Bail on Balancing: An Alternative Approach to the Physical Design of Field-coupled Nanocomputing Circuits

Marcel Walter[1] Robert Wille[2,3,4] Frank Sill Torres[5] Rolf Drechsler[1,3]

[1]Group of Computer Architecture, University of Bremen, Germany
[2]Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
[3]Cyber Physical Systems, DFKI GmbH, Bremen, Germany
[4]Software Competence Center Hagenberg GmbH, Hagenberg, Austria
[5]Department for the Resilience of Maritime Systems, DLR, Bremerhaven, Germany
{m_walter, drechsler}@uni-bremen.de, robert.wille@jku.at, frank.silltorres@dlr.de

***Abstract*—*Field-coupled Nanocomputing* (FCN) devices emerged as a post-CMOS alternative that promises highest computation capabilities with lowest power dissipation. To meet the strict FCN design rules, most existing physical design techniques still rely on conventional methods for almost two decades now – rendering the realization of large scale functions infeasible. In this work, we propose an alternative approach to the physical design of FCN circuits which bails on ineffective methods like balancing. By this, we are able to generate FCN circuit layout descriptions of functions for which state-of-the-art methods timeout.**

I. Introduction

Despite the continuous advances of today's standard technology (CMOS) and its derivatives, one can clearly state that the approaching of physical barriers demands the research for alternatives. This challenge motivated the proposal of numerous technologies – even if, until today, no dominating candidate emerged. Several predictions even indicate a more diverse future reality with several technologies existing in parallel and serving different markets [1]. If aiming at future ultra-low power applications, the nanotechnology *Field-coupled Nanocomputing* (FCN) appears as a promising candidate [2]. In contrast to conventional technologies, information transfer and processing is implemented without any electric currents but via field forces. Hereby, FCN is not bound to a specific material and has been experimentally realized, amongst others, with dangling bonds [3], molecules [4], or nanomagnets [5].

Interestingly, despite its promising character, there is still a lack of automatic design methodologies for FCN circuits. One can determine two principal reason for this shortcoming – first, conventional design methods can only be partially applied and, second, most solutions presented thus far suffer from a complexity that increases drastically with design size, e.g. [6], [7], [8]. Both reasons are intensely discussed in this paper in order to pinpoint to the weaknesses of existing solutions. We clearly show that automatic design solutions for FCN circuits require a completely new approach that differs from classic strategies. That means, the conventional FCN design flow of – starting from a given logic network – balancing, crossing reduction, and crossing substitution leads to logic duplication and requires mapping to pre-clocked grids – eventually causing tremendously high design complexity. The consequences of this increase are manifold and include larger circuit areas, longer circuit delays, and extended design time.

Our contributions are (1) an extensive discussion about the aforementioned shortcomings in conventional FCN physical design which is more or less done the same way as a decade ago; and (2) the proposal of a new physical design algorithm which bails on classic strategies such as balancing to keep input sizes moderate. This enables us to realize larger functions than with those classical methods. Experimental evaluations confirm the benefits.

(a) Zero state ($P = -1$) (b) One state ($P = +1$)

Fig. 1: Elementary QCA cell devices

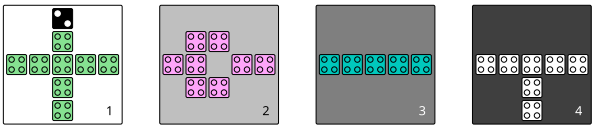
To keep this work self-contained, we introduce FCN circuits by means of the QCA implementation as a running example in Section II. There, we also discuss the physical design problem in this domain. In Section III, we discuss shortcomings of current solutions to this problem and develop a general idea of an alternative one which we then propose and discuss in detail in Section IV. We evaluate the resulting approach in Section V and conclude the paper in Section VI.

II. FCN Circuits and their Physical Design Problem

Field-Coupled Nanotechnologies (FCN) is an umbrella term for a class of several post-CMOS technologies such as *atomic Quantum-dot Cellular Automata* (aQCA, [9], [3]), *molecular Quantum-dot Cellular Automata* (mQCA, [4]), or *Nanomagnet Logic* (NML, [10]). While their physical implementations differ, their computational concepts are almost identical. Because of this, many of the existing design approaches apply to the entire class of field-coupled circuits. In order to focus on the main contributions of this work and without loss of generality, we will consider QCA-like technologies as a representative in the following. However, all mentioned aspects can easily be adopted for the other physical implementations of the FCN concept.

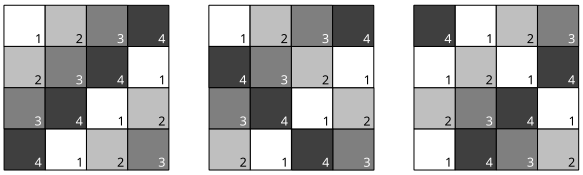
In general, FCN circuits are implemented using elements that interact via local fields (usually called *cells*). In QCA, a cell is composed of four *quantum dots* which are able to confine an electric charge and are arranged at the corners of a square [11]. Each cell contains two free and mobile electrons that are able to tunnel between adjacent dots and interact via Coulomb forces (note that 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.

