Design Automation for Field-coupled Nanotechnologies

Marcel Walter[1] Rolf Drechsler[1,2]

[1]Group of Computer Architecture, University of Bremen, Germany
[2]Cyber Physical Systems, DFKI GmbH, Bremen, Germany

{m_walter, drechsler}@uni-bremen.de

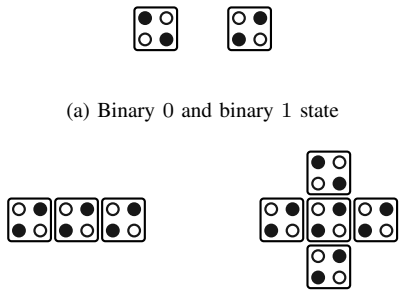
***Abstract*—Circuits based on complementary metal-oxide-semiconductors (CMOS) enabled the digital revolution and still provide the basis for almost all computational devices to this date. Nevertheless, the class of *Field-coupled Nanocomputing* (FCN) technologies is a promising candidate to outperform CMOS circuitry in various metrics. Not only does FCN process binary information inherently, but it also allows for absolute low-power in-memory computing with an energy dissipation that is magnitudes below that of CMOS. However, physical design for FCN technologies is still in its infancy. In this Student Research Forum Proposal, exact and heuristic techniques tackling design automation for FCN are presented.**

I. Motivation & Background

Worldwide energy consumption allotted to information and telecommunication systems is growing. Some scenarios predict that the sector could reach as much as 51 % of global electricity usage by 2030 and thereby contribute up to 23 % of the globally released greenhouse gases [13].

Consequently, there is an increasing interest in alternative technologies that enable fast computations with considerably lower energy dissipation compared to state-of-the-art CMOS transistors. *Field-coupled Nanocomputing* (FCN) [14] is a class of emerging technologies and is constantly gaining more attention. In contrast to conventional technologies, FCN conducts computations without any electric current flow – allowing operations with a remarkable low energy dissipation that is several magnitudes below current CMOS technologies [15], [16]. This promising outlook motivated explorations into its feasibility which led to several suitable contributions to the physical implementation of FCN technologies, many of them very recently (i. e. in the last 3–4 years) [17], [18], [19].

Generally, instead of transistors, FCN circuits use elements that are usually called *cells* that interact via mutual repulsion of local fields. In *Quantum-dot Cellular Automata* (QCA) [20], one possible implementation of the FCN concept, a cell is composed of four or six *quantum dots* which are each able to confine an electric charge [21], [22]. Adding two free and mobile electrons into each cell, that can tunnel between adjacent dots, yields two possible stable states due to mutual repulsion via Coulomb interaction. The resulting binary states are depicted in Fig. 1a where the circles indicate quantum dots and the black bullets indicate electrons. Fig. 1b shows how to arrange multiple cells in a row to build a wire segment. It transmits binary information from left to right or vice versa as the same field interactions happen across the cell boundaries and thereby affect the polarization of adjacent cells. This



(a) Binary 0 and binary 1 state

(b) Wire segment (c) Majority gate

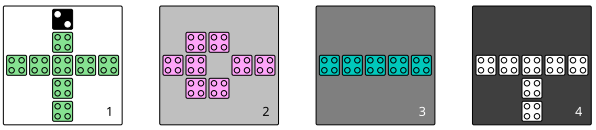
Fig. 1: Elementary QCA cell devices

formation is extended in Fig. 1c to construct a Majority gate where three input cells (e. g. top, left, and bottom) compete for the polarization of the center cell that eventually transmits its value to the output cell (e. g. the right one). By setting one input to a constant value, AND and OR gates can easily be constructed from Majority gates, too.

Several gate libraries have been proposed for various FCN technologies. Due to the brevity of this paper, the *QCA ONE* library [23] for the QCA technology will be used as a running example for visualizations. In this library, elementary gates are arranged in a *tile* of size 5×5 QCA cells. Implementations of an AND gate, an inverter, a wire segment and a fan-out are shown in Fig. 2. We kindly refer the inclined reader to the original paper for further information.

