DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Electrical Engineering

Signal Distribution Networks in Automatic QCA Standard Cell Placement and Routing

Benjamin Hien

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Electrical Engineering

Signal Distribution Networks in Automatic QCA Standard Cell Placement and Routing

Signalverteilungs-Netzwerke in automatisierter QCA Standard Zellen Plazierung und Verdrahtung

Author:	Benjamin Hien
Supervisor:	Prof. Dr. Robert Wille
Advisor:	Dr. Marcel Walter
Submission Date:	08.02.2023

I confirm that this master's thesis in electrical engineering is my own work and I have documented all sources and material used.

Munich, 08.02.2023 Benjamin Hien

Acknowledgments

Abstract

New technologies to compete with CMOS, one of them QCA
Placement and Routing as key to producibility.
Challenges of Placement and Routing in previous algorithms.
Goals of this work:
I minimizing area/tiles,
II making it possible to place majority-gates (which is a promising aspect of QCA),
III making it possible to P&R sequential circuits
This is done by introducing several signal distribution networks
Results are compared with already existing algorithms...

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**
1.1 Motivation 1
1.2 Objective 1

2 Preliminaries **2**
2.1 Representation of Logic Circuits 2
2.1.1 Boolean Functions 2
2.1.2 Logic Networks 3
2.2 QCA Technology 7
2.2.1 Cells 7
2.2.2 Clocking 8
2.2.3 Gates 13
2.3 P&R problem 17

3 State of the Art **20**
3.1 Combinational P&R Algorithms 20
3.2 Sequential P&R Algorithms 26

4 Methodology **28**
4.1 Input Network 28
4.2 Majority Gates Placement Network 28
4.3 Sequential Circuits Placement 28

5 Experimental Evaluation **29**
5.1 Benchmarks 29
5.2 Results 29

List of Figures **30**

List of Tables **31**

Bibliography **32**

1 Introduction

1.1 Motivation

About the technology, why its promising and important.
Lack of automated algorithms for P&R
P&R as sign of producibility
Why the distribution networks are able to make QCA better producible / cheaper

1.2 Objective

The thesis is divided in ...

2 Preliminaries

2.1 Representation of Logic Circuits

The Boolean Algebra, formed by mathematician George Boole in 1847, shows that every digital circuit can be represented by logic functions, independent of their underlying technology. Until today his work is the foundation for defining and discussing about them.

2.1.1 Boolean Functions

A definition of the Boolean calculus was first provided by Edward V. Huntington in 1993. From the *set of independent postulates for the algebra of logic* and his own correction [12, 11], the following definition can be derived:

Definition 2.1.1 (Basis for Boolean algebra)**.** Given a finite set S, two binary functions $\cdot : S \times S \to S$ and $+ : S \times S \to S$, and one unary function $\neg : S \to S$, the tuple $(S, \cdot, +, \neg)$ is called a Boolean algebra iff the following constraints hold for all $a, b, c \in S$ and the *universe element* $u \in S$:

(1)	$a \cdot b = b \cdot a$	$a + b = b + a$
(2)	$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$	$a + (b \cdot c) = (a + b) \cdot (a + c)$
(3)	$\exists u \in S : a \cdot u = a$	$\exists \neg u \in S : a + \neg u = a$
(4)	$\exists \neg u \in S : a \cdot \neg a = \neg u$	$\exists u \in S : a + \neg a = u.$

With the constraints describing (1) *commutativity*, (2) *distributivity*, (3) *neutrality*, and (4) *complementarity*.

The most common Boolean algebra is defined by the 6-tuple $(\mathbb{B}, \vee, \wedge, \neg, 0, 1)$, where $\vee$ and $\wedge$ are another denotations for the binary operands for disjunction $+$ and conjunction $\cdot$ in $\mathbb{B}$, while $\neg$ is the known unary negation function. The set S holds two distinct elements $\{0, 1\}$, with $u = 1$ and $\neg u = 0$ respectively. Negation $\neg u$ is also commonly notated as $\bar{u}$.

Since this definition is restricting the use of only the three Boolean functions $(\vee, \wedge, \neg)$, we want to extend it by the following definition [26]:

Definition 2.1.2. A function $f : \mathbb{B}^n \to \mathbb{B}$, where $n \in N$, is called a Boolean function. Analogously, a function f : $\mathbb{B}^n \to \mathbb{B}^m$, where $n, m \in N$, is called multi-output Boolean function and can be interpreted as $f_v = (f_{v1}, ..., f_{vm})$, where $f_{vi} : \mathbb{B}^n \to \mathbb{B}$, for all $1 \leq i \leq m$.

A common notation for Boolean functions are the *conjunctive normal form* (CNF) and *disjunctive normal form* (DNF), using literals.

Definition 2.1.3. A literal is either an atom a (positive literal) or the negation of an atom $\neg a$ (negative literal).

Definition 2.1.4. A propositional Boolean formula is said to be in CNF if it is a conjunction of *clauses*, each of which is a disjunction of literals [29]:

$$\bigwedge_i \bigvee_j (\neg) v_{ij},$$

where $v_{ij} \in \mathbb{B}$.
A propositional Boolean formula is said to be in DNF if it is a disjunction of clauses, each of which is a conjunction of literals:

$$\bigvee_i \bigwedge_j (\neg) v_{ij},$$

where $v_{ij} \in \mathbb{B}$.

Using the CNF or rather the DNF and *De Morgan's laws* following from the definitions in 2.1.1, it follows that any Boolean Algebra can be reduced to only two operands, e.g. conjunction ($\vee$) and negation ($\neg$). Any set of such two Boolean functions is called *universal*.

2.1.2 Logic Networks

There are many ways of representing Boolean Functions. But most of them, including e.g. truth tables or reduced sum of products, suffer from drawbacks like exponential representations of basic functions like the parity (XOR) function. Even if a reasonable representation exists for a given function, simple operations like forming the complementary could yield an exponential function representation. Logic networks overcome these restrictions and have proven to be very useful transforming a circuit described by logic functions into a gate representation. This process is also referred to as *logic synthesis*. An approach for logic networks is given in [3], describing *function graphs*:

Definition 2.1.5 (Function graph)**.** A function graph is a rooted, directed graph with vertex set V containing two types of vertices. A *nonterminal* vertex v has as attributes an argument index $index(v) \in \{1, ..., n\}$, and two children $low(v), high(v) \in V$. A *terminal* vertex v has as attribute a value $value(v) \in \{0, 1\}$.

Furthermore, for any nonterminal vertex v, if $low(v)$ is also nonterminal, then we must have $index(v) < index(low(v))$. Similarly, if $high(v)$ is nonterminal, then we must have $index(v) < index(high(v))$.

Definition 2.1.6 (Function Graph Boolean Functions)**.** A function graph G having root vertex v denotes a function f_v defined recursively as:

1. If v is a terminal vertex:
   a) If $value(v) = 1$, then $f_v = 1$
   b) If $value(v) = 0$, then $f_v = 0$
2. If v is a nonterminal vertex with $index(v) = i$, then f_v is the function $f_v(x_1, ..., x_n) = \bar{x}_i \cdot f_{low(v)}(x_1, ..., x_n) + x_i \cdot f_{high(v)}(x_1, ..., x_n)$.

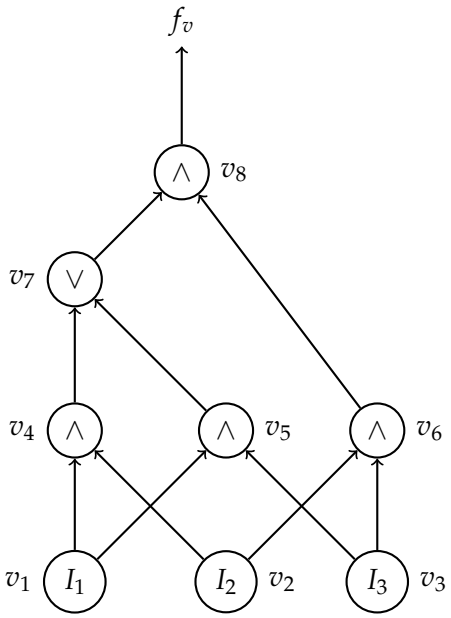
Since the definition reduces the number of children connected to a vertex to two, therefore only allowing binary Boolean Functions, a custom definition is given:

Definition 2.1.7 (Logic Network)**.** A logic network $N(V, E)$ is a rooted, directed graph with vertex set V and edge set E. For any vertex $v \in V$, vertices connected by incoming edges $e_{inc} \in E$ are called children. A vertex connected by and outgoing edge $e_{out} \in E$ is called parent. V contains two types of vertices. A *nonterminal* vertex v has as attributes an argument index $index(v) \in \{1, ..., n\}$, and l children $child_1(v), ..., child_l(v) \in V$. A *terminal* vertex v has as attribute a value $value(v) \in \{0, 1\}$.