Example 1. *Fig. 1 shows the schematic representations of elementary QCA devices called cells. The four circles denote the quantum dots where the two filled ones represent quantum dots containing electrons. The square shape illustrates the potential barrier shielding the cell from the outside world and, by this, preventing tunneling to nearby quantum dots. With an external electric field applied, these are the only two stable states a cell can assume.*



(a) AND (b) NOT (c) Wire (d) Fan-out

Fig. 2: Tiles in QCA implementation [12]



(a) 2DDWave [14] (b) USE [15] (c) RES [16]

Fig. 3: Clocking schemes for QCA circuit layouts

When composing several FCN cells next to each other, field interactions cause the polarization of one cell to influence the polarization of the others. This allows to implement structures that transmit and process information by arranging cells in patterned arrays – causing mutual repulsion in electrons in adjacent cells. Those structures are usually grouped together in *tiles* of uniform size – creating a gate library.

Example 2. *In Fig. 2, example implementations of some entities in tiles of* 5×5 *QCA cells are depicted. Fig. 2a realizes a Majority structure with the inputs being in the north, west, and south as well as with the output being in the western direction. Polarizations of the three inputs compete for the center cell where the strongest combined forces win. The result is then propagated to the output cell. However, the black cell in the north misses two quantum dots and, therefore, is in a constant zero state changing the gate function to an AND because* $\langle ab0 \rangle \equiv a \wedge b$*. Analogously, an OR gate can be constructed by changing one input to a constant one state.*

Fig. 2b realizes an Inverter with the input in the western and the output in the eastern direction. During signal propagation, the cells eventually interact over the corners causing a negation.

Fig. 2c implements a simple wire where information is propagated straight-forward. Lastly, Fig. 2d splits the signal in the center cell, realizing a fan-out structure.

However, in the preceding example, the input and output cells were arbitrarily assigned. Due to the meta-stability reasons and, in order to control the data flow within a design, FCN cells and, hence, also FCN tiles cannot be connected arbitrarily [13]. In fact, all cells and, thus, also the tiles, must be associated to an external clock that controls the initialization, holding, and resetting of the states of the cells. In case of QCA, an external electric clock controls the tunneling within the cells. Depending on the technology, each cell changes during a complete clock cycle between up to four different phases, i. e., a *switch*, a *hold*, a *reset*, and a *neutral* phase. In case of four phases, usually four external clocks numbered from 1 to 4 are applied, whereby each clock controls a selected adjacent set of cells (i. e. a tile). Furthermore, information flows from cells controlled by clock 1 to cells controlled by clock 2, from cells controlled by clock 2 to cells controlled by clock 3, and so on.

Various clocking schemes laid-out as floor plans have been proposed over the years [14], [15], [16]. Fig. 3 exemplifies the concept by providing a selection: Here, tiles are represented in different gray-scales which represent the respective clocks controlling them. We additionally highlight the different clocks with a small number in the bottom-right corners as well as by drawing the cells within the tiles in different colors (as it is the case, e.g., in Fig. 2).

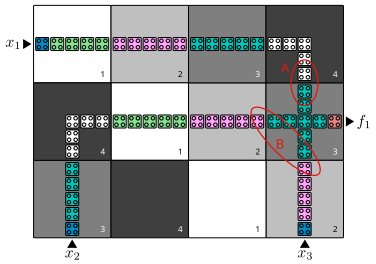


Fig. 4: Synchronization issues in FCN technologies

Example 3. *Consider the* 2DDWave *clocking scheme as sketched in Fig. 3a [17]. It forms one of the simplest floor plans where each counter diagonal is assigned the same clock number. This way, the incoming information flow to a tile is solely possible from the northern and western directions while the outgoing information flow from a tile always has to utilize the eastern or southern directions. That inherently restricts the scheme in multiple ways, since e. g. (1) sequential circuits cannot be realized due to the lack of feedback loops and (2) neither Majority gates nor 3-output fan-outs are possible due to the maximum input and output degree of* 2 *for each tile.*