Unfortunately, the physical design task in FCN does not simply boil down to a classical placement of operations and routing of wires because much tighter domain-specific constraints apply. One of them is *clocking* that, in FCN, is a critical component of combinational and sequential circuits alike because it directs the data flow and, at the same time, controls information synchronization, too, ultimately leading to restrictions like the enforcement of signal path balancing throughout the entire design. Among other obstacles, these *clocking constraints* are a limiting factor of design automation for FCN circuitry – as they prevent the applicability of conventional VLSI approaches –, despite significant efforts in the development of corresponding methods, e. g. [24], [25].

In this student research forum proposal, novel and applicable methods for the physical design of FCN circuits are presented.



(a) AND (b) INV (c) Wire (d) Fan-out

Fig. 2: Tiles in QCA ONE implementation

II. Contribution

This section briefly presents the findings and results of the published papers associated with the thesis. Due to space limitations, the reader is referred to the respective papers for further information.

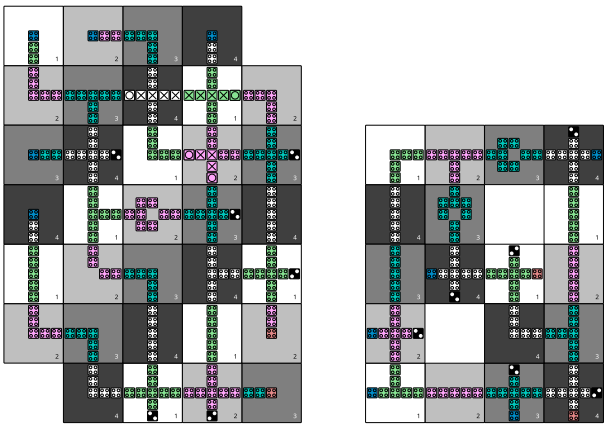
A. Theoretical Consideration

So far, algorithms to tackle the physical design problem for FCN have been developed without a clear understanding of the problem's complexity. Furthermore, no findings about related problems that could have been beneficial could be utilized in that process.

While conventional logic design methods are applicable to the FCN domain, the physical design constraints change tremendously compared to conventional VLSI as mentioned above. Nevertheless, the inputs and goals are almost identical. A physical design process always gets a logic description as some kind of network (AIG, MIG, XAG, etc.) as input and should eventually output a placed and routed circuit description that complies to all physical design constraints and can be physically simulated and sent to production.

In [1], the first theoretically sound definition of the physical design problem for FCN is given. It enabled to reason about the problem's complexity in various configurations and to link it to other well-studied problems in graph theory. Therefore, the FCN design task was formulated both as a decision problem (FCNPR) and an optimization problem (FCNOPR). It could be proven that both variants are intractable under the assumption that $\mathcal{P} \neq \mathcal{NP}$, i. e. FCNPR is $\mathcal{NP}$-complete and FCNOPR is $\mathcal{NP}$-hard. This was achieved by a reduction from the well-known *Hamiltonian Path Problem*. Further proofs showed that the intractability does not only hold for the most general cases of both problems in question but still applies when restricting the search space by e. g. using predefined clocking schemes. This way, several problem configurations inspired by actual design choices could be proven to be intractable. Thereby, it was shown that state-of-the-art FCN design methods did not suffer from insufficient methodology when trying to obtain optimal FCN circuits from specifications but were restricted by complexity reasons.

However, there are also positive results. The formal definition provides a basis to apply reasoning engines like SMT solvers to obtain exact solutions (cf. Section II-B). Additionally, two related problems could be identified, namely *Subgraph Homeomorphism* (SHP) [26], [27] and *Orthogonal Graph Drawing* (OGD) (cf. [28] for an overview) which have been extensively studied in the literature. Especially for OGD, poly-time approximations exist that can be utilized to develop scalable algorithms for FCN design and thereby circumvent intractability by accepting certain trade-offs (cf. Section II-C).



(a) 2DDWave clocking, primary I/O pins placed at the layout borders, and crossings enabled

(b) USE clocking, crossings disabled, and signal path balancing disabled

Fig. 3: Two differently layouted variants of c17

B. Exact Solution

Even though scalability of existing solutions for FCN physical design was rather limited already, no optimality with respect to some cost metric could be guaranteed. Therefore, the absolute quality of these solutions remained unclear.

