Majority-based Design Flow for AQFP Superconducting Family

Giulia Meuli [*], Vinicius Possani [*], Rajinder Singh [*], Siang-Yun Lee [†], Alessandro Tempia Calvino [†], Dewmini Sudara Marakkalage [†], Patrick Vuillod [*], Luca Amaru [*], Scott Chase [*], Jamil Kawa [*], Giovanni De Micheli [†]
[*] Digital Design Group, Synopsys Inc., Mountain View, CA, USA
[†] Integrated Systems Laboratory, EPFL, Lausanne, Switzerland

***Abstract*—Adiabatic superconducting devices are promising candidates to develop high-speed/low-power electronics. Advances in physical technology must be matched with a systematic development of comprehensive design and simulation tools to bring superconducting electronics to a commercially viable state. Being the technology fundamentally different from CMOS, new challenges are posed to design automation tools: library cells are controlled by multi-phase clocks, they implement the majority logic function, and they have limited fanout. We present a product-level RTL-to-GDSII flow for the design of Adiabatic Quantum-Flux-Parametron (AQFP) electronic circuits, with a focus on the special techniques used to comply with these challenges. In addition, we demonstrate new optimization opportunities for graph matching, resynthesis, and buffer/splitter insertion, improving the state-of-the-art.**

***Index Terms*—Superconducting circuits, AQFP, EDA flow, Logic optimization.**

I. Introduction

Superconducting electronic (SCE) devices are ideal candidates to build high-speed and low-power electronic circuits. With respect to standard CMOS, such devices rely on different materials and different physical phenomena. It follows that design and simulation tools must integrate dedicated strategies to enable the creation of complex superconducting circuits.

Josephson Junctions (JJ) and superconductive inductors are the basic building blocks of SCE devices, used to generate superconducting loops with quantized magnetic fields. Adiabatic Quantum-Flux-Parametron (AQFP) is a family of superconductive devices characterized by high energy efficiency [1]. Thanks to AC excitation currents driving the adiabatic switching operations, they have reduced dynamic energy dissipation: an advantage with respect to other superconducting devices, e.g., Rapid-Single-Flux-Quantum (RSFQ). Every logic cell in the AQFP library is sequential and controlled by a multi-phase clock, hence requiring advanced timing schemes and clock distribution. Besides, all paths to the input of a cell must be balanced to guarantee that excitation inputs are available at the clock signal. An additional challenge is posed by the limited fanout of superconducting cells. Specifically, an AQFP cell can drive 3 other cells, even if a smaller limit is often enforced to reduce power. For larger fanouts, special splitter cells are used to distribute the signal. AQFP logic cells natively implement the majority (MAJ) operation by combining three buffers. Other functions, e.g., AND/OR, are derived from the MAJ gate by choosing a specific buffer producing a constant logic value. New circuit optimization and mapping strategies can be developed, based on this highly expressive Boolean function, especially since both 3-input (MAJ3) and 5-input (MAJ5) gates are available. We will use MAJ and MAJ3 interchangeably. To represent the logic efficiently, it is ideal to use a network representation that suites the specific application. In the case of AQFP, this representation is Majority-Inverter Graphs (MIG) [2].

In this work, we describe how Synopsys' Fusion Compiler (FC), an industrial design automation tool originally designed for classical CMOS, has been adapted to work for AQFP superconducting circuits and delivers a complete RTL-to-GDSII flow. This flow, as the previously proposed solution for the RSFQ technology [3], is part of the company's effort to create efficient EDA tools for superconducting circuits. Starting from how the AQFP library cells are handled, this manuscript presents majority-based synthesis and optimization, technology mapping, and post processing, which includes splitter and buffer insertion. Our result section presents an analysis of how the circuit resources (number of cells) and area vary during the synthesis steps of the design flow.

