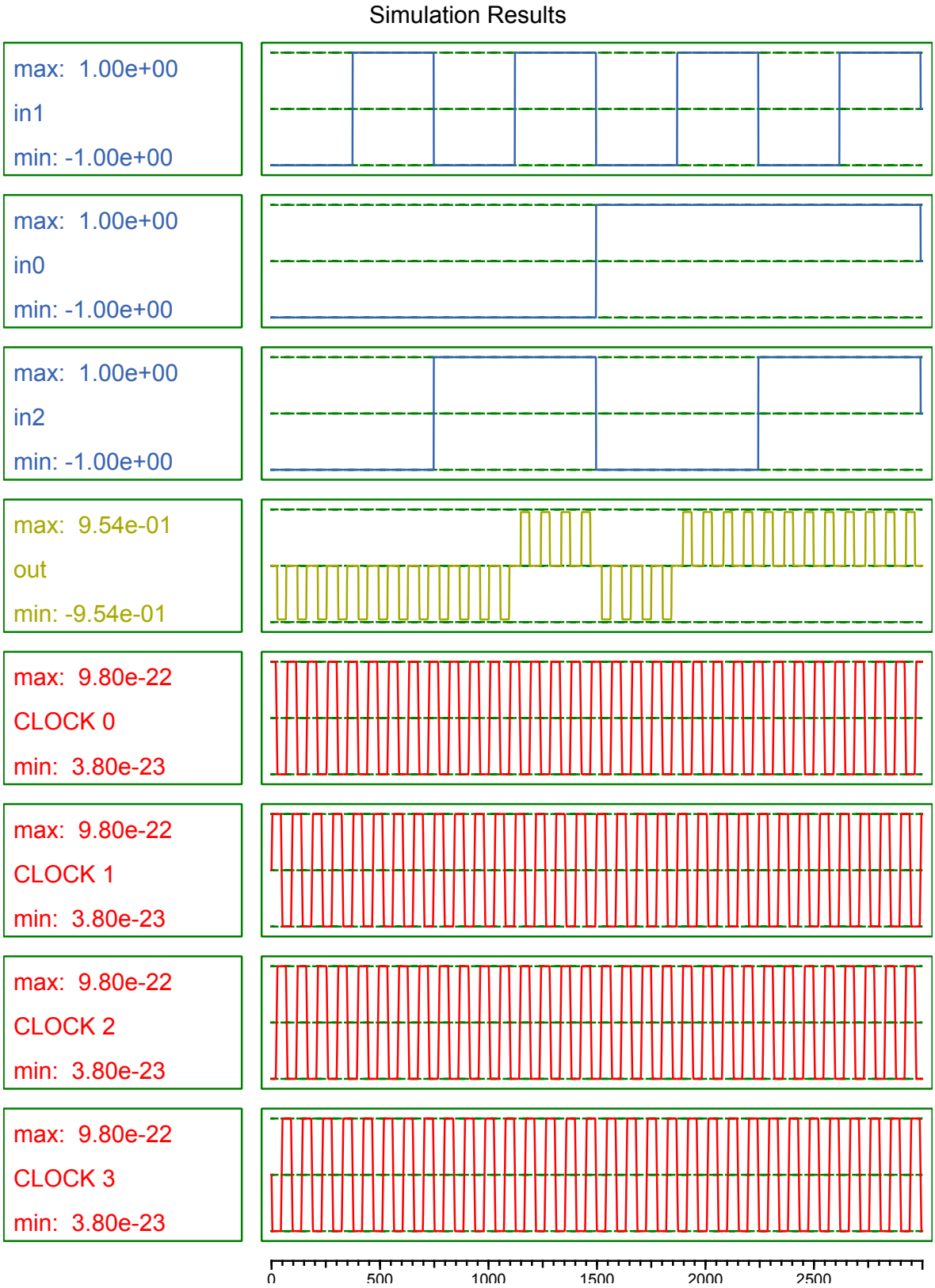


Figure A.14: Simulation results for the majority gate standard cell

A.8 MUX 2-1

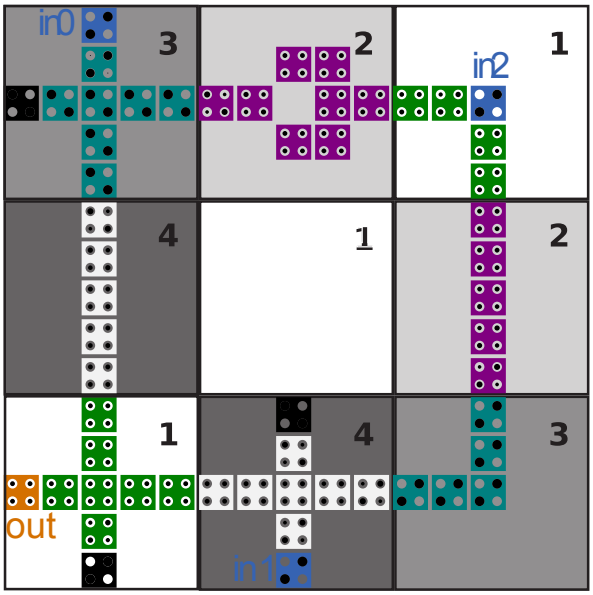


Figure A.15: Standard cell of an 2-to-1 multiplexer inside *USE* grid

name: mux

n_input: 3
n_output: 1
input: in0 in1 in2
output: out