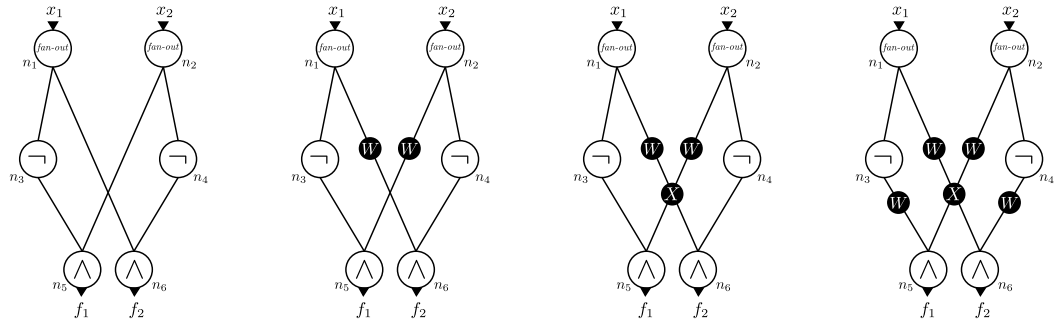
This last issue is a problem with the USE *clocking scheme sketched in Fig. 3b as well [15]. While USE indeed allows feedback, its tiles' maximum input and output degree is also* 2*. The* RES *scheme sketched in Fig. 3c overcomes this restriction and allows for feedback, Majority gates, and 3-output fan-outs in certain tiles. However, due to the increased degree in some tiles, the degree in other tiles must naturally be lower as we are still facing a 2-dimensional grid structure. Therefore, circuit layouts tend to become more widespread in the RES scheme and consequently have higher area costs and longer critical paths [16].*

With those clocking schemes/floor plans defining possible data flow connections for logic elements, arbitrary functionality can be realized in FCN circuits – assuming the respectively given functionality to be realized can properly be layouted onto a grid. At a first glance, a corresponding physical design task, i. e., generating FCN layouts from logic networks, seems to boil down to the placement of gates to be realized so that the data flow always obeys those clocking restrictions. However, there are in fact two further issues a designer additionally needs to address:

1) **Local Synchronization:** Data is only propagated in a way where tiles controlled by a clock i are followed by tiles controlled by a clock $(i + 1) \bmod C$ (with C being the number of different clocks used in the technology; i. e., $C = 4$ for QCA), and
2) **Global Synchronization:** The number of passed tiles for any two signals traveling from primary inputs to the same gate have to be equal. Otherwise, data will arrive desynchronized – leading to different or even undefined behavior.

Example 4. *Consider Fig. 4 which depicts a QCA circuit realized with* 4×3 *tiles, where the upper left and lower right tiles have the coordinates* $(0, 0)$ *and* $(3, 2)$*, respectively. At a first glance, one could assume that the circuit implements the Majority function* $f_1 = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$*, because the tile at position* $(3, 1)$ *contains a Majority gate, while all the other tiles have wire segments assigned.*

However, one can identify two problems. First, the tile at position $(0, 3)$ *is controlled by clock 4 while the tile at position* $(3, 1)$ *is controlled by clock 3 (see letter A). This violates the*



(a) Initial logic network (b) Balancing (c) Crossing substitution (d) Another balancing

Fig. 5: Balancing and crossing substitution

local synchronization, and thus, no data flow from the former to the latter tile is possible.

Consider now the wires connecting the primary inputs x_2 and x_3 with the Majority gate at position $(3, 1)$ (see also letter B). Here, the local synchronization is satisfied. However, information from the primary input x_2 at position $(0, 2)$ needs to pass 4 tiles to arrive at the joint tile $(3, 1)$, while the signal from x_3 at tile position $(3, 2)$ only needs to pass 1 tile. This way, x_2's signal information arrives with a delay of one full clock cycle – violating the global synchronization.

Overall, this leads to the physical design problem for FCN circuits which is considered in detail in this work.

III. Drawbacks of Current Solutions and General Idea for an Alternative Approach

Since the discovery of the FCN concept as a circuit technology, design automation engineers came up with numerous techniques to map logic representations to FCN circuit layouts, e. g. [18], [8], [6], [19]. However, the core ideas did not fundamentally change over the years. One of the main obstacles is to achieve global synchronization within the layout as discussed in the previous section. To this end, those approaches employed a preprocessing step on the originally given function to be realized first.

More precisely, at the beginning of the layout process, the logic function to be realized is provided in terms of a logic network, i. e., a graph where nodes are assigned Boolean operations and edges represent signals connecting them. Examples for commonly used logic networks are *And-inverter graphs* (AIGs) and *Majority-inverter graphs* (MIGs) as their nodes represent AND and Majority functions, respectively, which can directly be translated into corresponding uniform QCA tile representations using a gate library like the one introduced in the previous section. However, having such a graph description, performing a direct mapping to a clocking scheme is rarely possible without inserting additional wire elements to respect local and global synchronization (which, in contrast to the conventional design, may cause substantial costs [20], [21]). Furthermore, crossings in the realization are to be avoided because they lead to additional physical crossing elements which require one extra tile each. Overall, these restrictions formulate an $\mathcal{NP}$-complete problem [22].