In [2], the problem formulations introduced in [1] were picked up and modeled in first-order logic. Using an SMT solver [29], it became possible to generate optimal FCN circuits in terms of area while meeting all design constraints by a sequence of incremental solver calls. Various options and toggles allow for the use of, e. g., arbitrary predefined clocking schemes, crossings, primary input/output locations, unbalanced paths (cf. Section II-C), synchronization elements (cf. Section II-C), and wire-length restrictions. Additionally, optimization targets can be set to minimize, e. g., the number of wire segments or crossings used. Symmetry breaking and sophisticated encoding mechanisms are applied to reduce solving time as much as possible. For the first time, true design exploration of FCN circuits becomes possible and thereby provides a baseline to evaluate future heuristic solutions.

Fig. 3 shows two FCN circuit layouts implemented in the QCA ONE library. They have been generated from the same logic network benchmark *c17* taken from [30] using the discussed SMT-based exact physical design approach. For the layout in Fig. 3a, the *2DDWave* clocking scheme [31] was enforced, primary inputs (blue) and outputs (orange) were to be realized by designated pins and placed at the layout's borders, wire crossings were enabled, and signal path balancing was enforced. For the layout in Fig. 3b, however, the *USE* clocking scheme [32] was applied, primary inputs and outputs were not constrained, crossings were disabled, and signal path balancing was disabled, too.

The resulting layouts differ in various design metrics. The layout in Fig. 3a uses an area of 5×7 tiles, needed 18

TABLE I: Evaluating the exact solution

Benchmark			Previous s-o-t-a [33]			Proposed solution [2]		
Name	**#Gates**	**I / O**	**Dimension**	**CP**	**t in s**	**Dimension**	**CP**	**t in s**
2:1 MUX	5	3 / 1	4 × 5	5	9	3 × 3	5	< 1
XOR	6	2 / 1	4 × 7	7	11	3 × 3	5	< 1
XNOR	8	2 / 1	6 × 6	8	13	3 × 5	9	< 1
Half adder	10	2 / 2	7 × 6	8	55	5 × 5	13	10
c17	11	5 / 2	10 × 6	13	15	3 × 5	9	< 1
ParGen	14	3 / 1	9 × 10	14	27	3 × 8	16	6
4:1 MUX	16	6 / 1	11 × 8	19	9612	5 × 7	15	55
ParCheck	21	4 / 1	10 × 14	14	3014	6 × 7	15	224

#Gates Gate count plus fan-outs **CP** Critical path in tiles
Dimension Occupied area in tiles **t in s** Runtime in seconds

wire segments (plus 7 additional tiles for the primary I/O pins) and 3 wire crossings, has a critical path of 11 tiles and highest throughput of ⅛ which means that new signals can be applied to the primary inputs in every clock cycle. The layout in Fig. 3b uses an area of only 4×5 tiles, needed 7 wire segments (and no additional ones for primary I/O pins) and no crossings, but has a critical path of 13 tiles and lower throughput of ⅓ which means that all input signals need to be applied for 3 consecutive clock cycles until the output signals stabilize due to the violated path balancing.

Table I provides a brief overview about further obtained results compared to a former state-of-the-art solution [33]. The columns *Benchmark* list the names of the used benchmark circuits, the number of gates (including fan-out nodes) and the number of primary inputs and outputs respectively. The further columns *Previous s-o-t-a* and *Proposed solution* both list the dimension, i. e. the area in tiles, of the resulting FCN circuits, the critical path, and the needed runtime in seconds to obtain the results respectively. To ensure comparability between the two approaches, the same settings have been chosen, i. e., the USE clocking scheme, enabled wire crossings, no constrained primary I/O pins, and enforced signal synchronization.

As the design objective in both cases was area, the critical path value can differ. As it can be clearly seen, the proposed SMT-based exact solution does not only obtain smaller FCN circuit descriptions in every case but does so in significantly less runtime than a previous state-of-the-art [33].

