FPIA: Field-Programmable Ising Arrays with In-Memory Computing

G. Higgins Hutchinson, E. Sifferman, T. Bhattacharya, & D. B. Strukov

UC Santa Barbara, Department of Electrical and Computer Engineering

Santa Barbara, California, USA

{hutch,ethanjsifferman,tinish,dimastrukov}@ucsb.edu

ABSTRACT

Ising Machine is a promising computing approach for solving combinatorial optimization problems. It is naturally suited for energy-saving and compact in-memory computing implementations with emerging memories. A naïve in-memory computing implementation of quadratic Ising Machine requires an array of coupling weights that grows quadratically with problem size. However, the resources in such an approach are used inefficiently due to sparsity in practical optimization problems. We first show that this issue can be addressed by partitioning a coupling array into smaller sub-arrays. This technique, however, requires interconnecting sub-arrays; hence, we developed in-memory computing architecture for quadratic Ising Machines inspired by island-type field programmable gate arrays, which is the main contribution of our paper. We adapt open-source tools to optimize problem embedding and model routing overhead. Modeling results of benchmark problems for the developed architecture show up to 60x area improvement and faster operation than the baseline approach. Finally, we discuss algorithm/circuit co-design techniques for further improvements.

1 INTRODUCTION

Many practical combinatorial optimization (CO) problems may be represented as Quadratic Unconstrained Binary Optimization (QUBO) [12, 24]. Various physics-based and -inspired platforms have attracted recent interest in building QUBO-solving accelerators — the majority of such systems are collectively known as quadratic Ising Machines (IM) due to the QUBO merit function's similarity to Lenz' and Ising's Hamiltonian for spin glass systems [24]. In the most general, fully connected IMs, the focus of this paper, N spins corresponding to variables in the QUBO function are coupled via $O(N^2)$ matrix of tunable couplers that moderate local interaction among pairs of variables (Fig. 1). Solving a CO problem involves initializing IM states and then updating spin values over time, e.g., at discrete time steps in discrete-time IMs. A new spin state is computed by applying nonlinear (e.g., step) function over a dot product between the current IM state and the spin's coupling weights. With the help of annealing techniques, often implemented in the spin circuitry, IM states can evolve to a ground state, representing a solution to the underlying QUBO problem. IM extensions include high-order IMs with coupling among more than two spins [9], high-dimensional IMs with spin taking on more than two values [36], and IMs with probabilistic spin updates [11]. The functionality of such more general IMs is similar to other computing approaches for solving CO problems, such as Hopfield Neural Network [16], Boltzmann machines [33], and p-bit computing circuits [10].

Optical, electronic, quantum, and hybrid device technologies have been explored for implementing hardware accelerators of IMs
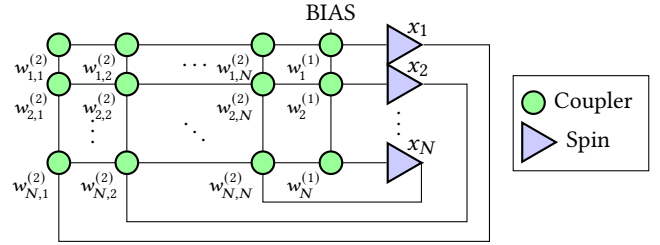

Figure 1: Quadratic Ising Machine for solving $\frac{1}{2}\sum_{i,j}^{N} W_{ij}^{(2)} x_i x_j + \sum_{i}^{N} W_i^{(1)} x_i + W^{(0)}$ QUBO problem. IMs are similar to Hopfield Neural Networks in which "coupler" and "spin" are called "synapse" and "neuron", respectively.

and related concepts [27]. Because the most common IM operation is a vector-by-matrix multiplication (VMM), In-Memory Computing (IMC) circuit implementations are especially promising. In this work, we will assume that spins are realized with CMOS circuits, while coupling weights are implemented with crossbar-integrated emerging or conventional memory arrays, similar to the previous work on neuromorphic inference accelerators [5].

Larger CO problems with $N > 1000$ are typically of practical interest. However, the memory array dimensions in IMC circuits are limited, primarily due to IR drop issues [2]. A solution to this problem in neuromorphic accelerators was to break up a larger single "logical" crossbar into multiple smaller size physical crossbar circuits [6, 34]. For example, in ISAAC architecture, a tile hosting a smaller physical memory array generates analog partial dot products. Partial products are sent via shared interconnect for the final accumulation performed in the digital domain[34].

On the other hand, practical QUBO-formulated problems are also very sparse (Fig. 2a), with the sparsity increasing with the problem size. (The studied benchmark problems are competition 3SAT [3, 4], random uniform 3SAT [15], and custom-generated semiprime factoring [30], converted to corresponding QUBO problems using Rosenberg approach [24].) In addition, the maximum spin "fan-in" (i.e., the maximum number of non-trivial elements in a single row of a coupling matrix) is much smaller than N (Fig. 2b). Hence, all nontrivial coupling weights of a spin might be implemented within a single physical crossbar, avoiding the need for expensive ADC circuits. In fact, sparseness and limited fan-in are necessary attributes for hard (most relevant) decision-type SAT problems. For example, the clause-to-variable ratio for hard 3SAT problems is close to 4.5, translating to the linear sparseness scaling with the size of equivalent QUBO problems.

