Ignore Clocking Constraints: An Alternative Physical Design Methodology for Field-Coupled Nanotechnologies

Robert Wille[1,3] Marcel Walter[2] Frank Sill Torres[2,3] Daniel Große[2,3] Rolf Drechsler[2,3]

[1]Johannes Kepler University Linz, Austria
[2]Group of Computer Architecture, University of Bremen, Germany
[3]Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
robert.wille@jku.at, {m_walter, frasillt, grosse, drechsler}@uni-bremen.de

***Abstract*—*Field-Coupled Nanocomputing* (FCN) allows for conducting computations with a power consumption that is magnitudes below current CMOS technologies. Recent physical implementations confirmed these prospects and put pressure on the *Electronic Design Automation* (EDA) community to develop physical design methods comparable to those available for conventional circuits. While the major design task boils down to a place and route problem, certain characteristics of FCN circuits introduce further challenges in terms of dedicated clock arrangements which lead to rather cumbersome clocking constraints. Thus far, those constraints have been addressed in a rather unsatisfactory fashion only.**

In this work, we propose a physical design methodology which tackles this problem by simply ignoring the clocking constraints and using adjusted conventional place and route algorithms. In order to deal with the resulting ramifications, a dedicated *synchronization element* is introduced. Results extracted from a physics simulator confirm the feasibility of the approach. A proof of concept implementation illustrates that ignoring clocking constraints indeed allows for a promising alternative direction for FCN design that overcomes the obstacles preventing the development of efficient solutions thus far.

I. Introduction

Field-Coupled Nanocomputing (FCN) [1] is a class of technologies that, in contrast to conventional technologies, conducts computations without any electric current flow. Consequently, FCN allows for operations with a remarkable low energy dissipation that is a number of magnitudes below current CMOS technologies [2], [3]. Considering that energy consumption is a constant threat for the advancement of integrated circuit technologies, FCN accordingly positioned itself as a promising alternative to these conventional technologies. This is confirmed by the fact that, just recently (i. e. in the recent 3-4 years), several suitable contributions to the physical realization of FCN technology have been presented, e. g. *molecular Quantum-dot Cellular Automata* (mQCA) [4], *atomic Quantum-dot Cellular Automata* (aQCA) [5], [6], or *Nanomagnet Logic* (NML) [7].

These technological advancements put pressure on the *Electronic Design Automation* (EDA) community and motivate the development of design automation for FCN technologies. However, certain characteristics of FCN make the development of automatic design methods for FCN circuits a highly *non-trivial* task. More precisely, the basic element in FCN are nanoscale cells that represent information via its polarity or magnetization. These cells interact amongst each other via local fields which enables information transfer as well as processing [8], [9]. In order to comply with metastability issues, FCN technologies apply external electric or magnetic clocks that control the ability of a cell to change its state. Therefore, cells are organized in groups, also known as *tiles*, that are controlled by different external clocks. These groups must be arranged such that the relation of the clocks is in accordance with the required data flow – yielding very cumbersome *clocking constraints*.

These clocking constraints eventually pose significant challenges to the physical design of FCN circuits because they e. g.

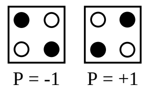
- overcomplicate the mapping of a certain functionality (i. e. the placement and routing of gates and corresponding connections) into a proper FCN description since gate connections have to be properly timed (see e. g. [10], [11]) or
- make sequential circuits very expensive since either corresponding latch structures are costly (see e. g. [12]) or are represented as wires which may lead to complex synchronization tasks to be addressed (see e. g. [13]).

Overall, while the major design task boils down to a typical place and route problem, the clocking constraints make the physical design of FCN circuits a rather cumbersome task. Despite significant efforts on the development of corresponding physical design methods for FCN (as shown e. g. in [10], [11], [13], [14]), no really satisfying solution is available yet (all this is discussed in more detail later in Section III).

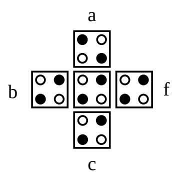
In this work, we propose an alternative physical design methodology for FCN circuits which avoids these problems. To this end, we make a rather bold yet effective proposal: Simply ignore the clocking constraints in FCN! This immediately solves all the problems mentioned above and suddenly allows for the straight-forward utilization of (adjusted) place and route algorithms available for conventional circuitry – enabling for the exploitation of more than 30 years of experience. On the downside, simply ignoring such essential constraints of the FCN technology will likely lead to broken realizations. Hence, we counter the ramifications of ignoring the clocking constraints by the introduction of a dedicated *synchronization element* which adapts the way how external clocks are used in FCN technologies.

Results extracted from a physics simulator confirm the feasibility of the approach. Furthermore, we provide proof of concepts that show how the new paradigm together with the synchronization element allows to utilize established and sophisticated EDA methods for the design of FCN circuits. Overall, this overcomes the obstacles that prevented the development of efficient solutions for FCN design thus far.

