Efficient Multi-Path Signal Routing for Field-coupled Nanotechnologies

Marcel Walter
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
marcel.walter@tum.de

Robert Wille*
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
robert.wille@tum.de

ABSTRACT

Establishing itself among the vanguard of beyond-CMOS candidates, *Field-coupled Nanocomputing* (FCN) has advanced in recent times due to fabrication breakthroughs of *Silicon Dangling Bonds* (SiDBs). At the foundation of these breakthroughs, experimental demonstrations showcase the feasibility of FCN logic components and wire segment implementations at the physical limits of scaling. However, automatic design methods for this highly-promising technology remain scarce, as they are impeded by the necessity to conform to particular constraints that differ from those in CMOS technologies. Previously proposed approaches are restricted by their inability to overcome scalability limitations and/or their failure to generate results of adequate quality. In this work, we aim to improve this state of the art by addressing the epicenter of performance inadequacy and proposing a distinctive multi-path FCN routing algorithm that is explicitly adjusted to the design constraints dictated by FCN technologies. The resulting approach can be parameterized to generate signal routings for almost arbitrary FCN placements or, in case this is impossible, pinpoint the designer to the unsatisfied connections. Experimental evaluations confirm these abilities on an established benchmark set and demonstrate a runtime advantage of several orders of magnitude over a state-of-the-art physical design algorithm.

CCS CONCEPTS

• **Hardware → Quantum dots and cellular automata**; **Wire routing**; **Physical synthesis**; *Placement*; *Clock-network synthesis*.

1 INTRODUCTION & MOTIVATION

Field-coupled Nanocomputing (FCN) [5] serves as an umbrella term for a class of beyond-CMOS emerging nanotechnologies that do not rely on electrical current flow but on the repulsion of physical fields to transmit information and conduct computations. FCN technologies promise energy dissipation capabilities below the *Landauer Limit* [20, 21, 24] or clock frequencies in the terahertz regime [26, 36] while inherently possessing logic-in-memory abilities.

In recent times, the FCN domain has advanced from being a mainly theoretical concept to an experimentally proven aspirant for a beyond-CMOS era of computation. In particular, the fabrication breakthroughs of *Silicon Dangling Bonds* (SiDBs) [1, 2, 4, 14, 17–19, 29, 31, 47] and their commercialization, e. g., by the research enterprise *Quantum Silicon Inc.* that secured multi-million dollar funding, recently confirmed their potential. SiDBs are atomically-sized entities that can act as quantum dots and have successfully been used to implement FCN logic components and wire segments with footprints below 30 nm^2 using coupled pairs of SiDBs; a scheme dubbed *Binary Dot Logic* (BDL) [17]. Furthermore, physical simulations of SiDBs [28] enabled the creation of various gate libraries [7, 27, 40, 46] as well as adaptations of the *Quantum-dot Cellular Automata* (QCA) [22] concept; enabling a transfer of knowledge from that domain while offering greater flexibility at its core.

In addition to their differences in physical properties, FCN and CMOS technologies vary greatly in their design constraints that must be obeyed when generating circuit layouts from specifications. The main differences in this physical design stage are planarity, clocking, and signal synchronization constraints [15, 34, 35]. While these hardly affect the gate placement (which can be handled rather similarly to CMOS and, hence, is already well-researched and understood; cf. [3] for an overview), they particularly result in routing problems—requiring complex wirings that quickly congest. Consequently, the physical design (and, here, routing in particular) is substantially harder for FCN than for the conventional realm [34, 35, 42].

However, despite these different levels of intricacy, established physical design methods for FCN, e. g., [10, 38, 41, 43], have considered placement *and* routing as a single holistic problem thus far—leading to an explosion in the search space and, hence, amplifying the problem's intractability. Accordingly, these algorithms are not scalable [10, 38, 41, 45] or of such low result quality that they cannot be realistically applied for the obtainment of practical circuitry [43]. If it were possible to separate the gate placement from the wire routing part—as it is done in conventional physical design—, the complexity of each stage would become manageable.

In this paper, we propose a solution to decouple the peculiar signal routing from the gate placement. Instead, we focus our attention on the actual epicenter of performance inadequacy in the physical design of FCN, namely, the multi-path signal routing problem under technology constraints. To explicitly address this problem, we adopt techniques that are established in the conventional domain (namely graph coloring and SAT solving) but, at the same time, explicitly adjust them to FCN's constraints.

Thereby, a physical design methodology results that offers multiple advantages over established solutions, namely it (1) decouples gate placement from the peculiar FCN wire routing, (2) is freely parameterizable to cover the entire spectrum between completeness and scalability, (3) is able to generate signal routings for almost arbitrary FCN placements or, in case this is impossible, pinpoint the designer to the unsatisfied connections, and (4) is independent of the layout's clocking floorplan and topology, i. e., can be used for arbitrarily-clocked Cartesian, triangular, or hexagonal grids.

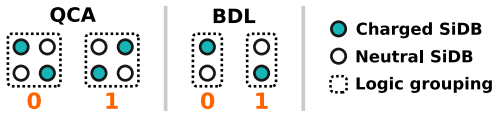
Experimental evaluations confirm the applicability of the proposed approach and show that it outperforms a state-of-the-art physical design approach in terms of runtime by several orders of magnitude, while being on par in terms of result completeness. Furthermore, when sacrificing result completeness, the proposed approach is able to generate routings for larger layouts that are outside the realm of what the state-of-the-art algorithm can handle.

