Versatile Signal Distribution Networks for Scalable Placement and Routing of Field-coupled Nanocomputing Technologies

Marcel Walter*, Benjamin Hien*, and Robert Wille*†

*Chair for Design Automation, Technical University of Munich, Germany
Email: {marcel.walter, benjamin.hien, robert.wille}@tum.de
https://www.cda.cit.tum.de/research/fcn/
†Software Competence Center Hagenberg GmbH, Austria

***Abstract*—*Field-coupled Nanocomputing* (FCN) is a promising beyond-CMOS technology that leverages physical field repulsion instead of electrical current flow to transmit information and perform computations, potentially leading to energy dissipation below the *Landauer Limit* and clock frequencies in the terahertz regime. Despite recent progress in the experimental realization of FCN using *Silicon Dangling Bonds* (SiDBs), the physical design of FCN circuits remains a challenging task due to different design constraints compared to CMOS technologies. In this paper, we present three core contributions to the FCN physical design problem, building on top of the fastest heuristic algorithm in the FCN literature, *ortho*. Via special routing structures called *Signal Distribution Networks* (SDNs), we 1) reduce area overhead, wire costs, and the number of wire-crossings in routing solutions by approximately $25\,\%$, $10\,\%$, and $17\,\%$, respectively; 2) allow the use of Majority gates to quantify their routing costs, which occur to be immense; and 3) enable the automatic placement and routing of sequential logic for the first time in the literature. Our approach can potentially pave the way for the practical implementation of the FCN technology and its advancement as a viable green alternative to conventional computing technologies.**

I. Introduction & Motivation

Field-coupled Nanocomputing (FCN) [1] refers to a category of beyond-CMOS nanotechnologies that leverage physical field repulsion instead of electrical current flow to transmit information and perform computations. This approach offers the potential for energy dissipation below the *Landauer Limit* [2]–[4] and clock frequencies in the terahertz regime [5], [6], while also incorporating logic-in-memory capabilities. FCN technologies have the potential to significantly enhance energy efficiency and computational performance compared to conventional CMOS technologies. These unique properties have made FCN an attractive area of research for the development of next-generation computing systems.

The field of FCN has made significant progress in recent years, moving from being a purely theoretical concept to an experimental reality. This progress has been driven by the fabrication breakthroughs of *Silicon Dangling Bonds* (SiDBs) and their commercialization [7]–[16]. SiDBs are atomically-sized entities that can act as quantum dots and have been used to implement FCN logic components and wire segments with footprints below $30\,\mathrm{nm}^2$ using the *Binary Dot Logic* (BDL) approach [11]. Physical simulations of SiDBs have enabled the development of various gate librariess [17]–[21] and adaptations of the *Quantum-dot Cellular Automata* (QCA) concept [22], providing a transfer of knowledge from the QCA domain while offering greater flexibility. The recent multi-million dollar funding secured by research enterprise *Quantum Silicon Inc.* confirms the potential of SiDBs and FCN technologies for future computing systems.

In addition to their distinct physical properties, FCN and CMOS technologies have different design constraints that must be followed during the circuit layout generation process. The main differences in this stage are planarity, clocking, and signal synchronization constraints [23]–[25]. These differences result in complex and highly interdependent placement and routing problems that can quickly lead to congestion. This makes the physical design of FCN circuits much more challenging than their conventional CMOS counterparts. The complexity of placement and routing in FCN technologies has been extensively documented in the literature [24]–[26].

Nevertheless, several exact and heuristic physical design algorithms for the FCN domain have been proposed in the literature [27]–[31]. While exact algorithms provide optimality guarantees for their produced layouts, their exponential runtime overhead limits their usage to rather small circuits. On the other hand, heuristic algorithms are able to process large specifications and generate circuit layouts quickly. However, their result quality often lacks behind the respective optimal solutions by several orders of magnitude [28], [32]. Additionally, heuristics are mostly specialized in handling a certain kind of circuitry only, e. g., purely combinational or low-input degree (Majority-free) logic [27], [29], [30], [33]. In fact, to the best of the authors' knowledge, no algorithm—exact or heuristic—for placement and routing of sequential logic has yet been proposed in the literature. Furthermore, established heuristic algorithms have no way of controlling or limiting the amount of wire-crossings they produce—a critical measure for most FCN systems as they are costly to fabricate and lead to unstable signal propagation and sensitivity to environmental factors.

To overcome these limitations, this work presents three core contributions to the FCN physical design problem. On top of *ortho* [30], the fastest heuristic placement and routing algorithm in the FCN literature, we introduce *Signal Distribution Networks* (SDNs) to

1) reduce area overhead, wire costs, and the number of wire-crossings in routing solutions by approximately $25\,\%$, $10\,\%$, and $17\,\%$, respectively;
2) allow for the use of Majority gates, leading to an initial quantification of their routing overhead, which occur to be immense; and
3) enable the automatic placement and routing of sequential logic for the first time, making it possible to design and implement FCN-based sequential circuits.

These contributions have the potential to pave the way for the practical implementation of FCN technology and to drive its advancement towards becoming a viable alternative to conventional computing technologies.

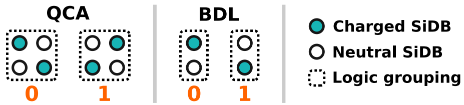
The remainder of this article is structured as follows: Section II discusses preliminaries and related work in the domain of field-coupled nanotechnologies, their fabrication, and physical design. The main contributions regarding signal distribution networks and their advantages over the state of the art are outlined in Section III. To demonstrate the applicability of the proposed method, an experimental study is conducted in Section IV. Finally, Section V summarizes and concludes this work.