Furthermore, for any nonterminal vertex v, if $child_i(v)$ with $1 \leq i \leq l$, then we must have $index(child_i(v)) < index(v)$ respectively.

Definition 2.1.8 (Logic Network Boolean Functions)**.** A set of nary Boolean Functions $x_1, ..., x_n \in \mathbb{B}$ is assigned to every vertex via the argument index $index(v) = i$. The graph function f_v is defined recursively as:

1. If v is a terminal vertex:
   a) If $value(v) = 1$, then $f_v = 1$
   b) If $value(v) = 0$, then $f_v = 0$
2. If v is a nonterminal vertex with $index(v) = i$, then f_v is the function $f_v(v_i) = x_i(f_{child_1(v)}(v_{i-1}), ..., f_{child_l(v)}(v_{i-n}))$.



The corresponding recursive Boolean functions read:

$$
\begin{array}{ll}
f_v = f_v(v_8) & f_v(v_6) = f_v(v_2) \vee f_v(v_3) \\
f_v(v_8) = f_v(v_7) \wedge f_v(v_6) & f_v(v_5) = f_v(v_3) \vee f_v(v_1) \\
f_v(v_7) = f_v(v_4) \wedge f_v(v_5) & f_v(v_4) = f_v(v_2) \vee f_v(v_1)
\end{array}
$$

with primary inputs: $f_v(v_1), f_v(v_2), f_v(v_3) \in \{0, 1\}$

Figure 2.1: Binary Logic Network of Majority Function

The binary Logic Network of the ternary majority function is depicted figure 2.1.

Definition 2.1.9 (Majority Function)**.** The ternary Boolean majority function is defined as: $\langle a, b, c \rangle = ab + ac + bc$, so that the function value equals the majority of it's incoming values.
It follows: $\langle a, b, 0 \rangle = a \cdot b$ and $\langle a, b, 1 \rangle = a + b$.

Adapting the names used in the underlying libraries used to program the algorithms proposed subsequently following in this work (chapter 4), a terminal vertex is referred to as *primary input* (PI) with their set denoted as I. The set of nonterminal vertices referred to as *nodes* is denoted as Λ. From the definition follows $I \cap \Lambda = \emptyset$. An edge connecting a children v_i and parent vertex v_j is called a *signal*. With $i < j$ the notation of a signal is given as (v_i, v_j). The set of all signals is denoted as Σ. If an edge is dangling, so it doesn't point to another vertex, it is called *primary output* (PO) and their set is denoted as O. Therefore also $\Sigma \cap O = \emptyset$ holds true. From the definition of a logic network we can now describe it as acyclic directed graph $N = (\Lambda, I, \Sigma, O)$.
As already mentioned in subsection 2.1.1, a universal set of two Boolean functions can form any Boolean algebra. As long as this universality is contained the set of node functions in a logic network can be extended arbitrarily. Common logic networks containing only two network functions are e.g. *AND-Inverter Graphs* (AIGs) allowing only conjunction and negation. Another widely used binary logic network is the *Majority-Inverter Graphs* (MIGs) utilizing the ternary majority function and negation. But there also exists a wide range of logic networks permitting more than just two node functions. One widely used example is the *XOR-AND-Inverter Graph* (XAG) with the parity function, conjunction and negation functions respectively.

As part of the logic synthesis, a suitable logic network representation of the combinational circuit has to be determined. Because, even though these logic networks can implement any Boolean function given in a specification, not every logic network can be synthesized into any given technology. Looking at the current standard technology *complementary metal-oxidesemiconductor* (CMOS), the logic network is then synthesized by using building blocks consisting of *metal-oxide-semiconductor field-effect transistors* (MOSFETs), the elemental unit in this technology.

One drawback, that sticks to this definition of logic networks and holds also true for the aforementioned representations is that they are all *non-canonical*, which means that a given function can be represented by different logic networks. This property can be explained by the fact that nodes with the identity function are allowed, which can be inserted everywhere in the logic network, while the function representation of the logic network stays the same. Even the exclusion of such identity nodes has no

impact, since simple node combinations, like two negotiation nodes, collapse to the identity function. Following this argumentation, there exists an infinite number of logic networks representing one Boolean Function, resulting in the widely accepted assumption, that the determination of an optimal logic network is a $\mathcal{NP}$-complete problem [26]. Attempts to create canonical logic networks, seem to evade this problem, but include co$\mathcal{NP}$-complete problems in itself [3]. Algorithms used for logic synthesis are therefore based on approximate solutions .

2.2 QCA Technology

Following the well known Moores law, CMOS technology is facing a multitude of challenges. Well-known are e.g. short channel effect, impurity variations, and most importantly the heat, resulting from static and dynamic power losses. To tackle these challenges the International Roadmap for Devices and Systems (IDRS), former ITRS, proposes solutions within the semiconductor domain, e.g. new materials and multi-core architectures. But also new technologies are researched including Quantum computing and the domain of *Field-Coupled Nanocomputing* (FCN). This work focuses on one of the most promising FCN technologies, namely *Quantum dot cellular automata* (QCA). The main difference of this technology compared to CMOS is the representation of logical modes, using the location of electron pairs in QCA-cells, instead of voltage levels. Data between cells is transferred based on Coulomb repulsion, utilizing electromagnetic fields. This enables the technology to achieve high performance in terms of device density, clock frequency and power consumption [18].

2.2.1 Cells

As already mentioned, the elemental unit of this technology is a QCA-cell. Since there in no uniform way of build the quantum dots and connecting them to cells, we look at a rather lower-level abstraction depicted in figure 2.2. The four circles in the corners of the QCA-cells show quantum-dots, that can be implemented by any charge container with discrete electrical energy states. Further a cell contains two excess electrons, which can be localized by the quantum dots. The energy barriers or the quantum dots are able to trap the charge of the electron. If an electron is trapped inside a quantum-dot it is filled black. Due to the Coulomb repulsion the electrons occupy diagonally opposed quantum dots, resulting in two possible stable cell configurations and one unstable cell configuration [20, 15, 16].
A stable states indicates, that it is well distinguishable of the usual energy band. Therefore the energy difference between two consecutive energy states must be well above the thermal noise energy ($k_B T$). Only such states are suited for information

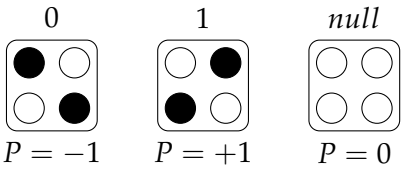


Figure 2.2: QCA-Cell sates

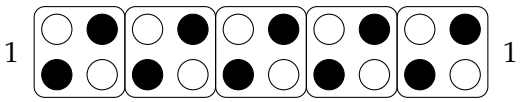


Figure 2.3: Adjacent QCA-cells forming a wire segment

transfer. The stable states can be derived from the cell Polarization, which can be $+1$ and -1 or *null* in the unexcited state. The two stable states contain the same electrostatic energy and are used to encode the binary values 0 and 1 [20].
In order to transfer information, cells are placed side by side, whereby the polarization of the driver cell, which is the left most cell inputting the information, changes the polarization of the adjacent cell. When the adjacent cell is polarized it can give its state to the next cell and so on [15]. This simple structure is representing a wire in QCA-technology and its function is depicted in Figure 2.3.

2.2.2 Clocking

As already mentioned, the data transfer in the QCA paradigm is accomplished by cell-to-cell interaction. Given a fixed polarization of a cell, the next cell reacts to the Coulomb repulsion and changes its polarization accordingly. Looking at the wire segment in figure 2.3, the leftmost cell has a fixed polarization and is called the input. After some time the information propagates through to the right most cell, representing the output of the simplified QCA-circuit. The depicted state of the cell, where all neighboring cells have the same polarization is called ground state. Since the logic is deterministic, there exists exactly one ground state configuration of cell polarization for one choice of inputs. When in ground state, the circuit has minimum state energy.