These observations motivate our work — to develop more advanced Field-Programmable Ising Machine ("FPIA") architecture based on efficient IMC circuits and relevant design automation
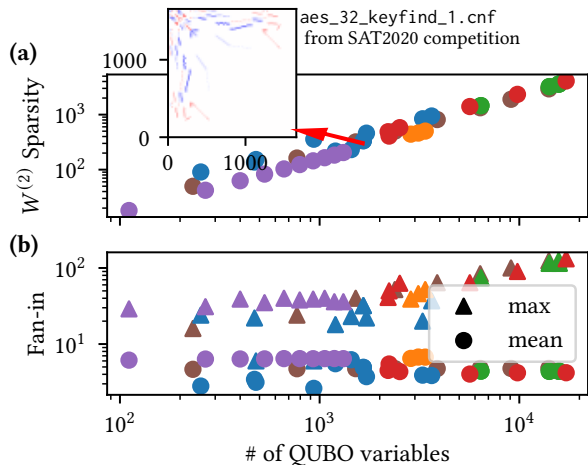
Figure 2: (a) Sparsity of the studied benchmark problems (see Fig. 3 legend), defined as N^2 / (# nonzero weights). The inset shows a down-sampled coupling matrix with color-coded weight density for one of the problems. (b) Maximum and average spin fan-in (number of incoming non-trivial connections to a spin) for the same problems.

algorithms that take advantage of sparsity and limited spin fan-in of practical QUBO problems. Furthermore, we focus on high-performance solutions that could exploit massively spin update parallelism and rapid convergence of IMs [14]. While there have been prior reports on IMC-enabled emerging memory IM architectures[29, 35], and, separately, on reconfigurable spintronics-based IMs[28], we believe that our work is the first to report programmable IM architecture that efficiently integrates IMC and rich FPGA-like interconnect to optimally implement locally dense, globally sparse connectivity of practical QUBO problems.

2 WEIGHT UTILIZATION IMPROVEMENT

To make FPIA's motivation more concrete, we investigate the prospects of partitioning problems' weight arrays into smaller sub-arrays such that all non-zero weights of each spin are located in the same sub-array. The key feature to exploit in a packing algorithm is the flexibility in mapping QUBO variables to IM spins, i.e., a QUBO variable can be mapped to any hardware spin without affecting IM's functionality. We use a quadratic-time greedy algorithm that is similar to the well-known First-Fit-Decreasing (FFD) algorithm for integer bin-packing algorithm because performing packing optimally is known to be NP-hard [18]. The algorithm starts by creating a list of (unpacked) spins sorted by their fan-in and a list of initially empty sub-arrays representing the clusters. Each spin steps in order through the list of clusters, and is packed into the first valid cluster. A candidate spin may be invalid to add to a cluster either because the cluster already contains O total spins, or because the cardinality of the union of spin inputs after adding the candidate would exceed I. We test this approximate packing for several QUBOs, holding $I = 256$ and varying O.

The results show that partitioning significantly improves weight utilization, especially for larger QUBO problems because of larger available sparsity (Fig. 2a). As expected, it is the largest for $O = 1$;
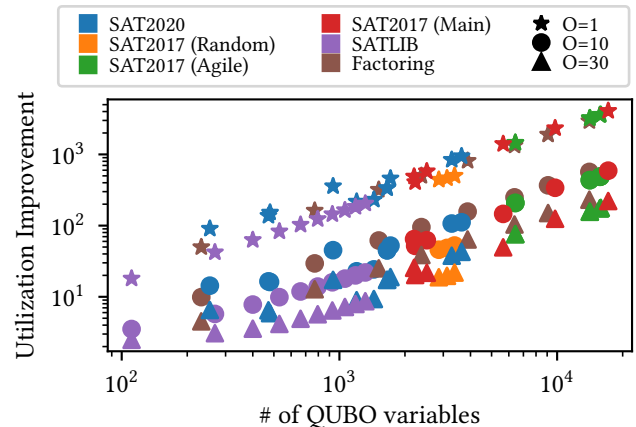

Figure 3: Coupling weight utilization improvement by tiling, defined as $N^2/(\sum_j I_j \cdot O_j)$, where $I_j \leq I$ and $O_j \leq O$ are the used input and output sizes of a cluster, and j steps through all clusters created by the FFD algorithm.

however, this is not necessarily representative of higher circuit density because such an analysis only includes memory array area but neglects array periphery and routing overheads, which would be higher for smaller sub-arrays. We next introduce the FPIA architecture and perform its detailed modeling to understand these tradeoffs better.