II. Preliminaries & Related Work

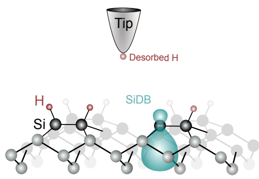
This section provides an overview of the foundational concepts of FCN technologies and delves into the existing state of the art in its physical design methods. The aim of this section is to present the necessary background information to fully appreciate the contributions and innovations described in the subsequent sections of this paper. This work aims to be self-contained and, thus, provides a brief review of FCN in Section II-A and its current physical design landscape in Section II-B.

A. Field-coupled Nanocomputing

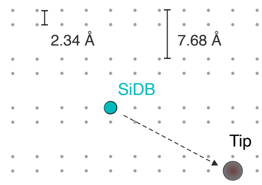
The central component of computation and information representation in FCN technologies is referred to as a *cell* [1]. Despite variations in implementation among FCN technologies, cells possess certain fundamental characteristics: 1) they exhibit two distinct states when observed that can be labeled binary 0 and binary 1, respectively; 2) these logic



(a) QCA and BDL elementary cells fabricated from SiDBs.



(b) SiDB fabrication process on the H-Si(100)-2×1 lattice (side view).



(c) The H-Si(100)-2×1 lattice with atom distances specified (top view).

Fig. 1: Elementary FCN devices and SiDB fabrication.

states are associated with the manifestation of a physical field, such as an electric or magnetic field, reflecting the state; and 3) adjacent cells are coupled through their fields, leading to an alignment of their states [1], [22], [34]–[36]. In other words, the presence of an electric charge can disturb a charged-based FCN cell and cause it to change its state [22], a phenomenon that can then be transmitted through further adjacent cells, resulting in logic-in-memory behavior and information transmission without the flow of electric current [34].

Example 1. *Figure 1a presents two variations of charge-based FCN cells, namely* Quantum-dot Cellular Automata *(QCA) [22] and* Binary Dot Logic *(BDL) [11]. Both cell types can be created via SiDB fabrication [7]–[10] and are composed of quantum dots. QCA and BDL cells differ in the number of quantum dots used. In the case of QCA, each cell consists of four quantum dots placed at the corners of a square shape. On the other hand, a BDL cell only requires two dots. Both cells exhibit two possible charge distributions when supplied with one charge per two dots, which are then interpreted as binary states, either* 0 *or* 1*, due to the Coulomb interaction. This interpretation is established in previous works on QCA [22] and BDL [11].*

It is to be noted that FCN circuitry is usually restricted to one certain type of elementary cell. While it is not impossible—and in fact has been explored in recent works [37]—it is uncommon to mix FCN fabrication styles. Instead, a circuit would for instance either be implemented as a QCA or a BDL layout.[1]

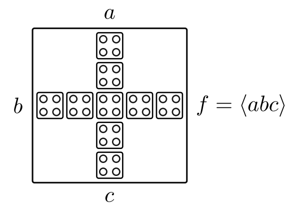
The creation of SiDBs is achieved through the use of a *Scanning Tunneling Microscope* (STM) applied to a *Hydrogen-passivated Silicon* (H-Si) surface. The STM voltage severs the bond between a silicon and hydrogen atom, leading to the desorption of the hydrogen atom and the formation of an open valence bond.

Example 2. *The fabrication process on the surface is depicted in Figure 1b and involves the movement of the STM tip to each successive silicon dimer to generate the next SiDB as seen in Figure 1c.*

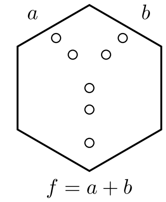
Each SiDB acts as a chemically identical quantum dot and its energy levels of the charge transition are within the band gap, making their states stable unless disturbed by other charges. In n-doped systems, SiDBs tend to be negatively charged and retain their quantum dot character, properties that are leveraged to create logic elements such as gates and wire segments. The first experimental demonstration of a working sub-$30\,\mathrm{nm}^2$ OR gate and multiple wire segments using SiDBs were achieved by Huff *et al.* [11].

FCN logic gates are composed of elementary cells on a surface and operate through the same field interactions as individual cells [11],

[1]Several other FCN implementations such as *Nanomagnet Logic* (NML) [36] and *molecular QCA* (mQCA) [35] have been proposed in the literature. Due to the brevity of this work, we refer the interested reader to the respective primary sources.



(a) QCA MAJ gate [38].



(b) BDL OR gate [11].

Fig. 2: FCN logic gates.

[34], [38]. Several gate and wire segment implementations have been proposed for different FCN technologies, such as [17], [18], [20], [38], [39]. Figure 2a and Figure 2b illustrate a QCA Majority (MAJ) gate and a BDL OR gate, respectively, with their input and output signals annotated. The inputs in these examples apply Coulombic pressure to the gates' centers, causing the output cells to align their polarization and perform the specified computation.

B. Physical Design

Composing an FCN layout from elementary gates and connecting them with wire segments to yield a circuit that is functionally equivalent to a given logic-level specification is called *physical design*. The specification is usually provided in the form of gate-level netlists, resulting from a preceding logic synthesis flow.

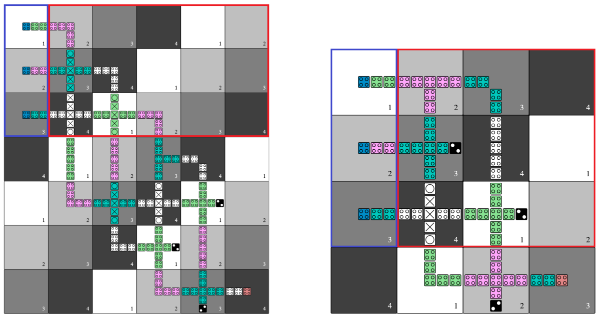
Several technology constraints enforce restrictions on the FCN circuit layouts generated this way. Most FCN technologies are planar with limited crossing capabilities, making it challenging to route wires without congestion. Furthermore, the lengths of such wire segments must be balanced throughout the layout to guarantee signal synchronization. Most importantly however, to ensure signal stability and regulate the direction of information flow, FCN circuits must be partitioned into uniform regions that are activated and deactivated at periodic intervals by external fields [23], [34].