Besides what is already industrially available, there is plenty of room for improvement. We unveil novel techniques in circuit rewriting, mapping and splitter insertion, which combined improve over the state-of-the-art in terms of area and delay. As publicly available logic synthesis tools, dedicated to CMOS, mostly rely on AND-inverter Graphs (AIG) [4], in this work we describe a mapper to MIGs that also optimizes logic sharing. Several works focused on optimizing superconductive circuits for area and delay and developed specific buffer and splitter insertion methods [5]–[7]. In this work, we present a rewriting algorithm that leverages exact synthesis and supports both 3-input and 5-input majority. Furthermore, we present a buffer and splitter insertion method that, in contrast with existing techniques, does not perform logic transformation. Instead it re-schedules the network to obtain a reduction of about 15% in JJ count

Overall, this work wants to highlight the challenges that EDA tools must overcome to design for the AQFP superconducting family, describe the industrial state-of-the-art, and shed light on new improvement opportunities.

II. Background

The use of Josephson junctions (JJ) in logic devices was originally proposed in [8]. They constitute the building blocks

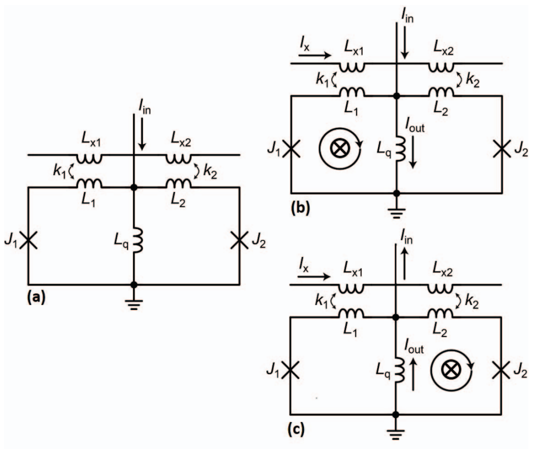

Fig. 1. Quantum Flux Parametron (QFP). (a) Circuit representation; (b) State "1"; (c) State "0".

of any superconducting electronic circuit. A Josephson junction consists of two superconductors coupled by a tunneling barrier. The junction operates at high switching speed between two distinct states: a pair tunneling state in which current flows through the barrier without any voltage drop, called superconducting state; and a single-particle tunneling state in which the current, above the critical level, flows with a voltage drop across the junction, called voltage pulse state. Josephson devices can be divided into two categories: flux transfer and voltage transfer devices. Among the first ones, AQFP devices [9] exploit adiabatic switching in the logic operation to drastically reduce the dynamic energy consumption of conventional superconducting devices. Superconducting adiabatic devices are realized by operating the Quantum Flux Parametron (QFP) [10] in adiabatic mode. A QFP, whose structure is shown in Figure 1(a), has the high switching speed of JJ devices but with a lower energy dissipation. It consists of a two-junction Superconducting QUantum Interference Device (SQUID) with two Josephson junctions J_1 and J_2, loop inductances L_1 and L_2, and an output inductance L_q. An initial current I_{in} flows in the device. The excitation current I_x is applied through inductances L_{x1} and L_{x2}, which are magnetically coupled with L_1 and L_2 respectively. If the excitation current causes the total applied flux to exceed the half flux quantum with an inwards input current I_{in}, a single quantum flux enters the left superconductive loops, generating a large output current. This state, shown in Figure 1(b), is encoded as state "1" of the logic device. State "0" is shown in Figure 1(c), where a quantum flux enters the right loop. It follows that the polarity of the initial current determines the final logic state of the cell. QFP devices are connected through superconducting transformers to form AQFP logic gates. Signal propagation is controlled using a multi-phase clock.

