Equality Saturation for Datapath Synthesis: A Pathway to Pareto Optimality

Ecenur Ustun[1], Cunxi Yu[2], and Zhiru Zhang[1]
[1]Cornell University, [2]University of Utah

***Abstract*—Equality saturation, originally developed in the late 1970s for use in automated theorem provers, has been recently advanced to perform scalable rule-based rewriting for optimizations in various domains, such as program synthesis, compiler optimization, and datapath synthesis. Constructing an e-graph using rewrite rules that preserve program functionality, equality saturation addresses phase ordering problems in rewriting-driven optimizations. This promising approach shows significant potential for achieving Pareto optimality. This paper provides a brief introduction to equality saturation and the open-source tool `egg`, and highlights its potential application in optimizing datapaths in both RTL and high-level synthesis. We include case studies and outline the opportunities for future work in both datapath and logic synthesis using equality saturation.**

I. Introduction

Optimizing datapaths in digital circuits is a complex multi-objective process that involves reconciling potentially conflicting objectives and constraints, such as power consumption, performance, area, and accuracy. By exploring the Pareto-optimal solutions that achieve a desirable trade-off among these costs, hardware designers can make informed decisions and reduce design time and cost. The existing body of literature on identifying Pareto frontier for datapath optimizations is rich and varied, encompassing a multitude of techniques, algorithms, and design metrics. Prior research on automated datapath synthesis has primarily focused on heuristic search methods, statistical methods, and machine learning techniques [1]–[5]. However, existing solutions face a challenge in compactly representing a large set of functionally equivalent design points and extracting the Pareto frontier set with formal guarantees.

Equality saturation is a technique used in formal methods and compilers to optimize logic or code by applying a series of rules to simplify expressions and reduce redundancy [6]. More specifically, given an input specification such as a dataflow graph or program syntax tree, equality saturation constructs an *e-graph*, a graph-based data structure, by iteratively applying a set of rewrite rules that preserve the program's functionality. In this context, a rewrite can be a compiler transformation or a decomposition rule for integer multiplication. As these rewrites only augment the e-graph, meticulous phase ordering is not necessary. Upon reaching saturation or a specified timeout, the resulting e-graph compactly encodes a vast set of equivalent expressions for the input program, facilitating fast exploration of Pareto-optimal design points.

With the state-of-the-art open-source equality saturation tool called `egg` [7], this technique has been successfully applied in linear algebra [8], tensor computation, DSP compilation [9], and floating-point arithmetic [10], [11], among other areas, using rule-based rewriting. In the rest of this paper, we introduce a few recent efforts that apply equality saturation to datapath synthesis, and outline the opportunities for future work in both datapath and logic synthesis using this approach.

II. Case Studies

Recent studies have shown that equality saturation, particularly with the `egg` framework, is a promising technique for achieving scalable Pareto-optimal design space exploration and facilitating domain-specific or cross-domain optimizations. This section examines recent research that employs equality saturation to optimize datapaths, in the contexts of both RTL synthesis and high-level synthesis (HLS).

Coward *et al.* have proposed a new approach for optimizing RTL designs by representing them as data-flow graphs and utilizing e-graphs and equality saturation techniques [12]. The authors introduce a set of rewrites that enable efficient design space exploration and an automated method for optimizing architectures based on bitwidth parameters. They also quantify the noise floor in logic synthesis results to understand the limits of theoretical cost metrics. The results demonstrate that the automated rewriting technique can match the performance of skilled hardware engineers, while generating different architectures for various bitwidth designs that reflect bitwidth-dependent trade-offs. Additionally, the authors discuss potential future directions, such as expanding to floating-point operations, enhancing automated design verification, and tackling scalability limits through intelligent design space search procedures.

In their subsequent work [11], Coward *et al.* present a novel theory that enhances hardware design optimization by introducing the `ASSUME` operator to the e-graph for constraint-aware datapath optimization. This operator encodes sub-domain equivalences and takes two operands — an e-class containing equivalent expressions to evaluate, and a set of e-classes containing expressions that represent conditions assumed to be true during evaluation. The incorporation of `ASSUME` into e-graphs enables the representation of multiple equivalence relations, thereby allowing for the exploitation of branch-specific optimizations and the computation of tight approximations to intermediate signals. This new extension expands the applicability of equality saturation to a broader range of datapath optimization techniques.

IMpress [13], the first work to optimize large integer multiplication in HLS by performing equality saturation over many decomposition rules, produces various equivalent

expressions of integer multipliers with different hardware costs while avoiding the phase ordering problem. Although the e-graph is constructed to be as compact as possible, it contains a tremendous number of expressions which makes optimal extraction a nontrivial task. For example, decomposing a 2048-bit multiplication using 25 rewrite rules results in an e-graph with 645,935 nodes, corresponding to 1.59e6179 expressions. To address this challenge, IMpress builds a cost model that accurately estimates the resource utilization of any multiplication expression contained in the e-graph and offers ILP-based constrained and multi-objective extraction strategies tailored to optimize and balance the utilization of different FPGA resource types. IMpress has shown its effectiveness in exploring the Pareto frontier in multiple cryptography applications, resulting in up to 33% better utilization of a large FPGA device.

III. Opportunities

As demonstrated in the previous section, equality saturation presents a multitude of opportunities for enhancing datapath optimization in both RTL synthesis and HLS. We also posit that this method can be applied to various conventional and emerging logic synthesis challenges. In this section, we suggest several potential directions for exploration.