This mechanism is referred to as *clocking*, which is a critical aspect of every FCN implementation, as both combinational and sequential circuits must be clocked to maintain signal stability and control information flow direction. The black outlines surrounding the gates shown in Figure 2a and Figure 2b represent the clock partitioning, which employs square tiles in the case of QCA and hexagonal tiles in the case of BDL [17], [40].

In the default clocking system, four consecutive clock signals are used, numbered from 1 to 4. The clocking system facilitates a pipeline-like flow of information where signals are transmitted from tiles that are under the control of clock 1 to those under clock 2, then clock 3, and finally clock 4, before cycling back to clock 1 [23], [34]. However, this creates challenges with respect to signal propagation and synchronization, which must be carefully managed to ensure that adjacent tiles are clocked consecutively, and that wire lengths are balanced throughout the circuit to prevent delay differences and subsequent desynchronization [24].

The distribution of a clock signal to each tile has been the subject of much discussion in the literature, with a general agreement that the clock signals can be transmitted through buried electrodes in the substrate of the circuit. A range of regular clock zone arrangements, referred to as *clocking schemes*, have been proposed, such as [41]–[43]. In support of clocking schemes, various standard gate libraries have been developed, providing single-tile implementations of common logic functions and wire segments, such as [17], [39].

Finally, several physical design algorithms enable the automatic obtainment of FCN circuit layouts from specifications using standard gate libraries and established clocking schemes while respecting all technology constraints; a problem that was proven to be $\mathcal{NP}$-complete [26]. As a consequence, existing solutions are either optimal in their result quality, but do not work on larger input specifications [28], [32], or



(a) 2:1 MUX layout of size 6 × 7 as yielded by *ortho* [30].

(b) An equivalent layout, but with ordered inputs, of size 4 × 4.

Fig. 3: The impact of input ordering. The blue highlighting marks primary inputs; red shows the area affected by their ordering.

are scalable to a high degree, but produce layouts that lack in key cost metrics [27], [29], [30].

To the best of the authors' knowledge, the most scalable algorithm for FCN physical design in the literature is *ortho* [30], which is based on an approximation of *Orthogonal Graph Drawing*. While this approach is able to automatically design layouts with several hundred million tiles, its result quality is sub-par in terms of area usage, wire lengths, and crossing count. Furthermore, it is not able to process 3-input MAJ gates or sequential logic.

In the following section, these shortcomings of the *ortho* algorithm are addressed as this work's main contribution.

III. Signal Distribution Networks

This section constitutes the main contribution of this paper. As outlined above, existing scalable placement and routing algorithms for FCN technologies are limited in their capabilities and result quality. We propose overcoming these limitations via the introduction of *Signal Distribution Networks* (SDNs) which are distinct wire segment arrangements within FCN layouts that aid in the proper routing of complex scenarios. We utilize SDNs on top of the existing *ortho* algorithm [30] discussed above. In the following, Section III-A presents an SDN structure with the ability to reduce wire crossings and area costs by applying a primary input ordering. Afterward, Section III-B proposes an SDN that can incorporate the more expressive MAJ gates; a feat that previous algorithms lacked and that sheds light on MAJ's routing costs. Finally, Section III-C conceptualizes an SDN that enables the automatic placement and routing of sequential FCN logic for the first time.

A. Input Ordering SDNs

In the existing *ortho* algorithm, the vertical order in which primary input (PI) pins are placed at the layout border is arbitrary. However, it can be easily shown that careful adjustment of the PI order leads to fewer wire crossings and less area overhead. This benefit is magnified when designing complex circuitry, as the wiring complexity grows exponentially with the number of gates. Proper PI ordering can help reduce wiring and minimize wire crossings, leading to simpler and more efficient designs.

Example 3. *Consider the QCA circuit layout of a simple 2:1 MUX function, that was automatically generated by the* ortho *algorithm, depicted in Figure 3a. It is of size 6×7 tiles and possesses 5 wire crossings. On the other hand, the layout shown in Figure 3b represents the same logic function and is only of size 4×4 tiles with a single wire crossing. This reduction has been achieved purely by altering the input permutation highlighted in blue.*

To delve into the concept at hand, the first step is to determine the portion of the logic network that is pertinent to the *Input Ordering SDN*. This can be accomplished by analyzing the tiles located in the input area, which comprise those directly linked to PIs as well as those connected to such (red areas in Figure 3). PIs that are on the fan-in cone of the same gate are placed consecutively, prioritizing the placement of PIs connected to fan-outs and minimizing routing distance, thereby reducing the likelihood of wire crossings. Once gates and wires are accurately positioned on these tiles, conflicts are resolved, enabling the *ortho* algorithm to operate normally from there. A schematic of resulting layouts is depicted in Figure 4a.

After the PIs have been reordered, the remaining logic network is topologically sorted, allowing the improvements derived from the ordering to affect the entire subsequent circuit, which can be processed by *ortho* without further modifications.