The remainder of this paper is structured as follows: in an effort to keep this paper self-contained, Section 2 reviews the basics of FCN and the current state of the art of their physical design. Section 3 introduces the general idea of the proposed multi-path FCN routing algorithm, while Section 4 provides the corresponding implementation details. In Section 5, we summarize the results of an experimental evaluation. Finally, Section 6 concludes the paper.

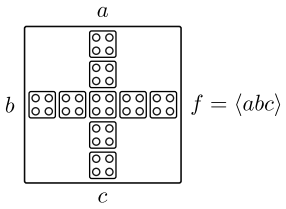
2 PRELIMINARIES & RELATED WORK

In an effort to keep the work at hand self-contained, this section reviews FCN preliminaries as well as the current state of the art of their physical design.

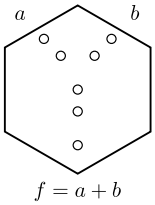
*Also with Software Competence Center Hagenberg (SCCH).



(a) QCA and BDL elementary cells.



(b) QCA Majority gate [37].



(c) BDL OR gate [17].

Figure 1: Elementary FCN devices and logic gates.

2.1 Field-coupled Nanocomputing

The basic unit of binary information representation in FCN technologies is the *cell* [5]. FCN cells are implemented differently across technologies but share some fundamental properties: (1) when observed, a cell is either in the binary 0 or binary 1 state, (2) logic states are linked to the exhibition of a physical field, i. e., an electric or magnetic one that reflects the state, and (3) adjacent cells are coupled through their fields which causes them to align their states accordingly [5, 8, 22, 23, 25]. In other words, in the presence of an electric charge, a charged-based FCN cell could be disturbed to change its state [22]; an effect that can be propagated through further adjacent cells. This way, information transmission without the flow of electric current becomes possible [25].

Example 1. *Figure 1a exemplarily depicts two types of charge-based FCN cells, namely QCA [22] and BDL [17], which can both be fabricated from SiDBs [1, 14, 18, 29]. While a QCA cell is composed of four quantum dots that are arranged at the corners of a square, a BDL cell requires just two dots. In both cells,* Coulomb *interaction causes exactly one of two possible charge distributions when the cell is provided with one charge per two quantum dots. These charge distributions are interpreted as the aforementioned states and are labeled binary* 0 *and binary* 1 *respectively [17, 22].*

The careful spatial arrangement of FCN cells on the surface of a substrate yields logic gates and wire segments that process information via the same field interactions [17, 25, 37]. In fact, a plethora of gate and wire segment implementations for various FCN technologies have been proposed in the literature, e. g., [11, 32, 37, 46]. As examples, Figure 1b and Figure 1c depict a QCA Majority gate and a BDL OR gate, respectively, with their input and output signals highlighted. In both cases, the inputs exert Coulombic pressure on the gate's center which causes the output cells to align their polarization in accordance and, consequently, to conduct the specified computation. A general restriction in this regard is FCN's *planarity* that constrains all gates and wires to be placed in the 2D plane with only short wire-wire crossings available [37].

Furthermore, for reasons of signal stability and information flow direction, FCN circuits of a certain size must be subdivided into uniform tiles that are periodically activated and deactivated by external fields [15, 25]. This concept is called *clocking* with the external coupling fields being the *clocks*. Clocking has a different purpose in FCN technologies than it has in CMOS. Both combinational and sequential FCN circuits must be clocked for the aforementioned reasons. The black outlines framing the gates depicted in Figure 1b and Figure 1c represent the clock tiling [16]. In the case of QCA, square tiles that are oriented in a Cartesian grid are utilized, while BDL employs a hexagonal tiling [46].

The default clock system for QCA circuitry involves four consecutive clock signals, which are numbered 1 to 4, of length $\frac{\pi}{2}$ each, resulting in a full 2π *clock cycle* [15, 25]. Assigning adjacent clock numbers to adjacent tiles enables a pipeline-like information flow where signals are propagated from tiles controlled by clock 1 to ones controlled by clock 2, to 3, to 4, and finally back to 1 again, each of which taking 1 timestep [15, 25]. This results in more obstacles that require careful attention: signal *propagation* and *synchronization*. It must be ensured that adjacent tiles are consecutively clocked and that the length of all wires is balanced throughout the circuit to avoid delay differences and, consequently, desynchronization [34].

Clock signal distribution networks have been discussed in the literature with the consensus that clock signals can be transmitted via electrodes that are buried in the circuit's substrate [15, 33]. Various regular clock zone arrangements, which are referred to as *clocking schemes*, have been proposed, e. g., [9, 13, 39]. Respective *gate libraries* offer single-tile implementations of common logic functions and wire segments, e. g., [32, 46].

2.2 Physical Design

The task of *physical design* is to generate an FCN circuit layout from a logic-level specification by placing FCN gates onto clocking schemes and connecting them with wire segments such that the resulting layout is functionally equivalent to its specification. Commonly, these specifications are provided by means of *logic networks*, i. e., synthesized gate-level netlists. From the previous discussions, the following technology constraints arise for FCN physical design: (1) FCN circuit layouts are planar with very limited crossing capabilities, thus, wire routing quickly congests, (2) information flow directions dictated by the clocking scheme must be obeyed, and (3) wire lengths must be balanced throughout the layout to assure signal synchronization.