Given a circuit in ground state, when the input of the system is changed, the systems energy rises as depicted in figure 2.5. It results from the so called *kink*-energy, which describes the energy difference between two QCA-cells with opposing polarization. Since the polarization of the cells changes one cell after the other, also the *kink* moves along the cells. After time τ the system dissipates again into the ground state, with newly computed outputs. While the kink-energy stays the same with an increasing

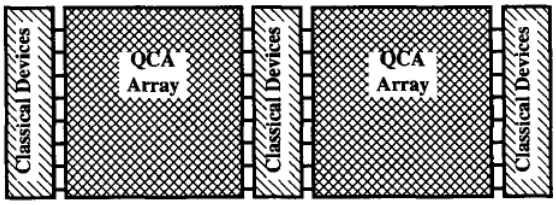


Figure 2.4: Schematic of a combined QCA and CMOS sytem [16].

number of cells, degeneracy of the excited states rises. This again means, that the system can be illegally in an excited state at non-zero temperature [16].

The described process is called abrupt switching with dissipative coupling to the environment and can be summarized in the following three steps [16]:

1. Write the input bit by fixing the polarization state of cells along the input edge.
2. Allow the array to relax to its ground state while the new inputs are kept fixed.
3. Non-invasively read the results of the computation by sensing the polarization state of cells.

In the second step, highly complex dissipation mechanisms like phonon and plasmon emission take place, making it nearly impossible to get a complete theoretical description of the system. The dissipation time τ is therefore determined via experiments. The first and third step require an environment, which provides the features for fixing inputs and sensing the output polarization. Such a system has to be integrated with classical CMOS devices as seen in Figure 2.4.

In order to implement a full QCA-system without CMOS-components, the QCA-Array has to be divided into smaller decoupled sub-regions. Therefore an adiabatic switching or rather a clocking is introduced. The clocked regions are referred to as clocking zones. The clocking utilizes an external signal, the clock, to activate and deactivate said clocking zones. The approach first used, decreases the interdot barriers of all cells in a clocking zone, when applying a new input. When all cells in the region are stable, the barriers are raised again, while the barriers of the subsequent clocking zone are lowered simultaneously. This way the ground state gradually propagates through the whole circuit [15]. Today's used approaches create electrical fields with an external clock generator and distribute it to the cells through the device substrate using embedded electrodes. Thereby the energy level of the *null* state can be controlled,

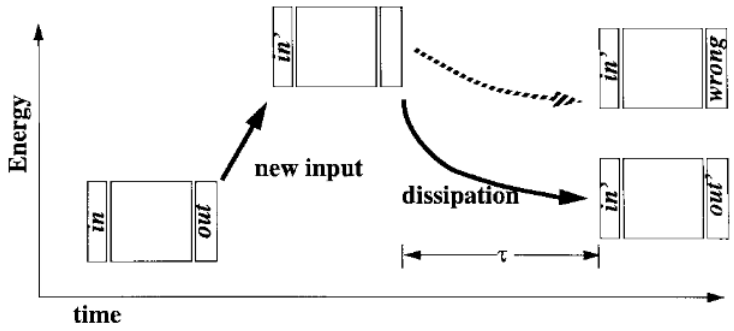


Figure 2.5: Schematic representation of a metastable state. Instead of relaxing correctly to the new ground state, a system may be delayed in an excited state due to an inability to tunnel through a kinetic barrier [15].

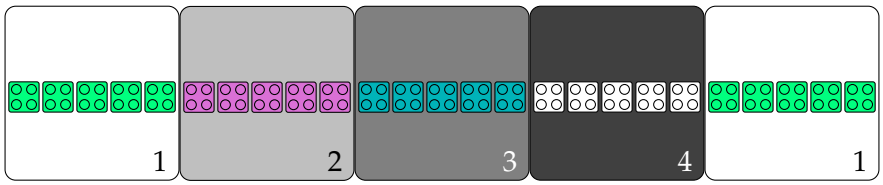


Figure 2.6: QCA-Cell wire with corresponding clock zones

resulting in a equivalent effect as in the former approach [26].

A wire, divided in such clocking zones is shown in 2.3. The colors of the zones and cells as well as the zone number show redundant information about the type of clocking zone. They differ in the external applied electrical field and therefore the energy of the cells. In QCA the clocking is divided into the four consecutive states, *switch*, *hold*, *release* and *relax*. They are aligned in a pipeline like structure, where each of these state is phase shifted by $\pi/2$, forming a 2π clock cycle. In the switch-phase cells start getting polarized dependent on the polarization of the driving cell. When the cells are polarized they get fixed in the hold-phase. Afterwards in the release-phase the excitation gradually decreases, resulting in the unexcited relax-state [20]. The scheme of such a pipeline like clocking is depicted exemplary on a wire segment in Figure 2.7.

The described clocking is named *Landauer clocking*. The inventor Rolf Landauer himself pointed out the vast power dissipation of this clocking mechanism. This impairing property is already well known from the CMOS technology, resulting in several drawbacks including extreme cases of switched off chip areas referred to as dark silicon. One common approach in CMOS is the lowering of circuit frequency utilizing

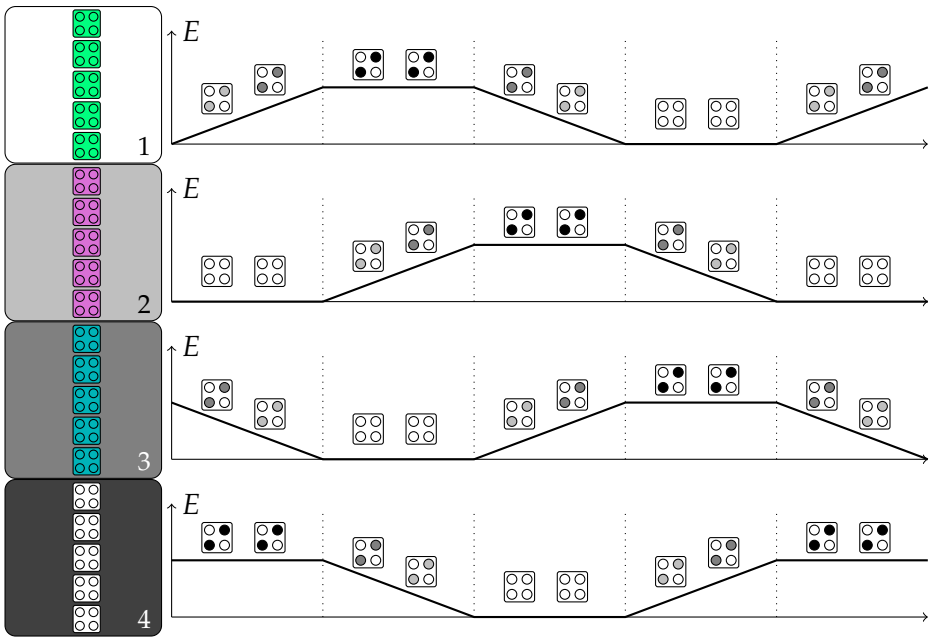
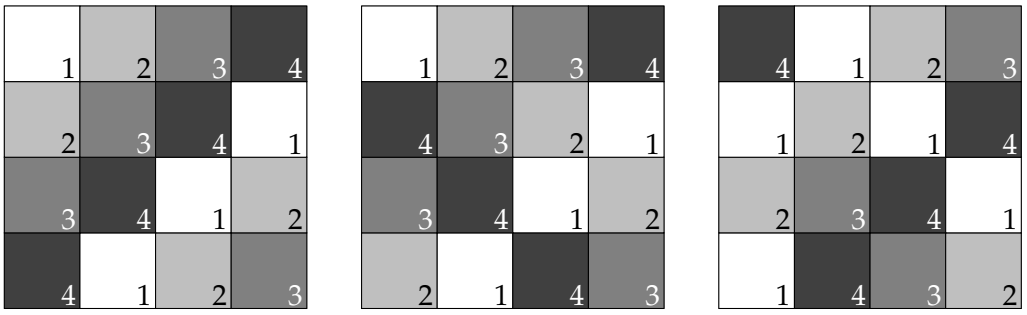