Logic functions natively implemented by the AQFP technology are Buffer, Inverter (NOT), and Majority (MAJ) gate. The inverter is easily obtained using an output transformer with a negative coupling coefficient. The 3-input Majority gate (MAJ) is obtained connecting three Buffer gates to a 3-to-1 Branch cell. Inverted inputs are implemented without area or delay overhead. The majority gate of three inputs x, y, and z implements the following Boolean function:

$$\mathrm{MAJ}(x,y,z) = \langle xyz \rangle = (x \vee y) \wedge (x \vee z) \wedge (y \vee z) \quad (1)$$

From Eq. 1 it is clear how $\mathrm{MAJ}(xy0) = x \wedge y$ and $\mathrm{MAJ}(xy1) = x \vee y$. It follows that AND/NAND and OR/NOR gates can be derived from the MAJ gate by fixing one of the inputs to a constant value. To optimize the logic of an AQFP device it is then convenient to use a multi-level logic network capable of efficiently representing the majority operation. Perfect candidates are Majority-Inverter Graphs (MIG) [2], in which each node implements the majority operation and edges can be inverted. MIGs have a dedicated complete algebra, hence the minimum MIG for a given Boolean function is always reachable using the transformations defined in the algebra. Despite that, finding the sequence of transformations for the minimum MIG is not trivial. Since the majority function is self-dual, inverters can be propagated through gates, which provides additional optimization opportunities.

III. Industrial AQFP Majority Synthesis Flow

As described in the introduction, there are several features of the AQFP technology that require dedicated handling in an industrial design flow. The first challenge is to handle the AQFP library, in which every cell is sequential. To adapt the library to the requirements of Fusion Compiler, a CMOS-based industrial synthesis engine, clock pins are removed to create new CMOS-equivalent combinational cells. The function expression of the cell is converted from sequential to combinational. Constraint timing data between clock and data inputs are removed. Sequential delay data from clock to output are transformed to combinational delay for all the input pins. In addition to the described special handling of library cells, logic optimization steps have been modified to reduce gates' fanout. Since AQFP maps into MAJ gates, a specific logic optimization step has been developed, which runs on a MIG representation of the logic. Once we have a functional netlist, the artificial library references are 'swapped' back to the original ones. Post-processing steps are required to add splitters and buffers and they are described in details in the next sections. Despite FC delivers a complete RTL-to-GDSII flow, including place&route and clock distribution, this work focuses on the initial synthesis flow steps.

A. Fanout Optimization

Despite the AQFP technology would enable a cell load of 3, since large fanouts are in conflict with adiabatic operation, here we enforce a maximum fanout of 1. To obtain larger fanouts, trees of splitter cells are inserted in the circuit. It follows that large-fanout cells have a negative impact on the area of the device due to the overhead given by the splitters' cells. In addition, since splitter cells are sequential, all paths to the loads need to be re-balanced after splitter insertion, causing operating frequency degradation. Thus, it is paramount to consider the fanout as a key cost function for the optimization of superconducting circuit.

To address the area impact of splitter cells, the proposed flow performs iterative optimization driven by a new cost function that considers the area cost of hypothetical splitter trees at the cell fanout. This cost function uses information from the cell library to scale the area impact of the splitters with respect to the other cells. Besides, logic sharing optimizations with low area impact have been removed from the flow, to avoid the unnecessary generation of large-fanout cells. Following the same motivation, aggressive depth optimization has been substituted by more selective techniques.

B. Majority Synthesis and Majority Mapping

One of the main challenges in AQFP-based design is to reduce the circuit latency since the combinational cells are clocked and have a unitary delay cost in this technology. In logic synthesis, it is well established that the logic depth in a combinational circuit correlates with the circuit performance. When it comes to the multi-level logic optimization targeting the AQFP technology, the minimization of logic depth plays a fundamental role for timing optimization. In this sense, the recent advances in the majority-based synthesis provide a robust alternative for depth-oriented optimization due to the ability for reshaping the circuit structure as well as for encapsulating logic into majority nodes [2].