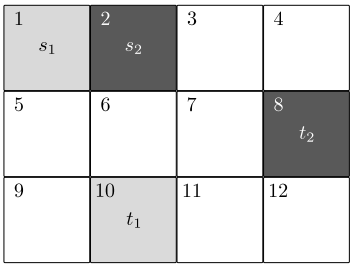
Since the circuit's clocking scheme inherently defines its information flow capabilities and directions, most established physical design methods are tailored toward one particular scheme only, e. g., [10, 38, 43]. To this end, these techniques consider gate placement and wire routing as a single interwoven problem with the goal to transfer knowledge about the processes from one step to the next, which, however, leads to an explosion in search space size [42, 45]. Nonetheless, the discussed technology constraints that must be obeyed in the FCN domain can be considered to be mainly routing-related. In fact, the gate placement merely differs from its conventional counterpart.

Notwithstanding, to the best of the authors' knowledge, an FCN routing algorithm that is decoupled from a placer and instead is able to determine the wiring for a given gate placement on an arbitrary clocking scheme and layout topology does not yet exist.

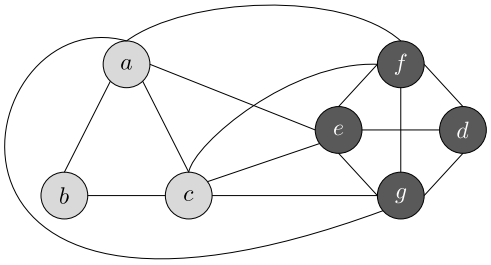
We define the multi-path routing problem on an FCN circuit layout with a placement given as its starting point. The inputs to the problem are a clocked partial FCN layout L_p—that may already contain some placed gates and wire segments—as well as a set of *routing objectives*[1] $R := \{(s_1 \mapsto t_1), (s_2 \mapsto t_2), \dots\}$ that specify source-target tile pairs to be connected. The output is an FCN circuit layout L obtained from L_p by additionally inserting wire segments to form connections between each specified source-target pair. In other words, a router has to find one connecting wire path per objective, i. e., source-target tile pair, that must not conflict with any other wire path or gate.

A naive approach to this task could come to mind, namely iterative *single-path* construction, i. e., creating one path at a time, and performing backtracking, i. e., *rip-up and re-route*, when a conflict occurs. However, not only could this procedure lead to an exponential runtime behavior, but it does also not guarantee the satisfaction of the FCN design constraints.

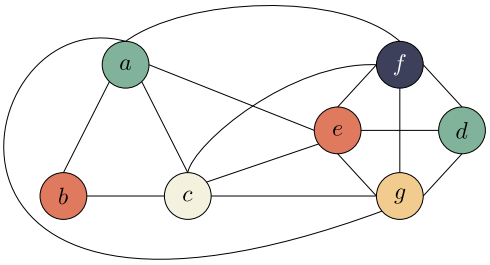
[1] This term is not to be confused with the *objective function* of an optimization problem.



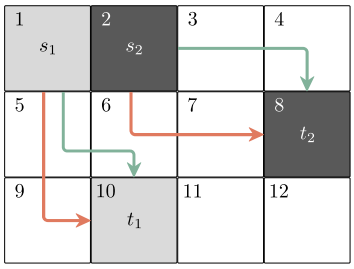
(a) The routing objectives are highlighted on the Cartesian layout in light and dark gray, respectively.



(b) EPG extracted from the routing objectives. Each clique is color-coded to match their objective from Figure 2a.



(c) A 5-coloring of the EPG. Each set of identically colored vertices of size 2 corresponds to a valid routing: $\{a, d\}$ (green) and $\{b, e\}$ (orange).



(d) Two valid routings have been extracted from the coloring (green and orange) of which either can be picked.

Figure 2: Two routing objectives $s_1 \mapsto t_1$ and $s_2 \mapsto t_2$ on a 4×3 Cartesian layout are getting satisfied via the proposed algorithm.

Thusly, in this work, we focus on *multi-path* FCN routing, i. e., the simultaneous construction of a plethora of paths that are aware of conflicts between each other. To this end, we tune our algorithms to inherently respect all technology constraints.

3 GENERAL IDEAS

Motivated by the state of the art review in the previous section, this work provides a solution that focuses on the generation of a multi-path FCN routing for a given placement—explicitly addressing the shortcomings in the physical design of FCN thus far. This section first sketches and illustrates the main ideas, before further details on the corresponding implementation are provided thereafter.

We propose encoding this problem as a graph coloring instance and applying a SAT solver (*complete* configuration) or coloring heuristic (*scalable* configuration) to determine solutions. To this end, the proposed algorithm consists of the following steps:

(1) **Enumerate potential paths for each routing objective:** Create a collection of feasible cycle-free wire segment connections between each source-target pair in the set of routing objectives. These can be enumerated with adjusted path finding algorithms that obey the layout topology and information flow directions dictated by the underlying clocking scheme while also taking blockades on the layout (due to already placed gates and routed wire segments) into account. It is necessary to generate a fair amount of potential paths to select from in the following steps.
(2) **Create an edge intersection graph from the enumerated paths:** To represent routing conflicts between the individual paths from step 1, an *edge intersection graph of paths on a grid* (EPG) [12] is generated. In this graph, each vertex represents a path and each edge between two vertices indicates an existing routing conflict.
(3) **Color the edge intersection graph:** Perform a vertex coloring such that no two adjacent vertices share the same color. The closer the number of utilized colors is to the chromatic number of the graph, the more routing objectives can be satisfied this way. We, thus, propose to apply a SAT solver for this step if complete solutions are to be ensured.
(4) **Extract a (partial) routing from the coloring:** All vertices of the same color correspond to a set of conflict-free paths. Consequently, the largest of such sets represents the one that satisfies the most routing objectives. These can be applied to the layout to obtain a (partial) routing. If all possible paths have been enumerated in step 1 and if a SAT solver has been utilized in step 3, the routing solution is *complete*, i. e., there exists no other set of paths that satisfies more routing objectives in R on L.