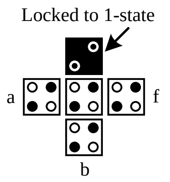
The remainder of this work is as follows: Section II reviews fundamental aspects of FCN and the clocking constraints, while the following Section III discusses the restrictions of current FCN design approaches. Section IV introduces the new synchronization element and Section V presents its exemplary integration into an automatic design algorithm (serving as proof of concept of the proposed methodology). Finally, Section VI concludes this work.



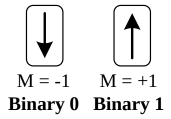
(a) States in QCA



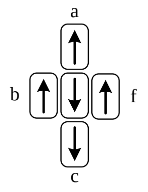
(b) QCA Majority



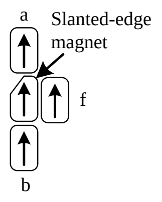
(c) QCA OR



(d) States in NML

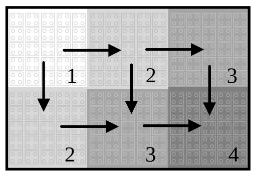


(e) NML Majority

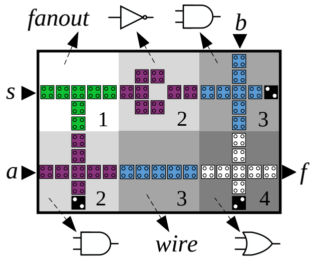


(f) NML OR

Fig. 1: FCN realizations of basic operations



(a) Grid of clock zones with arrows indicating allowed data flows and possible QCA cell positions



(b) QCA circuit implementing the function $f(a, b, s) = a \cdot s + b \cdot \bar{s}$

Fig. 2: QCA circuit design

II. FCN Circuits, Clocking Constraints, and the Resulting Design Method

Field-coupled Nanocomputing (FCN) utilizes cells that interact via local fields and, thus, allows for information transfer and the realization of logic functions [1]. Common incarnations of this technologies are *Quantum-dot Cellular Automata* (QCA) in terms of *molecular QCA* [4] or *atomic QCA* [5], [6] as well as *Nanomagnet Logic* (NML) [7], [15].

In the former case of QCA, these cells are based on molecules [4] or dangling bonds [5] and composed of four *quantum dots* which are able to confine an electric charge and are arranged at the corners of a square [16], [8]. Adding two free and mobile electrons into each cell that are able to tunnel between adjacent dots yields a stable state due to interaction (tunneling to the outside of the cell is prevented by a potential barrier). Then, because of the mutual repulsion, the two electrons tend to locate themselves at opposite corners of the cell – eventually leading to two possible *cell polarizations* (namely $P = -1$ and $P = +1$ which can be defined as binary 0 and binary 1, respectively). In contrast, NML cells utilize nanomagnets [7]. Here, an NML *cell* is a single domain nanomagnet that can assume only the two stable magnetization states $M = -1$ and $M = +1$, which also can represent the binary values 0 and 1 [1].

Example 1. *Fig. 1a shows two QCA cells with their four quantum dots (denoted by circles) and two electrons (illustrated by black dots). Due to the electrostatic Coulomb interaction, those are the only stable states which the QCA cell can assume. Usually, the state shown in the left-hand side of Fig. 1a is defined as binary 0, while the state shown in the right-hand side of Fig. 1a is defined as binary 1. Similarly, Fig. 1d depicts two NML cells where the arrow indicates the current magnetization of each cell. Commonly, the magnetization perpendicular to the* down *direction is defined as binary 0, while the magnetization to the opposite direction is defined as binary 1.*

When composing several FCN cells next to each other, field interactions cause the polarization or magnetization of one cell to influence the polarization or magnetization of the others. This allows to realize Boolean functions such as AND, OR, NOT, Majority, etc.

Example 2. *Fig. 1b shows the QCA realization of the Majority function, where e. g. a binary 0 from input a competes with two binary 1s coming from inputs b and c. The output follows the majority of the input values, which is a binary 1 in this case. Locking one of the three inputs to the 0-state turns this cell into an AND gate, while locking one of the inputs to the 1-state results into an OR gate as shown in Fig. 1c. In a similar fashion, the structure in Fig. 1e depicts the NML implementation of the Majority function where the inputs a, b and c compete with each other. AND and OR gates can be constructed using so-called* slanted-edge magnets *[15] that give preference to a magnetization, as shown in Fig. 1f.*

Due to the issue of metastability and in order to control the data flow within a design, FCN circuits must apply external clocking fields. In case of QCA, the external electric clock controls the tunneling within a cell, while in NML the external magnetic clock controls the switching ability of the nanomagnets. Depending on the technology, each cell changes during a complete clock cycle between four (QCA) or three (NML) different phases, i. e. a *switch*, a *hold*, a *reset* and an *neutral* phase (the latter only in case of QCA). For the sake of simplicity and without loss of generality, we will consider a four-phase technology in the following.