Our majority-based optimization works on top of a generic Boolean network where each internal node can represent an arbitrary Boolean function. Initially, the proposed depth-oriented optimization decomposes the network into MAJ nodes and inverters e.g., majority-inverter graph (MIG). In the sequence, it performs multiple iterations of depth-first search (DFS) traversals trying to optimize the logic depth of each internal node by applying majority transformation rules based on the MIG algebra proposed in [2]. The algorithm optimizes the nodes in the transitive-fanin cones of each primary output by successively applying associativity, distributivity, relevance, and substitution transformations on the majority nodes. This iterative optimization works as a delay-driven approach by costing and updating the logic depth of each node. After the depth-oriented optimization, we do a pass of area recovering based on the distributive rule. We have observed that performing around of twenty iterations of depth-oriented optimization typically leads to good QoR (Quality of Results). Besides that, we have early bailouts to stop the iterative loop when the network structure is not improving for successive iterations.

After performing multiple passes of majority-based optimization, we end up with a network comprising MAJ nodes and explicit inverters. Therefore, we apply a post processing on top of the optimized network to ensure that we can fully benefit from the intrinsic properties of the AQFP cells to implement complemented signals. For instance, in AQFP technology, the AND, OR, and MAJ cells can implement inverted inputs without any extra area and delay cost. In this sense, the absorption of input inverters into these cells is essential to avoid the extra area and depth cost of explicit inverters. The proposed post processing performs three main operations: (i) collapsing input inverters into MAJ nodes; (ii) normalizing the pseudo MAJ nodes with a constant 0/1 input into their equivalent AND/OR representation; (iii) and, finally, permuting the inputs of the MAJ, AND, and OR nodes as needed to match the right polarities of the corresponding AQFP cells from the library, a process called MAJ-npn absorption. In practice, such a post processing delivers a logic network where each internal node has a direct correspondence to an existing combinational cell from the library. This way, the technology mapping for the generic Boolean network is straightforward done by a 1-1 mapping from network nodes to AQFP cells.

C. Post-Synthesis Steps

Once we have a fully-functional netlist, the following post-processing steps are required.

1) Reference swapping: Synthesis uses synthetic library cells while creating the synthesized netlist. The reference of each instance in the design needs to be swapped to the equivalent superconducting library cell. This is achieved by using a naming convention for the synthetic library cells. For example, if the original library cell is named “dff_h01”, the synthetic library cell is named “dff_h01_cmos_equiv”. To swap the reference of an instance, the correct real library cell is searched by name, by stripping of “_cmos_equiv” from the name of the existing reference.

2) Splitter Insertion: We have already discussed how splitter cells are required to distribute the signal when the cell fanout exceeds 1. The available splitter cells have one input and 2 or 3 outputs. For any net with multiple loads, a balanced splitter tree is created to ensure that each net in the design has one driver and one load. For example, supposed net A, has 4 loads. A 2 level splitter cell tree will be created, assuming splitters with 2 outputs are used. The original net connected to the input of first splitter cell and its output pins connected to next level of 2 splitters. The 4 output pins of the 2 splitters, now drive the original 4 loads of the net.

3) Buffer Insertion: The signal moves from the input of a cell to its output with each clock cycle. This implies that to ensure that input signals of a logic cell are correct, all paths to the cell must be of same length. The primary ports of the design are also considered as clocked elements for this purpose. The length is defined as number of combinational cells in the path. Buffering is done in two steps: (i) calculate the longest path, and (ii) add buffers so that the all path lengths equal the longest path. For example, if the longest path to a cell has a length of 10 and the path length from a primary input A to the cell is 6, 4 new buffers are required to ensure design correctness.

IV. Novel Synthesis Opportunities with AQFP

The previous section presented how we reformulated an industrial EDA flow to provide a complete solution for the AQFP technology. In this section we are giving a special focus to the logic synthesis step by proposing novel solutions that natively address the AQFP requirements during logic optimization and technology mapping.

A. Graph Mapping

As AQFP is majority-based, converting a given logic circuit into a good initial MIG structure is essential to exploit the expressive potential offered by the majority operation and obtain compact circuits. This process enables specialized mappers for AQFP technology to benefit from a favorable initial structure improving the final area and delay results. We developed a versatile mapper that maps from any network type into a technology-dependent or -independent (graph mapping) representation [11].