The following example illustrates this procedure.

Example 2. *Consider the empty 4×3 Cartesian layout in Figure 2a. Its tiles are numbered 1 to 12 and two routing objectives are highlighted as $s_1 \mapsto t_1$ and $s_2 \mapsto t_2$. It is assumed that the source and target tiles of any objective are blocked for all other routing attempts because there would naturally be a gate already placed at that position.*

It is possible to enumerate all cycle-free paths for each of the two objectives, which leads to the following two sets:
$s_1 \mapsto t_1$:

$$(1, 5, 6, 10) \quad \text{(a)}$$
$$(1, 5, 9, 10) \quad \text{(b)}$$
$$(1, 5, 6, 7, 11, 10) \quad \text{(c)}$$

$s_2 \mapsto t_2$:

$$(2, 3, 4, 8) \quad \text{(d)}$$
$$(2, 6, 7, 8) \quad \text{(e)}$$
$$(2, 6, 7, 3, 4, 8) \quad \text{(f)}$$
$$(2, 6, 7, 11, 12, 8) \quad \text{(g)}$$

Figure 2b depicts the corresponding EPG that can be constructed from these paths. The two cliques $\{a, b, c\}$ and $\{d, e, f, g\}$ belong to different objectives and are, thus, emphasized via matching shades of gray. Since all paths for the same objective have to share at least their first and last tile by definition, they are all mutually exclusive in the solution space. Hence, they form cliques in the EPG. Any edges between cliques indicate conflicting paths amid objectives.

Hereinafter, we are looking for any set of non-conflicting vertices equal in size to the number of routing objectives (here 2). A coloring of the EPG reveals precisely that, namely, as can be seen in Figure 2c, two colors have been used twice, i. e., green for the set $\{a, d\}$ and orange for the set $\{b, e\}$. Therefore, we have found two valid solutions to the initial routing problem, which are depicted in Figure 2d.

Using such a procedure allows generating a complete routing for arbitrary FCN placements in case one exists if an optimal coloring is found. This can be guaranteed via the application of SAT solvers since they entirely cover the respective search space. Moreover, even if no complete routing is possible, e. g., because of a suboptimal placement and conflicting objectives, the proposed procedure still proves to be beneficial. Since it is inherently aware of all potential conflicts, it can optimize for the subset of paths that successfully connects the most source-target pairs conflict-free. This, in turn, pinpoints the designer (or corresponding placement methods for that matter) to objectives that could not be realized—allowing them to accordingly adjust the placement until a working solution has been determined.

The following section elaborates on each of the proposed algorithm's steps in greater detail.

4 IMPLEMENTATION DETAILS

The following sections shed light on individual aspects of the proposed multi-path FCN routing algorithm. Section 4.1 explains how routing candidates, i. e., potential partial solutions, are enumerated; Section 4.2 highlights the use of a special kind of edge intersection graph of paths within the algorithm; Section 4.3 goes into details on the utilized graph coloring technique; finally, Section 4.4 discusses how routing solutions are extracted from a coloring.

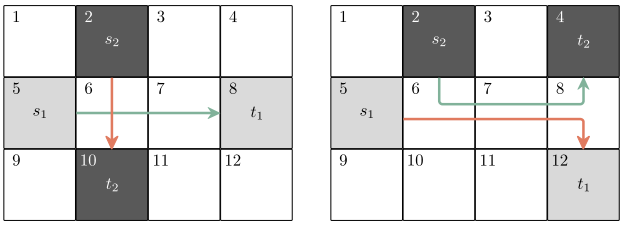
4.1 Enumerating Routing Solutions

For the enumeration of routing solutions for the problem in question, we are particularly interested in the shortest possible paths for each routing objective because shorter connections equal lower area requirements and signal delays.

Hence, the intuitive solution to the routing problem that comes to mind could be to determine only the shortest path for each objective via the A* algorithm. This approach has the apparent benefit of low complexity but reaches its limits fairly early on: if only one possible route for each objective exists, there is no alternative connection available when conflicts inevitably occur.

Alternatively, it could appear tempting to enumerate *all* potential paths for each routing objective to be guaranteed to have a solution available if one exists. While this approach works fine for small to medium-sized layouts, it is also limited in its applicability due to the huge complexity that arises when large amounts of paths have to be enumerated and processed in the subsequent steps. Additionally, just because a path is *possible* does not mean it is *desirable*. Exhaustively enumerating the solution space would include the consideration of obscure, high-cost routings which would not be part of any realistic layout in the overwhelming majority of cases. As the distances between source and target tiles grow, the number of these "redundant" connections increases rapidly.