In case of four phases, usually four external clocks numbered from 1 to 4 are applied, whereby each clock controls a selected set of cells. For fabrication purposes, cells are usually grouped in a grid of square-shaped *tiles* such that all cells within a tile are controlled by the same external clock [17], [18][1]. Each tile might contain cells realizing a logic function (e. g. as depicted in Fig. 1), denoted as FCN *gates* or routing elements, like wires, fanouts or wire-crossings [3]. It is important to note that FCN cells can only process new data when its controlling clock is in the *switch* phase. Furthermore, these data must come from FCN cells that are in a stable state, i. e. whose related clock is in the *hold* phase. That means, a correct data flow is only possible between tiles controlled by clocks with a phase difference of one, i. e. that are consecutively numbered. In other words, tiles controlled by clock 1 can only pass its data to tiles controlled by clock 2 etc. and, finally, from clock 4 to clock 1. In the following, we call these kinds of demand *clocking constraint*, i. e. the proper propagation of data among all tiles. Consequently, the data flow between tiles is conducted in a pipeline-like fashion controlled by the four external clocks.

Example 3. *Fig. 2a shows possible groupings of QCA cells into tiles. Each square shape represents a tile, in which QCA cells realizing a gate may be placed (possible cell positions are hinted in each tile). Each tile is related to one of the four external clocks (denoted by the numbers in the bottom-right corners and a corresponding grayscale) which control all QCA cells within the respective clock zone. The arrows indicate possible data flow directions. For example, a logic gate located in the upper-middle tile which is controlled by clock 2, might connect its outputs to a gate or a wire located in the neighboring tiles that are controlled by clock 3, while the input data must come from a gate or wire located in the upper-left tile that is controlled by clock 1.*

[1]The independent clocking of single cells should be avoided due to limited control of the magnetic and electric fields [19], [20].

Because of these clocking constraints, a design methodology for FCN circuits which basically involves the following steps is usually applied: In a first step, the function to be realized is decomposed into Boolean sub-functions for which corresponding FCN gate realizations are available (using methods for conventional logic or majority logic synthesis such as [21] and [22], [23]). Next, the resulting description is mapped to the corresponding FCN realizations by using a given technology-depended gate library [3], [12]. Afterwards, the gates have to be placed on a grid of tiles (e. g. as depicted in Fig. 2a) and the corresponding routing has to be executed [24]. The latter step requires that the routing elements are added in a fashion so that the clocking constraints are satisfied. That means, all inputs of a gate or routing element must connect to structures located in a preceding tile, while all outputs must connect to elements in subsequent tiles.

Example 4. *Fig. 2b depicts the QCA implementation of the multiplexer function with three variables $f(a, b, s) = a \cdot s + b \cdot \bar{s}$. This function has been decomposed into logic gates, i. e. AND, Inverter and OR gates, which then have been placed in the grid of tiles. Next, wires and fanouts have been added, such that the clocking constraints are satisfied. For example, the inverter in the upper-middle tile receives its input data from the left-hand fanout controlled by clock 1, and it sends its data to the right-hand AND that is controlled by clock 3.*

The placement and routing problem under all given constraints for FCN technologies has been proven to be a computational hard problem even under certain simplifications [25].

III. Motivation and Proposed Idea

While the design methodology reviewed in the previous section constitutes the established approach, it comes with some severe challenges which limit the potential of today's design automation for FCN circuits. Those challenges, which are to a great extend caused by the clocking constraints, provide the motivation of this work and are discussed in the first part of this section. Afterwards, we sketch how to overcome these challenges by an alternative design methodology which eventually breaks with the clocking constraints and, by ignoring them, provides a promising alternative direction for FCN design.

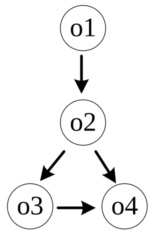
A. Motivation: Always Trouble with the Clocking Constraints

The physical design of FCN circuits basically boils down to a place and route problem. However, although some efforts have been spent on this (as shown e. g. in [10], [11], [13], [14]), no really satisfying solution is available yet. This is mainly caused by the need to satisfy the clocking constraints reviewed in the previous section. In fact, the clocking constraints pose a challenge e. g. for the actual physical design (the basis of every layout methodology) and for the synchronization in sequential circuits (essential for practically relevant applications).

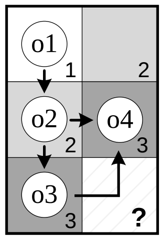
1) Overcomplicating the Mapping: As sketched in Section II, when mapping a certain functionality (i. e. the placement and routing of gates and corresponding connections) into a proper FCN layout, it must be assured that data only passes between tiles controlled by consecutively numbered clocks. Furthermore, FCN is a planar technology, i. e. logic and routing elements are located in the same layer [8].[2] All this strongly poses a severe challenge to design methods for FCN as illustrated in following example.

Example 5. *Fig. 3a depicts a graph of a simple netlist with the 4 operations o1 to o4. These operations have been placed on a grid of tiles in an attempt to satisfy the clocking constraints (see Fig. 3b). However, as one can see, this placement and routing leads to a conflict as there is no simple way to connect the output of operation o3 to o4. A possible solution is depicted*

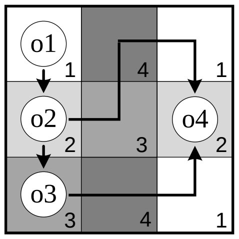
[2]Note that an exception are wire-crossings for QCA that can utilize a second layer for short sections [26]. However, this does not change the main premise that planar routing is needed.