Figure 2.7: QCA clocking pipeline

e.g. multi-chip architectures. To tackle this problem in the QCA domain, Landauer pointed out that the *erase* function has to be eliminated from the clocking. This function is logically irreversible and describes the erasing of information in a clock zone, when passed to the next. He argues that every erased bit dissipates at least $k_B T \ln(2)$ in heat. Exemplary if a QCA-cell has size $1nm \times 1nm$ and operating frequency of $100GHz$, the corresponding density of devices results to $10^{14}\ cm^{-2}$. Further a dissipation of $0.1eV$ per electron every clock cycle is assumed, resulting in a total power dissipation of $160\ kW\ cm^{-2}$. This directly yields to the statement, that a device operating with this clocking would be inoperable (it would evaporate due to the heat) [14]. The *Bennet clocking* tackles exactly this problem by altering the timing of the clocking signals. Just as in the Landauer clocking the clocking-wave moves from left to right, but leaving no trailing edge, when information is passed. Instead the cells will be held in the excited state until the information propagates through the whole QCA-array. When the output was read, the excitation is released in reverse order resulting in no erase functions. This means, that this *quasi-adiabatic* clocking leads to a minimal power dissipation but with two constraints. The effective clock rate is at least halved due to the additional backwards propagation and since only one signal vector can be transmitted through the system, the pipeline capabilities are reduced [14].
In order to apply Bennet clocking, it was already stated that the QCA-array has to be



(a) 2DDWave (b) USE (c) RES

Figure 2.8: Different clocking Schemes in QCA

divided into clock zones. Allowing an arbitrary number of cells in one clock zone gives lots of freedom in designing clock zones with variable geometries. This clocking is referred to as *cell-based* and increases the fabrication process due to its variety in clock zone geometries. Assuming the necessity of an uniform fabrication in order to fabricate circuits with millions of cells, this clocking gets infeasible for large circuits. Also the scheme supports clocking of single cells, which means that electrodes of the same size have to be fabricated. Since this is also not feasible, this design is obsolete.
In order to attain uniform clocking zones with a possible distribution of clocking signals, the *tile-based* clocking is introduced. The approach of this design is to provide uniform square tiles of the size 3×3 or 5×5. For clocking tiles bigger than this, the information propagation was suggested to be erroneous, also following in an argument against cell-based clocking [22].

The tile-based clocking leads to several proposals of clocking-schemes, which give a certain distribution of clock zones. Since they follow an uniform pattern they can be extended easily for every size of the circuit. In 2.8 three clocking schemes are showed, each of them based on a different idea. Since information is only allowed in ascending clock order (except 4 to 1), the 2DDWave clocking scheme in figure 2.8(a) only allows information to propagate in two directions, south and north [25]. This simplicity allows no back propagation, prohibiting the placement of sequential circuits. Also it restricts gates in the scheme to have a maximum input size of two. The USE scheme 2.8(b) tackles the first problem by introducing clocking loops into the scheme, giving the possibility to place sequential circuits [5]. To tackle the second problem, the RES scheme 2.8(c) gives an opportunity to place gates of input size three. Since one tile is restricted to four adjacent cells of which one has to output the information of the cell, this gives the maximum input size allowed. This is especially important for the placement of

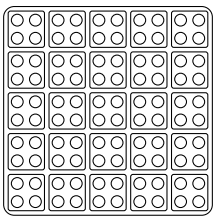
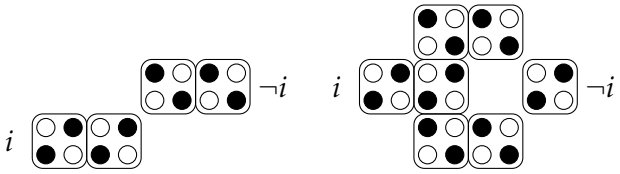


Figure 2.9: Cell representation of tile



(a) 45° Inverter (b) Standard Inverter

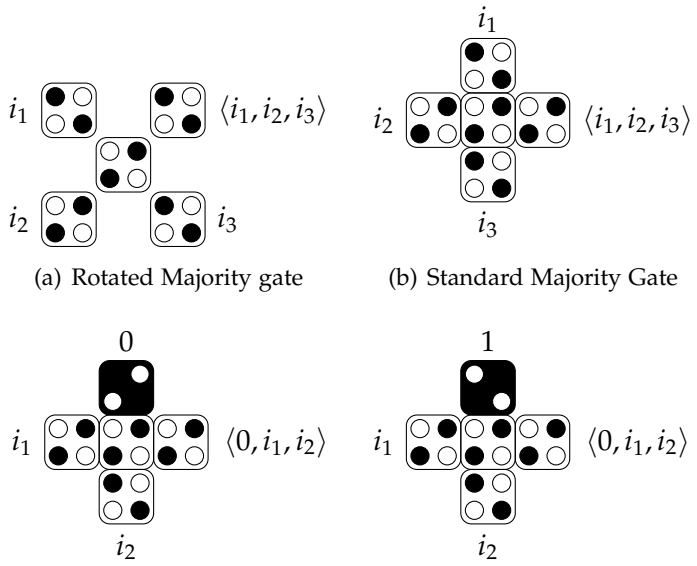
Figure 2.10: Different QCA Inverter representations

majority-gates [9]. In QCA-technology they can be represented by only one tile, making it to a huge advantage over CMOS technology. This is further evaluated in the next subsection on gates.

2.2.3 Gates

In this chapter a library of gates is introduced, which are later used in the placement and routing algorithms. Some of these gates are inherited from the QCA ONE library [19], which is used as base for some works on placement and routing CITE. The QCA ONE library proposes gates formed by one tile as well as gates formed by multiple tiles. A major drawback of this library is the prerequisite of a clocking scheme (USE) in order to form multiple tile gates. This restricts the underlying placement and routing algorithm in the clocking domain. Also manual changes of the standard cells clock zones, size or positioning is not allowed [19], imposing the designer with even more restrictions. For this work, the standard cell library should only contain gates occupying one tile. Every other logic function is composed out of these standard cells. The tiles used are of the dimension 5×5, which means that all standard gates are reduced to this area (figure 2.9).

The first gate in the library is the inverter or NOT gate. The simplest implementation is shown in figure 2.10(a). It consists of two wire segments which are shifted by exactly one cell height, so that the polarization is transferred diagonally resulting in an inversion of the input [20]. In order to get a more robust gate regarding disturbance, the



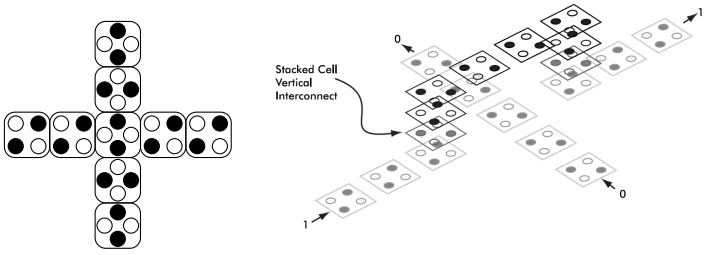
(a) Rotated Majority gate (b) Standard Majority Gate

(c) Standard AND Gate (d) Standard OR Gate

Figure 2.11: The QCA Majority gate

C-shaped inverter shown in figure 2.10(b) is introduced. This gate is used as standard in many libraries and works CITE but it still has to be mentioned that this implementation is really prone to complex single electron faults [17] and even common displacement faults [21], suggesting the addition of an inverter leg resulting to a E-shaped gate structure. Nevertheless the C-shaped inverter gate is part of the QCA ONE library and is also selected as standard cell for this work.