expression:
$out = (in0 \ \& \ -in2) + (in0 \ \& \ in1)$

zone_dimension: 5

width: 15
height: 15
layers: 1
l_reference_zone: 3
r_reference_zone: 2

delay_table:
/ in0 in1 in2

out 2 1 4

port_location:
in0 2 0 0
in1 7 14 0
in2 12 2 0
out 0 12 0

free:
5,9 5,9 0,0

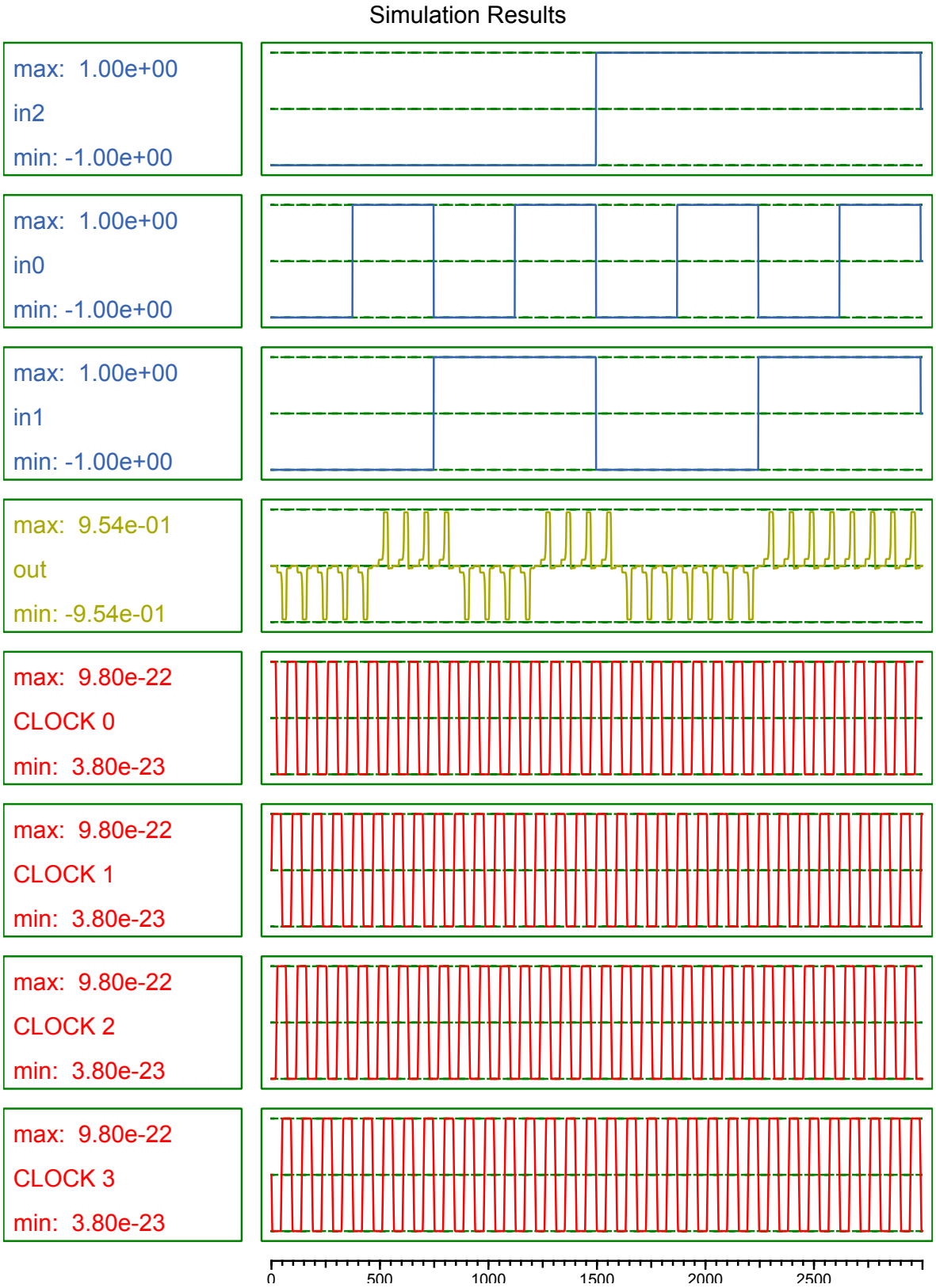