C. Heuristic Solutions

Besides the intractability proofs, in [1], several similar problems to FCNPR and FCNOPR have been identified of which *Orthogonal Graph Drawing* (OGD) has known linear-time approximations. These findings are utilized in [3] to scalably generate FCN circuit descriptions from logic networks with a 2000+ times improvement in the processable gate count and a runtime that is magnitudes faster compared to the state-of-the-art.

OGD is a problem from the domain of graph drawing, which often asks for a *visually pleasing* representation of given graphs. As this is often needed for UML and other diagrams, graph drawing looks for a mapping of each node and each edge in the graph to some positions on a topology (e. g. a plane) so that a human viewer is assisted best in understanding the diagram. Typically, the number of crossings and bends of the lines are to be reduced as far as possible. In OGD specifically, each node has to get some non-overlapping integer coordinates assigned and every edge is to be drawn as non-overlapping segments of vertical and horizontal lines only. Hence, OGD can only obtain valid drawings of graphs of degree 4 or less, that is graphs where each node has at most degree 4 (in-degree + out-degree), which are called *4-graphs*.

OGD has powerful approximations when being restricted to *3-graphs*, which are analogously defined [34]. If it was possible to input a logic network as the graph to be drawn to that algorithm, the FCN physical design problem would be partially solved. However, the algorithm presented in [34] works on undirected graphs and obviously does not respect the clocking mechanisms of FCN circuits. In [3], this could be solved by proposing an extension of the edge coloring algorithm from [34] to assign information flow directions in an FCN layout properly. Additionally, the clocking scheme had to be restricted to limit the number of possible paths [35]. As a consequence, the resulting algorithm yields an FCN layout to a given 3-graph logic network (i. e. one that does not utilize Majority gates) in linear time, where the resulting layout size is bounded by a polynomial in the size of the network.

Table II provides evaluation results of larger functions than in the previous section. The benchmark networks were obtained from [30], [36]. These could not have been handled by other FCN physical design algorithms. The columns are organized in the same way as above: for each benchmark, the respective name, gate count and primary inputs and outputs are listed in the columns *Benchmark*. The resulting FCN layouts are characterized in the columns *Proposed solution*, namely the area in tiles, the critical path, and the runtime are given respectively. As it can be seen, the runtime is in the range of several seconds even for logic networks with almost 40 000 gates.

The benefit of this methodology comes at the cost of additional wires. However, contrary to classical CMOS, in FCN technologies, wire segments induce the same costs as logic gates in the layout. Therefore, in [4] a case study is conducted to evaluate their energy-impact using a physical model [16]. Calculations on the post-synthesis and the post-layout level have been performed. These confirmed that the layout step indeed has the most significant impact on the energy dissipation, too, as it almost always has to add wire elements to comply to the clocking constraints by e. g. balancing paths.

Therefore, when creating an FCN circuit layout from a logic network, the number of inserted wire segments may provide a realistic cost metric for future design algorithms.

Since wire segments are mainly used for overcoming clocking constraints, a different solution may be to tackle the clocking mechanisms directly. In [5], a novel method for clock generation in FCN technologies is proposed that allows one to overcome synchronization issues in previous FCN circuits. To this end, additional clock generators with elongated phases are proposed that are able to stall signals and thereby *imitate* the delay of multiple wire segments in a single tile. Thereby, utilizing these stalling clocks that have been named *synchronization elements* (SE), wire overhead and area in FCN circuit layouts can be reduced at the cost of decreased

TABLE II: Evaluating the approximate OGD solution

Benchmark			Proposed solution [3]		
Name	**#Gates**	**I / O**	**Dimension**	**CP**	**t in s**
c432	551	36 / 7	426 × 161	584	< 1
c499	963	41 / 32	690 × 306	995	< 1
c1355	1515	41 / 32	1243 × 369	1611	< 1
c1908a	2043	33 / 25	1540 × 536	2077	< 1
c2670a	2455	155 / 64	1756 × 760	2511	< 1
c3540a	3588	50 / 22	2523 × 1111	3639	1
c5315a	5478	177 / 123	3857 × 1751	5577	2
c6288	6928	32 / 32	5714 × 1246	6957	2
ctrl	498	7 / 25	356 × 149	495	< 1
router	658	60 / 3	488 × 231	717	< 1
int2float	699	11 / 7	514 × 196	708	< 1
i2c	3508	133 / 127	2515 × 1123	3632	1
bar	8592	135 / 128	6547 × 2180	8724	6
sin	14314	24 / 25	10549 × 3828	14374	14
voter	39476	1001 / 1	30542 × 9935	40476	10