3 FPIA AND BASELINE ARCHITECTURES

The proposed FPIA architecture resembles an island-type FPGA [8], with the configurable logic blocks (CLBs) replaced by mixed-signal IMC blocks (Fig. 4). IMC block's I input and O output pins are connected to the routing channels via the nearest "connection block", but only to a fraction of wires (denoted with F_I and F_O, respectively) as is common in island-type FPGAs. Routing channels contain wires stretching R_{tile} tiles vertically or horizontally. Bends and extensions of the wire are implemented with the "switch blocks".

At the core of the IMC block is a $(I+O)\times O$ crossbar array of multi-bit memory cells and associated peripheral circuitry for implementing up to O spins, most importantly including two-quadrant (i.e., differential pair to encode negative weights) mixed-signal VMM, nonlinear activation, and annealing, and all spins' coupling weights. The digital inputs to the crossbar could be local, i.e., routed inside the block from local spins, or supplied externally from spins of other IMC blocks (Fig. 4b). Such an architecture allows up to I global and up to O local couplings implemented for each spin (though a lesser degree of coupling is assumed in modeling experiments, as discussed in section 4).

To provide a fair point of comparison for FPIA, we consider a baseline architecture inspired by ISAAC [34] in which a full logical coupling matrix is again partitioned into smaller ($S\times S$) physical sub-arrays but without previously considered optimization related to matrix sparsity (Fig. 5). The upside of such straightforward ("naive") implementation is very simple routing. As discussed in the introduction section, its downside is the need to add partial products from multiple sub-arrays to compute spin states. Similarly to FPIA, we assume IMC blocks with digital outputs. Block's partial dot-products
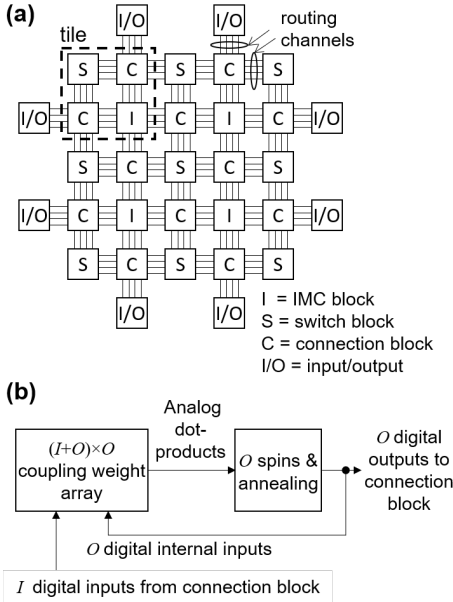
Figure 4: (a) FPIA's top-level and (b) mixed-signal IMC block architectures. Only four tiles are shown in (a) for simplicity.
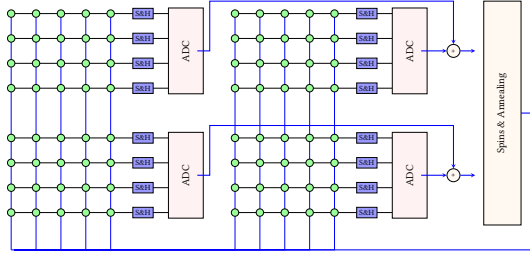

Figure 5: Baseline architecture.

are digitized, and top-level aggregation of results is performed in the digital domain. To further simply the routing and reduce ADC overhead, similar to ISAAC [34], we assume that IMC computations are run on a (relatively) slow clock, then sampled-and-held while a fast, pipelined shared ADC digitizes many intermediate values in a single IMC clock cycle.

4 MODELING FRAMEWORK

Many tools and algorithms for implementing circuits on FPGA can be re-purposed to implement FPIA, owing to the similarity in structure between the two. In this study, similarly to [28], we leverage the open-source VPR (Versatile Place and Route) tools [7, 25] to pack, place, and route QUBO problems in an FPIA. Specifically, a fictional "FPGA architecture" and "circuit netlist" are generated to encode, respectively, the capabilities of an FPIA and connection requirements of a QUBO — the process is visualized for a toy problem in Fig. 6. Each spin is mapped to a flip-flop to ensure a "circuit" without combinatorial loops. Each crossbar array row is mapped to an equivalent-width LUT.

An efficient routing architecture and algorithms require logical equivalence of FPGA CLB's inputs and outputs, i.e., the flexibility of permuting CLB's inputs and outputs. Such a feature is already available in our IMC block because its spins' location and external inputs

can be shuffled by appropriately setting up weights in the $(I+O) \times O$ sub-array. To ensure FPIA IMC block's logical equivalence in the VPR tools, the modeled CLB architecture includes a fully-populated crossbar switch with $I + O$ inputs and O outputs connecting CLB's I input pins and its O flip-flop outputs to O I-input LUTs. Note that we restrict the VPR packing step to only utilizing $I \times O$ sub-arrays, i.e., CLBs with O I-input LUTs in the modeled FPGA architecture, instead of using full $(I + O) \times O$ sub-arrays. This is because the VPR packer cannot be configured to limit the maximum number of external couplings to I (out of available $I + O$ total). With such limitation, CLB's fully-populated crossbar switch functionality is effectively implemented with $O \times O$ portion of sub-array.