B. Majority SDNs

In logic synthesis, it is well known that the 3-input MAJ function is more expressive than common 2-input gates, i. e., logic networks of complex functions can be realized using fewer gates when exclusively relying on MAJ gates as elementary building blocks in contrast to, e. g., AND gates [44]. However, this benefit does not translate to CMOS, where no cheap transistor implementation of the MAJ function exists. In many FCN technologies, on the other hand, MAJ is an elementary function that can be realized on a single tile and only requires a few cells to be implemented (as discussed in Section II-A).

Thus far, this benefit of logic reduction has yet to be quantified in the physical design domain. Due to their higher input degree, MAJ gates cannot be natively processed by the *ortho* algorithm. Consequently, no large scale design studies using MAJ gates in FCN layouts exist yet.

This section addresses the placement and routing of majority gates in FCN circuits using the *ortho* algorithm. To this end, an SDN called *Majority SDN* is proposed, allowing for the placement and routing of majority gates within the scalable algorithm and enabling a quantitative comparison of design metrics between functionally equivalent layouts with and without MAJ gates.

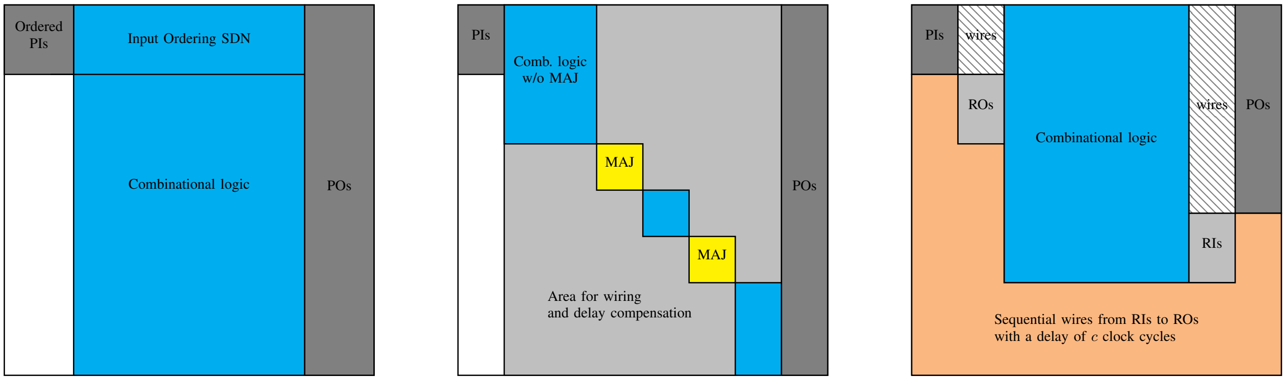
The *ortho* algorithm relies on a regular *2DDWave* clocking floorplan [41], which limits gates' input and output directions to only two options each: *north* and *west* for incoming signals, and *east* and *south* for outgoing ones. To introduce MAJ gates with three incoming signals into the layout, an *RES*-like clocking scheme [43] is necessary, which includes tiles with three incoming directions and one outgoing one.

However, changing the native clocking scheme of *ortho* to RES would be highly inefficient and difficult to implement, resulting in approximately double the area usage for sections of the circuit that exclusively utilize 2-input logic.

An alternative approach to benefiting from the higher degree that the RES scheme provides, is to only support it in certain evenly distributed regions, allowing MAJ gates to be placed in specific, permanently assigned locations. For example, the layout could be divided into 4×4 tile sub-regions, and every fifth sub-region would be RES clocked, while the rest would be occupied with the traditional 2DDWave scheme. On one hand, this approach should not produce as much area overhead since only some regions are inaccessible for 2-input logic gates. However, the permanent clocking assignment limits the placement of MAJ gates to specific locations, leading to larger overhead if a MAJ gate were to be placed distant from such a sub-region. For a specification consisting mainly of MAJ gates, this implementation would also waste most of the 2DDWave-clocked area. Another aspect to consider is that within a uniform 2DDWave-clocked layout signal synchronization is trivially satisfied; a feat that is disrupted by introducing RES clocking within the layout.

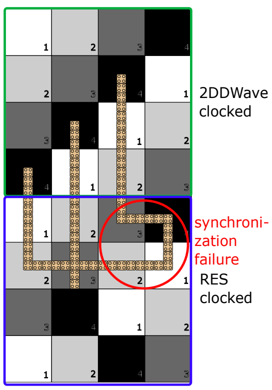
Example 4. *Figure 5a depicts a QCA layout, whose top four rows are 2DDWave-clocked, while the bottom four rows are RES-clocked. In the top 2DDWave part, all three wire paths are synchronized. The two left paths need exactly one clock cycle to reach the MAJ gate in clock zone 3. However, the rightmost path introduces a delay as soon as it arrives in the RES scheme, such that the signal travels for two clock cycles before reaching the MAJ gate, thereby violating signal synchronization.*

To overcome these complications, the proposed SDN uses a custom clocking scheme only in areas where MAJ gates are placed by *ortho* as

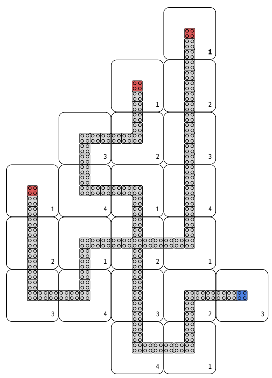


(a) Input Ordering SDN. (b) Majority SDN. (c) Sequential SDN.

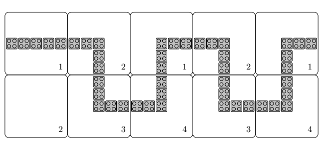
Fig. 4: Schematics of the proposed Signal Distribution Networks.



(a) Synchronization failure when stitching 2DDWave and RES together naively.



(b) Proposed Majority SDN with proper signal synchronization.



(c) Wire chain for delay compensation.