#Gates Gate count plus fan-outs **CP** Critical path
Dimension Occupied area in tiles **t in s** Runtime in seconds

TABLE III: Evaluating the approximate SE solution

Benchmark				Proposed solution [6]		
Name	**#Gates**	**I / O**	**#FF**	**Dimension**	**#SE**	**t in s**
c432	551	36 / 7	—	96 × 91	867	27
c499	963	41 / 32	—	169 × 167	2909	160
c880	816	60 / 26	—	116 × 116	1861	79
c1355	1515	41 / 32	—	190 × 183	3637	277
c1908	1111	33 / 25	—	169 × 164	2949	161
ctrl	498	7 / 25	—	92 × 92	935	26
router	658	60 / 3	—	103 × 104	928	45
int2float	699	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

#Gates Gate count plus fan-outs **#FF** Flip-flop count
#SE Synchronization elements **t in s** Runtime in seconds

throughput. Physical simulations confirmed the applicability of the proposed novel clocking mechanism.

Finally, in [6], synchronization elements are exploited to make conventional VLSI methods applicable to FCN while at the same time significantly shrinking wire lengths. An experimental implementation was developed that used a *simulated annealing* [37] placement combined with a rip-up and reroute A^* [38] routing. Since this technique naturally caused several design rule violations regarding the FCN clocking constraints, synchronization elements are applied to the circuit layouts in a post-processing step to fix them. The probabilistic nature of the applied methods like simulated annealing could not guarantee to find a valid circuit layout in every single run. However, the application of conventional methods had the additional benefit of enabling the design of sequential FCN circuits which could not be realized by previous FCN-specific approaches.

Table III provides evaluation results in the same fashion as in the previous sections. The columns entitled *Benchmark* list the name of the used logic networks, the gate count including fan-out nodes, the number of primary input and outputs, and, additionally this time, the number of flip-flops respectively. The columns entitled *Proposed solution* shows the resulting circuit dimensions in tiles, the number of needed synchronization elements to fix clocking violations induced by the conventional design algorithms, and the runtime in seconds it took to obtain a solution. The combinational benchmark networks were again taken from [30], [36] and the sequential ones from [39].

Compared to the OGD solution [3], the obtained FCN circuit layouts are smaller in terms of area. However, it did take longer to compute them and the technique is based on the probabilistic simulated annealing which means that it is not guaranteed to yield a solution in every run. Additionally, the number of necessary synchronization elements grows fast with the network size, leading to an overhead in both the throughput and the external clock generator.

D. Formal Verification

The approaches presented so far generate FCN circuit layouts from specifications in terms of logic networks. However, there is naturally no guarantee, that the algorithm design and implementation are flawless. As a consequence, a designer might want to verify the correctness of the resulting layout, i. e. the preservation of the given logic function. Due to the rather complex clocking mechanisms in the FCN domain, pipeline-like behavior occurs in the circuits that can cause unexpected delay differences. These can ultimately lead to faulty outputs even if a circuit layout *appears* to be correct on a pure-logic level. This behavior renders conventional combinational equivalence checks non-applicable in the FCN domain.

In [7], a formal verification technique based on SAT and ILP solvers is introduced that deals with FCN clocking and thereby circumvents the discussed obstacles. To this end, a two-phase approach is proposed.

In the first phase, a conventional equivalence check on a *miter* structure of the specification logic network and the obtained FCN circuit layout is performed. In this phase, clocking and delay information is ignored as the pure-logic level is evaluated. If the primary outputs of the specification and the layout differ for any primary input, the two cannot be equivalent for any clocking. As a consequence, no further checking is required and it can be returned that the FCN circuit layout does not conduct the same functionality as its specification.