Instead, we propose to explore a middle ground by enumerating the *k shortest paths* via *Yen's algorithm* [48] where k is a value that can be passed as a parameter making the search more or less exhaustive by trading-off completeness and runtime.



(a) Legal wire crossing. **(b) Illegal routing conflict.**

Figure 3: Wire crossing vs. routing conflict.

4.2 Edge Intersection Graph of Paths

In [12], the general notion of an *edge intersection graph of paths*—which was usually defined on a tree—was extended to the *edge intersection graph of paths on a grid* (EPG). Given a set of paths $\mathcal{P}$ of coordinates on a grid (interpreted as tiles of layouts for the use case in this work), $EPG(\mathcal{P}) = (V, E)$ is defined as an undirected graph with $V := \mathcal{P}$ and $E := \{\{u, v\} \mid u \cap v \neq \emptyset\}$ where $u, v \in \mathcal{P}$. Intuitively, each vertex in the edge intersection graph represents a path from the set $\mathcal{P}$ and two vertices share an edge iff their respective paths share at least one coordinate, i. e., layout tile.

The notion of two paths sharing layout tiles is, in this domain, to be interpreted as routing conflicts. Hence, the EPG encodes all enumerated paths and their pairwise conflicts.

However, as formerly established, FCN does permit wire crossings as a technology feature which means that it is possible to relax the definition of routing conflicts. If only single coordinates are shared along two paths, they are not mutually exclusive. This realization is illustrated in Figure 3a, where a legal wire crossing of two paths is depicted. Thereby, we can adjust the definition of the edge set E of EPGs in order to exclude edges between crossing paths but preserve them between conflicting ones, i. e., only create edges between paths that share both their source and target coordinate or more than one *consecutive* coordinate. More formally expressed, $E := \{\{u, v\} \mid (u_* = v_* \wedge u_\dagger = v_\dagger) \vee \exists i, j \in \mathbb{N} : u_i = v_j \wedge u_{i+1} = v_{j+1}\}$ where $u, v \in \mathcal{P}$ and p_* denotes the first, $p_\dagger$ the last, and p_n the nth element of path p. The difference between a legal wire crossing and an illegal routing conflict is depicted in Figure 3.

While being mentioned in Section 3, it is imperative to further discuss that the EPG arising from these routing objectives possesses a particular structure: since all paths that belong to the same routing objective are mutually exclusive by definition (since they share at least their source and target tiles), their respective vertices form a clique, i. e., a complete subgraph, in the EPG. Hence, the EPG is a collection of $|R|$ cliques, one for each routing objective. In case there are no possible paths available for an objective, its respective clique is empty.

4.3 EPG Vertex Coloring

For the multi-path routing problem discussed in this work, a graph coloring of the EPG represents valuable information: since edges in the EPG represent routing conflicts, any two vertices that receive the same color can be routed together conflict-free. Moreover, in a colored EPG that uses the minimum amount of colors (χ-coloring), the largest set of vertices with equal colors represents the largest possible set of non-conflicting paths, i. e., the wire routing that satisfies the most objectives.

It is possible to apply a heuristic in this step, but no maximality of the routing set's cardinality can be guaranteed this way, thus, mostly partial routings would result even if complete routings were possible. The determination of χ-colorings require the application of an exhaustive search for which we apply a SAT solver. A SAT

formulation for the graph coloring problem greatly benefits from the knowledge of cliques generated in the previous step as they can be used for strong symmetry breaking.

4.4 Routing Extraction

Any vertices in the EPG that are assigned the same color can be routed together conflict-free. Since, by definition, no two vertices within a clique can have the same color, these vertices must belong to different cliques and, thus, different routing objectives. The largest subset of vertices of identical color represents the largest set of objectives that can be simultaneously satisfied. In case multiple solution sets exist in the EPG coloring, the respective solutions can be selected based on secondary optimization criteria like signal delay or bounding box size.

If all possible paths have been enumerated and if the coloring is minimal, e. g., determined by a SAT solver, then it is guaranteed to have found a complete solution. Otherwise, it might be partial. However, even if the coloring is minimal, cases are possible in which not all objectives could be satisfied because no satisfying routings exist, to begin with. The benefit of the proposed approach is to generate partial solutions even in those cases whereas other SAT-based techniques would simply return `UNSAT`. Furthermore, it is precisely known which objectives caused issues such that the corresponding source or target gates can be moved with the confidence that there existed no solution that would have allowed their routability.

In other words, the algorithm provides insight into necessary adjustments to the input placement. This allows for a quick convergence toward a fully satisfiable layout.

5 EXPERIMENTAL EVALUATION

In this section, the results of experimental evaluations are summarized which investigated the practical applicability of the proposed multi-path routing algorithm. To this end, we first discuss the setup that we applied for all subsequent experiments in Section 5.1. Afterward, we compare the obtained results against the state of the art in Section 5.2 and we investigate the proposed algorithms' parameterizability in terms of scalability vs. completeness in Section 5.3.

5.1 Setup

The proposed algorithm was implemented in C++ for the open-source FCN framework *fiction* and made publicly available on GitHub.[2] The utilized SAT solver is *Glucose 4.1* [6]. For all experiments, established QCA benchmarks from [10, 38] were used for reference. All evaluations in the following sections were run on a Manjaro 21 machine with an AMD Ryzen 7 PRO 5850U CPU with 1.90 GHz (up to 4.40 GHz boost) and 32 GB of DDR4 main memory.