(a) Netlist graph with 4 operations o1 to o4

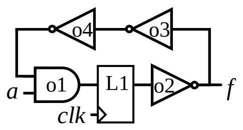


(b) Layout design and conflict

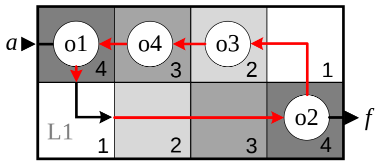


(c) Conflict free layout

Fig. 3: Placement and routing leading to conflict



(a) Schematic representation



(b) Layout design leading to a synchronization conflict

Fig. 4: Synchronization conflict in sequential circuit (red lines indicate the path that should have a maximum length of 4 tiles)

in Fig. 3c. Despite the fact that determining such a solution is a complex task for which no scalable solutions has been found yet (see [10]), it also frequently yields rather complex designs including e. g. wire elements that need to realize several "detours" until they eventually reach the desired gate in the correct clock cycle.[3]

2) Synchronization in Sequential Circuits: As extensively discussed in [13], a common solution for the design of sequential circuits is to avoid any sequential elements like latches. Instead, one should exploit the inherent pipeline-like behavior of FCN and assure that all signals are synchronous. Nevertheless, several latch structures have been proposed, e. g. in [12]. However, these gates have considerable area costs, easily more than 5 times of a basic gate, and questions regarding control signal distribution remain unanswered. Hence, sequential elements are commonly represented as wires, and it must be assured that only data is processed together which belong to the same cycle. However, this can become critical for circuits containing feedback paths, as new input data as well as data coming from the feedback paths must be synchronous. Again, an example illustrates the problem.

Example 6. *Fig. 4a shows a schematic representation of a sequential circuit that contains the operations o1 to o4, a latch element $L1$ which is triggered by the control signal clk, and a feedback path. A minimal implementation is depicted in Fig. 4b where all elements have been placed on the grid such that the data passes only between tiles controlled by consecutively numbered external clocks. The latch element $L1$ is represented as wire. That means, is has to be assured that data coming from input a and the output signal of $L1$, that is fed back to $L1$, are synchronous. This design has a critical path length of 7 tiles (indicated by the red line in Fig. 4b). Consequently, the data fed back from $L1$ will only arrive after two cycles. This follows from the observation that a signal can travel four tiles in one clock cycle, during which each tile enters*

[3]Note that long wire elements indeed are a problem since, as investigated in [27], wire elements cause significant costs with considerable higher impact as in conventional circuitry.

once in switch *mode (see also Section II). Thus, it cannot be assured that the feedback signal from* $L1$ *and the input signal from* a *are processed synchronously. Possible solutions are the addition of a latch cell, accompanied by high area costs and the challenge of control signal distribution, or the costly insertion of delaying buffers for the input signal from* a*.*

Overall, the clocking constraints currently limit the physical design and synchronization in sequential circuits. By this, it prevents the utilization of well-known and established place and route methods (see e. g. [28], [29] for an overview). This explains why the design of FCN circuits still is mainly conducted manually (as done e. g. in [18]) and while all approaches for automatic design of FCN circuits are, thus far, still far away from the efficiency we take for granted in the design of conventional circuitry (see e. g. [14], [10] covering EDA methods for FCN circuits that are severely limited to rather small circuits).

B. Proposed Idea: Ignoring the Clocking Constraints

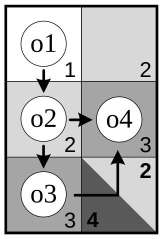
Based on the discussions from above, we came to the conclusion that, in order to tackle the limits of FCN circuit design, no iterative improvement of the state-of-the-art is sufficient as they do not address the core of the problem: satisfying the clocking constraints causes challenges which are hard to overcome. Consequently, we propose an alternative methodology which breaks with the current approach and simply ignores the clocking constraints – at least at the beginning. In fact, if the clocking constraints are ignored, both problems discussed above disappear:

- Physical design can employ a direct routing and connect all operations without the need for any "detours" and
- Sequential circuits can easily be synchronized as it is commonly done in conventional circuitry.

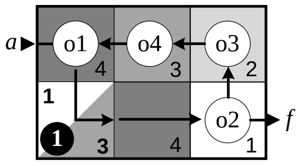
However, of course simply ignoring these essential constraints of the FCN technology will likely lead to FCN realizations that will not work properly. After all, the clocking constraints for FCN circuits do not simply disappear when we choose to ignore it. Though, we can fix the resulting problems afterwards by introducing dedicated *synchronization elements*. Those are FCN elements which are supposed to be added to the grid and which can stall a signal for a given number of clock cycles. By this, the corresponding synchronization is achieved through these dedicated elements rather than through costly and complex to design circuit/wire structures. Two examples illustrate the idea:

Example 7. *Consider again the problem discussed before in Example 5 and the placement as shown in Fig. 3b. In order to avoid the synchronization problem (how to connect* $o3$ *with* $o4$ *while, at the same time, satisfying the clocking constraint), a synchronization element is placed in the right-bottom corner of the grid (denoted by a two-colored tile in Fig. 5a). The function of this synchronization element is two-fold: it has to be able to process data coming from a tile controlled by clock 3, and it must forward data to a tile also controlled by clock 3. That means, the tile must work as controlled by clock 4, in order to receive data from a tile controlled by clock 3. Further, when passing data, it must act as a tile controlled by clock 2, in order to pass data to clock 3. More precisely, the clock of this new element must be configured such that it enters in the* switch *phase together with clock 4, and it must enter in the* hold *phase together with clock 2 (indicated by respective numbers on the "input-side" and "output-side" of the two-colored tile in Fig. 5a).*

Example 8. *In similar fashion, the problem of the sequential circuit considered in Example 6 and shown in Fig. 4 can be resolved with the proposed synchronization elements. A corresponding solution is shown in Fig. 5b. Here, the synchronization element (again denoted by a two-colored tile) must enable the synchronization between two tiles controlled by clock 4. That means, it must receive data as tiles controlled by clock 1, and pass data as tiles controlled by clock 3. Furthermore, the synchronization element must stall the signals for one additional clock cycle (additionally indicated by the circled* 1 *in Fig. 5b), such that data coming from input* a *and feedback data from* $L1$ *are synchronous.*



(a) Layout of netlist depicted in Fig. 3a using *synchronization element*



(b) Layout of sequential circuit with design *synchronization element*

Fig. 5: Application of *synchronization elements* (represented as two-colored tiles and the respective clock characteristics) for the design problems presented in Fig. 3 and Fig. 4.

However, this idea obviously only works if a proper realization of such a synchronization element is possible in FCN. Hence, the final contribution of this work is the provision of such a design which is described in the next section. While this eventually will yield some additional costs, it provides a suitable alternative to the state-of-the-art design methodology which, thus far, is stuck to be applicable for rather small circuits and hardly relevant circuits only. Afterwards, a proof of concept summarized in Section V demonstrates the applicability of the proposed alternative methodology.

IV. FCN Realization of the Synchronization Element

The alternative methodology based on ignoring the clocking constraints during the design of FCN circuits only works, if an FCN realization of the synchronization element is available. This section introduces such a realization, confirms why this realization indeed works, and briefly discusses its costs.

A. Synchronization Element

As discussed in the previous section, the synchronization element must be able to transfer data between tiles controlled by (electric or magnetic) clocks that are not consecutively numbered. Therefore, we propose the application of some additional external clocks with an asymmetric shape. While similar clocks have already been discussed before in [30], they have not been exploited yet to eventually resolve the clocking constraint challenges discussed here. More precisely, in contrast to standard clocks, these additional clocks have a *hold* phase that is longer than the remaining phases of the clock signal. During this extended *hold* phase, information stored in a tile controlled by the respective clock is stable and can be passed to neighboring tiles [3], [9]. However, neighboring tiles can only process this data, if they are in the *switch* phase. The principal idea is to apply clocks that enable the synchronization of the *hold* phase of the synchronization element and the *switch* phase of the following tile [4]. Consequently, the synchronization element can be followed by a tile controlled by any clock. One can even extend the *hold* phase of the synchronization element such that data is stalled by more than one clock phase, e. g. as required in Example 6.

However, the extended *hold* phase delays the time until the synchronization element can receive new data from its predecessor. Consequently, data sent from the predecessor during the extended *hold* phase are not processed and have to be sent again. As shown in [31], [32], this requires that the frequency of the input data is halved – leading to a design throughput that is reduced by factor 2.

[4]Usually, this synchronization is given if the clocking constraint is satisfied.

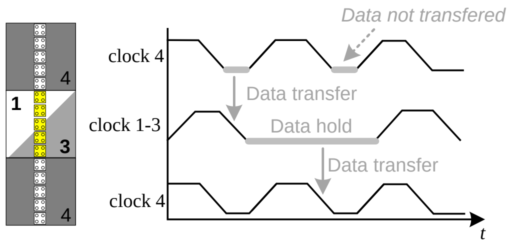


Fig. 6: Synchronization Element applied in QCA circuit enabling data transfer between two tiles controlled by clock 4

Example 9. *Fig. 6 depicts a simple QCA circuit with two tiles controlled by clock 4 and an intermediate tile which is the synchronization element. The latter is controlled by clock 1-3 which enables the synchronization between tiles controlled by clock 4. The depicted curves demonstrate how data are transfered and hold, depending on the state of the respective clock signal. More precisely, during a falling slope, the cells within a tile are in the* switch *phase, i. e. new input data can be processed. In case of a low clock signal, the cells are in the* hold *phase, i. e. the data are stable and can be passed to neighboring tiles controlled by clocks in the* switch *phase.*