The next gate, which has to be investigated, is the majority gate, being the most important gate in QCA technology. Definition 2.1.9 suggests that the implementation of this function in CMOS technology requires multiple AND and OR gates to be placed and routed. In QCA technology on the other hand a majority function can be represented by exactly one gate, making it one of the major advantages over CMOS technology. There are two main implementations of the majority gate. The rotated majority gate in figure 2.11(a), which is used in QCA ONE, and the +-majority gate 2.11(b). Both of these implementations have their advantages and drawbacks. On the one hand the rotated majority gate exhibits sufficiently high degree of fault-tolerance against cell displacement or misalignment but has very poor degree of fault-tolerance against single cell omission or extra cell deposition [13]. The +-majority gate on the other hand is very prone to cell displacement but is also used as building block for AND and OR gates in most works. This means that the fabrication process for all these gates is very

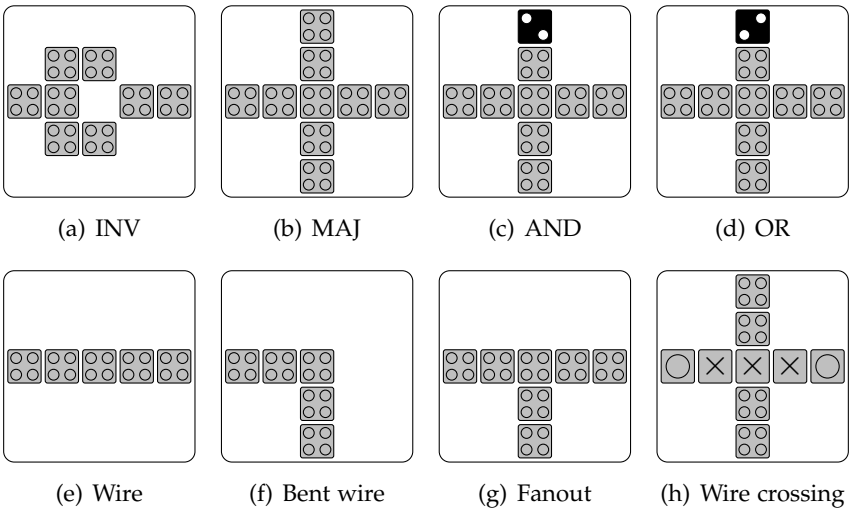


(a) Coplanar wire crossing (b) Multi-layer wire crossing

Figure 2.12: Different wire QCA wire crossing implementations

similar and since this work is more aimed to enhance the production of QCA circuits, the +-majority gate is chosen as standard gate for this work.
Following Definition 2.1.9 the AND gate can be derived by fixing one input of the majority gate to logic 0, while the OR gate is obtained by fixing one input to logic 1. The resulting gates, which are also part of the standard gate library of this work are shown in figure 2.11(c).

Another major topic regarding gates are wires. Until now only straight planar wires have been introduced. In order to distribute information on the 2D grid, provided by the tile based layout, also bent wires have to be introduced. They are depicted in Figure 2.13(f) and show a bend of 90 degrees. Given that all tiles can be rotated by 90°, 180° and 270° respectively a tile connected to a bent wire can be routed to each adjacent tile of a bent wire. Also since all gates introduced so far have a fan-out of one we need a fan-out node to multiply signals. This is done by adding a bent wire to a straight wire resulting in the fan-out shown in Figure 2.13(g).
The last special case of wires are crossing cases. By rotating the cells of one wire string by 45° the rotated cells don't have crosstalk with non-rotated cells [21] as shown in figure 2.12(a). This solution is very handy because it supports the planar structure of the circuit and is therefore called *coplanar crossover*. Further the possibility of multi-layer QCA has been investigated and found especially useful in the case of wire crossings. To use this, one wire string is raised to an additional higher layer, which is connected with a vertical interconnect as in figure 2.12(b). The signal transmission in the vertical stacked cells works just as in horizontal direction. To impede any crosstalk between the wire strings two intermediate layers of cells are used in vertical direction. Theoretically the added layer cannot only be used as wire but since the signal distribution works just as in the ground layer also gates can be placed in these multi-layers. Simulations



(a) INV (b) MAJ (c) AND (d) OR

(e) Wire (f) Bent wire (g) Fanout (h) Wire crossing

Figure 2.13: QCA Standard Library

have shown that coplanar crossovers reduce the coupling between the horizontal string segments significantly. This makes the horizontal interconnect very sensitive to crosstalk and therefore highly prone to cell displacements. Multi-layer circuits on the other hand show a high robustness and therefore are used as standard in this library. In the gate 5×5 representation of the multi-layer interconnect, the raised wire string is described with a $\times$, while the vertical layers are described with a circle. In Figure 2.13 all gates used in this work are summarized.

Latches and Registers

In order to implement sequential circuits, storage elements need to be implemented in QCA technology. Although they have to be rethought in QCA, the main characteristics can be derived from the well-known CMOS technology. Therefore, first the ideas of storage elements and their implementation is discussed. The simplest element, which can store one bit is an Eccles–Jordan flip-flop (FF). It is formed by connecting the output of one inverter to the input of another inverter and vice versa. Because this element is so simple it comes with some drawbacks. First of all the latch output is directly connected to the input causing noisy behavior, called transparency [10]. Further a high voltage shift is needed at the FF value and the FF is transparent in the transition region, requiring a stable voltage level during the transition. To tackle these Problems the inverters are replaced by NOR gates, reducing transparency between the input and output. Also this change introduces clear states. In the set state the latch output is set

to $Q = 1$ and in the reset phase it is set to $Q = 0$. Furthermore there exists a hold phase, where the current value is held (latched) and an undefined state. These states give the latch the name *Set-Reset-Latch* (SR-Latch). In order to use latches in synchronous circuits the Set and Reset inputs are clocked. This results in the gated SR-Latch. Because this latch still has an undefined state, which is not allowed the set and inverted reset inputs are connected together excluding the undefined state. Now the value is held when the clock $clk = 0$ and the D-latch is transparent, when $clk = 1$. Since transparency is still present a flip flop is formed by connecting two D-latches to a master-slave D-flip-flop producing only stable outputs.
The goal of a storage element in QCA is to have all the properties of a D-FF. In the QCA ONE library an effort was made to translate the D-FF into QCA by just replacing the CMOS gates and wires with the corresponding QCA gates. But since clocking plays an important role in the functionality of storage elements and the clocking has some major differences from QCA to CMOS as already observed in 2.2.2, the sense of purpose is questionable.
This means that storage elements should be completely rethought in the QCA domain. To create a QCA element which is able to store one bit the solution is rather simple. Since every tile is clocked on its own, a simple latch can be formed by a wire segment held in the hold phase of the clocking. In Figure !! such a wire segment is depicted. It is still adjacent to one ascending and one descending clock number (mod N_{clk}). The storage property is reached by extending the hold phase a the required number of clock cycles [26]. Because this implementation needs a different clock generator for every latch it is not sure if it is possible to implement. Though it can be shown that the same properties can be achieved by placing buffering wire segments instead of the introduced wire-latch. For every clock cycle the hold phase would be extended in the wire-FF, four adjacent wire segments are added delaying the information arrival by exactly the same time. Since wires have to be used to route the loops created by sequential circuits this idea is implemented for this work and explained more accurate in the next chapters.

2.3 P&R problem

As seen in the last sections of this chapter placement and routing are strongly related to the clocking inside the QCA domain. In this section the denotation and constraints of placement and routing in the QCA domain are introduced. The P&R problem evolves from a grid enabling tile based design in conjunction with a logic network.

Definition 2.3.1. A *layout* is defined by a $w \times h$ grid $\Gamma_{w,h}$ and a graph $G(V, E)$, which is placed on the grid. Every *tile* of the layout can be accessed via its x and y coordinates.

The set of tiles is denoted as T with $t = (x, y) \in T$. For any vertex of the graph $v(x, y)$ is restricted to the boundaries $x < w$ and $y < h$. For edges $\{(x, y), (x^*, y^*)\}$ it holds $|x - x^*| + |y - y^*| = 1, 0 \leq x, x^* \leq w, 0 \leq y, y^* \leq h$.

Definition 2.3.2. A *gate-level* layout describes a layout grid in combination with a logic network $N = (\Lambda, I, \Sigma, O)$. Besides the already known mapping *placement* p, which assigns nodes to tiles, there are two additional mappings. The *routing* r, which assigns logic network signals to layout paths (connected tiles) and a *clocking* c assigning clock numbers to tiles. The gate-level layout is therefore described as $L = (\Gamma, N, p, r, c)$.
Further, nodes placed on the gate-layout are referred to as *gates*. Two tiles $t_i = (x_i, y_i)$ and $t_j = (x_j, y_j)$ where $|x_i - x_j| + |y_i - y_j| = 1$ are called *adjacent*. A path, which is wired through adjacent tiles is called *wire*. In this context one tile corresponds to a *wire segment*. If neither a gate nor a wire segment is placed on a tile, it is empty. It follows that a layout with only empty tiles is also empty. A layout is said to be S-clocked if it follows a clocking scheme S. Otherwise it is irregularly clocked. Moreover an adjacent tile of a tile $t \in T$, where T is the set of all tiles, is incoming t^- if $c(t) - c(t^-)$ mod $clk = 1$. This means that the incoming tile is able to forward information to the viewed tile according to pipelined clocking. For outgoing tiles t^+ it holds $c(t^+) - c(t)$ mod $clk = 1$ accordingly. For QCA it was already stated that the clock number $clk = 4$.