Numerous crossbar-compatible memory cells have been explored for IMC applications [32]. In this study, we focus on three representative technologies — embedded flash (eFlash) with optimistic and pessimistic area scaling models, corresponding to the original [1] and redesigned eFlash [13], respectively, and SRAM [40]. For the latter, since the memory cell is digital, a differential 2-bit coupling weight, a sufficient precision for the studied benchmark problems, is assumed to be implemented with four SRAM memory cells with 1x- and 2x -width read transistors to encode the bit significance, similarly to [19].

Parameters used for the area and performance modeling are summarized in Table 1, while die areas of $M \times M$-tile FPIA and baseline architectures implementing N-spin IM are estimated as

$$A_{FPIA} \approx M^2 \left(I \cdot O \cdot A_{cell} + IA_{wl} + O \left(A_{bl} + A_{sense} \right) \right)$$
$$+ A_{routing} + \frac{M \cdot I \cdot A_{pewl}}{S_{pe}} + \frac{M \cdot O \cdot A_{pebl}}{S_{pe}} \quad (1)$$

$$A_{baseline} \approx \left\lceil \frac{N}{S} \right\rceil^2 \left(S^2 A_{cell} + SA_{wl} + SA_{bl} + A_{ADC} \right)$$
$$+ \frac{N \cdot A_{pewl}}{S_{pe}} + \frac{N \cdot A_{pebl}}{S_{pe}} \quad (2)$$

Here, eFlash and SRAM area assumptions are based on the work in [1, 13] and [26], respectively. The selected S value for baseline architecture modeling is optimistic, though achievable with proper bootstrapping to address IR issues [5]. Also very optimistic is $A_{ADC} = 10^6 \lambda^2$, based on area-optimized SAR ADCs [20] that would bottleneck speed if used, though ADC area overhead is also used as a variable parameter in a broader study. Sensing and driving overhead assumptions are similar to [5, 6]. Programming circuitry follows the design described in [6] and is assumed to be shared between IMC blocks, with A_{pewl} / A_{pebl} area overhead required to support a word line or bit line, respectively. We assume, however, that some time-multiplexing of programming circuitry can be exploited without bottle-necking overall operation (S_{pe}). Note that we neglect overheads of digital processing and annealing circuitry (that can be packed in the unused corners of a tile), as well as sample-and-hold circuitry (which is much smaller compared to ADC overhead according to [34]) and digital final summation in baseline architecture.

To explore the trade-off space of FPIA architectures, we search for optimal FPIA tile parameters. To optimize such a process and increase the scale at which we could study these problems, we used SMAC3 [23], an implementation of a Sequential Model-Based
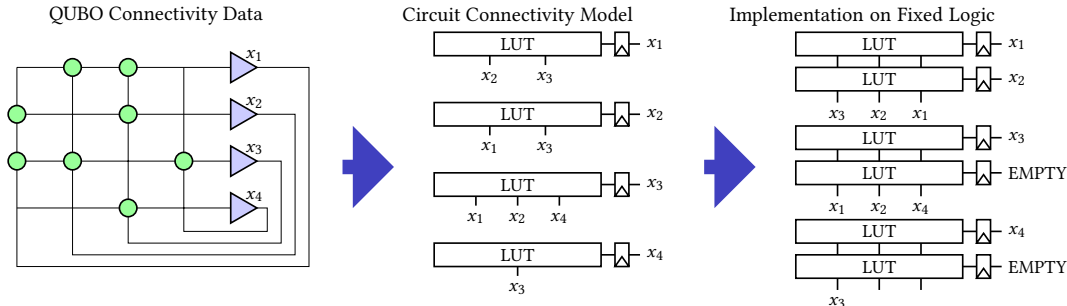
Figure 6: Mapping of an original, 4×4 **in this example, quadratic IM (left) to an equivalent FPGA circuit model preserving QUBO problem connectivity (center), which is then mapped to target FPIA represented by FPGA architecture with** 3×2 **CLBs (right). Equivalent nodes must subsequently be shorted by local and global routes, as appropriate. Note that the toy problem presented here is relatively densely connected (50%), and the tiling technique does not achieve any advantage.**

	eFlash optimistic	eFlash pessimistic	SRAM
A_{fetmin}	50	50	50
A_{cell}	60	180	600
A_{sense}	500	2500	2500
A_{wl}	1500	1500	1500
A_{bl}	1550	1550	1550
A_{pewl}	11300	11300	1550
A_{pebl}	7700	7700	1550
S_{pe}	1000	1000	1000
S	256	256	256

Table 1: Representative parameters (areas in units of λ^2**, the square minimum feature area of the technology) of important reference components of FPIA.**

Optimization [17]. In the algorithm configuration variant, random forests are used as the surrogate models, which appears to improve the ability to learn useful trends for loss functions that may return ∞ over BO methods with a Gaussian Process surrogate model. We use $R_{tile} = 4$ similar to area-optimal architectures described in Ref. [31], and use a Wilton topology with each wire segment connected to three other wire segments as described in [8], since the results are not sensitive to these parameters. Once an FPIA is packed, placed, and routed, the required array and routing size can be extracted, and we define the figure of merit *Tiling Advantage* as the ratio of baseline (Eq. 2) to FPIA (Eq. 1) area.