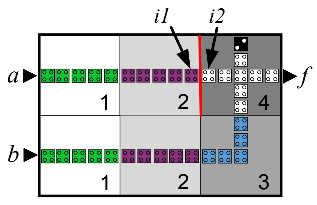
One can note that the clock of the synchronization element enters the switch *phase, when clock 4 is in the* hold *phase. This behavior is the same for clock 1 which also is in the* switch *phase when clock 4 remains in the* hold *phase. Next, the clock 1-3 of the synchronization element has an extended* hold *phase such that it is still in this phase when clock 4 enters the* switch *phase. Thus, clock 1-3 passes data to clock 4 in the same way as clock 3 which also remains in the* hold *phase when clock 4 is in the* switch *phase. However, the synchronization element can only receive new data every second clock cycle due to the longer* hold *phase of clock 1-3.*

B. Feasibility Analysis

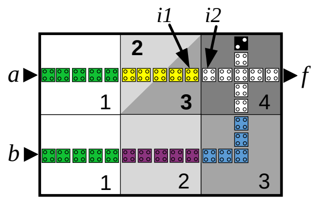
In order to evaluate the feasibility of such additional clocks, we integrated the ability to represent asymmetric clocks into the physics simulator QCADesigner-E[5]. The tool is based on the widely applied QCADesigner which enables the detailed simulation of the behavior of QCA on the quantum level [34]. Next, we implemented two versions of a simple circuit consisting of wires and an OR. The first version misses the clocking constraints by using neighboring tiles that are controlled by clocks that are not consecutive, which is indicated by a red line in Fig. 7a. The second version, depicted in Fig. 7b uses a synchronization element that enables the synchronization between tiles controlled by clock 1 and clock 4 without adding any additional hold time. The simulation results are shown in Figs. 7c and 7d, respectively. Here, *Cx* denotes clock x, *i1* and *i2* are the polarization values of the cells indicated in each circuit, and a, b as well as f are the polarization values of the cells connected with the inputs and outputs of the circuits.

In case of the circuit that fails the clocking constraints, the wrong output is determined for the input combination $a = 0$ and $b = 0$ (in fact, $P = +1$ is determined, i. e. binary 1). The reason for this error is the wrong data transfer between the QCA cells $i1$ and $i2$, provoked by the missing clock constraints (indicated by the red circle in Fig. 7c). In contrast, the circuit using the synchronization element determines the correct output value for all input combinations of a and b – assuming that only every second result is obtained. The curves in Fig. 7c reveals how cell $i1$, which belongs to the synchronization element, holds its polarization long enough such that cell $i2$ can read the correct value (indicated by the red circle in Fig. 7d).

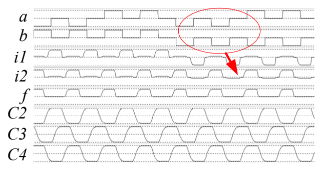
[5]The tool is freely available at [33].



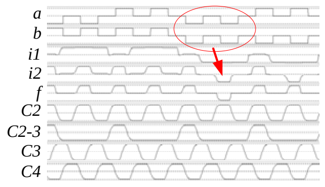
(a) Failed clocking (red line)



(b) Application of sync. element



(c) Waves for failed clocking



(d) Waves with sync. element

Fig. 7: Simulations with extended physic simulator of circuits with and without synchronization element. Red arrows indicate where data are propagated erroneous (c) and correctly (d).

V. Proof of Concept

Ignoring the clocking constraints and addressing the resulting ramifications with synchronization elements as introduced in the previous sections, indeed allows to overcome existing problems of FCN design and opens the path for utilizing conventional and established EDA methods. In order to demonstrate this, we conducted a proof of concept evaluation for which we implemented an initial design flow for FCN circuits following the newly proposed methodology on top of the *fiction* framework [35]. This section summarizes the main steps within this new design flow as well as the obtained results.

Based on the premise of the propose idea, the implemented design methodology now heavily relies on established and highly-optimized EDA methods for conventional circuitry. First, *ABC* [36] is used to synthesize corresponding gate level descriptions out of a given functional description (including both, combinational as well as sequential functions). More precisely, ABC converts the functional representation into an AND-Inverter-Graph out of which a gate netlist composed of 2-input AND gates, 2-input OR gates, NOT gates, as well as flip-flops is retrieved afterwards.

Then, those gates can directly be mapped to the corresponding FCN structures such as the ones shown in Fig. 1 or as presented in [3]. Since no clocking constraint need to be considered anymore, the placement and routing of the respectively resulting FCN structures can now be conducted using conventional P&R methods (see e. g. [28], [29] for an overview). To this end, only the respective sizes of the FCN structure plus the size of a placeholder for a possible synchronization element have to be additionally defined. In our proof of concept implementation, we utilized simulated annealing [37] to determine a placement and a customized version of the A*-algorithm [38] to determine the routing (using the Manhattan distance as a cost function in both cases). Finally, synchronization elements are instantiated as needed in order to resolve all violations of the clocking constraints that might have been caused by the conducted P&R.