Fig. 5: Majority SDN.

illustrated schematically in Figure 4b. As a result, the placement and routing of 2-input gates does not produce any area overhead. Figure 5b illustrates the proposed *Majority SDN*. The red cells indicate the three inputs for the MAJ gate. The output marked blue allows the algorithm to wire it in the eastern or southern direction, eliminating any restrictions for subsequent gates. It is also important to note that the input tiles as well as the output tiles have the same clocking number as in a regular 2DDWave scheme, allowing for seamless integration. Although the proposed SDN does not produce any area overhead for the placement and routing of 2-input gates, it can be seen that it does produce excess area for MAJ gates due to its complex wiring, resulting from signal synchronization that it has to satisfy. However, no wire crossings are used in its design. Assuming that wire crossings have a high fabrication cost, the proposed *Majority SDN* may be considered less costly than a functional decomposition into 5 AND gates (plus inverters and fan-outs) which does require wire crossings. Nevertheless, a meaningful comparison of the two implementations can only be performed if informed by fabrication.

Finally, to compensate for the delay introduced by the MAJ gate, a compensation in form of a wire chain must be inserted on signal paths that converge with the MAJ output further down in the layout. The simplest form of such a structure can be seen in Figure 5c.

C. Sequential SDNs

To the best of the authors' knowledge, there is currently no solution for automatic placement and routing of sequential FCN circuitry available in the literature. While a limited number of manual approaches for addressing sequentiality do exist, these approaches are less than ideal, as they rely on the incorporation of an additional clock signal via FCN wires, a methodology that runs counter to the underlying clocking paradigm that is intrinsic to FCN. In light of these challenges, we put forth a novel proposal for a *Sequential SDN* that leverages the *ortho* algorithm to facilitate automatic placement and routing, all while properly taking into account the vital importance of signal synchronization.

Sequential circuits can be perceived as a combinational logic module that is interconnected to storage elements, such as *latches* and *flip-flops*. Corresponding to the primary inputs (PIs) and primary outputs (POs), the signals that feed information *into* the storage components are referred to as *register inputs* (RIs), while those transmitting information back *out of* the storage and into the combinational part are referred to as *register outputs* (ROs). However, unlike conventional circuitry, where information can be linked back from storage to the inputs of the combinational logic arbitrarily, i.e., from RIs to ROs, the wiring in FCN involves placing wire segments, each of which delays the information by one additional clock phase, acting in of itself as a partial flip-flop.

Contrary to the approach suggested in the extant literature, we propose a novel idea that obviates the requirement for additional sequential elements in FCN circuits, as the wire segments in themselves, by virtue of the delay they introduce, function as inherent storage elements. We therefore advocate the use of this natural delay, engendered by sequential wiring, to emulate storage elements, thereby curbing the need for bespoke clocking mechanisms and minimizing the accompanying area overhead.

The functional equivalence of PIs/ROs and ROs/RIs presents an opportunity for leveraging their similarities in the automatic placement and routing of sequential circuits. In this adaptation, once the combinational logic has been placed and routed, the ROs are treated as PIs, and the RIs as POs, with the exception that PIs and POs are positioned at the borders to ensure their external accessibility. Subsequently, routing from the RIs to the ROs must be established, while preserving signal synchronization. A resulting schematic of this process is depicted in Figure 4c.

In our proposed method, we first rewire and sort the RIs in the same order as the ROs. This ordering ensures that all RIs are placed on the same diagonal, and because the gates are uniformly clocked with 2DDWave, their signals are synchronized. Using this starting position, we then route wires of the same length between each RI-RO pair. However, finding the right wiring is not arbitrary since the wires must

TABLE I: Obtained results using *Input Ordering SDNs*.

	Benchmark Circuit				State of the Art [30]				Proposed Approach			
	Name	PI	PO	Gates	Area	Wires	Cross.	t in s	Area	Wires	Cross.	t in s
[45]	c17	5	2	8	130	70	11	<1	70	44	7	<1
	c432	36	7	414	96 928	36 025	4352	<1	84 942	36 139	3577	<1
	c499	41	32	816	392 256	89 974	5824	<1	289 980	79 103	5353	<1
	c880	60	26	639	245 344	70 312	8659	<1	185 148	66 082	8110	<1
	c1355	41	32	1064	580 944	111 966	6215	<1	454 314	103 758	5512	<1
	c1908	33	25	813	381 060	99 968	9001	<1	287 471	89 571	8047	<1
	c2670	233	64	1463	1 343 280	323 106	32 710	1.05	1 040 175	304 108	22 145	1.55
	c3540	50	22	1987	2 037 028	448 336	42 795	1.41	1 758 488	436 795	39 638	2.33
	c5315	178	123	3628	7 409 772	1 658 707	120 748	5.3	5 897 752	1 602 367	90 543	8.21
	c6288	32	32	6467	15 065 444	847 982	33 758	3.15	7 599 620	705 183	34 994	6.1
	c7552	207	107	4501	10 331 370	2 258 129	186 432	7.59	7 938 000	2 028 163	137 000	11.08
[46]	dec	8	256	320	317 656	161 537	6871	<1	194 788	103 010	6772	<1
	ctrl	7	25	409	92 214	27 263	2667	<1	61 623	22 867	2149	<1
	router	60	3	490	143 149	53 419	7954	<1	112 699	51 103	5402	<1
	int2float	11	7	545	145 580	47 469	4870	<1	127 218	44 749	4199	<1
	cavlc	10	11	1600	1 097 544	283 873	25 100	1.44	932 576	263 297	23 565	1.29
	priority	128	8	2349	2 454 192	665 069	57 918	3.3	1 729 644	592 805	38 685	2.81
	adder	256	129	2541	3 577 363	790 224	83 316	4.33	2 398 532	859 067	82 809	4.47
	i2c	136	127	2728	4 407 440	1 087 757	94 027	5.71	3 419 640	1 028 301	80 798	5.21
	max	512	130	6110	20 644 180	5 320 238	651 642	30.12	18 334 810	4 851 155	351 515	28.1
	bar	135	128	6672	23 452 764	3 981 858	363 230	25.85	16 360 140	3 514 868	285 585	20.91
	sin	24	25	11 437	60 004 375	8 216 004	514 313	75.72	44 001 198	7 010 038	469 482	49.25
Average relative change (lower is better)									**-25.39%**	**-10.18%**	**-17.04%**	**+9.81%**

