Late Breaking Results:
Wiring Reduction for Field-coupled Nanotechnologies

Simon Hofmann, Marcel Walter, and Robert Wille*
Technical University of Munich
Chair for Design Automation
Munich, Bavaria, Germany
https://www.cda.cit.tum.de/research/nanotech/
{simon.t.hofmann,marcel.walter,robert.wille}@tum.de

ABSTRACT

The emergence of *Field-coupled Nanocomputing* (FCN) as a green and atomically-sized post-CMOS technology introduces a unique challenge for the development of physical design methods: unlike conventional computing, wire segments in FCN entail the same area and delay costs as standard gates. Hence, it is imperative to reconsider physical design strategies tailored for FCN to effectively address this distinctive characteristic. This paper unveils a recent breakthrough in minimizing the number of wire segments by an average of 20.13 %, which, due to the high cost associated with wires, also leads to an average decrease of 34.10 % in overall area and 19.84 % in critical path length. Furthermore, unlike existing post-layout optimization algorithms, the proposed method maintains scalability even for layouts encompassing millions of tiles.

CCS CONCEPTS

• **Hardware** → **Quantum dots and cellular automata**; **Placement**; **Wire routing**.

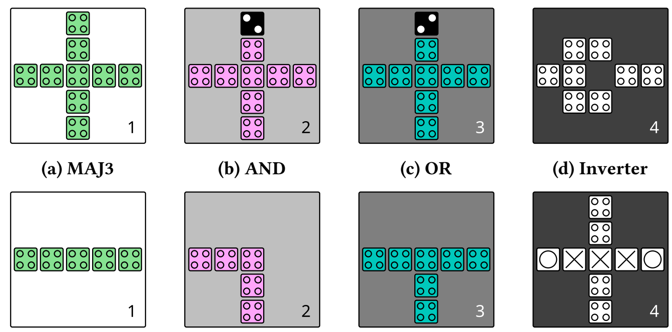
1 INTRODUCTION

Due to recent advances in atomically precise manufacturing [11] of *Silicon Dangling Bonds* (SiDBs, [1]) making *Field-coupled Nanocomputing* (FCN, [3]) a reality, efficient physical design methods are needed to generate gate-level layouts for this emerging technology.

One technology that implements the FCN concept is *Quantum-dot Cellular Automata* (QCA, [10]), where a cell consists of four *quantum dots* located in a square frame on a substrate. Multiple cells are then arranged on a 5×5 grid to construct standard gates such as the majority-of-three (MAJ3) function, AND, OR, inverter, and wire segments, as illustrated in Fig. 1. These gates can be activated by an external coupling signal, also called *clock*.

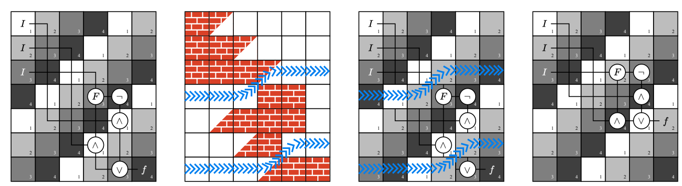
Unfortunately, wire segments, as seen in Fig. 1e to 1h, possess the same area and delay costs as the standard gates in Fig. 1a to 1d. Therefore, not only the positioning of gates has a huge impact on the resulting layout characteristics like area and critical path length, but also the number of wire segments connecting them. As a consequence of this co-dependence of placement and routing, reducing the number of wire segments not only improves circuit delay, but also circuit area.

Current approaches aim to minimize area overhead by determining advantageous placements of logic gates in a layout. Achieving this objective can be accomplished through two main approaches: The first involves employing SAT-based solvers [14] to calculate the optimal placement, albeit feasible only for smaller instances. Alternatively, heuristics can be utilized to swiftly identify suboptimal placements [6, 16], which can then be refined by relocating them to better positions during a post-layout optimization phase [7].



(a) MAJ3 (b) AND (c) OR (d) Inverter
(e) Wire (f) Bent wire (g) Fan-out (h) Crossing

Fig. 1: The QCA ONE gate library [12].



(a) Layout for the 2:1 multiplexer created by *ortho* [16] **(b) Added obstructions and possible cuts for wiring removal.** **(c) Wires on the cut paths are deleted, leaving behind gaps.** **(d) Resulting gaps are closed by pushing the layout together.**

Fig. 2: One iteration of the proposed wiring reduction algorithm.

This work proposes a novel post-layout optimization algorithm for wiring reduction which is highly scalable and achieves average area savings of 34.10 % simply by finding and deleting excess wiring in a layout. Due to a recently discovered connection between Cartesian layouts suitable for QCA and hexagonal layouts suitable for SiDBs [8], the proposed algorithm is also technology-independent.