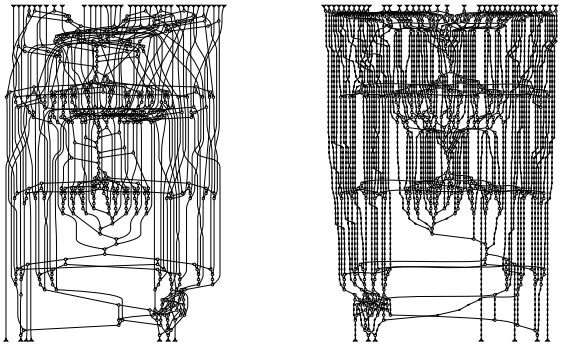
Example 5. *Consider the Boolean function represented in terms of an AIG (with explicit fan-out and inverter nodes) with inputs x_1, x_2 and outputs f_1, f_2 as shown in Fig. 5a. Here, the AND node labeled n_5 gets one input from node n_3, which gets its input from node n_1 making for a total path length of 3. On the other hand, node n_5's second input is fed by node n_2 only, making it a total path length of just 2 (analogously for node n_6). A direct mapping to a layout would therefore violate global synchronization. Furthermore, the network also contains a crossing of two connections in its current representation, which needs to be taken into account as well.*

Almost all solutions for the design of FCN circuits tackled this problem by performing preprocessing steps which alter the structure of the logic network in order to avoid these problems.[1] More precisely, the following steps are usually performed:

1) **Path balancing:** First, a levelizing is computed which assigns a *depth* d to each node in the graph. By this, each primary input gets $d = 0$ and each other node with predecessors $p_1, \ldots, p_n$ gets $d = max(p_1, \ldots, p_n) + 1$, respectively. Then, for all adjacently connected nodes n_1, n_2 and which differ in their depth by more than 1, auxiliary balance nodes are introduced by subdividing the respective connection until the depth difference is evened out.
2) **Crossing reduction:** All nodes with the same depth value are being grouped together in a *rank* r_d. An ordering is computed for each rank so that the connections intersecting with each other in a drawing of these ranks is minimized. Due to the $\mathcal{NP}$-completeness of this problem [26], such an ordering can only be approximated in reasonable time.
   Further reduction is sometimes accomplished by logic duplication that, however, increases the logic network's size exponentially in the worst case.
3) **Crossing substitution:** Each remaining intersection of connections is being substituted by a designated crossing node so that it can be handled properly during physical design. This however may cause the need to redo the balancing and thus insert even more auxiliary nodes.

Example 6. *Consider again the logic network shown in Fig. 5a. Applying the balancing step as reviewed above yields a network as shown Fig. 5b which resolves the problem concerning global synchronization illustrated in Example 5 on the logic level by inserting two auxiliary wire nodes drawn in black with a white W label. Since the network contains a crossing of two connections, a substitution step is performed resulting in the network shown in Fig. 5c which now contains a designated crossing node drawn in black with a white X label. However, thereby, another balancing becomes necessary to equalize paths lengths which finally yields the balanced and substituted logic network shown in Fig. 5d which could finally be processed by a conventional FCN physical design algorithm.*

[1] There are alternative approaches that introduce additional clock signals to circumvent this preprocessing, e. g., [23], [24]. However, to the best of our knowledge, of all non-alternative approaches not relying on formal methods, only [25] does not make use of any of the following techniques.



(a) Unbalanced (478 nodes) (b) Balanced (3275 nodes)

Fig. 6: Logic network of c432 [27]

Obviously, applying those preprocessing steps significantly increases the number of nodes and, by this, the input size to the subsequent physical design process that realizes the FCN layout. While this might be negligible for the small logic networks discussed in the examples above, this quickly sums up for larger functions. As a consequence, the runtime increases disproportionately much (as we are facing an $\mathcal{NP}$-complete problem), rendering most existing approaches unusable.

Example 7. *Fig. 6a sketches an original graph representation for the function* c432 *[27] composed of 478 nodes.*[2] *After employing just the balancing step, this graph gets extended to the version sketched in Fig. 6b composed of 3275 nodes where the wires again are represented by smaller black nodes. In other words, satisfying the global synchronization constraints following the currently conducted state-of-the-art frequently leads to a drastic blow-up in the number of nodes and, hence, leads to intractability in the worst case, when certain methods for crossings reduction are incorporated like the ones explained above. As a consequence, many conventional algorithms for FCN physical design do not even terminate (cf. Section V).*

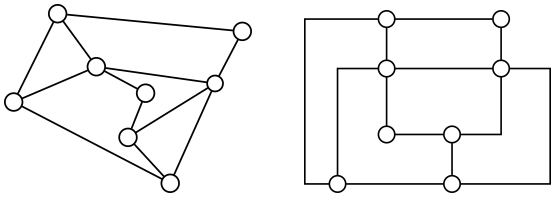
Furthermore, in the bottom center of Fig. 6b, long connections that span from one side of the network to other can be spotted. Even though, their source and target nodes are just one level apart, during a mapping to an FCN layout, these gates could not be directly connected as they will most likely not be adjacent in the layout. Consequently, even more wire nodes have to be introduced to realize these wire connections.