close the loop between the RIs in the bottom-right corner and the ROs in the upper left corner of the layout. We thus perform this routing in a dedicated sequential wiring area that is southern to the placed combinational logic.

Another obstacle that arises is that clocking cannot be selected independently for each back loop, as these loops may cross each other. Therefore, we clock this sequential wiring area with a 2DDWave scheme that is rotated by 180° and, thus, enables wires to run in the opposite direction. This approach does introduce a large delay that is due to the fact that the *ortho* algorithm generates the combinational logic only in the south-eastern direction, increasing the distance between PIs/ROs and POs/RIs. This delay grows with the size of the combinational logic, ultimately deteriorating the circuit's throughput.

Further research may include developing a folding operation for the *ortho* algorithm to decrease the distance between RIs and ROs, reducing the delay introduced by this SDN.

IV. Experimental Evaluations

In this section, the results of experimental evaluations are summarized, which investigated the applicability of the proposed SDNs. To this end, we first discuss the setup that we applied for all subsequent experiments in Section IV-A. Afterward, we first compare the quantitative impact of *Input Ordering SDNs* against the state of the art in Section IV-B. Subsequently, we investigate the placement and routing costs of MAJ gates using the proposed *Majority SDNs* Section IV-C. Finally, we present first results of sequential FCN layouts by applying the proposed *Sequential SDNs* in Section IV-D.

A. Experimental Setup

The proposed SDNs were individually implemented in C++ on top of *ortho* in the open-source FCN framework *fiction* and made publicly available on GitHub.[2] As benchmarks to place and route, the established *ISCAS85* [45] and *EPFL* [46] combinational circuits were utilized for the comparison of *Input Ordering SDNs* against the state of the art. For the evaluation of *Majority SDNs*, randomly generated *Majority-inverter graphs* (MIGs) [44] of varying sizes were considered due to the lack of standardized MAJ benchmark functions. For a comparison against the state of the art, they were decomposed into *And-inverter graphs* (AIGs). Finally, for the evaluation of *Sequential SDNs*, the *PoliTo ITC99* benchmarks from [47] were applied. All evaluations in the following sections were run on a Windows 11 machine with an AMD Ryzen 5 PRO 3500U CPU with 2.10 GHz (up to 3.70 GHz boost) and 16 GB of DDR4 main memory.

B. Input Ordering SDNs

This section compares the effects of primary input ordering on layout characteristics using key cost metrics. We evaluate the state-of-the-art algorithm, *ortho*, and our custom adaptation using *Input Ordering SDNs* on a common benchmark set. The resulting layout data is presented in Table I. The table includes three sections: 1) *Benchmark Circuit* with the benchmark name, primary input/output counts, and number of gates including fan-outs and inverters; 2) *State of the Art* with the layout characteristics for *ortho*; 3) *Proposed Approach* with the layout characteristics for our adaptation. The layout characteristics compared are the bounding box area in number of tiles, number of wire segments, number of wire crossings, and runtime in seconds.

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

TABLE II: Obtained results using *Majority SDNs*.

Benchmark Circuit					State of the Art [30]				Proposed Approach			
Name	PI	PO	Gates	MAJ	Area	Wires	Cross.	t in s	Area	Wires	Cross.	t in s
r01	3	2	10	5	1856	709	92	<1	11 439	1803	58	<1
r02	4	3	10	7	2160	731	92	<1	20 570	2712	56	<1
r03	6	11	40	30	43 884	11 838	1205	<1	449 320	37 240	750	<1
r04	10	28	100	93	382 200	100 222	8479	<1	4 796 737	378 818	4498	3.75
r05	10	51	200	190	1 510 875	371 272	31 667	1.88	20 993 856	1 525 594	15 967	18.2
r06	10	26	90	85	321 290	79 105	6803	<1	3 254 776	256 686	3592	2.43
r07	23	71	277	267	3 108 966	744 287	63 270	3.88	41 437 414	2 976 412	31 276	32.11
r08	16	86	340	324	4 538 956	1 050 366	86 195	5.65	59 912 010	4 199 694	40 406	49.83
r09	5	21	72	61	162 679	40 424	3907	<1	1 597 944	131 478	2059	1.37
r10	8	22	88	83	284 715	74 636	6861	<1	3 052 612	253 936	3349	2.38
r11	12	47	149	140	882 640	221 230	19 547	1.16	12 289 965	913 040	8905	10.26
Average relative change (lower is better)									**+1022.70%**	**+257.38%**	**-46.75%**	**+1153.83%**

The ultimate advantages of the proposed approach can be effectively demonstrated through a summary of the relative changes in the four key metrics, which are listed in the last row. By employing *Input Ordering SDNs*, the average reduction in the resulting circuit layout area is 25.39 %. Additionally, the number of wire segments was decreased by 10.18 %, and the number of crossings by 17.04 %. While on average the runtime increased slightly by 9.81 %, in absolute numbers, it can be seen that on larger benchmarks the runtime mainly decreases (as a result of the creation of comparably smaller layouts), leading to a net benefit over all benchmarks.