Our mapper is cut-based and utilizes a database of small circuit structures or a technology library to map and rewrite the circuit. Compared to other approaches such as LUT-based rewriting [12] and rewriting [13], our method offers better logic sharing and global optimization. Boolean matching is used to bind the cuts to the available structures or primitives so that decomposition costs are known during mapping and are used to drive the cuts selection process. Our approach executes multiple mapping refinement iterations from global to local optimization. In this way, the mapper first generates the cover globally accounting for shared logic, using local delay and 'area flow' [14] as cut selection criteria, and then optimizes it locally, in the MFFCs, using 'exact area' [15]. Structural hashing is also used in graph mapping to find common logic between the structures and further reduce the size of the mapped circuit.

As a result, we can obtain more compact and shallow logic than other graph mapping or rewriting approaches. Moreover, our mapper supports depth-oriented mapping that is crucial for the AQFP technology to limit the length of unbalanced paths. For AQFP synthesis, we are interested in mapping an initial network type into a MIG targeting both depth and size reduction. To do so, we used the mapper in the delay-oriented setting using a library composed of optimum size MIG structures for each 4-input NPN class.

B. Exact Resynthesis Considering AQFP-Specific Costs

Buffers and splitters in an AQFP circuit often take up a significant amount of the area and delay. To consider the cost of buffers and splitters during logic synthesis, we develop a rewriting algorithm utilizing an exact database of buffer-and-splitter-inserted MIGs [16]. The exact database contains, for each 4-input NPN class, a set of optimum AQFP structures under different input arrival time patterns.

To generate the database, we enumerate all single-output 4-input MIG DAG structures (without inverters) with predefined bounds on the number of gates and levels. Then, using a depth bounded search with backtracking together with dynamic programming, the optimum buffer-splitter insertion for each MIG structure is computed for a set of different input arrival time patterns. Then, for each MIG structure, we find the NPN classes computable by that DAG, and the database is updated accordingly. This database generation is a one-time computation that only depends on the AQFP library.

The rewriting algorithm first maps the input network to a 4-LUT network using ABC [17] and replaces the LUTs with optimum structures from the database considering their input arrival times, in a bottom-up manner. For each 4-LUT node, the algorithm computes the arrival times of its fanin signals by augmenting the levels of the respective fanins in the already rewritten part of the network with the additional number of levels required if balanced splitter trees were to be connected to those fanins. Such augmentations are needed only for fanins that have multiple fanouts. Then the algorithm replaces the LUT with the AQFP structure from the database that would give the minimum number of JJs, also taking the additional buffer requirements into account in case the input arrival time pattern does not exactly match the arrival patterns available in the database.

C. Buffer and Splitter Insertion

After logic optimization, we identify that buffer and splitter insertion can be further minimized by re-configuring the schedule of the logic network without logic transformation [18]. A schedule of a network is a level assignment to each logic gate. Using a linear-time algorithm and given a schedule, the minimum number of buffers and splitters needed to legalize the network can be computed. Thus, fine-grain optimization on the final JJ count of an AQFP circuit is achieved by finding a better schedule for the optimized MIG.

We first apply two simple scheduling algorithms, *as-soon-as-possible scheduling* (ASAP) and *as-late-as-possible scheduling* (ALAP), which try to place each logic gate at the lowest (or highest, respectively) possible level considering enough levels for splitters at the output of multi-fanout gates. The one which leads to a smaller buffer and splitter count is taken as the initial schedule. Then, we attempt to optimize the schedule by moving groups of closely-connected gates, called *chunks*, together. The set of all chunks forms a partition of all gates in the network, and the interfaces between chunks are where flexibility exists in the configuration of buffers and splitters. On average, our scheduling-based buffer-and-splitter optimization is capable of reducing about 15% and up to 40%, depending on the benchmark and the technology assumptions used, of the number of buffers and splitters [18].