Obviously, this analysis showed the main drawbacks of current solutions for FCN design: The preprocessing step employed by almost all existing solutions thus far frequently "blows-up" the size of the instance to be layouted – a critical development for an $\mathcal{NP}$-complete problem. Motivated by this, we are proposing an alternative approach for the physical design of FCN circuits. The main idea is to bail on the troublesome preprocessing steps reviewed above and, instead, work on the originally given input network. This obviously requires an alternative method to deal with the synchronization problems which is described in the next section.

IV. Resulting Alternative Approach

Bailing on balancing and crossing reductions allows us to work on the originally given input network. While this avoids "blowing up" the instance size as sketched by the examples in the previous section, it also puts us back to square one in dealing with how to satisfy the synchronization constraints reviewed in Section II. However, we observed that, by additionally using knowledge from graph theory, those constraints can be dealt with "on-the-fly" during the layouting process.

[2]Note that the details of this graph are not relevant; the figure should only illustrate the size of a non-trivial function.



(a) Straight-line drawing (b) Orthogonal drawing

Fig. 7: Graph drawing examples

More precisely, instead of determining an *absolute* positioning of nodes as it is done by the preprocessing steps of almost all existing approaches thus far, we propose to compute *relative* positions of them first (for which we do not need to "blow up" the input network). To this end, we utilize methods addressing the *Graph Drawing* problem [28] – a common problem in graph theory which tries to find a visually pleasing representation of a graph by assigning proper positions to vertices and edges and which finds applications e. g., in visualizing information systems such as UML diagrams, VLSI circuits, etc. For our case, we particularly utilize the *Orthogonal Graph Drawing* (OGD) [29] problem, in which vertices get integer positions assigned and edges must exclusively be drawn as segments arranged in a 90°-fashion.

Example 8. *Fig. 7 depicts two examples of graph drawings for the same given input graph. In Fig. 7a, a* straigh-line drawing *has been computed whose only restriction is that all edges must be drawn as straight lines. Fig. 7b instead shows an* orthogonal drawing.

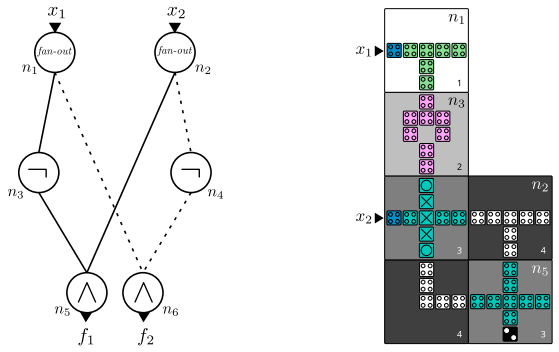
The task to find an orthogonal drawing already resembles a physical design problem remarkably close. In both cases, non-overlapping integer positions for the nodes are to be determined, and also in both cases, the connections are to be realized in a 90°-fashion as wire segments. Therefore, one could assume that an orthogonal drawing of a logic network can be directly translated into an FCN circuit layout.

Unfortunately, this is not the case because the clocking and synchronization restrictions are not considered in OGD. It is not even guaranteed that for any OGD of any directed graph, i. e., logic network, a clocking can be computed that satisfies the local synchronization constraint. In fact, most existing algorithms for OGD work on undirected graphs and thereby disregarding information flow directions, making it unlikely to find a valid clocking for a given orthogonal drawing.

In the following, we present a technique which enables us to generate orthogonal drawings of logic networks which always can be clocked to satisfy FCN design constraints. Thereby, such drawings are directly isomorphic to FCN circuits and can be seamlessly translated. By this, our approach can *draw* logic networks as FCN circuits without an intermediate step.

To this end, we incorporated a clocking strategy into the drawing process. We achieve this by relying on a *direction assignment* to the network edges similar to *edge coloring* used in OGD algorithms [30]. More precisely, we restrict incoming or outgoing directions of each node mapped to the FCN layout prior to placement. We explain the procedure with two different outgoing directions in the following. The directions we pick are the eastern and southern directions for outgoing connections (likewise, the northern and western directions for incoming connections). It must be ensured that for an assignment $d : E \rightarrow \{east, south\}$ to the edges of the input logic network,

1) each outgoing edge of any node must have a different direction assigned, and



(a) Direction assignment (dashed = east)

(b) Resulting layout (for the first four nodes)

Fig. 8: Physical design using direction assignment

2) each incoming edge to any node must have the same direction assigned.