The resulting flow is capable of completely realizing combinational as well as sequential circuits without the need to explicitly address the challenges discussed in Section III. This has been confirmed by applying standard benchmarks such as *ISCAS85* [39] and *EPFL* [40] for combinational functionality as well as *PoliTo ITC99* [41] for sequential circuitry to the resulting implementation. Table I summarizes some of the obtained results. The first four columns provide details of the applied benchmarks, i. e. their name, number

TABLE I: Obtained results

Benchmark [39], [40], [41]				Proposed results		
Name	#Gates	I / O	#FF	Dimension	#SE	t in s
c432	542	36 / 7	—	96 × 91	867	27
c499	1125	41 / 32	—	169 × 167	2909	160
c880	816	60 / 26	—	116 × 116	1861	79
c1355	1381	41 / 32	—	190 × 183	3637	277
c1908	1111	33 / 25	—	169 × 164	2949	161
ctrl	521	7 / 26	—	92 × 92	935	26
router	655	60 / 30	—	103 × 104	928	45
int2float	706	11 / 7	—	109 × 107	1624	56
b04	1526	12 / 8	66	200 × 194	5474	294
b07	1024	2 / 8	49	159 × 163	3033	119
b08	430	10 / 4	21	84 × 83	758	17
b10	489	12 / 6	17	93 × 88	985	22
b13	731	11 / 10	53	112 × 108	1495	54

of gates, as well as primary inputs and outputs (in case of sequential functionality, the number of flip-flops is provided as well). The column *Dimension* shows the size in x- and y-dimension of the resulting layout in tiles. In the column *#SE*, the number of inserted synchronization elements is listed (note that this also includes one synchronization element per flip-flop in case of sequential circuits). Finally, column *t in s* provides the overall runtime in CPU seconds required for the complete flow, i. e. from the initially given functionality to the final layout. All experiments have been conducted on an Intel Xeon E5-2630 v3 machine with 2.40 GHz (up to 3.20 GHz boost) and 64 GB of main memory running Fedora 22.

The results confirm that, following the proposed methodology of simply ignoring the clocking constraints, indeed allows to utilize established and sophisticated EDA methods for FCN design. This eventually unleashes significant potential: While most of the related work heavily relies on manual labor or rather limited automatic methods (c.f. the discussions in Section III-A), the proposed design methodology can already in a simple implementation as presented here realize functions composed of dozens of inputs and hundreds of gates. As for sequential circuits, we were able to present first automatically generated results at all. While surely more dedicated and optimized implementations of the proposed methodology are possible (a more detailed treatment of this is left for future work), this already shows the promising potential of the proposed approach.

VI. Conclusions

In this work, we proposed to ignore the clocking constraints – a fundamental concept which usually has to be considered when realizing FCN circuits but which also prevented the utilization of established and sophisticated place and route methods thus far. To deal with the ramifications of ignoring these constraints, we proposed to use *synchronization elements* which are easy to implement since they constitute a single-tile wire structure only which is additionally triggered by external clocks. The feasibility of the synchronization element has been confirmed using a physics simulator and a proof of concept implementation showed that ignoring the clocking constraints indeed resolves the obstacles we currently see in FCN design. By this, the proposed methodology provides a promising new direction for an alternative FCN design methodology which certainly will trigger corresponding future work building on that.

References

[1] N. G. Anderson and S. Bhanja, *Field-coupled Nanocomputing: Paradigms, Progress, and Perspectives*, 1st ed. New York: Springer, 2014.

[2] J. Timler and C. S. Lent, "Power Gain and Dissipation in Quantum-dot Cellular Automata," *J. Appl. Phys.*, vol. 91, no. 2, pp. 823–831, 2002.

[3] F. S. Torres, R. Wille, P. Niemann, and R. Drechsler, "An Energy-aware Model for the Logic Synthesis of Quantum-Dot Cellular Automata," *TCAD*, vol. PP, no. 99, 2018.

[4] C. S. Lent *et al.*, "Molecular Cellular Networks: A non von Neumann architecture for molecular electronics," 2016, pp. 1–7.

[5] S. Bohloul, Q. Shi, R. A. Wolkow, and H. Guo, "Quantum Transport in Gated Dangling-Bond Atomic Wires," *Nano Letters*, vol. 17, no. 1, pp. 322–327, 2017.

[6] T. R. Huff, H. Labidi *et al.*, "Atomic White-Out: Enabling Atomic Circuitry through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface," *ACS Nano*, vol. 11, no. 9, pp. 8636–8642, 2017.

[7] X. K. Hu *et al.*, "Edge-Mode Resonance-Assisted Switching of Nanomagnet Logic Elements," *IEEE Trans. Magn.*, vol. 51, no. 11, pp. 1–4, 2015.

[8] C. S. Lent and P. D. Tougaw, "A Device Architecture for Computing with Quantum Dots," *Proceedings of the IEEE*, vol. 85, no. 4, pp. 541–557, 1997.

[9] D. Giri, G. Causapruno, and F. Riente, "Parallel and Serial Computation in Nanomagnet Logic: An Overview," *TVLSI*, pp. 1–11, 2018.

[10] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler, "An Exact Method for Design Exploration of Quantum-dot Cellular Automata," in *DATE*, 2018, pp. 503–508.