From this definition we can outline the difficulty of placing and routing a logic network onto a two dimensional grid, with exception of wire crossings, which however are really costly and therefore should be minimized. One major challenge for P&R algorithms is the signal synchronization, which results the strong dependency of clocking and signal distribution. As already pointed out, for every signal path it has to hold true that information can only propagate from a tile with clocking number i to an outgoing tile with clocking number $(i + 1 \mod clk)$. This property is called the *local synchronization constraint*. The existence of possible signal paths can be assured by using predefined clocking schemes, but however can comprise some constraints. Further the *global synchronization constraint* states that every two signal paths leading to the same tile need to pass the same amount of tiles starting at their primary input. Since this constraint has to hold true for every gate the complexity increases rapidly with growing network sizes. Therefore the combination of all these challenges forms a P&R problem, which is commonly accepted to be $\mathcal{NP}$-hard [27].
After reaching a gate-level layout still a technology has to be mapped onto it. For this work the in subsection 2.2.3 proposed standard library is used for the mapping. Although the definitions in this work are held quite generic because they are based on the book [26], defining the P&R problem for the field-coupled nanotechnologies domain. This means that for example a change of clock to $clk = 3$ also allows a

placement and routing for *Nanomagnet Logic* (NML). Even though the algorithm in this work is designed only for QCA, the ideas may also be derived for other FCN technologies.

3 State of the Art

In this chapter various approaches trying to solve the placement and routing problem for QCA are reviewed. In the first part algorithms, which are able to work only with combinational circuits, are investigated under the theoretical groundwork done in chapter 2. Thereby both algorithms determining *optimal* and those who determine *scalable* solutions exist. In the second part ideas and challenges of sequential placement and routing algorithms are investigated.

3.1 Combinational P&R Algorithms

In section 2.3 the placement and routing problem was stated to be $\mathcal{NP}$-hard. This means, that on a deterministic Touring Machine there exists no algorithm that can determine a solution in polynomial time, but the proofs of solutions with an answer "yes" can be verified on it. This can be written as $\mathcal{P} \neq \mathcal{NP}$. In order to enable optimal solutions for the placement and routing algorithms so called *Satisfiability Solvers* are used. The general principle can be seen in figure !!. First an instance of a satisfiability problem is encoded from a problem instance in such a way that they are equisatisfiable. This means that if there exists a solution to the satisfiability problem instance there also exists a solution to the original problem instance. In the second step, the satisfiability instance is passed to a specialized solver returning an assignment to each variable in the encoding if such a solution exists or otherwise UNSAT. Lastly a solution to the original problem can be derived from the assignment [26]. It has to be mentioned that these solvers even though they are improving average case performance drastically, they are still bound to the complexity barrier of their underlying system, meaning $\mathcal{P} \neq \mathcal{NP}$ [1].
With this knowledge optimal placement and routing algorithms can be discussed. Two approaches from [26] are reviewed for this. The first algorithm "Exact Placement and Routing" finds a valid placement, routing and clocking, also described as (p, r, c), given an empty layout L and a logic network N. In order to find an optimal solution, the minimum layout size $w \times h$ has to be determined for which the constraints of (p, c, r) hold true. Therefore all possible sizes of layouts are encoded and passed to a satisfiability solver iteratively and the first layout for which the solver returns true is the minimum or rather optimal solution. The satisfiability solver utilized by this algorithm

is a *boolean satisfiability*-solver (SAT-solver), which is sufficient information for this work. The experimental results show that the determined layouts of the algorithm are many times smaller than the compared state of the art CITE. But due to the complexity of the algorithm utilizing satisfiability solvers, the algorithm times out for quite small circuits already, making it insufficient for the manufacturing of commercial QCA circuits.
In the book [26] another exact P&R algorithm is proposed. The idea is to create a *one-pass synthesis*, which combines logic synthesis and physical design in a single run. Therefore this algorithm has to tackle two $\mathcal{NP}$-hard problems relying again on the power of satisfiability solvers. This particular algorithm uses a *satisfiability modulo theories*-solver (SMT-solver). The idea is to eliminate some shortcomings of the two-step synthesis derived from CMOS. This includes treating wires as gates since the costs are equal in QCA and including data synchronization, which is dependent on the tiles passed. In this manner a SAT problem can be formed and passed to a SAT-solver. The instances are now created only passing a empty layout L of size $w \times h$. Even though this algorithm is able to find *truly minimal* solution since the non-optimal logic networks are eliminated, the experimental results show the same problems as in the exact P&R approach. This means that the high complexity of the satisfiability solver leads to a time-out of the algorithms for circuits with a gate size $|N| \geq 30$.
These results lead to the usage of scalable placement and routing algorithms. These approaches trade optimality of the circuit, like layout size for computing time. This makes them scalable in the time domain and therefore applicable for the manufacturing of commercial QCA circuits. All algorithms reviewed in the following are based on the original VLIS process, meaning they treat logic synthesis and physical design as their own problem.
Starting at the logic synthesis many works present a preprocessing of logic networks to make them suitable to translate them into gate level representations. There are several steps which are widely used to modify logic networks. The first of them is the node duplication or rather dummy node insertion. The idea of this process is to minimize wire-crossings, which we have analyzed to be very costly in QCA. Also the fanouts of the nodes are reduced, leading to a reduction of the place and rout complexity. One simple algorithm for this is to visit every node iteratively from the primary outputs to the primary inputs. If the current node hasn't been visited its marked as visited and if a already marked node is visited it is duplicated. But this means that not only one node is duplicated but also all the nodes included in te sub tree rooted by it [23]. From this simple example we can already suggest that the insertion of dummy nodes can lead to uncontrollable growth of the logic network. Other algorithms which are not that dedicated, don't have such a high overhead in dummy nodes but therefore can't eliminate all wire crossings making it necessary to include nodes for crossings, so called *crossing edge insertions* [6]. Another preprocessing steps including the insertion of

so called *buffer nodes* is the synchronization- Since the global timing constraint requires two paths leading to the same node to pass the same amount of tiles, this step makes sure that every two paths leading to a node include the same amount of gates. A buffer node therefore indicates that wires in QCA have to be viewed as gates as well. Also due to the insertion of buffers different partitions of a logic network can be generated [4]. Some approaches lead to even higher insertion of nodes indicating wasted area. This arises from the idea of a complete ternary logic network representation of a QCA circuit, since the gates are based on majority functions GRAPH . When extra area is produced due to the insertion of nodes, also wire lengths can be increased. For gates with two or three inputs, this means that if the longest wire has to be split into more than one clock zone, also the shortest wire has to be split into the same amount of clock zones in order to preserve the signal synchronization [23]. Another big problem of these algorithms is the requirement of cell-based clocking, which we have already showed to be insufficient. Even though there also exist algorithms using tile-based clocking they are limited by the general drawbacks of preprocessing [24] leading to exploding logic networks and even use greedy placement and routing algorithms limiting the approach to small and simple reconvergent patterns [8].

All these reasons lead to the proposal of *ortho*, an algorithm implementing a scalable placement, routing and clocking without preprocessing steps [28]. Since this algorithms forms the base of this work, the algorithm is explained detailed in the following.
First of all, a proper representation of the logic network is needed. Therefore in some works already the idea of an orthogonal embedding, have been proposed [4]. This means that the logic network is mapped onto a two-dimensional grid, so it can be seen as an assignment of the tuple (p, r). For ortho this is done by orthogonal graph drawing (OGD), which is described in [7].

Definition 3.1.1 (Orthogonal Graph Drawing)**.** An OGD maps a graph $G = (V, E)$ onto a plane grid with size $w \times h$. The mapping assigns vertices $v \in V$ with coordinates (x, y) to grid points, with $1 \leq x < w$ and $1 \leq y < h$. Edges $e \in E$ are assigned to paths in the grid, so they consist only from horizontal and vertical segments. The paths are non-overlapping, meaning that they are not allowed to cross any vertices.

Figure 3.1 shows an example OGD. The dots in the graph represent vertices and are connected via straight line paths. Therefore the graph is drawn orthogonal.