The assignment d can be computed in linear time and needs only one additional bit of memory overhead per edge. Utilizing this assignment is the basis for generating the aforementioned drawing, i. e., the FCN layout because the assigned directions express the spatial relation between each two nodes.

Example 9. *Consider the running example network in Fig. 5a one last time. A direction assignment using only the two directions* south *and* east *yields a representation as shown in Fig. 8a, where solid lines represent* south *and dashed lines represent* east*. Since the edge (n_1, n_3) is annotated with the southern direction, we can infer that in the final layout, node n_3 will be located south of node n_1.*

Thereby, a relative positioning is achieved which allows incorporating auxiliary balance wires, crossings, and the clocking on-the-fly during physical design, i. e., layout drawing/-generation as we demonstrate in the following. The relative positions generate dependencies of the placement order. As shown in the previous example, node n_3 cannot be placed before node n_1. To resolve these dependencies, we process the nodes in topological order, i. e., starting with the primary inputs and never processing any node which has unprocessed predecessors.

Starting with an empty layout, the first logic node n_1 can be placed at position $(0, 0)$ with clock zone 1. For each following node n_i it is guaranteed that it does not have unplaced predecessors. Hence, n_i can be placed in either a new column (incoming edges are assigned *east*) or a new row (incoming edges are assigned *south*) so that a wire connection with at most one bend is possible [30]. By this, the 2DDWave clocking naturally emerges (cf. Fig. 3a).

Example 10. *Processing the nodes of Fig. 8a in topological order, i. e. $n_1, n_3, n_2, n_5, n_4, n_6$, provides a proper placement sequence. The benefits of this order can be seen by considering e. g. node n_5: this node has incoming connections from nodes n_3 and n_2, where node n_3 on its part has an incoming edge from node n_1. Placing node n_5 therefore requires to first place all nodes it depends on, which we ensure using the topological sorting.*

Based on that, the actual placement is conducted as follows (illustrated in Fig. 8b): We start with an empty grid and place fan-out node n_1 on position $(0, 0)$. Next to process is the NOT gate n_3, which has an incoming connection labeled south*, meaning we place it south of its predecessor, i. e., at position $(0, 1)$. Node n_2 however does not have any labeled incoming edges and is therefore placed in a new column and new row to not interfere with other connections, i. e., at position $(1, 2)$. There, we can add a wire to the layout's border to make the primary input accessible. Now, that all of its predecessors have been placed, we can find a position for node n_5 as well. It has two incoming connections labeled* south*, meaning that it needs to be located in the south of its predecessors too. The position that fulfills this requirement with the least wire overhead is $(1, 3)$, because n_2 and n_5 can be connected directly, and from n_3 to n_5 there are only 2 wire segments that need to be realized (including 1 crossing).*

One can notice how a clocking scheme naturally emerges during placement. Nodes n_4 and n_6 can then be placed in the same fashion and are left to the reader as an exercise due to page limitations.

Overall, this scheme allows obtaining FCN layouts from given logic networks *without* the need for conventional preprocessing steps. By this, the mapping can be conducted on the originally given graph and does not require balancing as most previously proposed methods. Also, global synchronization becomes controllable because all primary input pins are located on the layout's western border and all primary output pins on the layout's eastern border. Therefore, the path discrepancy can easily be measured and adjusted by the pins' distance on the layout. Eventually, this allows to conduct the physical design substantially more efficient than related work. This is also confirmed by experiments whose results are summarized next.

V. Experimental Evaluation

In order to demonstrate the applicability of the proposed solution, we implemented the approach described in the previous section in C++ on top of the publicly available FCN design framework *fiction* [31] as command `ortho -b`.[3] In this section, we discuss the experimental setup and summarize the results.

We took standard benchmarks in the form of synthesized Verilog netlist files from [27], [32] and converted them to AIGs. In all AIGs, we replaced complemented edges with inverter nodes and also substituted high-degree outputs by designated fan-out nodes; applying a breadth-first strategy in case of cascading fan-outs. We then compared our proposed approach against the conventional FCN design process, i. e., applying preprocessing for which we used *fiction*'s `balance` command. All benchmark runs were conducted on a Fedora 30 machine with an Intel Xeon E5-2630 v3 CPU @ 2.40 GHz (up to 3.20 GHz boost) with 20 MB of L3 cache and 64 GB of main memory. The results have been verified using the method from [33].

The obtained results are summarized in Table I. The column *Benchmark* lists information about the networks' name, its number of nodes (including inverters and fan-outs), and its number of primary inputs and primary outputs. The columns *Conventional approach* and *Proposed approach* list the resulting QCA layout's area usage in μm^2, its energy dissipation for slow (25 GHz) and fast (100 GHz) clocking in meV, as well as the runtime in CPU seconds it took the algorithm to obtain the result. The amount of additionally needed balancing nodes for the conventional preprocessing method is given in column *+W*. For area calculations, we took standard cell dimension values from *QCADesigner* [34], i. e., $18\,nm \times 18\,nm$ for a single cell with vertical and horizontal cell spacing of 2 nm. Calculations for energy dissipation are obtained from [35].