5 MODELING RESULTS

Figure 7 shows a motivation for using Bayesian Optimization. While the tiling advantage is not perfectly monotonic when sweeping I and O parameters, there appears to be a distinct optimal region, e.g., around $I \approx 80$ and $O \approx 40$ for the shown problem. Further investigation using Bayesian optimization with fixed connection block frequencies ($F_I = 0.2$ and $F_O = 0.2$) reveals a shared plateau ($O \in [30, 50]$, $I \in [80, 140]$) of approximate optima for other studied benchmark problems.

We next studied the impact of F_I and F_O on the routing area while fixing $I = 80$ and $O = 40$, i.e., using quasi-optimal values determined from the previous experiments. The grid search confirms previously chosen $F_I \approx 0.15$ optimal input frequency, close to typically used in FPGA [7], while does not reveal strong patterns in output connection occupancy (Fig. 8).
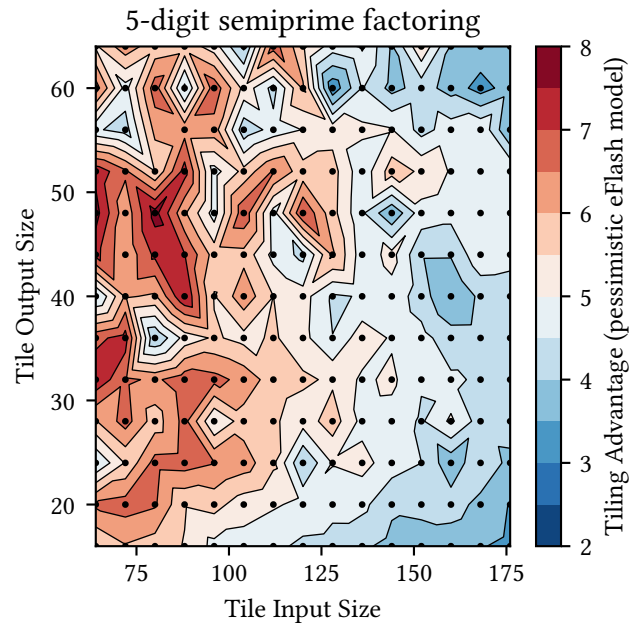

Figure 7: The landscape of tiling area advantage for a 5-digit prime factoring QUBO problem. The input and output connection fractions are fixed for this analysis, $F_I = F_O = 0.2$**.**
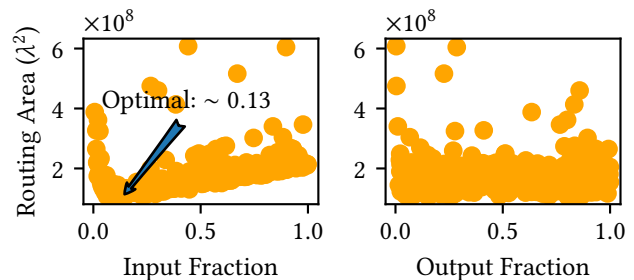

Figure 8: The impact of f_I **and** f_O **on** *routing* **area (not including IMC tile area) for 5-digit semiprime factoring problem. Routing area is reciprocal of the tiling advantage since the IMC block area does depend on connection block occupancy.**
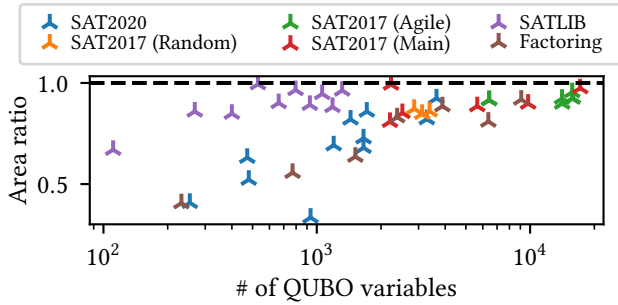
Figure 9: The decrease in area for a shared FPIA architecture (with the same parameters across all mapped problems) as compared to customized FPIA architectures (with parameters optimized per problem) for SRAM implementation.
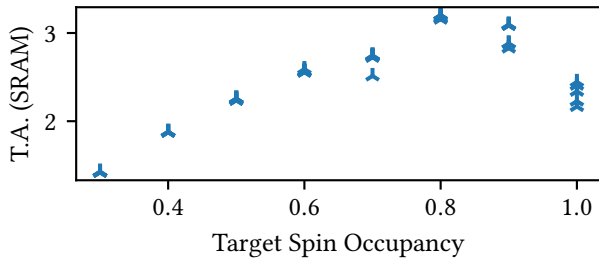

Figure 10: An impact of under-utilizing IMC on tile advantage, studied on the random 3SAT problems for SRAM implementation.