If on the other hand, the layout was found to be logically equal to its specification, the delay information is investigated. Therefore, every path on the layout is traversed and an ILP instance is created from the gathered structural information about their length, clocking, and possible synchronization elements. Eventually, one free variable for each primary input and output is introduced where it is enforced that the output values have to be equal to all path delays plus some unassigned value for the primary inputs.

Consequently, when passed to an ILP solver, one of three cases happens: (1) the solver returns `UNSAT`, i. e. the layout tremendously violates clocking and synchronization constraints so that the specified function will never actually be computed (extracting an UNSAT core can assist in debugging as it can point towards the location of failure), (2) the solver returns `SAT` with all primary input variables equal to 0, i. e.

TABLE IV: Evaluating the formal verification solution

Benchmark				Equivalent Cases			Non-equiv. Cases	
Name	**#Gates**	**I / O**	**Dimension**	**Equiv.**	$d(pi)$	**t in s**	**Equiv.**	**t in s**
2:1 MUX	5	3 / 1	3 × 4	Strong Eq.	0	< 1	Not Eq.	< 1
XOR	6	2 / 1	3 × 6	Strong Eq.	0	< 1	Not Eq.	< 1
XNOR	8	2 / 1	3 × 6	Strong Eq.	0	< 1	Not Eq.	< 1
Half adder	10	2 / 2	4 × 5	Strong Eq.	0	< 1	Not Eq.	< 1
ParGen	14	3 / 1	5 × 7	Strong Eq.	0	< 1	Not Eq.	< 1
4:1 MUX	16	6 / 1	7 × 8	Strong Eq.	0	< 1	Not Eq.	< 1
ParCheck	21	4 / 1	7 × 8	Strong Eq.	0	< 1	Not Eq.	< 1
c17	11	5 / 2	8 × 4	Weak Eq.	2	< 1	Not Eq.	< 1
c499	963	41 / 32	690 × 306	Weak Eq.	177	14	Not Eq.	< 1
c1355	1515	41 / 32	1243 × 369	Weak Eq.	225	22	Not Eq.	< 1
c1908	1111	33 / 25	852 × 400	Weak Eq.	133	14	Not Eq.	< 1
c3540	2997	50 / 22	2200 × 842	Weak Eq.	577	86	Not Eq.	< 1
ctrl	498	7 / 25	356 × 149	Weak Eq.	96	3	Not Eq.	< 1
int2float	699	11 / 7	514 × 196	Weak Eq.	182	6	Not Eq.	< 1
cavlc	2294	10 / 11	1666 × 638	Weak Eq.	301	50	Not Eq.	< 1
adder	3434	256 / 129	2541 × 1149	Weak Eq.	823	104	Not Eq.	< 1
i2c	3508	133 / 127	2515 × 1123	Weak Eq.	815	148	Not Eq.	< 1
bar	8592	135 / 128	6547 × 2180	Weak Eq.	2343	967	Not Eq.	< 1

#Gates Gate count plus fan-outs — **Dimension** Occupied area in tiles
$d(pi)$ Number of clock cycles to stabilize all signals — **t in s** Runtime in seconds

the layout is properly synchronized along every path and new signals can be applied at the primary inputs in every clock cycle, and (3) the solver returns `SAT` with some primary input variables not equal to 0, i. e. any signal must be applied to these inputs for a sequence of clock cycles equal to the variable's value in case to achieve equivalence (the throughput drops accordingly). Case (2) is called *strong equivalence* while case (3) is called *weak equivalence*.

Table IV provides evaluation results in the same fashion as in the previous sections. The exact and approximate OGD approach were utilized to generate (faulty) FCN circuit layouts which were then tested for equivalence against their logic network specification. The columns entitled *Benchmark* again list the name, the number of gates plus fan-outs, and primary inputs and outputs of the used logic networks that are the same as in the previous sections. This time, additionally, the dimension in tiles of the resulting layout is listed as part of the benchmark. The further columns distinguish between *Equivalent Cases* that contain the unchanged layouts as obtained by [2], [3] and *Non-equivalent Cases* that have been modified by inserting random faults, e. g. by changing gate functions.