5.2 Comparison Against the State of the Art

In this section, we demonstrate the flexibility and evaluate the completeness of the proposed multi-path routing algorithm via comparison against the most generic state-of-the-art physical design algorithm [41].

Table 1 contains placements of the aforementioned benchmark functions on various clocking schemes—namely *2DDWave* [39], *USE* [9], and *RES* [13]—that we know to be routable such that all objectives are satisfied simultaneously. Each clocking scheme exhibits different information flow possibilities, making it algorithmically challenging to determine solutions for any given input layout. In fact, as mentioned in Section 2, most algorithms are tuned to a single clocking scheme only and cannot produce any results on different ones. The correctness of obtained solutions is validated via formal verification [44].

Altogether, the recorded runtime data prove that the state-of-the-art algorithm struggles with the USE and RES clocking schemes. In contrast, the proposed approach is able to generate complete wire routings—as it consistently experiences zero unsatisfied objectives—for all given benchmarks on all clocking schemes in virtually zero time and, thereby, outperforms the state of the art by several orders of magnitude.

Table 1: Comparison against the state of the art

Benchmark					Physical Design		
					SOTA[a]	Proposed	
	Name	#Obj.[b]	Clocking	$w \times h = A$	t in s	t in s	#UNSAT[c]
[38]	par_gen	21	2DDWAVE	$4 \times 8 = 32$	0.74	0.00	0
		21	USE	$4 \times 8 = 32$	8.48	0.00	0
		21	RES	$4 \times 8 = 32$	5.30	0.00	0
	HA	16	2DDWAVE	$4 \times 6 = 24$	0.39	0.00	0
		16	USE	$4 \times 7 = 28$	4.21	0.00	0
		14	RES	$7 \times 6 = 42$	8.58	0.00	0
	FA	16	2DDWAVE	$4 \times 7 = 28$	0.54	0.00	0
		16	USE	$4 \times 6 = 24$	1.71	0.00	0
		16	RES	$4 \times 6 = 24$	1.29	0.00	0
	par_check	31	2DDWAVE	$6 \times 8 = 48$	2.76	0.00	0
		31	USE	$5 \times 11 = 55$	4179.12	0.00	0
		30	RES	$8 \times 7 = 56$	265.14	0.00	0
[10]	1_bit_add_aoig	30	2DDWAVE	$5 \times 11 = 55$	3.16	0.00	0
		30	USE	$5 \times 10 = 50$	4573.72	0.00	0
		30	RES	$8 \times 7 = 56$	914.98	0.00	0
	t	22	2DDWAVE	$5 \times 6 = 30$	0.72	0.00	0
		22	USE	$6 \times 6 = 36$	106.27	0.00	0
		22	RES	$4 \times 8 = 32$	7.53	0.00	0
	t_5	22	2DDWAVE	$5 \times 6 = 30$	0.78	0.00	0
		22	USE	$6 \times 6 = 36$	128.29	0.00	0
		22	RES	$4 \times 8 = 32$	9.67	0.00	0
	c17	19	2DDWAVE	$4 \times 7 = 28$	0.51	0.00	0
		19	USE	$5 \times 7 = 35$	17.20	0.00	0
		19	RES	$7 \times 5 = 35$	9.15	0.00	0
	b1_r2	26	2DDWAVE	$5 \times 8 = 40$	3.29	0.00	0
		26	USE	$5 \times 9 = 45$	907.26	0.00	0
		25	RES	$11 \times 5 = 55$	136.48	0.00	0
	majority_5_r1	30	2DDWAVE	$4 \times 11 = 44$	3.18	0.00	0
		30	USE	$6 \times 8 = 48$	1586.57	0.00	0
		30	RES	$7 \times 8 = 56$	1138.82	0.00	0
	clpl	21	2DDWAVE	$2 \times 16 = 32$	0.67	0.00	0
		21	USE	$2 \times 21 = 42$	152.97	0.00	0
		21	RES	$2 \times 19 = 38$	18.16	0.00	0

[a] State-of-the-art physical design algorithm [41].
[b] Number of routing objectives, i. e., source-target pairs.
[c] Number of unsatisfied objectives.

5.3 Tradeoff Between Scalability and Completeness

In the second part of the evaluation, we select more challenging layouts to route from the same benchmark set and compare the proposed algorithm in two configurations: *complete* and *scalable*. Hereby, *complete* refers to the same configuration that was applied in the previous section, namely utilizing a full path enumeration and applying a SAT solver for the coloring step; *scalable* makes two adjustments: (1) it lowers the k parameter to not enumerate all possible paths anymore, and (2) applies a heuristic for the coloring step. Without loss of generality, we chose $k = 75$ and applied the *MCS* coloring algorithm [30].

Table 2 lists the collected data. It can be seen that the proposed approach in its complete configuration starts to experience some higher runtime values and also times out on four layouts before determining a solution. When switched over to the scalable configuration, however, runtime consistently falls below one second again at the expense of completeness as the algorithm could not find a full routing for every layout anymore.

However, the proposed algorithm pinpoints the designer (or corresponding placement methods for that matter) to objectives that could not be realized—allowing them to accordingly adjust the placement until a working solution has been determined.