D. AQFP Synthesis Flow

Overall, our AQFP synthesis flow consists of three steps: First, a good initial MIG is obtained by applying 3 rounds of graph mapping as introduced in Section IV-A. Then, the MIG is optimized considering buffer and splitter costs using the AQFP-specialized rewriting algorithm proposed in Section IV-B repeatedly for 10 iterations. Finally, buffers and splitters are inserted using the scheduling-based approach described in Section IV-C.

V. Experimental Results

We demonstrate Synopsys' Fusion Compiler AQFP-dedicated flow using a combinational design from the random-control EPFL benchmark [20] 'i2c controller', which has 147 inputs and 142 outputs. We report in Table II the number of logic cells, splitters and buffers at the different flow steps: (i) logic optimization, (ii) splitter insertion, and (iii) balancing (buffer insertion). Our MIG-based optimization and mapping

TABLE I
RESULTS OF THE PROPOSED AQFP SYNTHESIS FLOW.

Benchmark	Reference [19]		Only MAJ3 gates are used					Both MAJ3 and MAJ5 gates are used					
	Area (#JJs)	Delay (Levels)	MAJ3 count	Area (#JJs)	Delay (Levels)	Area impr. %	Delay impr. %	MAJ3 count	MAJ5 count	Area (#JJs)	Delay (Levels)	Area impr. %	Delay impr. %
5xp1	824	8	106	670	8	18.69	0.00	73	16	630	8	23.54	0.00
c1908	5242	53	391	3716	32	29.11	39.62	290	57	3684	31	29.72	41.51
c432	2198	50	201	1696	35	22.84	30.00	138	32	1636	32	25.57	36.00
c5315	18932	49	1244	10136	32	46.46	34.69	1065	91	9766	29	48.42	40.82
c880	4520	36	321	2304	22	49.03	38.89	212	56	2224	21	50.80	41.67
chkn	4022	28	426	2736	18	31.97	35.71	214	106	2506	16	37.69	42.86
count	1426	18	126	866	15	39.27	16.67	110	8	840	14	41.09	22.22
dist	4208	17	538	3356	12	20.25	29.41	345	96	3146	11	25.24	35.29
in5	4312	20	449	3008	15	30.24	25.00	201	124	2738	13	36.50	35.00
in6	3472	17	375	2424	12	30.18	29.41	167	104	2160	11	37.79	35.29
k2	18294	29	1915	14872	18	18.71	37.93	697	618	13250	17	27.57	41.38
m3	3118	13	411	2616	12	16.10	7.69	255	78	2468	11	20.85	15.38
max512	5536	19	706	4562	14	17.59	26.32	431	139	4234	14	23.52	26.32
misex3	14996	29	1500	10320	19	31.18	34.48	822	338	9528	17	36.46	41.38
mlp4	3622	19	443	2828	13	21.92	31.58	288	78	2640	11	27.11	42.11
prom2	28774	22	3641	23446	16	18.52	27.27	2362	661	22186	14	22.90	36.36
sqr6	1102	11	128	792	8	28.13	27.27	89	20	758	7	31.22	36.36
x1dn	1296	15	152	990	11	23.61	26.67	57	46	886	10	31.64	33.33
Total	125894	453	13073	91338	312	**27.45**	**31.13**	7816	2668	85280	287	**32.26**	**36.64**

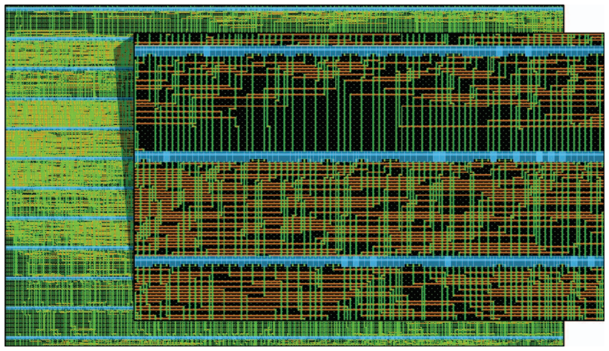
Fig. 2. GDSII picture of the design and a zoom of a selected area.