E. Holistic Design Framework

All proposed approaches from the previous sections have been made publicly available in the extensible open-source framework *fiction* [8] that was developed for this purpose. The *fiction* framework thereby provides a complete flow for the physical design (i. e. synthesis, placement, routing, clocking/timing, verification, and debugging) of FCN circuits on different abstraction levels, e. g., the gate-level which is independent of any technological implementation and can be compiled down to cell-level by applying a gate library. Several algorithms from the literature have been re-implemented here too, e. g. *fan-out substitution* and *network balancing*. FCN circuit layout descriptions generated by *fiction* can also be directly exported to physic simulators [40], [41] and other design tools [42].

Utilizing the EPFL Logic Synthesis Libraries [43], *fiction* provides an *ABC*-like [44] shell interface for user interaction, parses established file formats like gate-level Verilog and AIGER, and provides powerful scripting and benchmarking features. Due to its extensibility for further FCN technologies, gate libraries, and physical design algorithms, *fiction* enables future research in the domain by providing an open platform. For this vision, *fiction* was decorated with the *Best Research Demo Award* at ISVLSI 2019.

III. Accomplishments

Within three years of enrollment in a Ph.D. program, several accomplishments have been made that are summarized in the following.

- Nine papers in the FCN domain have been published in conference proceedings and journals, namely DATE 2018 [2], DSD 2018 [4], NANO 2018 [5], ASP-DAC 2019 [3], JETC 2019 [1], ISVLSI 2019 [6], MICPRO 2020 [9], ISVLSI 2020 [10], DAC 2020 [7]. Two additional ones are currently under blind review.
- Cooperation with international researchers from Johannes Kepler University Linz, Austria, Universidade Federal de Minas Gerais, Brazil, Politecnico di Torino, Italy, and École polytechnique fédérale de Lausanne, Switzerland, was established.
- Interdisciplinary work combining the expertise from nanotechnology and computer science has been conducted.
- All findings from the papers above have been made accessible and easily reproducible as the open-source framework *fiction* [8].[1]
- The *fiction* framework was decorated with the *Best Research Demo Award* at ISVLSI 2019.
- Alongside the work in the FCN domain, papers in the fields of *Quantum Computing* and *Reversible Logic* have been published at ASP-DAC [11] and DATE [12] respectively. Another one in the field of *Machine Learning* is currently under blind review.

IV. Conclusion

Saving power in computational devices might be a critical step towards a greener, more sustainable future. Field-coupled Nanocomputing (FCN) enables to conduct binary in-memory computations with tremendously low energy dissipation and is a promising post-CMOS candidate concept.

In this Student Research Forum Proposal, design automation techniques from the domain of physical design for FCN circuit layouts have been proposed that enable to map conventional logic descriptions like AIGs to FCN layouts while complying with the complex technological constraints like clocking and synchronization.

To this end, the underlying theoretical problem was investigated first and proven to be intractable under the assumption $\mathcal{P} \neq \mathcal{NP}$ for various configurations that are inspired by actual design decisions. Second, exact and heuristic solutions to tackle the FCN physical design problem were proposed. The exact technique is based on incremental SMT solver calls and allows for design space exploration. A scalable approximate solution based on *Orthogonal Graph Drawing* was presented next. The scalability, however, comes at the cost

[1]The code is available at GitHub: https://github.com/marcelwa/fiction

of additional wire segments. By introducing novel clocking mechanisms that enabled *synchronization elements* to stall signals, the wire overhead could be reduced. Synchronization elements also made conventional VLSI methods applicable to FCN as demonstrated by a prototype implementation that uses a *simulated annealing* placer combined with an A^* router. To validate correctness of designed circuits, a formal verification technique based on SAT and ILP solvers was proposed that does not only perform combinational but also delay equivalence checks to incorporate the complex clocking and synchronization behavior. All presented approaches have been made publicly available in the open-source framework *fiction* that was developed to enable future research in the FCN domain.

These results were achieved in the course of three years of enrollment in a Ph.D. program, several works have been presented at venues like DATE, ASP-DAC, and DAC, and the resulting *fiction* tool was decorated with the *Best Research Demo Award* at ISVLSI 2019.

Own References

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

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

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

[4] 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.

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

[6] 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.