6 CONCLUSIONS

Most established physical design methods for FCN have considered gate placement *and* wire routing as a single interwoven problem thus

[2] https://github.com/marcelwa/fiction

Table 2: Tradeoff between scalability and completeness

	Benchmark				Proposed Physical Design			
					Complete		Scalable	
	Name	#Obj.[a]	Clocking	$w \times h = A$	t in s	#UNSAT[b]	t in s	#UNSAT[b]
	mux21	9	2DDWAVE	6 × 7 = 42	0.01	0	0.00	1
	xor2	10	2DDWAVE	5 × 7 = 35	0.00	0	0.00	0
	xnor2	12	2DDWAVE	6 × 8 = 48	0.00	0	0.00	0
[38]	par_gen	21	2DDWAVE	9 × 13 = 117	0.05	0	0.00	1
	HA	16	2DDWAVE	9 × 8 = 72	0.00	0	0.00	0
	FA	16	2DDWAVE	8 × 10 = 80	0.00	0	0.00	1
	par_check	31	2DDWAVE	12 × 19 = 228	16.27	0	0.02	7
	xor	11	2DDWAVE	6 × 7 = 42	0.00	0	0.00	0
	1_bit_add_aoig	30	2DDWAVE	12 × 18 = 216	TO	–	0.03	4
	t	22	2DDWAVE	10 × 16 = 160	5.70	0	0.01	5
	t_5	22	2DDWAVE	10 × 16 = 160	0.67	0	0.01	1
[10]	c17	19	2DDWAVE	10 × 13 = 130	1.19	0	0.02	1
	b1_r2	29	2DDWAVE	13 × 17 = 221	0.11	0	0.00	1
	majority	30	2DDWAVE	9 × 24 = 216	TO	–	0.05	6
	majority_5_r1	30	2DDWAVE	10 × 23 = 230	1.21	0	0.02	6
	newtag	29	2DDWAVE	12 × 25 = 300	TO	–	0.21	3
	clpl	29	2DDWAVE	17 × 25 = 425	TO	–	0.89	4

[a]Number of routing objectives, i. e., source-target pairs.
[b]Number of unsatisfied objectives.

far. As the consequence, this consideration leads to deliberate scalability issues. However, it is the wire routing that specifically suffers from peculiar technology constraints like planarity, information flow directions, and signal balancing. In this work, we presented an efficient multi-path routing algorithm for FCN technologies that is decoupled from gate placement. This approach is freely parameterizable to tradeoff runtime and result completeness and is independent of the utilized clocking scheme and layout topology.

Experimental evaluations confirmed the applicability of the proposed algorithm. On an established benchmark set of layouts on the three most common clocking schemes, it consistently outperformed the state of the art by several orders of magnitude in terms of runtime while being able to generate *complete* wire routings. On a set of larger benchmarks that are mostly outside the realm of the state-of-the-art algorithm, the proposed approach demonstrated its ability to significantly speed up its runtime by sacrificing result completeness. While this may lead to incomplete solutions, the unsatisfied routing objectives can be adjusted accordingly until a complete solution has been determined.

Thereby, the multi-path routing algorithm proposed in this work has the potential to become an integral part of future FCN design automation flows.

ACKNOWLEDGMENTS

We would like to express our gratitude toward Bella Gardner for her insightful remarks on the SAT formulation and the manuscript.