As it can be seen, the proposed approach has a significant advantage compared to the conventional technique. As our approach does not "blow up" the logic network with – apparently unneeded – wire nodes, it achieves better results in area consumption, energy dissipation, and runtime. It even is capable of obtaining results for benchmarks for which the conventional approach timeouted after 30 min.

[3] The source code is available at https://github.com/marcelwa/fiction.

TABLE I: Comparison to conventional approach

Benchmark			Conventional approach (including preprocessing)					Proposed approach			
Name	**#Nodes**	**I / O**	**+W**	**Area**	***E* (25 GHz)**	***E* (100 GHz)**	**Time**	**Area**	***E* (25 GHz)**	***E* (100 GHz)**	**Time**
c432	478	36 / 7	2 797	6 704.66	16 850.68	152 994.88	9.98	673.57	2 903.88	25 921.11	0.09
c499	1 021	41 / 32	3 521	18 226.40	36 599.97	332 446.41	29.96	2 969.23	9 008.83	81 061.34	0.42
c880	670	60 / 26	3 402	12 811.37	30 922.86	280 916.43	19.98	1 480.56	5 678.16	50 904.80	0.23
c1355	1 229	41 / 32	4 785	27 082.75	47 957.47	435 633.71	46.60	3 823.24	10 868.45	97 710.04	3.63
c1908	1 024	33 / 25	5 327	26 045.93	56 440.93	513 297.67	42.49	2 817.90	10 404.72	93 826.96	3.47
c2670	1 684	233 / 63	5 204	55 128.61	127 089.76	1 156 392.81	98.47	11 261.18	34 459.66	312 396.18	15.21
c3540	2 408	50 / 22	8 379	103 028.81	202 801.57	1 845 927.47	205.12	14 755.96	48 407.64	439 050.86	21.04
c5315	4 336	178 / 123	26 008	—	—	—	*TO*	52 990.26	1 65 679.22	1 506 855.03	93.86
c6288	6 066	32 / 32	18 178	—	—	—	*TO*	78 303.12	1 01 417.73	918 122.90	107.38
c7552	5 042	207 / 107	54 084	—	—	—	*TO*	70 587.60	1 63 835.11	1 489 006.48	117.46
ctrl	461	7 / 25	1 277	3 090.52	12 460.83	113 166.57	3.10	565.95	4 111.17	37 061.62	0.11
router	556	60 / 3	5 582	15 460.42	30 163.55	274 099.89	27.05	1 026.74	2 742.06	24 259.51	1.12
int2float	609	11 / 7	988	3 409.62	10 596.54	95 949.28	3.80	938.32	4 812.47	43 229.05	0.15
dec	907	8 / 256	51	3 482.90	25 595.19	232 967.48	5.96	3 223.50	24 929.33	226 894.55	4.89
cavlc	1 711	10 / 11	3 072	28 295.12	106 961.25	973 666.87	51.87	6 956.60	41 728.29	379 170.14	13.33
adder	2 170	256 / 129	161 170	—	—	—	*TO*	15 688.10	62 584.89	568 113.76	26.36
priority	2 186	128 / 8	86 028	—	—	—	*TO*	13 319.89	63 593.25	577 339.47	23.06
max	6 772	512 / 130	159 581	—	—	—	*TO*	136 817.83	407 646.74	3 712 404.33	227.93
bar	7 437	135 / 128	14 852	—	—	—	*TO*	139 227.53	343 744.64	3 131 973.20	235.15
sin	12 608	24 / 25	205 045	—	—	—	*TO*	364 025.97	657 525.96	5 985 854.67	416.84

#Nodes Logic nodes plus fan-outs **Area** Layout area in μm^2 **Time** CPU time in seconds
+W Auxiliary wire nodes due to balancing ***E*** Energy dissipation in meV *TO* Timeout of 30 min reached

VI. Conclusion

In this work, we investigated the design of Field-coupled Nanocomputing which is an emerging post-CMOS concept that promises astonishing energy dissipation characteristics. In particular, we discussed the shortcomings of conventional physical design approaches in the domain. These often unnecessarily "blow-up" their input logic networks with auxiliary balance wire nodes leading to worse performance and obtained layout quality. We proposed a novel approach to tackle this problem in an alternative fashion. We propose to bail on network balancing and only insert wire elements on the layout-level where needed utilizing findings from graph theory. By this, we were able to obtain significantly improved results in less runtime than the current state of the art.