Current logic synthesis methods rely on graph-level transformations using directed acyclic graph (DAG) representations of Boolean networks [14]. Local optimizations are limited by Boolean algebraic transformations, while global transformations can be identified using formal methods like SAT and BDDs, but scalability issues arise. Equality saturation's rule-based rewriting and e-graph compactness offer three avenues to improve QoRs, runtime, and integration challenges.

(1) *Can equality saturation offer more efficient and compact representations for global rewriting?* By representing transformations that go beyond individual decomposition rules like Shannon expansion in BDDs [15] and Reed-Muller in FDDs, equality saturation can potentially exceed the current Pareto frontier on QoRs produced by existing methods. Additionally, it surpasses basic local structural hashing like AIGs while remaining a manageable size. One potential direction for exploration is whether equality saturation can establish a new (semi-)global DAG-aware rewriting engine with formal equivalence checking.

(2) *Can equality saturation facilitate the coupling of word-level and Boolean synthesis?* As a rule-based rewriting engine, equality saturation has the potential to explore modularized domains during the rewriting process. This involves combining word-level rules with Boolean algebra, which may reveal new optimization opportunities in both logic and datapath optimizations. Traditionally, these optimizations are applied separately, resulting in suboptimal solutions, or combined at the Boolean level, which requires high computational complexity [16]. Given the extensibility of equality saturation tools such as `egg`, it is possible to efficiently handle such multi-domain synthesis tasks. Moreover, the synthesis process could be further optimized using new abstraction operators, such as the recently introduced `ASSUME` operator [11].

(3) *Can equality saturation facilitate the adoption of conventional synthesis techniques in emerging technologies such as quantum computing?* There is a considerable overlap between emerging synthesis techniques and conventional state-of-the-art approaches. Equality saturation can effectively repurpose state-of-the-art methods by tailoring them to specific domains for new application areas. Moreover, it can be integrated into existing design tools and workflows, enabling a seamless transition to new synthesis paradigms.

Acknowledgement

This work is funded in part by NSF FMiTF awards #2019306 and #2019336, and by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] A. K. Verma, P. Brisk, and P. Ienne, "Challenges in Automatic Optimization of Arithmetic Circuits," *Symp. on Computer Arithmetic (ARITH)*, 2009.

[2] V. Krishnan and S. Katkoori, "A Genetic Algorithm for the Design Space Exploration of Datapaths During High-Level Synthesis," *IEEE Trans. on Evolutionary Computation (TEVC)*, 2006.

[3] D. H. Ram, M. Bhuvaneswari, and S. Logesh, "A Novel Evolutionary Technique for Multi-Objective Power, Area and Delay Optimization in High Level Synthesis of Datapaths," *IEEE Computer Society Annual Symp. on VLSI (ISVLSI)*, 2011.

[4] S. Xydis, K. Pekmestzi, D. Soudris, and G. Economakos, "Compiler-in-the-Loop Exploration during Datapath Synthesis for Higher Quality Delay-Area Trade-Offs," *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, 2013.

[5] R. Roy, J. Raiman, N. Kant, I. Elkin, R. Kirby, M. Siu, S. Oberman, S. Godil, and B. Catanzaro, "PrefixRL: Optimization of Parallel Prefix Circuits using Deep Reinforcement Learning," *Design Automation Conf. (DAC)*, 2021.

[6] R. Tate, M. Stepp, Z. Tatlock, and S. Lerner, "Equality Saturation: A New Approach to Optimization," *Symp. on Principles of Programming Languages (POPL)*, 2009.

[7] M. Willsey, C. Nandi, Y. R. Wang, O. Flatt, Z. Tatlock, and P. Panchekha, "egg: Fast and Extensible Equality Saturation," *Proc. of the ACM on Programming Languages (POPL)*, 2021.

[8] Y. R. Wang, S. Hutchison, J. Leang, B. Howe, and D. Suciu, "SPORES: Sum-Product Optimization via Relational Equality Saturation for Large Scale Linear Algebra," *Proc. of the VLDB Endowment*, 2020.

[9] A. VanHattum, R. Nigam, V. T. Lee, J. Bornholt, and A. Sampson, "Vectorization for Digital Signal Processors via Equality Saturation," *Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2021.

[10] B. Saiki, O. Flatt, C. Nandi, P. Panchekha, and Z. Tatlock, "Combining Precision Tuning and Rewriting," *Symp. on Computer Arithmetic (ARITH)*, 2021.

[11] S. Coward, G. A. Constantinides, and T. Drane, "Automating Constraint-Aware Datapath Optimization using E-Graphs," *Design Automation Conf. (DAC)*, 2023.

[12] ——, "Automatic Datapath Optimization using E-Graphs," *Symp. on Computer Arithmetic (ARITH)*, 2022.

[13] E. Ustun, I. San, J. Yin, C. Yu, and Z. Zhang, "IMpress: Large Integer Multiplication Expression Rewriting for FPGA HLS," *Symp. on Field Programmable Custom Computing Machines (FCCM)*, 2022.

[14] A. Mishchenko, S. Chatterjee, and R. Brayton, "DAG-Aware AIG Rewriting a Fresh Look at Combinational Logic Synthesis," *Design Automation Conf. (DAC)*, 2006.

[15] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers (TC)*, 1986.

[16] C. Yu, M. Ciesielski, M. Choudhury, and A. Sullivan, "DAG-aware Logic Synthesis of Datapaths," *Design Automation Conf. (DAC)*, 2016.