An open-source implementation on top of the *fiction* framework [15] is available as part of the *Munich Nanotech Toolkit* (MNT) [18].[1] Furthermore, the generated layouts have been included in the benchmark suite *MNT Bench* [9].[2]

2 PROPOSED WIRING REDUCTION APPROACH

The core concept revolves around the selective removal of excess wiring by cutting them from a layout, contingent upon the ability to restore functional correctness by realigning the remaining layout fragments. Given the complexity of identifying these cuts, obstructions are strategically inserted into the layout to safeguard against the inadvertent deletion of standard gates or wire segments essential for the layout's integrity. Leveraging the obstructed layout as a basis,

*Also with Software Competence Center Hagenberg (SCCH).

DAC '24, June 23–27, 2024, San Francisco, CA, USA
 The definitive Version of Record was published in *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA*, https://doi.org/10.1145/3649329.3663491.

[1] The code is publicly available at https://github.com/cda-tum/fiction.
[2] https://www.cda.cit.tum.de/mntbench

Table 1: Comparative experimental evaluation of the proposed wiring reduction approach.

Benchmark Circuit [2, 4]			Ortho [16]				Proposed Wiring Reduction					Difference		
Name	I / O	$\|N\|$	$w \times h$	A	$\|W\|$	CP	$w \times h$	A	$\|W\|$	CP	$t[s]$	ΔA	$\Delta\|W\|$	ΔCP
c17	5 / 2	8	10 × 13	130	63	21	8 × 11	**88**	**51**	**17**	0.00	−32.31 %	−19.05 %	−19.05 %
c432	36 / 7	414	208 × 466	96928	35982	673	193 × 389	**75077**	**31369**	**581**	0.75	−22.54 %	−12.82 %	−13.67 %
c499	41 / 32	816	454 × 864	392256	89901	1317	309 × 638	**197142**	**65089**	**946**	18.86	−49.74 %	−27.60 %	−28.17 %
c880	60 / 26	639	328 × 748	245344	70226	1075	272 × 624	**169728**	**58293**	**895**	4.55	−30.82 %	−16.99 %	−16.74 %
c1355	41 / 32	1064	494 × 1176	580944	111893	1669	383 × 935	**358105**	**90494**	**1317**	34.94	−38.36 %	−19.12 %	−21.09 %
c1908	33 / 25	813	435 × 876	381060	99910	1310	352 × 678	**238656**	**80163**	**1029**	12.68	−37.37 %	−19.76 %	−21.45 %
c2670	233 / 64	1463	807 × 1701	1372707	323910	2498	649 × 1357	**880693**	**255517**	**1996**	86.29	−35.84 %	−21.11 %	−20.10 %
c3540	50 / 22	1987	931 × 2188	2037028	448264	3118	856 × 1828	**1564768**	**396497**	**2683**	142.02	−23.18 %	−11.55 %	−13.95 %
c5315	178 / 123	3628	1926 × 4019	7740594	1695255	5908	1565 × 3240	**5070600**	**1370685**	**4768**	1833.76	−34.49 %	−19.15 %	−19.30 %
c6288	32 / 32	6467	2273 × 6628	15065444	847918	8900	2215 × 5385	**11927775**	**752370**	**7599**	3700.07	−20.83 %	−11.27 %	−14.62 %
c7552	207 / 107	4501	2139 × 4830	10331370	2257823	6963	1753 × 3710	**6503630**	**1796980**	**5457**	4256.77	−37.05 %	−20.41 %	−21.63 %
ctrl	7 / 25	409	218 × 423	92214	27231	640	160 × 366	**58560**	**22098**	**525**	1.79	−36.50 %	−18.85 %	−17.97 %
router	60 / 3	490	257 × 557	143149	53356	813	245 × 391	**95795**	**42511**	**635**	3.22	−33.08 %	−20.33 %	−21.89 %
int2float	11 / 7	545	251 × 580	145580	47451	828	230 × 514	**118220**	**42975**	**741**	1.32	−18.79 %	−9.43 %	−10.51 %
dec	8 / 256	320	673 × 472	317656	161273	1144	256 × 465	**119040**	**66307**	**720**	66.25	−62.53 %	−58.89 %	−37.06 %
cavlc	10 / 11	1600	658 × 1668	1097544	283852	2325	617 × 1453	**896501**	**257646**	**2069**	45.21	−18.32 %	−9.23 %	−11.01 %
priority	128 / 8	2349	988 × 2484	2454192	664933	3471	961 × 1892	**1818212**	**575032**	**2852**	235.69	−25.91 %	−13.52 %	−17.83 %
adder	256 / 129	2541	1279 × 2797	3577363	789839	4075	769 × 2038	**1567222**	**526696**	**2806**	995.04	−56.19 %	−33.32 %	−31.14 %
Average Difference												−34.10 %	−20.13 %	−19.84 %

I, O and $|N|$ are the number of inputs, outputs and nodes in the logic network, respectively; w, h and A are the width, height and resulting area (in tiles) of the layout, respectively; $|W|$ and CP indicate the number of wire segments and the length of the critical path, respectively; $t[s]$ is the runtime in seconds; the area, number of wire segments and critical path length difference ΔA, $\Delta|W|$ and ΔCP, compare the layout before and after optimization, lower is better.