Nonetheless an OGD also only respects the placement and routing, leaving the clocking to be addressed. The problem of insufficient clocking out of an OGD representation can be showed from the example in [26]. Figure 3.2 shows, that for a given OGD (3.2(b)) there has to be no clocking which can resolve the timing constraints (3.2(c)). It stands out that for the down right corner no clocking zone can be found that is satisfies

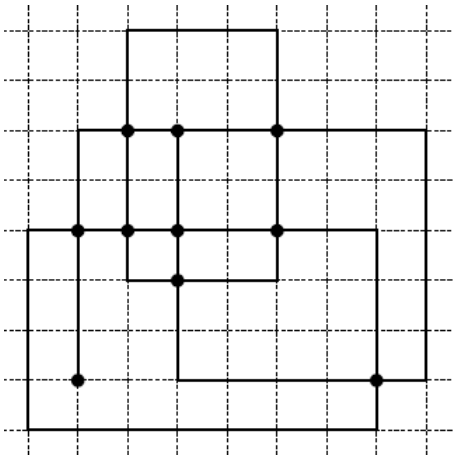
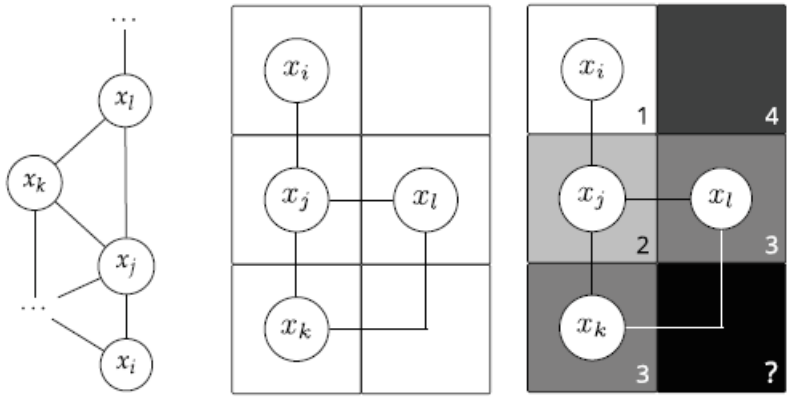


Figure 3.1: Example OGD drawing

the local synchronization constraint but also the global synchronization constraint is violated. Since the clocking or rather signal synchronization was a main task of the preprocessing now some other solution has to be found.

The idea used for the ortho algorithm comes from an extension to OGDs , which allows to determine a special OGD from a logic network in polynomial time being the constraint needed for a scalable approach. The base used here is formed by Therese Biedl [2], who proposes a OGD with an additional edge coloring. Although the effectiveness and complexity bounds in her work were proven on the restriction of *undirected 3-graphs* and as we already examined from the precious chapter a logic network is neither containing only nodes of most degree 3 nor undirected. To overcome the fist restriction, a custom logic network can be created by assigning own nodes for fanouts and inverters. This way the maximum node degree gets decreased to three, while the expressiveness of the logic network representation is maintained. The second restriction can be overcome by a custom coloring built on the original approach, which also serves as direction assignment. Given a logic network converted to a 3-graph, the coloring in form of edge directions $d : \Delta \rightarrow \{east, south\}$ is assigned. The coloring can be understood as relative position arrangement. If an edge (v_i, v_j) is colored *east*, means that the vertex v_j is positioned east of v_i, so that $x_j > x_i$. The color *south* for an edge (v_i, v_j) assigns v_j a relative position of v_i, so that v_j is south of v_i or $y_j > y_i$. In order to color a graph with only these two colors the following *assignment constraints* must hold true:

1. All **incoming** edge of a vertex has to be painted with the **same** color.
2. All **outgoing** edges of a vertex have to be painted with **opposite** colors.

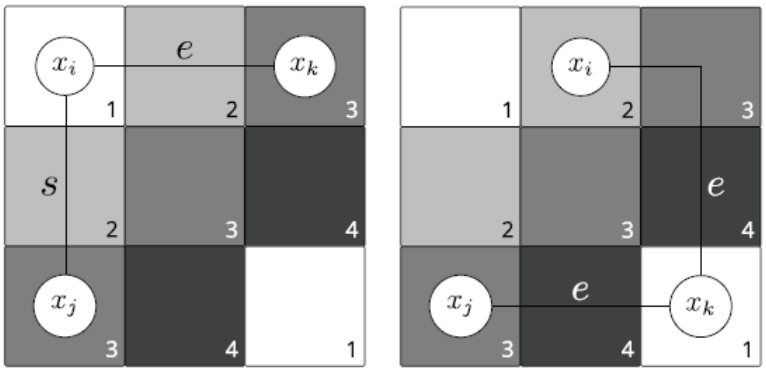


(a) Partial representation of a logic network (b) OGD representation of the logic network (c) No sufficient clocking for this OGD can be examined

Figure 3.2: Insufficient timing constraints of a OGD representation [26]

The relative position assignment under the proposed constraints can be seen at an arbitrary example in figure 3.3. In the example for outgoing edges (figure 3.3(a)), it is clear that the assignment constraint makes sure that two outgoing edges of the same vertex are routed without conflict. Due to the definition of the colors, the layout is increased in x-direction for an east-coloring and analogously extended in the y-Direction for a south-coloring. Figure 3.3(b) depicts the assignment constraint for the incoming edges of one vertex. Since this implies only one color, also the layout only is extended in one direction, here in x-direction due to the east-coloring. The clocking used in the example is 2DD-Wave, because it also supports only signal flow in the directions *east* and *south*. This means that the local synchronization constraint is always true if we use this scheme. But due to its uniformity the scheme also maintains the global synchronization constraint for vertices placed after the proposed direction assignment. This is why ortho utilizes the 2DD-Wave scheme. However, there exist logic networks for which no coloring holding the constraints can be found. Figure 3.4 shows such a coloring conflict. For the edge with a "?", no direction can be assigned for which the formulated constraints hold true. So an auxiliary node is introduced resolving the conflict.
The ortho algorithm can be described as follows.

Figure !! depicts the pseudo code for it. The example depicted in figure !! shows the (p, r, c) of a 2:1 mux.



(a) Color assignment of outgoing signals

(b) Color assignment of incoming signals

Figure 3.3: Relative positions of an OGD graph with correct color assinment [26]

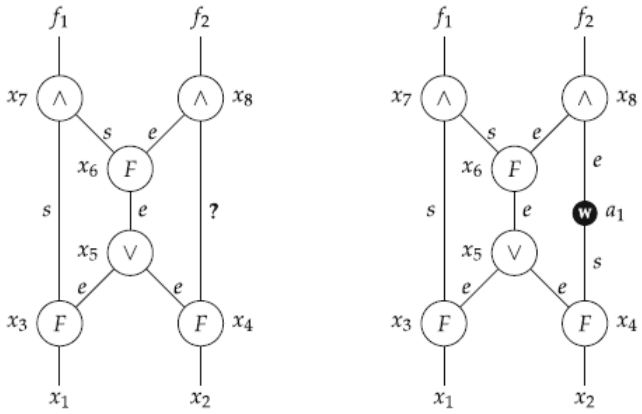


Figure 3.4: Solution for a coloring conflict

Algorithm 1 Ortho algorithm

Input: Logic network N
Input: Clock number clk
Output: Gate level layout L

Convert N to a 3-graph by substitution
$L \leftarrow$ empty 2DDWave-clocked layout of size $w = 0 \times (h = 0)$
Generate direction assignment $d : \Delta \rightarrow \{east, south\}$ and subdivide signals if necessary
Compute topological ordering $v_1, ..., v_i \in N$
Extend L by one column and reserve it for primary inputs
for each primary input $v_1, ..., v_l \in N$ **do**
    Extend L by one row
    Place v at position $(0, h-1)$
end for
for each primary input $v_1, ..., v_l \in N$ **do**
    Extend L by one column and one row
    Wire the primary input to position $(w-1, h-1)$
end for
for each vertex $v_l + 1, ..., v_i \in N$ with at most two incoming signals σ_1, σ_2 **do**
    if $d(\sigma_1) = d(\sigma_2) = east$ **then**
        Extend L by one column
        $h_p \leftarrow$ max. vertical position of v's predecessors
        Place v at position $(w-1, h_p)$
    else signals are labeled $south$
        Extend L by one row
        $w_p \leftarrow$ max. horizontal position of v's predecessors
        Place v at position $(w_p, y-1)$
    end if
    Extend L by one column and one row
    Wire the primary input to position $(w-1, h-1)$
end for
Draw orthogonal wire segments to connect v with its predecessor(s) accordingly
Connect the primary outputs to the respective borders **return** L

3.2 Sequential P&R Algorithms

Present the ideas in papers for QCA standard cell placement and routing.
Point out why they are not actionable: Reasons like clocking or cells aren't producible,

no automated algorithms

4 Methodology

4.1 Input Network

Reduces area and wire crossings.
Sorts the inputs.
Idea is simple place Fanout nodes at the beginning since they produce the most crossings
Needs a conditional east west coloring.