TABLE II
RESULTS OF THE INDUSTRIAL AQFP FLOW

	logic synthesis	splitter insertion	buffer insertion
#logic cells	907	907	907
#splitters	–	611	611
#buffers	–	65	5339
total	907	1583	6857
norm. area	1	1.60	3.54

methods synthesize a total of 907 logic cells. After logic synthesis, the references of these cells are swapped from the CMOS equivalent library to the original AQFP library, as explained in Section III-C1. The next flow step inserts splitter trees to guarantee that no net exceeds the fanout limit of 1. Both 1-2 and 1-3 splitters are available in the used AQFP library. As shown in Table I, 611 splitter cells and 65 buffer cells are inserted. The buffers are used to balance all paths of the splitter trees. During the last post-processing step, buffers are inserted to balance every path to the input of logic cells, for a total of 5339 buffers. Finally, Table I shows the area increase during each flow step. We use the normalized area to avoid disclosing information on the library cells. Although this manuscript focuses on logic synthesis and post-processing, FC provides a complete RTL-to-GDSII flow, hence including place&route. A portion of the chip area is shown in Fig. 2. It is possible to recognize logic cells in light blue and interconnects in green and orange.

We evaluate the AQFP synthesis flow described in Section IV-D using the MCNC benchmark suite [21] and compare our results against [19]. We assume that primary input signals are always available (i.e. do not need to be path-balanced with buffers) and strong enough (i.e. do not need to be branched with splitters). Primary outputs are branched and balanced. To guarantee a fair comparison we set the splitting capacity of splitters to 4 as in [19]. The method described in Section IV-C reduces the JJ count of 15% in this experiment. Table I summarizes the results focusing on the impact of MAJ-5 gates. On average, our synthesis flow improves over the state-of-the-art method by 27.5% in area and 31.1% in delay when only MAJ-3 is used. Moreover, if MAJ-5 gates are available, our flow further reduces area by 32.3% and delay by 36.6%.

VI. CONCLUSIONS

In this paper we detailed the challenges that EDA must face to design circuits for the AQFP technology family. A first challenge is to adapt tools that have been originally designed for standard CMOS. We presented Synopsys' solution, which provides a complete RTL-to-GDSII flow. Another challenge is to develop specific MIG-based optimization techniques for the new technology. We presented the special optimization and technology mapping methods integrated in Fusion Compiler. Besides, we focused on novel optimization opportunities and introduced a synthesis flow designed for AQFP. The flow is based on majority logic and performs splitter/buffer optimization by considering the technology requirements during logic synthesis. We evaluated our approach against the state-of-the-art method, reducing area by 27.5% and delay by 31.1% on average. Finally, results are further improved by leveraging 5-input majority gates.

ACKNOWLEDGMENT

Part of this work was supported by the Swiss National Science Foundation (SNSF) grant "Supercool: Design methods and tools for superconducting electronics" under Grant 200021_1920981. We thank Eric Mlinar and Yalan Zhang for their valuable support.

REFERENCES

[1] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Superconductor Science and Technology*, vol. 26, no. 3, p. 035010, 2013.

[2] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Majority-inverter graph: A new paradigm for logic optimization," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, pp. 806–819, 2016.

[3] L. Amarú, A. Ajami, S. Chen, Y. Zhang, T.-L. Tung, T. Arifin, T. Liu, M. Pan, G. Naveen, J. C. Vujkovic, P. Moceyunas, L. Clark, S. Whiteley, E. Mlinar, S. Lu, R. Singh, J. Chase, A. Belov, D. Rawlings, S. Anderson, A. Salz, R. Freeman, J. Kawa, and S. Chase, "First demonstration of a superconducting electronics microcontroller RTL-to-GDSII flow," in *Government Microcircuit Appl. Crit. Technol. Conf. (GOMACTech)*, 2021, p. 1–4.