A^* *Search* [5] is employed to systematically identify feasible cuts either from left to right or top to bottom. Subsequently, these identified cuts are removed from the layout to minimize not only the number of wire segments, but also the area and critical path length.

In the following, the four main steps of the approach are explained using the 2:1 multiplexer from Fig. 2a as a running example.

2.1 Adding Obstructions

First, obstructions are added to the layout to restrict A^* to exclusively finding valid cuts. In Fig. 2b, standard gates are blocked completely, as they cannot be deleted, and bent wire segments are blocked halfway, as they can only be deleted if cut in a specific direction.

2.2 Determining Cuts

On the obstructed layout, A^* is applied to find cuts through the layout that represent slices of excess wiring that can be removed while preserving the layout's logical integrity. In Fig. 2b, two possible cuts are marked in blue, while the previously added obstructions ensure that only valid cuts are determined.

2.3 Deleting Wires

All wire segments contained in the feasible cuts are then removed from the original layout, as shown in Fig. 2c.

2.4 Repositioning Gates

To restore the operational integrity of the optimized layout, all tiles situated below the recently deleted ones are moved up, and gates are reconnected accordingly. This results in two empty rows at the bottom in Fig. 2d, which can then be deleted completely, effectively reducing the layout's area. This process is repeated iteratively until convergence, i. e., until no more feasible cuts are found.

3 EXPERIMENTS

Using the wiring reduction method proposed in this work, results from *any* physical design algorithm for Cartesian layouts using the *2DDWave* [13] clocking scheme can be optimized in terms of area, number of wire segments, and critical path length.

To demonstrate the resulting advantages, we took layouts created by the heuristic physical design approach *ortho* [16] for a broad variety of well-established benchmark circuits [2, 4], applied the proposed wiring reduction algorithm, and verified the correctness of the optimized layouts via formal verification [17]. The obtained data is summarized in Table 1, which lists the benchmark configurations as well as layout characteristics before and after the optimization.

On average, the number of wire segments was reduced by 20.13 %, resulting in an average area reduction and critical path shortening of 34.10 % and 19.84 %, respectively, while being highly scalable with a maximum convergence time of 4256.77 s even for layouts with millions of tiles.

4 CONCLUSION

In contrast to conventional computing, wire segments in FCN impose the same area and delay cost as standard gates. This work presents a novel post-layout wiring reduction algorithm, which effectively minimizes both the area overhead and the critical path length by an average of 34.10 % and 19.84 %, respectively, simply by reducing the number of wire segments.

REFERENCES

[1] R. Achal et al. 2018. Lithography for robust and editable atomic-scale silicon devices and memories. *Nat. Commun.* 9, 1 (2018).

[2] L.G. Amarù et al. 2015. The EPFL Combinational Benchmark Suite. In *IWLS*.

[3] N.G. Anderson and S. Bhanja (Eds.). 2014. *Field-Coupled Nanocomputing - Paradigms, Progress, and Perspectives.* Springer.

[4] F. Brglez et al. 1985. A neutral netlist of 10 combinational benchmark circuits and a targeted translator in FORTRAN. In *ISCAS*.

[5] P.E. Hart et al. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[6] S. Hofmann et al. 2023. Late Breaking Results From Hybrid Design Automation for Field-coupled Nanotechnologies. In *DAC*. 1–6.

[7] S. Hofmann et al. 2023. Post-Layout Optimization for Field-coupled Nanotechnologies. In *NANOARCH*.

[8] S. Hofmann et al. 2023. Scalable Physical Design for Silicon Dangling Bond Logic: How a 45 ° Turn Prevents the Reinvention of the Wheel. In *IEEE-NANO*. 872–877.

[9] S. Hofmann et al. 2024. MNT Bench: Benchmarking Software and Layout Libraries for Field-coupled Nanocomputing. In *DATE*.

[10] C.S. Lent et al. 1994. Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules. In *PhysComp*. 5–13.

[11] J. Pitters et al. 2024. Atomically Precise Manufacturing of Silicon Electronics. *ACS Nano* (2024).

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

[13] V. Vankamamidi et al. 2006. Clocking and Cell Placement for QCA. In *IEEE-NANO*, Vol. 1. 343–346.

[14] M. Walter et al. 2018. An Exact Method for Design Exploration of Quantum-dot Cellular Automata. In *DATE*. 503–508.

[15] M. Walter et al. 2019. fiction: An Open Source Framework for the Design of Field-coupled Nanocomputing Circuits. arXiv:1905.02477

[16] M. Walter et al. 2019. Scalable Design for Field-Coupled Nanocomputing Circuits. In *ASP-DAC*. 197–202.

[17] M. Walter et al. 2020. Verification for Field-coupled Nanocomputing Circuits. In *DAC*. 1–6.

[18] M. Walter et al. 2024. The Munich Nanotech Toolkit (MNT). In *IEEE-NANO*.