[11] F. Riente, G. Turvani, M. Vacca, M. R. Roch, M. Zamboni, and M. Graziano, "ToPoliNano: A CAD Tool for Nano Magnetic Logic," *TCAD*, vol. 36, no. 7, pp. 1061–1074, 2017.

[12] D. A. Reis, C. A. T. Campos, T. R. Soares, O. P. V. Neto, and F. S. Torres, "A Methodology for Standard Cell Design for QCA," in *ISCAS*, 2016, pp. 2114–2117.

[13] M. Vacca, J. Wang, M. Graziano, M. R. Roch, and M. Zamboni, "Feedbacks in QCA: A Quantitative Approach," *TVLSI*, vol. 23, no. 10, pp. 2233–2243, 2015.

[14] G. Fontes, P. A. R. L. Silva, J. A. M. Nacif, O. P. V. Neto, and R. Ferreira, "Placement and Routing by Overlapping and Merging QCA Gates," in *ISCAS*, 2018, pp. 1–5.

[15] E. Varga, M. T. Niemier, G. Csaba, G. H. Bernstein, and W. Porod, "Experimental Realization of a Nanomagnet Full AdderUsing Slanted-Edge Magnets," *IEEE Trans. Magn.*, vol. 49, no. 7, pp. 4452–4455, July 2013.

[16] W. Liu, E. E. Swartzlander Jr, and M. O'Neill, *Design of semiconductor QCA systems*. Artech House, 2013.

[17] J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi, and F. Lombardi, "Tile-based QCA design using majority-like logic primitives," *JETC*, vol. 1, no. 3, pp. 163–185, 2005.

[18] C. A. T. Campos, A. L. P. Marciano, O. P. V. Neto, and F. S. Torres, "USE: A Universal, Scalable, and Efficient Clocking Scheme for QCA," *TCAD*, vol. 35, no. 3, pp. 513–517, 2016.

[19] K. Hennessy and C. S. Lent, "Clocking of Molecular Quantum-dot Cellular Automata," *J. Vac. Sci. Technol. B*, vol. 19, no. 5, pp. 1752–1755, 2001.

[20] M. T. Alam *et al.*, "On-Chip Clocking of Nanomagnet Logic Lines and Gates," *TNANO*, vol. 11, no. 2, pp. 273–286, 2012.

[21] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, 1st ed. McGraw-Hill Higher Education, 1994.

[22] M. G. A. Martins, V. Callegaro, F. S. Marranghello, R. P. Ribas, and A. I. Reis, "Majority-based logic synthesis for nanometric technologies," in *IEEE-NANO*, 2014, pp. 256–261.

[23] K. Kong, Y. Shang, and R. Lu, "An Optimized Majority Logic Synthesis Methodology for Quantum-Dot Cellular Automata," *TNANO*, vol. 9, no. 2, pp. 170–183, 2010.

[24] 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*. ACM, 2019, pp. 197–202.

[25] 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," vol. 15, no. 3. New York, NY, USA: ACM, 2019, pp. 29:1–29:10. [Online]. Available: http://doi.acm.org/10.1145/3312661

[26] I. L. Bajec and P. Pečar, "Two-layer synchronized ternary Quantum-dot Cellular Automata wire crossings," *Nanoscale Research Letters*, vol. 7, no. 1, 2012.

[27] F. S. Torres, R. Wille, M. Walter, P. Niemann, D. Große, and R. Drechsler, "Evaluating the Impact of Interconnections in Quantum-Dot Cellular Automata," in *DSD*, 2018, pp. 649–656.

[28] C. Sechen, *VLSI Placement and Global Routing Using Simulated Annealing*. Springer Science & Business Media, 2012, vol. 54.

[29] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Science & Business Media, 2011.

[30] M. Ottavi *et al.*, "Partially Reversible Pipelined QCA Circuits: Combining Low Power With High Throughput," *TNANO*, vol. 10, no. 6, pp. 1383–1393, 2011.

[31] F. S. Torres *et al.*, "Exploration of the Synchronization Constraint in Quantum-dot Cellular Automata," in *DSD*, 2018, pp. 642–648.

[32] F. S. Torres, M. Walter, R. Wille, D. Große, and R. Drechsler, "Synchronization of Clocked Field-Coupled Circuits," in *2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO)*. IEEE, 2018, pp. 1–5.

[33] F. S. Torres, "QCADesigner-E." [Online]. Available: https://github.com/FSillT/QCADesigner-E

[34] K. Walus and G. A. Jullien, "Design tools for an emerging SoC technology: Quantum-dot Cellular Automata," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1225–1244, 2006.

[35] M. Walter, R. Wille, F. S. Torres, D. Große, and R. Drechsler, "fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019, arXiv:1905.02477.

[36] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 24–40.

[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[38] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[39] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *ISCAS*. IEEE Press, 1985, pp. 677–692.

[40] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *IWLS*, 2015.

[41] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 Benchmarks and First ATPG Results," *D&T*, vol. 17, no. 3, pp. 44–53, 2000.