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

[8] M. Walter, R. Wille, F. Sill Torres, D. Große, and R. Drechsler, "fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits," 2019, arXiv:2002.10541.

[9] F. Sill Torres, P. A. Silva, G. Fontes, M. Walter, J. A. M. Nacif, R. S. Ferreira, O. P. Vilela Neto, J. F. Chaves, R. Wille, P. Niemann, D. Große, and R. Drechsler, "On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Aautomata," *MICPRO*, vol. 76, pp. 103–109, 2020.

[10] M. Walter, R. Wille, F. Sill Torres, and R. Drechsler, "Bail on Balancing: An Alternative Approach to the Physical Design of Field-coupled Nanocomputing Circuits," in *ISVLSI*, 2020.

[11] R. Wille, O. Keszöcze, M. Walter, P. Röhrs, A. Chattopadhyay, and R. Drechsler, "Look-ahead Schemes for Nearest Neighbor Optimization of 1D and 2D Quantum Circuits," in *ASP-DAC*, 2016, pp. 292–297.

[12] R. Wille, O. Keszöcze, S. Hillmich, M. Walter, and A. Garcia-Ortiz, "Synthesis of Approximate Coders for On-chip Interconnects Using Reversible Logic," in *DATE*, 2016, pp. 1140–1143.

Other References

[13] A. S. G. Andrae and T. Edler, "On Global Electricity Usage of Communication Technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.

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

[15] 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.

[16] 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.

[17] 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.

[18] C. S. Lent *et al.*, "Molecular Cellular Networks: A non von Neumann Architecture for Molecular Electronics," in *ICRC*, 2016, pp. 1–7.

[19] H. N. Chiu, S. S. H. Ng, J. Retallick, and K. Walus, "PoisSolver: a Tool for Modelling Silicon Dangling Bond Clocking Networks," 2020, arXiv:2002.10541.

[20] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum Cellular Automata," *Nanotechnology*, vol. 4, no. 1, p. 49, 1993.

[21] C. S. Lent, B. Isaksen, and M. Lieberman, "Molecular Quantum-dot Cellular Automata," *Journal of the American Chemical Society*, vol. 125, no. 4, pp. 1056–1063, 2003.

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

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

[24] M. Vacca *et al.*, "Feedbacks in QCA: A Quantitative Approach," *TVLSI*, vol. 23, no. 10, pp. 2233–2243, Oct 2015.

[25] 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*, May 2018, pp. 1–5.

[26] A. S. LaPaugh and R. L. Rivest, "The Subgraph Homeomorphism Problem," pp. 133 – 149, 1980.

[27] S. Fortune, J. Hopcroft, and J. Wyllie, "The Directed Subgraph Homeomorphism Problem," *Theoretical Computer Science*, vol. 10, no. 2, pp. 111–121, 1980.

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

[29] L. De Moura and N. Bjørner, "Z3: An Efficient SMT Solver," in *TACAS/ETAPS*, Berlin, Heidelberg, 2008.

[30] 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.

[31] V. Vankamamidi, M. Ottavi, and F. Lombardi, "Two-Dimensional Schemes for Clocking/Timing of QCA Circuits," *TCAD*, vol. 27, no. 1, pp. 34–44.

[32] 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.

[33] A. Trindade *et al.*, "A placement and routing algorithm for Quantum-dot Cellular Automata," in *SBCCI*, 2016.

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

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

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

[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. 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.

[40] 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.

[41] F. Riente, U. Garlando, G. Turvani, M. Vacca, M. R. Roch, and M. Graziano, "MagCAD: Tool for the Design of 3-D Magnetic Circuits," *IEEE J. Explor. Solid-State Computat.*, vol. 3, pp. 65–73, 2017.

[42] F. Riente *et al.*, "ToPoliNano: A CAD Tool for Nano Magnetic Logic," *TCAD*, vol. 36, no. 7, pp. 1061–1074, 2017.

[43] M. Soeken, H. Riener, W. Haaswijk, and G. De Micheli, "The EPFL Logic Synthesis Libraries," 2018, arXiv:1805.05121.

[44] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *CAV*, 2010, pp. 24–40.