Figure A.16: Simulation results for the 2-to-1 multiplexer standard cell

A.9 SR-Latch

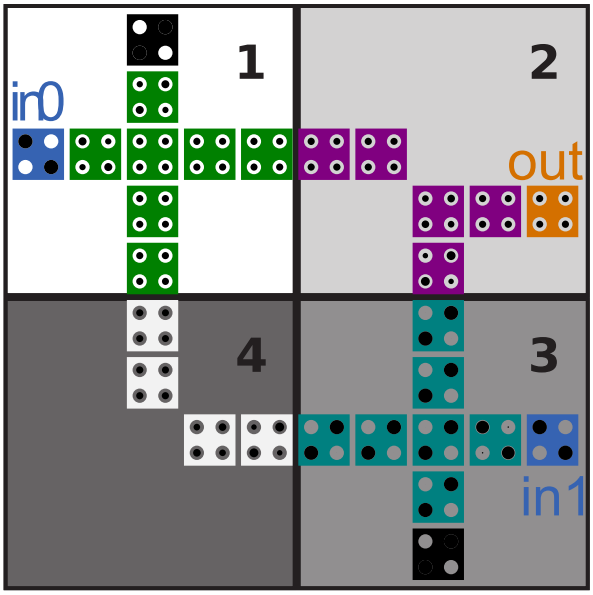


Figure A.17: Standard cell of an SR-Latch inside *USE* grid

name: sr_latch

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out[t] = -(in0 + -out[t-1]) + in1$