[4] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *International Conference on Computer Aided Verification*, 2010, pp. 24–40.

[5] E. Testa, S.-Y. Lee, H. Riener, and G. De Micheli, "Algebraic and boolean optimization methods for AQFP superconducting circuits," in *Asia and South Pacific Design Automation Conference (ASPDAC)*, 2021, pp. 779–785.

[6] C. L. Ayala, R. Saito, T. Tanaka, O. Chen, N. Takeuchi, Y. He, and N. Yoshikawa, "A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors," *Superconductor Science and Technology*, vol. 33, no. 5, p. 054006, 2020.

[13] A. Mishchenko, S. Chatterjee, and R. Brayton, "DAG-aware AIG rewriting: a fresh look at combinational logic synthesis," in *Design Automation Conference (DAC)*, 2006.

[7] R. Cai, O. Chen, A. Ren, N. Liu, N. Yoshikawa, and Y. Wang, "A buffer and splitter insertion framework for adiabatic quantum-flux-parametron superconducting circuits," in *IEEE Int'l Conference on Computer Design (ICCD)*, 2019, pp. 429–436.

[8] J. Matisoo, "The tunneling cryotron—a superconductive logic element based on electron tunneling," *Proceedings of the IEEE*, vol. 55, no. 2, pp. 172–180, 1967.

[9] O. Mukhanov, N. Yoshikawa, I. P. Nevirkovets, and M. Hidaka, *Josephson Junctions for Digital Applications*. Springer International Publishing, 2019, pp. 611–701.

[10] M. Hosoya, W. Hioe, J. Casas, R. Kamikawai, Y. Harada, Y. Wada, H. Nakane, R. Suda, and E. Goto, "Quantum flux parametron: a single quantum flux device for josephson supercomputer," *IEEE Trans. on Applied Superconductivity*, vol. 1, no. 2, pp. 77–89, 1991.

[11] A. Tempia Calvino, H. Riener, S. Rai, A. Kumar, and G. De Micheli, "A versatile mapping approach for technology mapping and graph optimization," in *Asia and South Pacific Design Automation Conference (ASPDAC)*, 2022.

[12] W. J. Haaswijk, M. Soeken, L. Amaru, P.-E. Gaillardon, and G. De Micheli, "LUT mapping and optimization for majority-inverter graphs," in *Int'l Workshop on Logic Synthesis (IWLS)*, 2016.

[14] V. Manohararajah, S. D. Brown, and Z. G. Vranesic, "Heuristics for area minimization in LUT-based FPGA technology mapping," *IEEE Trans. on Computer Aided Design*, 2006.

[15] J. Cong, C. Wu, and Y. Ding, "Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution," in *Int'l symposium on Field Programmable Gate Arrays (FPGA)*, 1999.

[16] D. S. Marakkalage, H. Riener, and G. De Micheli, "Optimizing adiabatic quantum-flux-parametron (AQFP) circuits using an exact database," in *ACM Int'l symposium on Nanoscale Architectures (NANOARCH)*, 2021.

[17] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–40.

[18] S.-Y. Lee, H. Riener, and G. De Micheli, "Irredundant buffer and splitter insertion and scheduling-based optimization for AQFP circuits," in *Int'l Workshop on Logic Synthesis (IWLS)*, 2021.

[19] E. Testa, S.-Y. Lee, H. Riener, and G. De Micheli, "Algebraic and Boolean optimization methods for AQFP superconducting circuits," in *Asia and South Pacific Design Automation Conference (ASPDAC)*, 2021, pp. 779–785.

[20] M. Soeken, H. Riener, W. Haaswijk, E. Testa, B. Schmitt, G. Meuli, F. Mozafari, and G. De Micheli, "The EPFL logic synthesis libraries," *arXiv preprint arXiv:1805.05121*, 2018.

[21] S. Yang, *Logic synthesis and optimization benchmarks user guide: version 3.0*. Microelectronics Center of North Carolina (MCNC), 1991.