Notable data include *c6288*, which achieved a remarkable area reduction of 49.56 %, *dec*, which achieved a significant decrease in wire segments by 36.23 %, and *max*, which demonstrated a considerable reduction in crossing count by 46.05 %. These remarkable findings highlight the efficacy of the proposed methodology, which achieves substantial improvements in important cost metrics through a simple yet ingenious modification to the existing algorithm.

C. Majority SDNs

This section presents an evaluation of the routing costs of 3-input MAJ gates in a large-scale scenario, which, to the best of the authors' knowledge, has not been previously investigated. Due to the unavailability of suitable MAJ benchmark circuits, this evaluation is conducted on randomly generated logic networks consisting of MAJ, AND, and OR gates. To be passed to the original *ortho* algorithm, the logic networks are decomposed into AIGs (where each MAJ gate is decomposed into 5 AND nodes plus inverters and fan-outs). In contrast, the proposed adaptation employs true MIG input. The obtained data is summarized in Table II, which has a similar structure as the table in the previous section. However, an additional *MAJ* column is included in the *Benchmark Circuit* section, which specifies the number of 3-input MAJ gates as a subset of all gates (excluding MAJ gates with constant inputs, which are equivalent to AND or OR gates).

As expected, the layout overhead for *Majority SDNs* is substantial despite the more concise logic representation of the input, resulting in an increase in the total area by over 1000 %, the number of wire tiles by almost 260 %, and the runtime by over 1150 % on average. The number of crossings, however, drops significantly by 46.75 %. Thus, it can be concluded that although MAJ is a highly expressive function that allows for more compact logic representation, these reductions in gate count do not translate to layout improvements, but rather the opposite. However, if crossing costs are determined to be more important than area, then it may be worthwhile to trade crossings for area by employing MAJ primitives.

D. Sequential SDNs

This section reports initial results of the automatic placement and routing of sequential FCN logic using the proposed *Sequential SDNs*. As there is currently no comparative material available, we present the obtained data without a state-of-the-art comparison. The layout characteristics are shown in Table III, where the same cost metrics as in

TABLE III: Obtained results using *Sequential SDNs*.

Benchmark Circuit [47]					Proposed Approach			
Name	PI	PO	Gates	Reg.	Area	Wires	Crossings	t in s
b01	2	2	127	5	7704	2837	226	<1
b02	1	1	68	4	2856	1214	127	<1
b03	4	4	420	30	132 352	47 514	4421	<1
b04	11	8	1866	66	1 516 008	440 650	28 841	<1
b05	1	36	2636	34	1 894 548	366 310	19 518	<1
b06	2	6	143	9	12 750	4660	425	<1
b07	1	8	1149	49	641 762	196 220	15 163	<1
b08	9	4	462	21	115 326	39 367	3428	<1
b09	1	1	426	28	124 465	44 597	4619	<1
b10	11	6	549	17	130 077	47 939	4546	<1
b11	7	6	1718	31	889 949	261 924	18 781	<1
b12	5	6	2854	121	3 945 978	1 195 053	74 904	<1
b13	10	10	878	53	510 540	174 757	14 746	<1
b14	32	54	24 900	245	156 416 376	26 976 087	876 527	134.95

the previous sections are specified. Additionally, the *Benchmark Circuit* section contains a *Reg.* column that indicates the number of sequential registers that each circuit contains.

Notably, it can be observed that the layout costs of the sequential circuits are generally higher than those of the comparable combinational circuits evaluated in Section IV-B. This is unsurprising, as the additional back wiring requires a significant portion of layout area and a large number of wires that increase with both the number of registers and the depth of the circuit.

Further research may include developing a folding operation to decrease the distance between RIs and ROs, reducing the delay and area introduced by this SDN.

V. Conclusions

In this paper, we presented three core contributions to the FCN physical design problem. We built upon the fastest heuristic algorithm in the FCN literature, *ortho* [30], and introduced *Signal Distribution Networks* (SDNs) to reduce area overhead, wire costs, and the number of wire-crossings by approximately 25 %, 10 %, and 17 %, respectively. We further enabled the use of MAJ gates in large-scale layouts to quantify their routing costs, which turn out to be immense. Finally, our SDNs enabled the automatic placement and routing of sequential logic for the first time in the literature. These advances bring practical implementation of FCN technology closer to reality and open up exciting possibilities for the future of design automation in the domain. In adherence to the principles of open science, the implementation has been made publicly available on GitHub.

References

[1] N. G. Anderson *et al.*, *Field-coupled Nanocomputing: Paradigms, Progress, and Perspectives*. New York: Springer, 2014.

[2] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[3] R. W. Keyes *et al.*, "Minimal Energy Dissipation in Logic," *IBM Journal of Research and Development*, vol. 14, no. 2, pp. 152–157, 1970.

[4] C. S. Lent *et al.*, "Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling," *Nanotechnology*, vol. 17, no. 16, pp. 4240–4251, 2006.

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

[6] L. Livadaru *et al.*, "Dangling-Bond Charge Qubit on a Silicon Surface," *New Journal of Physics*, vol. 12, no. 8, p. 083018, 2010.

[7] M. B. Haider *et al.*, "Controlled Coupling and Occupation of Silicon Atomic Quantum Dots at Room Temperature," *Physical Review Letters*, vol. 102, no. 4, p. 046805, 2009.

[8] 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.

[9] N. Pavliček *et al.*, "Tip-induced passivation of dangling bonds on hydrogenated Si(100)-2×1," *Applied Physics Letters*, vol. 111, no. 5, p. 053104, 2017.