zone_dimension: 5

width: 10
height: 10
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in1

out 3 1

port_location:
in0 0 2 0
in1 9 7 0
out 9 3 0

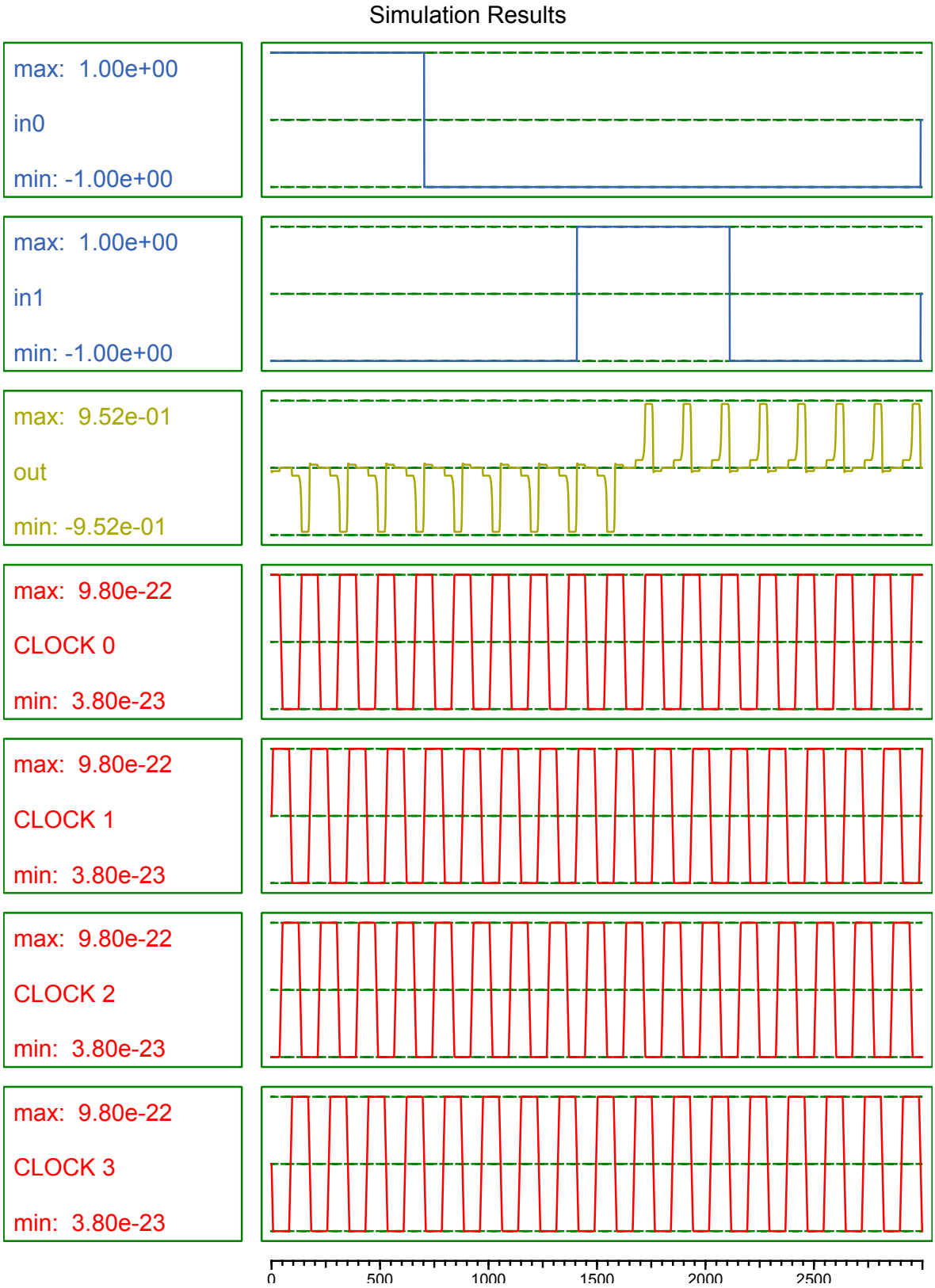


Figure A.18: Simulation results for the SR-Latch standard cell

A.10 D-Latch

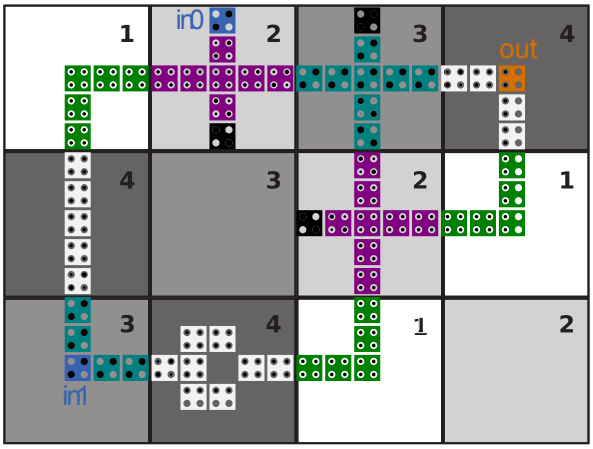


Figure A.19: Standard cell of an D-Latch inside *USE* grid

name: d_latch

n_input: 2
n_output: 1
input: in0 in1
output: out

expression:
$out[t] = (in0 \ \& \ in1) + (out[t-1] \ \& \ in1)$

zone_dimension: 5

width: 20
height: 15
layers: 1
l_reference_zone: 1
r_reference_zone: 2

delay_table:
/ in0 in2
out 2 5

port_location:

in0 7 0 0
in1 2 12 0
out 17 2 0

free:
5,9 5,9 0,0
15,19 10,14 0,0

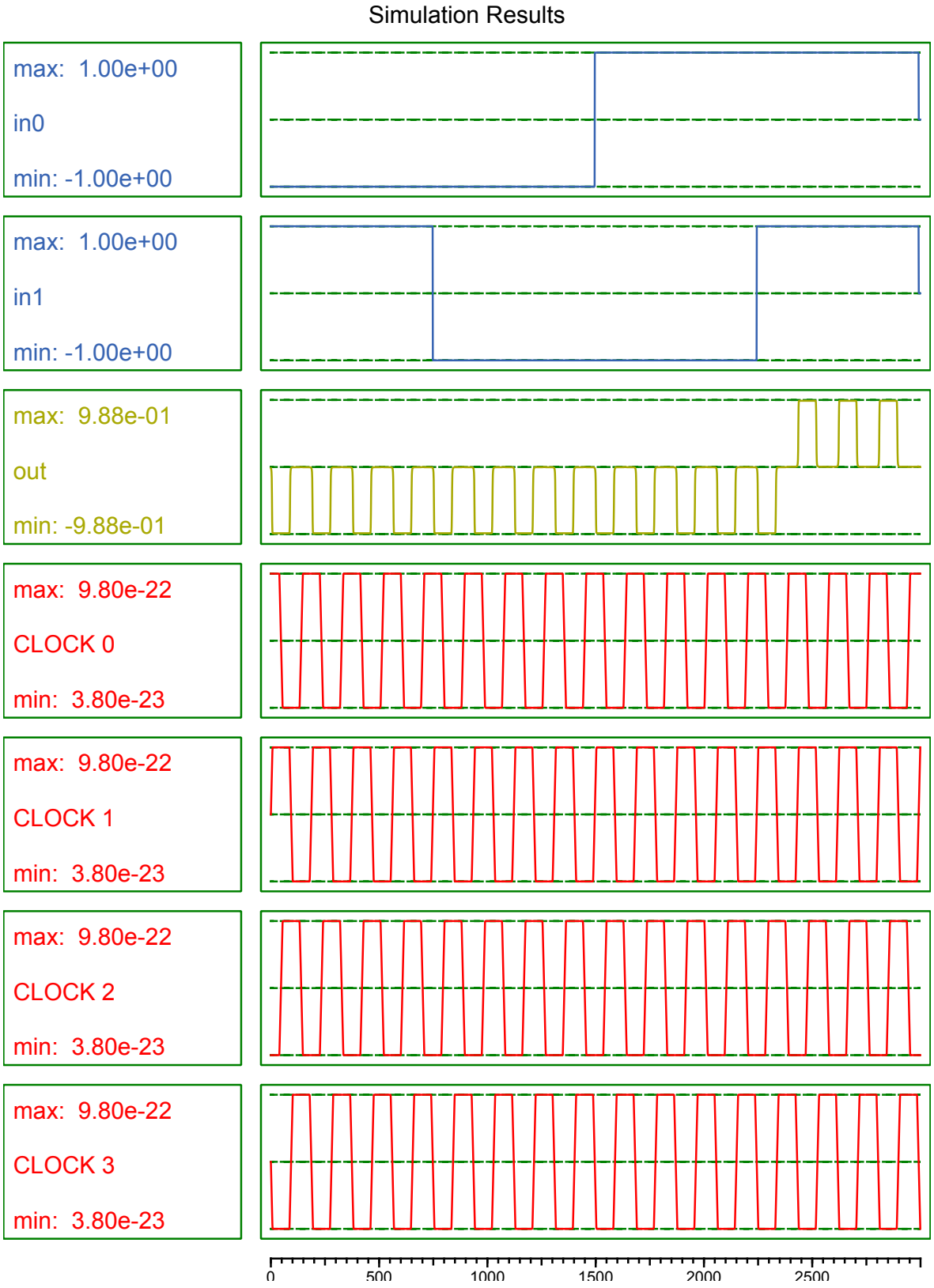


Figure A.20: Simulation results for the D-Latch standard cell

Bibliography

Ahmad, F., Bhat, G. M., and Ahmad, P. Z. (2014). Novel adder circuits based on quantum-dot cellular automata (qca). *Circuits and Systems*, 2014.

Akers, S. B. (1962). Synthesis of combinational logic using three-input majority gates. In *Switching Circuit Theory and Logical Design, 1962. SWCT 1962. Proceedings of the Third Annual Symposium on*, pages 149--158. IEEE.

Bernstein, G., Imre, A., Metlushko, V., Orlov, A., Zhou, L., Ji, L., Csaba, G., and Porod, W. (2005). Magnetic {QCA} systems. *Microelectronics Journal*, 36(7):619 – 624. ISSN 0026-2692. European Micro and Nano Systems {EMN} 2004.

Bernstein, K., Cavin, R. K., Porod, W., Seabaugh, A., and Welser, J. (2010). Device and architecture outlook for beyond cmos switches. *Proceedings of the IEEE*, 98(12):2169--2184.

Cho, H. and Swartzlander, E. E. (2007). Adder designs and analyses for quantum-dot cellular automata. *Nanotechnology, IEEE Transactions on*, 6(3):374--383.