The practical realization of FPIA hardware implies a fixed architecture that cannot be customized based on the mapped QUBO problem. Therefore, we next focus on "shared" FPIA architecture based on fixed quasi-optimal $I = 140$, $O = 40$, $F_I = 0.15$, $F_O = 0.2$ determined from previous modeling experiments. Note that the selected I in such shared FPIA must be not smaller than the largest maximum fan-in of the targeted benchmark problems. Naturally, a specifically-optimized architecture is capable of embedding any given problem at least as well as a shared architecture. The relative sub-optimality of the shared embedding is shown in Fig. 9. We note that in the worst case, a shared FPIA fabric would consume only about 3.4 times more area than an FPIA optimized for that specific problem.

It has been noted for traditional FPGAs [37] that minimum area is not always achieved for every mapped circuit at the point of maximum logic resource utilization. We observe similar behavior in FPIA by sweeping the target occupancy of each CLB's output pin set, i.e., limiting the number of spins in ICM block to less than O. 80%–90% occupancy is found to be optimal (Fig. 10). Finally, assuming shared architecture, we study tile advantage prospects when exploiting the under-utilization of IMC blocks (Fig. 11).

6 DISCUSSION AND SUMMARY

The modeling results for shared FPIA architecture show that as the problem size increases, so does the Tiling Advantage, with a slope strongly controlled by the coupling cell area. By design, FPIA architecture can be used to tile smaller problems, but its parameters
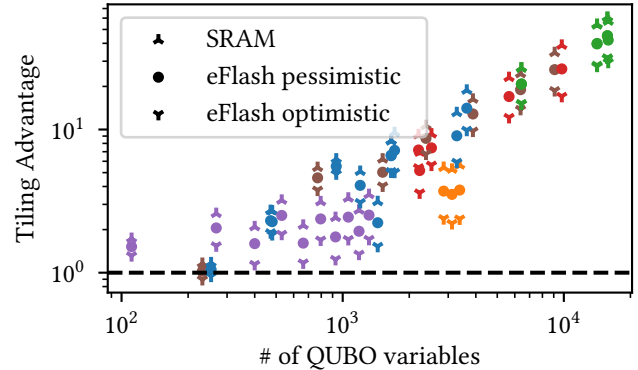

Figure 11: Tiling advantage of fixed ("shared") FPIA architecture for the studied benchmark problems (see Fig. 9 legend).

are not always optimal, and it is often less efficient compared to the baseline approach as smaller problems can fit onto a single physical crossbar of the baseline structure.

The detailed critical path routing delays, i.e., the longest delays of sending a spin value to a distant tile, assuming the optimal circuit parameters for $\lambda = 45$nm [21, 22], are almost independent of the problem size and always less than 1 ns. IMC latencies are expected to be significantly larger for considered block sizes [6], so that FPIA routing architecture adds negligible performance overhead. Therefore, FPIA is expected to be much faster than baseline architecture that relies on time-multiplexed read-out using shared ADC circuitry.

Because of the large contribution of ADC to the baseline design area, we studied the sensitivity of Tiling Advantage to the assumed A_{ADC}. Sweeping A_{ADC} and memory cell areas (hence indirectly varying practical values of S, another critical parameter in our baseline architecture) reveals a transition from crossbar-dominated to ADC-dominated scaling. (Fig. 12). An approximate expression for the Tiling Advantage is $\approx \frac{N^2(A_{cell}+\tilde{A}_{ADC})}{N^2CA_{cell}+A_{routing}}$, where $\tilde{A}_{ADC}$ is the amortized ADC area per cell, and C is the weight utilization improvement factor. Notably, in the ADC-dominated regime, the baseline design area is effectively constant w.r.t. cell area, but the FPIA area is increased, hence decreasing Tiling Advantage. Considering more broader options of ADC designs and memory technologies [38, 39] that meet the speed and precision requirements to run the baseline design at least 10 MHz, we find that realistic designs would be likely ADC-dominated.

Future work can improve FPIA directly in the architecture or through co-designing architecture and problem pre-processing. One caveat of shared FPIA architecture is that IMC sub-array minimum horizontal (I) dimensions cannot be smaller than the maximum fan-in of any spin of the targeted benchmark problems, which is expected to grow with problem size (Fig.2b). However, the average fan-in is almost flat (Fig.2b), and, therefore, a promising algorithmic approach is to break high fan-ins in the original problem by inserting new auxiliary variables. This technique changes the energy landscape of the problem. It hence further requires the characterization of possible changes in the solver's navigational efficiency and the impact on the time-to-solution. Another approach is to utilize heterogeneous IMC blocks containing multiple physical crossbars,
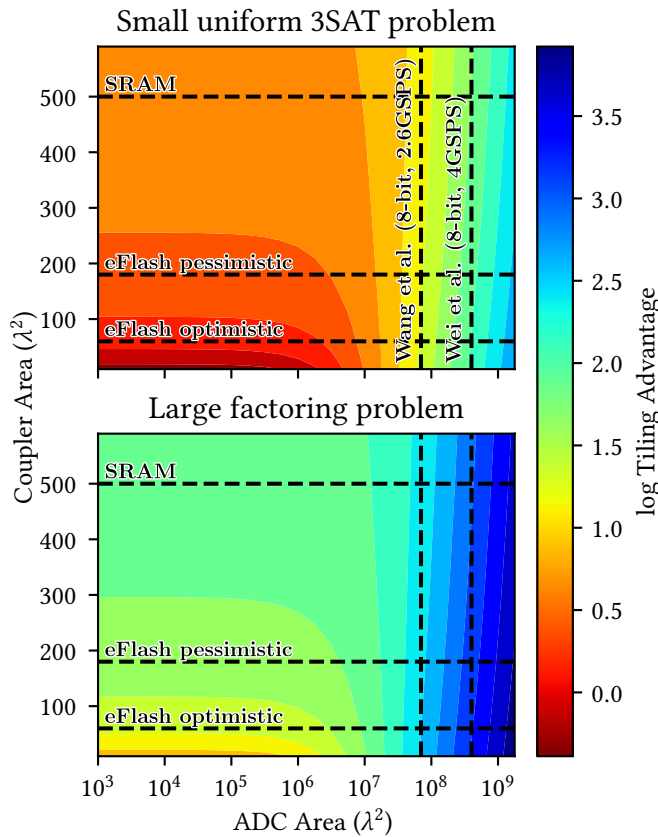
Figure 12: Tile advantage as a function of ADC and IMC unit cell sizes for two representative problems.