4.2 Majority Gates Placement Network

Reference to the importance of this part in the QCA technology.
Point out that 2DD-Wave is still used but clocking scheme is adjusted to place majority gates.
This is why buffers have to be used.
Point out the overhead that is produced by buffers.
Maybe assumption why it doesn't make sense to only use RES. (you lose really much space for the cells which are no majority gates)

4.3 Sequential Circuits Placement

Importance of sequential circuits.
Point out how the registers are implemented.
Show how the distribution network is generated, where Ris and Ros are placed and how they are treated within the network.
Make clear that this implementation is slowing down the circuit significantly.

5 Experimental Evaluation

5.1 Benchmarks

For combinational and sequential

5.2 Results

Everything

List of Figures

2.1 Binary Logic Network of Majority Function 5
2.2 QCA-Cell sates 8
2.3 Adjacent QCA-cells forming a wire segment 8
2.4 Schematic of a combined QCA and CMOS sytem [16]. 9
2.5 Schematic representation of a metastable state. Instead of relaxing correctly to the new ground state, a system may be delayed in an excited state due to an inability to tunnel through a kinetic barrier [15]. 10
2.6 QCA-Cell wire with corresponding clock zones 10
2.7 QCA clocking pipeline 11
2.8 Different clocking Schemes in QCA 12
2.9 Cell representation of tile 13
2.10 Different QCA Inverter representations 13
2.11 The QCA Majority gate 14
2.12 Different wire QCA wire crossing implementations 15
2.13 QCA Standard Library 16

3.1 Example OGD drawing 23
3.2 Insufficient timing constraints of a OGD representation [26] 24
3.3 Relative positions of an OGD graph with correct color assinment [26] . 25
3.4 Solution for a coloring conflict 25

List of Tables

Bibliography

[1] C. Bessiere. *Principles and Practice of Constraint Programming: CP 2007*. Springer, 2007.

[2] T. Biedl and G. Kant. "A better heuristic for orthogonal graph drawings." In: *Computational Geometry* 9.3 (1998), pp. 159–180.

[3] R. E. Bryant. "Graph-based algorithms for boolean function manipulation." In: *Computers, IEEE Transactions on* 100.8 (1986), pp. 677–691.

[4] M. Bubna, S. Roy, N. Shenoy, and S. Mazumdar. "A layout-aware physical design method for constructing feasible QCA circuits." In: *Proceedings of the 18th ACM Great Lakes symposium on VLSI*. 2008, pp. 243–248.

[5] C. A. T. Campos, A. L. Marciano, O. P. V. Neto, and F. S. Torres. "Use: a universal, scalable, and efficient clocking scheme for QCA." In: *IEEE Transactions on computer-aided design of integrated circuits and systems* 35.3 (2015), pp. 513–517.

[6] W.-J. Chung, B. Smith, and S. K. Lim. "Node duplication and routing algorithms for quantum-dot cellular automata circuits." In: *IEE Proceedings-Circuits, Devices and Systems* 153.5 (2006), pp. 497–505.

[7] M. Eiglsperger, S. P. Fekete, and G. W. Klau. "Orthogonal graph drawing." In: *Drawing Graphs*. Springer, 2001, pp. 121–171.

[8] G. Fontes, P. A. R. Silva, J. A. M. Nacif, O. P. V. Neto, and R. Ferreira. "Placement and routing by overlapping and merging qca gates." In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2018, pp. 1–5.

[9] M. Goswami, A. Mondal, M. H. Mahalat, B. Sen, and B. K. Sikdar. "An efficient clocking scheme for quantum-dot cellular automata." In: *International Journal of Electronics Letters* 8.1 (2020), pp. 83–96.

[10] C. Hawkins, J. Segura, and P. Zarkesh-Ha. *CMOS Digital Integrated Circuits: A First Course*. Materials, Circuits and Devices. Institution of Engineering and Technology, 2012. ISBN: 9781613530023.

[11] E. V. Huntington. "Boolean Algebra. A Correction." In: *Transactions of the American Mathematical Society* 35 (1933), p. 557.

[12] E. V. Huntington. "A New Set of Independent Postulates for the Algebra of Logic with Special Reference to Whitehead and Russell's Principia Mathematica*." In: *Proceedings of the National Academy of Sciences* 18.2 (1932), pp. 179–180. DOI: 10.1073/pnas.18.2.179. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.18.2.179.

[13] D. Kumar and D. Mitra. "A systematic approach towards fault-tolerant design of QCA circuits." In: *Analog Integrated Circuits and Signal Processing* 98 (Mar. 2019), pp. 1–15. DOI: 10.1007/s10470-018-1270-x.

[14] C. S. Lent, M. Liu, and Y. Lu. "Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling." In: *Nanotechnology* 17.16 (2006), p. 4240.

[15] C. S. Lent and P. D. Tougaw. "A device architecture for computing with quantum dots." In: *Proceedings of the IEEE* 85.4 (1997), pp. 541–557.

[16] C. S. Lent, P. D. Tougaw, and W. Porod. "Quantum cellular automata: the physics of computing with arrays of quantum dot molecules." In: *Proceedings Workshop on Physics and Computation. PhysComp'94*. IEEE. 1994, pp. 5–13.

[17] M. Mahdavi, M. A. Amiri, S. Mirzakuchaki, and M. N. Moghaddasi. "Single Electron Fault in QCA Inverter Gate." In: *2009 Fifth International Conference on MEMS NANO, and Smart Systems*. 2009, pp. 63–66. DOI: 10.1109/ICMENS.2009.23.

[18] M. Mohammadi, M. Mohammadi, and S. Gorgin. "An efficient design of full adder in quantum-dot cellular automata (QCA) technology." In: *Microelectronics journal* 50 (2016), pp. 35–43.

[19] D. A. Reis, C. A. T. Campos, T. R. B. S. Soares, O. P. V. Neto, and F. S. Torres. "A Methodology for Standard Cell Design for QCA." In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016, pp. 2114–2117. DOI: 10.1109/ISCAS.2016.7538997.

[20] T. N. Sasamal, A. K. Singh, and A. Mohan. "Quantum-Dot Cellular Automata Based Digital Logic Circuits: A Design Perspective." In: *Studies in Computational Intelligence*. 2020.

[21] G. Schulhof, K. Walus, and G. A. Jullien. "Simulation of Random Cell Displacements in QCA." In: *J. Emerg. Technol. Comput. Syst.* 3.1 (2007), 2–es. ISSN: 1550-4832. DOI: 10.1145/1229175.1229177.

[22] M. Taucer, F. Karim, K. Walus, and R. A. Wolkow. "Consequences of many-cell correlations in clocked quantum-dot cellular automata." In: *IEEE Transactions on Nanotechnology* 14.4 (2015), pp. 638–647.

[23] T. Teodósio and L. Sousa. "QCA-LG: A tool for the automatic layout generation of QCA combinational circuits." In: *Norchip 2007*. IEEE. 2007, pp. 1–5.

[24] A. Trindade, R. Ferreira, J. A. M. Nacif, D. Sales, and O. P. V. Neto. "A placement and routing algorithm for quantum-dot cellular automata." In: *2016 29th symposium on integrated circuits and systems design (SBCCI)*. IEEE. 2016, pp. 1–6.

[25] V. Vankamamidi, M. Ottavi, and F. Lombardi. "Two-dimensional schemes for clocking/timing of QCA circuits." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.1 (2007), pp. 34–44.

[26] M. Walter and R. Drechsler. "Design Automation for Field-Coupled Nanotechnologies." In: July 2020, pp. 176–181. DOI: 10.1109/ISVLSI49217.2020.00040.

[27] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler. "Placement and Routing for Tile-Based Field-Coupled Nanocomputing Circuits Is NP-Complete (Research Note)." In: *J. Emerg. Technol. Comput. Syst.* 15.3 (2019). ISSN: 1550-4832. DOI: 10.1145/3312661.

[28] M. Walter, R. Wille, F. S. Torres, D. Große, and R. Drechsler. "Scalable design for field-coupled nanocomputing circuits." In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 197–202.

[29] L. Zhang. "Solving QBF with combined conjunctive and disjunctive normal form." In: *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*. Vol. 21. 1. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 2006, p. 143.