Frank, D. J. (2002). Power-constrained cmos scaling limits. *IBM Journal of Research and Development*, 46(2.3):235--244.

Giri, D., Vacca, M., Causapruno, G., Rao, W., Graziano, M., and Zamboni, M. (2014). A standard cell approach for magnetoelastic nml circuits. In *Nanoscale Architectures (NANOARCH), 2014 IEEE/ACM International Symposium on*, pages 65--70. IEEE.

Haensch, W., Nowak, E. J., Dennard, R. H., Solomon, P. M., Bryant, A., Dokumaci, O. H., Kumar, A., Wang, X., Johnson, J. B., and Fischetti, M. V. (2006). Silicon cmos devices beyond scaling. *IBM Journal of Research and Development*, 50(4.5):339--361.

Haron, N. and Hamdioui, S. (2008). Why is cmos scaling coming to an end? In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 98–103.

Henderson, S. C., Johnson, E. W., Janulis, J. R., and Tougaw, P. D. (2004). Incorporating standard cmos design process methodologies into the qca logic design process. *Nanotechnology, IEEE Transactions on*, 3(1):2--9.

Hennessy, K. and Lent, C. S. (2001). Clocking of molecular quantum-dot cellular automata. *Journal of Vacuum Science & Technology B*, 19(5):1752--1755.

Huo, Z., Zhang, Q., Haruehanroengra, S., and Wang, W. (2006). Logic optimization for majority gate-based nanoelectronic circuits. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–1310.

ITRS (2000). International technology roadmap for semiconductors.

Janez, M., Pecar, P., and Mraz, M. (2012). Layout design of manufacturable quantum-dot cellular automata. *Microelectronics Journal*, 43(7):501--513.

Jorgenson, D. (2005). Moore's law and the emergence of the new economy. *Semiconductor Industry Association 2005 annual report: 2020 is closer than you think*, pages 16--20.

Keutzer, K. and Chinnery, D. (2000). Closing the gap between asic and custom: An asic perspective. In *Design Automation Conference*, pages 637--642. ACM.

Lent, C. S. (2000). Bypassing the transistor paradigm. *Science*, 288(5471):1597--1599.

Lent, C. S. and Tougaw, P. D. (1997). A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 85(4):541--557.

Lent, C. S., Tougaw, P. D., Porod, W., and Bernstein, G. H. (1993). Quantum cellular automata. *Nanotechnology*, 4(1):49.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8).

Niemier, M. T. (2004). *Designing digital systems in quantum cellular automata*. PhD thesis, University of Notre Dame.

Rairigh, D. (2005). Limits of cmos technology scaling and technologies beyond-cmos. *Institute of Electrical and Electronics Engineers, Inc.*

Sardinha, L. H., Costa, A. M. M., Neto, O. P. V., Vieira, L. F. M., and Vieira, M. A. M. (2013). Nanorouter: A quantum-dot cellular automata design. *Selected Areas in Communications, IEEE Journal on*, 31(12):825--834.

Sen, B., Dutta, M., and Sikdar, B. K. (2014). Efficient design of parity preserving logic in quantum-dot cellular automata targeting enhanced scalability in testing. *Microelectronics Journal*, 45(2):239--248.

Teja, V. C., Polisetti, S., and Kasavajjala, S. (2008). Qca based multiplexing of 16 arithmetic & logical subsystems-a paradigm for nano computing. In *Nano/Micro Engineered and Molecular Systems, 2008. NEMS 2008. 3rd IEEE International Conference on*, pages 758--763. IEEE.

Toth, G. (2000). *Correlation and coherence in quantum-dot cellular automata*.

Tougaw, P. D. and Lent, C. S. (1994). Logical devices implemented using quantum cellular automata. *Journal of Applied physics*, 75(3):1818--1825.

Vankamamidi, V., Ottavi, M., and Lombardi, F. (2008). Two-dimensional schemes for clocking/timing of qca circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(1):34--44.

Walus, K., Dysart, T. J., Jullien, G. A., and Budiman, R. A. (2004). Qcadesigner: A rapid design and simulation tool for quantum-dot cellular automata. *Nanotechnology, IEEE Transactions on*, 3(1):26--31.

Walus, K., Jullien, G., and Dimitrov, V. (2003). Computer arithmetic structures for quantum cellular automata. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pages 1435--1439. IEEE.

Welland, M. E. and Gimzewski, J. K. (1995). *Ultimate limits of fabrication and measurement*. Springer.