Acknowledgments

This work has partially been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

References

[1] V. Khanna, *Integrated Nanoelectronics: Nanoscale CMOS, Post-CMOS and Allied Nanotechnologies*, ser. NanoScience and Technology. Springer India, 2016.

[2] R. Wang, M. Chilla, A. Palucci, M. Graziano, and G. Piccinini, "An Effective Algorithm for Clocked Field-coupled Nanocomputing Paradigm," in *NMDC*, Oct 2016.

[3] T. R. Huff *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.

[4] C. S. Lent *et al.*, "Molecular Cellular Networks: A non von Neumann Architecture for Molecular Electronics," in *ICRC*, 2016.

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

[6] 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*. ACM, 2008, pp. 243–248.

[7] 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.

[8] G. Fontes *et al.*, "Placement and Routing by Overlapping and Merging QCA Gates," in *ISCAS*, 2018.

[9] 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.

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

[11] W. Liu, E. E. Swartzlander Jr, and M. O'Neill, *Design of Semiconductor QCA Systems*. Artech House, 2013.

[12] D. A. Reis *et al.*, "A Methodology for Standard Cell Design for QCA," in *ISCAS*, 2016, pp. 2114–2117.

[13] 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.

[14] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Clocking and Cell Placement for QCA," in *IEEE-NANO*, vol. 1, 2006, pp. 343–346.

[15] C. A. T. Campos *et al.*, "USE: A Universal, Scalable, and Efficient Clocking Scheme for QCA," *TCAD*, vol. 35, no. 3, pp. 513–517, 2016.

[16] M. Goswami *et al.*, "An Efficient Clocking Scheme for Quantum-dot Cellular Automata," *Electron. Lett.*, 2019.

[17] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Two-dimensional schemes for clocking/timing of QCA circuits," *TCAD*, vol. 27, no. 1, pp. 34–44.

[18] W.-J. Chung, B. Smith, and S. K. Lim, "Node Duplication and Routing Algorithms for Quantum-dot Cellular Automata Circuits," *IEE Proceedings-Circuits, Devices and Systems*, vol. 153, no. 5, pp. 497–505, 2006.

[19] A. Chaudhary, D. Z. Chen, X. S. Hu, M. T. Niemier, R. Ravichandran, and K. Whitton, "Fabricatable Interconnect and Molecular QCA Circuits," *TCAD*, vol. 26, no. 11, pp. 1978–1991, 2007.

[20] F. Sill 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.

[21] F. Sill Torres *et al.*, "On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Automata," *MICPRO*, vol. 76, pp. 103–109, 2020.

[22] M. Walter, R. Wille, D. Große, F. S. Torres, and R. Drechsler, "Placement & Routing for Tile-based Field-coupled Nanocomputing Circuits is NP-complete," in *JETC*, 2019.

[23] F. S. Torres, M. Walter, R. Wille, D. Große, and R. Drechsler, "Synchronization of Clocked Field-Coupled Circuits," in *IEEE-NANO*, 2018.

[24] R. Wille, M. Walter, F. Sill Torres, D. Große, and R. Drechsler, "Ignore Clocking Constraints: An Alternative Physical Design Methodology for Field-Coupled Nanotechnologies," in *ISVLSI*, 2019, pp. 651–656.

[25] M. Walter, R. Wille, F. S. Torres, D. Große, and R. Drechsler, "Scalable Design for Field-coupled Nanocomputing Circuits," in *ASP-DAC*, 2019, pp. 197–202.

[26] M. R. Garey and D. S. Johnson, "Crossing Number is NP-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 4, no. 3, pp. 312–316, 1983.

[27] 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.

[28] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for Drawing Graphs: An Annotated Bibliography," *Computational Geometry*, vol. 4, no. 5, pp. 235–282, 1994.

[29] M. Eiglsperger, S. P. Fekete, and G. W. Klau, "Orthogonal Graph Drawing," in *Drawing Graphs*. Springer, 2001, pp. 121–171.

[30] T. C. Biedl, "Improved Orthogonal Drawings of 3-graphs," in *CCCG*, 1996, pp. 295–299.

[31] 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.

[32] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL Combinational Benchmark Suite," in *Int'l Workshop on Logic Synth.*, 2015.

[33] M. Walter, R. Wille, F. S. Torres, D. Große, and R. Drechsler, "Verification for Field-coupled Nanocomputing Circuits," in *DAC*, 2020.

[34] K. Walus, T. J. Dysart, G. A. Jullien, and R. A. Budiman, "QCADesigner: A Rapid Design and Simulation Tool for Quantum-dot Cellular Automata," *TNANO*, vol. 3, no. 1, pp. 26–31, 2004.

[35] F. Sill Torres, R. Wille, P. Niemann, and R. Drechsler, "An Energy-Aware Model for the Logic Synthesis of Quantum-Dot Cellular Automata," *TCAD*, vol. 37, no. 12, pp. 3031–3041, 2018.