which would be chosen during packing to implement either several lower fan-in spins or fewer higher fan-in ones.

FPIA was studied in this work with respect to internally-analog, externally-digital, discrete-time optimization algorithms, but it is generalizable to continuous-time, fully-analog methods. This is because the routing architecture of FPIA requires no time-multiplexing, so wires could be directly used for analog signaling. However, this requires careful simulation of how much delay (and variation in delay) analog solver algorithms can tolerate.

In summary, we have proposed field-programmable architecture for efficiently implementing quadratic Ising Machines and related concepts. Leveraging open-source VPR design automation tools and Bayesian Optimization, we found a set of optimal parameters for the proposed hardware architecture for various practical benchmark combinatorial optimization problems. The modeled improvements are very encouraging, showing up to 60x area and faster operation compared to the baseline approach due to efficient exploitation of the benchmark problem sparsity.

ACKNOWLEDGMENTS

REFERENCES

[1] 2023. Superflash Technology Overview, SST, Inc. www.sst.com/technology/sst-superflash-technology.
[2] S. Agarwal, S.J. Plimpton, D.R. Hughart, et al. 2016. Resistive memory device requirements for a neural algorithm accelerator. In *IJCNN*. 929–938.
[3] T. Balyo, N. Froleyks, M.J.H. Heule, et al. 2020. Proceedings of SAT Competition 2020 : Solver and Benchmark Descriptions. http://hdl.handle.net/10138/318450
[4] T. Balyo, M.J.H. Heule, and M. Järvisalo. 2017. Proceedings of SAT Competition 2017 : Solver and Benchmark Descriptions. http://hdl.handle.net/10138/224324
[5] M. Bavandpour, M.R. Mahmoodi, H. Nili, et al. 2018. Mixed-Signal Neuromorphic Inference Accelerators: Recent Results and Future Prospects. In *IEEE IEDM*. 20.4.1–20.4.4.
[6] M. Bavandpour, M.R. Mahmoodi, and D.B. Strukov. 2020. aCortex: An Energy-Efficient Multipurpose Mixed-Signal Inference Accelerator. *IEEE JXCDC* 6, 1 (2020), 98–106.
[7] V. Betz and J. Rose. 1997. *VPR: a new packing, placement and routing tool for FPGA research*. Lecture Notes in Computer Science, Vol. 1304. Springer Berlin Heidelberg, Berlin, Heidelberg, 213–222.
[8] V. Betz, J. Rose, and A. Marquardt. 1999. *Background and Previous Work*. Springer US, Boston, MA, 11–35.
[9] C. Bybee, D. Kleyko, D.and Nikonov, A. Khosrowshahi, et al. 2023. Efficient Optimization with Higher-Order Ising Machines. *Nature Communications* 14 (12 2023), 6033.
[10] K.Y. Camsari, R. Faria, B.M. Sutton, and S. Datta. 2017. Stochastic p-bits for Invertible Logic. *Phys. Rev. X* 7, 3 (2017), 031014.
[11] S. Dutta, A. Khanna, A. Assoa, et al. 2021. An Ising Hamiltonian solver based on coupled stochastic phase-transition nano-oscillators. *Nature Electronics* 4 (07 2021), 502–512.
[12] F. Glover, G. Kochenberger, and Y. Du. 2019. Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4OR* 17, 4 (2019), 335–371.
[13] X. Guo, F. Merrikh-Bayat, M. Prezioso, et al. 2017. Temperature-Insensitive Analog Vector-by-Matrix Multiplier Based on 55 nm NOR Flash Memory Cells. In *IEEE CICC*. 1–4.
[14] M. Hizzani, A. Heittmann, G. Hutchinson, et al. 2023. Memristor-based hardware and algorithms for higher-order Hopfield optimization solver outperforming quadratic Ising machines. arXiv:2311.01171
[15] H. Hoos. 2011. SATLIB — Benchmark Problems. https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html
[16] J.J. Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *PNAS* 79, 8 (1982), 2554–2558.
[17] F. Hutter, H.H. Hoos, and K. Leyton-Brown. 2011. *Sequential Model-Based Optimization for General Algorithm Configuration*. Lecture Notes in Computer Science, Vol. 6683. Springer Berlin Heidelberg, Berlin, Heidelberg, 507–523.
[18] T. Izumi, T. Yokomaru, A. Takahashi, and Y. Kajitani. 1998. Computational complexity analysis of set-bin-packing problem. In *ISCAS*, Vol. 6. IEEE, Monterey, CA, USA, 244–247. https://doi.org/10.1109/ISCAS.1998.705257
[19] A. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy. 2019. 8T SRAM Cell as a Multibit Dot-Product Engine for Beyond Von Neumann Computing. *IEEE TVLSI* 27, 11 (2019), 2556–2567.
[20] L. Kull, T. Toifl, M. Schmatz, et al. 2013. A 3.1 mW 8b 1.2 GS/s Single-Channel Asynchronous SAR ADC With Alternate Comparators for Enhanced Speed in 32 nm Digital SOI CMOS. *IEEE JSSC* 48, 12 (2013), 3049–3058. https://doi.org/10.1109/JSSC.2013.2279571
[21] I. Kuon and J. Rose. 2008. Area and Delay Trade-Offs in the Circuit and Architecture Design of FPGAs. In *ACM/SIGDA FPGA* (Monterey, California, USA) *(FPGA '08)*. Association for Computing Machinery, New York, NY, USA, 149–158. https://doi.org/10.1145/1344671.1344695
[22] I. Kuon and J. Rose. 2008. iFAR – intelligent FPGA Architecture Repository. https://www.eecg.utoronto.ca/vpr/architectures/
[23] M. Lindauer, K. Eggensperger, M. Feurer, et al. 2022. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *JMLR* 23, 54 (2022), 1–9. https://www.jmlr.org/papers/v23/21-0888.html
[24] A. Lucas. 2014. Ising formulations of many NP problems. *Frontiers in Physics* 2 (2014).
[25] J. Luu, I. Kuon, P. Jamieson, et al. 2009. VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, Monterey California USA, 133–142.
[26] S. Mittal, J.S. Vetter, and D. Li. 2015. A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches. *IEEE Transactions on Parallel and Distributed Systems* 26, 6 (2015), 1524–1537.
[27] N. Mohseni, P. McMahon, and T. Byrnes. 2022. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics* 3 (04