REFERENCES

[1] R. Achal et al. 2018. Lithography for Robust and Editable Atomic-Scale Silicon Devices and Memories. *Nature Communications* 9, 1 (July 2018), 2778.
[2] R. Achal et al. 2019. Detecting and Directing Single Molecule Binding Events on H-Si (100) with Application to Ultradense Data Storage. *ACS Nano* 14, 3 (2019), 2947–2955.
[3] C. J. Alpert et al. 2008. *Handbook of Algorithms for Physical Design Automation.* CRC Press.
[4] F. Altincicek. 2022. Atomically Defined Wires on P-Type Silicon. *Bulletin of the American Physical Society* (2022).
[5] N. G. Anderson et al. 2014. *Field-coupled Nanocomputing: Paradigms, Progress, and Perspectives.* Springer, New York.
[6] G. Audemard et al. 2009. Predicting Learnt Clauses Quality in Modern SAT Solvers. In *IJCAI*. Citeseer.
[7] A. N. Bahar et al. 2020. Atomic Silicon Quantum Dot: A New Designing Paradigm of an Atomic Logic Circuit. *Transactions on Nanotechnology* 19 (2020), 807–810.
[8] G. H. Bernstein et al. 2005. Magnetic QCA systems. *Microelectronics Journal* 36, 7 (2005), 619–624.
[9] C. A. T. Campos et al. 2016. USE: A Universal, Scalable, and Efficient Clocking Scheme for QCA. *TCAD* 35, 3 (2016), 513–517.
[10] G. Fontes et al. 2018. Placement and Routing by Overlapping and Merging QCA Gates. In *ISCAS*.
[11] D. Giri et al. 2014. A Standard Cell Approach for MagnetoElastic NML Circuits. In *NANOARCH*. 65–70.
[12] M. C. Golumbic et al. 2009. Edge Intersection Graphs of Single Bend Paths on a Grid. *Networks: An International Journal* 54, 3 (2009), 130–138.
[13] M. Goswami et al. 2020. An efficient clocking scheme for quantum-dot cellular automata. *International Journal of Electronics Letters* 8, 1 (2020), 83–96.
[14] M. B. Haider et al. 2009. Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature. *Physical Review Letters* 102, 4 (2009), 046805.
[15] K. Hennessy and C. S. Lent. 2001. Clocking of Molecular Quantum-dot Cellular Automata. *Journal of Vacuum Science & Technology B* 19, 5 (2001), 1752–1755.
[16] J. Huang et al. 2005. Tile-based QCA Design Using Majority-like Logic Primitives. *JETC* 1, 3 (2005), 163–185.
[17] T. Huff et al. 2018. Binary Atomic Silicon Logic. *Nature Electronics* 1 (2018), 636–643. Issue 12.
[18] T. R. Huff et al. 2017. Atomic White-Out: Enabling Atomic Circuitry Through Mechanically Induced Bonding of Single Hydrogen Atoms to a Silicon Surface. *ACS Nano* 11, 9 (Sept. 2017), 8636–8642.
[19] T. R. Huff et al. 2019. Electrostatic Landscape of a Hydrogen-Terminated Silicon Surface Probed by a Moveable Quantum Dot. *ACS Nano* 13, 9 (2019), 10566–10575.
[20] R. W. Keyes et al. 1970. Minimal Energy Dissipation in Logic. *IBM Journal of Research and Development* 14, 2 (1970), 152–157.
[21] R. Landauer. 1961. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development* 5, 3 (1961), 183–191.
[22] C. S. Lent et al. 1993. Quantum Cellular Automata. *Nanotechnology* 4, 1 (1993), 49.
[23] C. S. Lent et al. 2003. Molecular Quantum-Dot Cellular Automata. *Journal of the American Chemical Society* 125, 4 (2003), 1056–1063.
[24] C. S. Lent et al. 2006. Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling. *Nanotechnology* 17, 16 (2006), 4240–4251.
[25] C. S. Lent and P. D. Tougaw. 1997. A Device Architecture for Computing with Quantum Dots. *Proc. IEEE* 85, 4 (April 1997), 541–557.
[26] L. Livadaru et al. 2010. Dangling-bond charge qubit on a silicon surface. *New Journal of Physics* 12, 8 (Aug. 2010), 083018.
[27] R. Lupoiu et al. 2022. Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning. *arXiv preprint arXiv:2204.06288* (2022).
[28] S. S. H. Ng et al. 2020. SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits. *TNANO* 19 (2020), 137–146.
[29] N. Pavliček et al. 2017. Tip-induced Passivation of Dangling Bonds on Hydrogenated Si(100)-2×1. *Applied Physics Letters* 111, 5 (2017), 053104.
[30] F. M. Q. Pereira et al. 2005. Register Allocation via Coloring of Chordal Graphs. In *Asian Symposium on Programming Languages and Systems*. Springer, 315–329.
[31] Mohammad Rashidi, Jeremiah Croshaw, and Robert Wolkow. 2022. Automated atomic scale fabrication. US Patent App. 17/429,443.
[32] D. A. Reis et al. 2016. A Methodology for Standard Cell Design for QCA. In *ISCAS*. 2114–2117.
[33] J. Retallick et al. 2021. Low-Energy Eigenspectrum Decomposition (LEED) of Quantum-Dot Cellular Automata Networks. *TNANO* 20 (2021), 104–112.
[34] F. Sill Torres et al. 2018. Synchronization of Clocked Field-Coupled Circuits. In *IEEE-NANO*.
[35] F. Sill Torres et al. 2020. On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Automata. *Microprocessors and Microsystems* 76 (2020), 103–109.
[36] J. Timler et al. 2002. Power Gain and Dissipation in Quantum-dot Cellular Automata. *Journal of Applied Physics* 91, 2 (2002), 823–831.
[37] P. D. Tougaw and C. S. Lent. 1994. Logical devices implemented using Quantum Cellular Automata. *Journal of Applied Physics* 75, 3 (1994), 1818–1825.
[38] A. Trindade et al. 2016. A Placement and Routing Algorithm for Quantum-dot Cellular Automata. In *SBCCI*.
[39] V. Vankamamidi et al. 2006. Clocking and Cell Placement for QCA. In *IEEE-NANO*, Vol. 1. IEEE, 343–346.
[40] M. D. Vieira et al. 2022. Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots. *IEEE Design & Test* (2022).
[41] M. Walter et al. 2018. An Exact Method for Design Exploration of Quantum-dot Cellular Automata. In *DATE*. 503–508.
[42] M. Walter et al. 2019. Placement & Routing for Tile-based Field-coupled Nanocomputing Circuits is $\mathcal{NP}$-complete. *Journal on Emerging Technologies in Computing Systems (JETC)* 15, 3 (2019), 10 pages.
[43] M. Walter et al. 2019. Scalable Design for Field-coupled Nanocomputing Circuits. In *ASP-DAC*. ACM New York, NY, USA, 197–202.
[44] M. Walter et al. 2020. Verification for Field-coupled Nanocomputing Circuits. In *DAC*.
[45] M. Walter et al. 2021. One-pass Synthesis for Field-coupled Nanocomputing Technologies. In *ASP-DAC*. ACM New York, NY, USA, 574–580.
[46] M. Walter et al. 2022. Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic. In *DAC*, Vol. 22.
[47] R. A. Wolkow et al. 2014. *Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics.* Springer, 33–58.
[48] Jin Y. Yen. 1970. An Algorithm for Finding Shortest Routes from all Source Nodes to a Given Destination in General Networks. *Quart. Appl. Math.* 27, 4 (1970), 526–530.