[10] R. Achal *et al.*, "Lithography for robust and editable atomic-scale silicon devices and memories," *Nature Communications*, vol. 9, no. 1, p. 2778, 2018.

[11] T. Huff *et al.*, "Binary Atomic Silicon Logic," *Nature Electronics*, vol. 1, pp. 636–643, 2018.

[12] R. A. Wolkow *et al.*, *Silicon Atomic Quantum Dots Enable Beyond-CMOS Electronics*. Springer, 2014, pp. 33–58.

[13] M. Rashidi *et al.*, "Automated Atomic Scale Fabrication," 2022, US Patent App. 17/429,443.

[14] T. R. Huff *et al.*, "Electrostatic Landscape of a Hydrogen-Terminated Silicon Surface Probed by a Moveable Quantum Dot," *ACS Nano*, vol. 13, no. 9, pp. 10 566–10 575, 2019.

[15] R. Achal *et al.*, "Detecting and Directing Single Molecule Binding Events on H-Si (100) with Application to Ultradense Data Storage," *ACS Nano*, vol. 14, no. 3, pp. 2947–2955, 2019.

[16] F. Altincicek, "Atomically Defined Wires on P-Type Silicon," *Bulletin of the American Physical Society*, 2022.

[17] M. Walter *et al.*, "Hexagons are the Bestagons: Design Automation for Silicon Dangling Bond Logic," in *DAC*, vol. 22, 2022.

[18] M. D. Vieira *et al.*, "Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots," *IEEE Design & Test*, 2022.

[19] R. Lupoiu *et al.*, "Automated Atomic Silicon Quantum Dot Circuit Design via Deep Reinforcement Learning," 2022.

[20] A. N. Bahar *et al.*, "Atomic Silicon Quantum Dot: A New Designing Paradigm of an Atomic Logic Circuit," *TNANO*, pp. 807–810, 2020.

[21] S. S. H. Ng *et al.*, "SiQAD: A Design and Simulation Tool for Atomic Silicon Quantum Dot Circuits," *TNANO*, vol. 19, pp. 137–146, 2020.

[22] C. S. Lent *et al.*, "Quantum Cellular Automata," *Nanotechnology*, vol. 4, no. 1, p. 49, 1993.

[23] K. Hennessy and C. S. Lent, "Clocking of Molecular Quantum-dot Cellular Automata," *Journal of Vacuum Science & Technology B*, vol. 19, no. 5, pp. 1752–1755, 2001.

[24] F. Sill Torres *et al.*, "Synchronization of Clocked Field-Coupled Circuits," in *IEEE-NANO*, 2018.

[25] ——, "On the Impact of the Synchronization Constraint and Interconnections in Quantum-dot Cellular Automata," *MICPRO*, vol. 76, pp. 103–109, 2020.

[26] M. Walter *et al.*, "Placement & Routing for Tile-based Field-coupled Nanocomputing Circuits is $\mathcal{NP}$-complete," *JETC*, vol. 15, no. 3, 2019.

[27] A. Trindade *et al.*, "A Placement and Routing Algorithm for Quantum-dot Cellular Automata," in *SBCCI*, 2016.

[28] M. Walter *et al.*, "An Exact Method for Design Exploration of Quantum-dot Cellular Automata," in *DATE*, 2018, pp. 503–508.

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

[30] M. Walter *et al.*, "Scalable Design for Field-coupled Nanocomputing Circuits," in *ASP-DAC*. ACM New York, NY, USA, 2019, pp. 197–202.

[31] M. Walter and R. Wille, "Efficient Multi-Path Signal Routing for Field-coupled Nanotechnologies," in *NANOARCH*, 2022.

[32] M. Walter *et al.*, "One-pass Synthesis for Field-coupled Nanocomputing Technologies," in *ASP-DAC*. ACM New York, NY, USA, 2021, pp. 574–580.

[33] S. Hofmann, M. Walter, and R. Wille, "Scalable Physical Design for Silicon Dangling Bond Logic: How a 45° Turn Prevents the Reinvention of the Wheel," in *IEEE-NANO*, 2023.

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

[35] C. S. Lent *et al.*, "Molecular Quantum-Dot Cellular Automata," *Journal of the American Chemical Society*, vol. 125, no. 4, pp. 1056–1063, 2003.

[36] G. H. Bernstein *et al.*, "Magnetic QCA systems," *Microelectronics Journal*, vol. 36, no. 7, pp. 619–624, 2005.

[37] G. Beretta *et al.*, "Multi-Molecule Field-Coupled Nanocomputing for the Implementation of a Neuron," *TNANO*, vol. 21, pp. 52–59, 2022.

[38] P. D. Tougaw and C. S. Lent, "Logical devices implemented using Quantum Cellular Automata," *J. Appl. Phys.*, vol. 75, no. 3, pp. 1818–1825, 1994.

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

[40] J. Huang *et al.*, "Tile-based QCA Design Using Majority-like Logic Primitives," *JETC*, vol. 1, no. 3, pp. 163–185, 2005.

[41] V. Vankamamidi *et al.*, "Clocking and Cell Placement for QCA," in *IEEE-NANO*, vol. 1. IEEE, 2006, pp. 343–346.

[42] 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.

[43] M. Goswami *et al.*, "An efficient clocking scheme for quantum-dot cellular automata," *International Journal of Electronics Letters*, vol. 8, no. 1, pp. 83–96, 2020.

[44] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization," in *Design Automation Conference (DAC)*. IEEE, 2014.

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

[46] L. Amarú *et al.*, "The EPFL Combinational Benchmark Suite," in *IWLS*, 2015.

[47] F. Corno *et al.*, "RT-level ITC'99 Benchmarks and First ATPG Results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.