2022), 363–379.

[28] A. Mondal, A.and Srivastava. 2021. Ising-FPGA: A Spintronics-based Reconfigurable Ising Model Solver. *ACM Transactions on Design Automation of Electronic Systems* 26, 1 (2021), 1–27.

[29] M. Nazm Bojnordi and E. Ipek. 2016. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *HPCA'16*. 1–13.

[30] P. Purdom and A. Sabry. 2018. CNF Generator for Factoring Problems. https://cgi.luddy.indiana.edu/~sabry/cnf.html

[31] K. Roy and M. Mehendale. 1992. Optimization of channel segmentation for channeled architecture FPGAs. In *IEEE CICC*. 4–4.

[32] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou. 2020. Memory devices and applications for in-memory computing. *Nature Nanotechnology* 15 (2020), 529–544.

[33] T. Sejnowski. 1987. Higher-Order Boltzmann Machines. 151 (Mar 1987).

[34] A. Shafiee, A. Nag, N. Muralimanohar, et al. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In

ACM/IEEE ISCA. 14–26.

[35] A. Sharma, R. Afoakwa, A. Ignjatovic, and M. C. Huang. 2022. Increasing Ising Machine Capacity with Multi-Chip Architectures. In *ACM ISCA'22*. 508–521.

[36] M. Strinati and C. Conti. 2022. Multidimensional hyperspin machine. *Nature Communications* 13 (11 2022), 7248.

[37] R. Tessier and H. Giza. 2000. Balancing Logic Utilization and Area Efficiency in FPGAs. In *Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing (Lecture Notes in Computer Science)*, R.W. Hartenstein and H. Grünbacher (Eds.). Springer, Berlin, Heidelberg, 535–544.

[38] D. Wang, X. Zhu, X. Guo, et al. 2019. A 2.6 GS/s 8-Bit Time-Interleaved SAR ADC in 55 nm CMOS Technology. *MDPI Electronics* 8, 33 (2019), 305.

[39] H. Wei, P. Zhang, B. Datta Sahoo, and B. Razavi. 2013. An 8-Bit 4-GS/s 120-mW CMOS ADC. In *IEEE CICC*. 1–4.

[40] C. Yu, T. Yoo, Tony T.-H. Kim, et al. 2020. A 16K Current-Based 8T SRAM Compute-In-Memory Macro with Decoupled Read/Write and 1-5bit Column ADC. In *IEEE CICC*. 1–4.