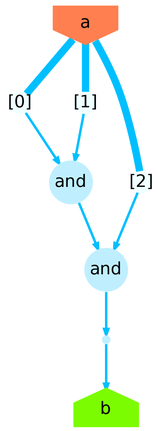
DECOMPILING HIGHER-ORDER FUNCTIONS IN DIGITAL CIRCUITS

Susan Clay, Sharanya Gehlot, Mihir Srivastava
Advisors: Prof. Jonathan Balkind, Zachary Sisco

UC SANTA BARBARA
Early Research Scholars Program

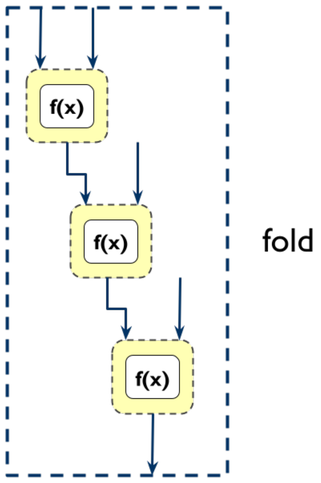
Introduction

Hardware decompilation recovers higher-level HDL code from netlists by identifying abstractions lost during synthesis. Previous work on loop rerolling focused on explicit iterative structures, but many hardware designs implement higher-order functions.



Background: Folds

Our work extends hardware decompilation to identify fold operations in netlists: Higher-order functions that sequentially reduce collections to a single result through repeated application of an operation.



Method: PyRTL and Egglog Fold Detection

Netlist Input

PyRTL Representation

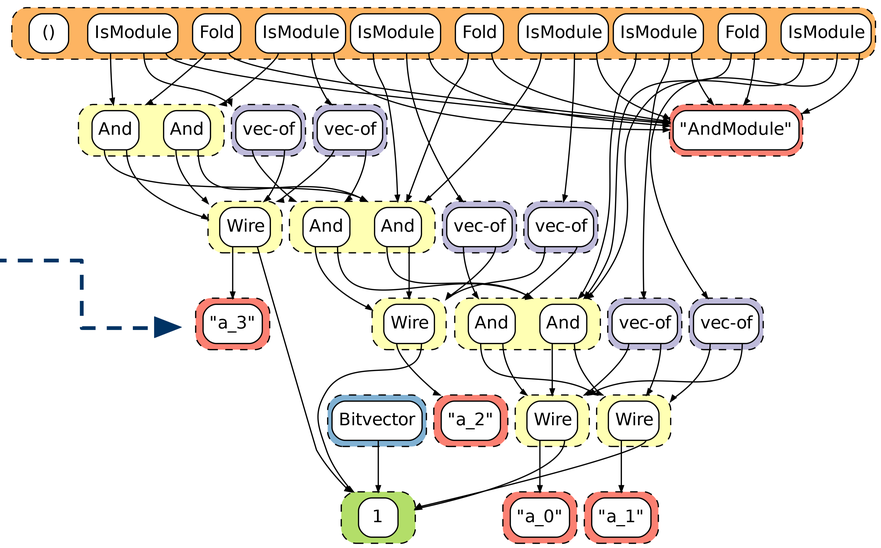
We leveraged PyRTL to model netlists with clear semantics, focusing on operations used in fold patterns like reducers, accumulators, and tree structures.

PyRTL Reducer Implementation

```
def reduce_and(bw) :
    in1 = pyrtl.Input(name="a_i", bitwidth=bw)
    out = pyrtl.Output(name="s_o", bitwidth=1)
    out <<= pyrtl.corecircuits.and_all_bits(in1)
```

Translation to Egglog

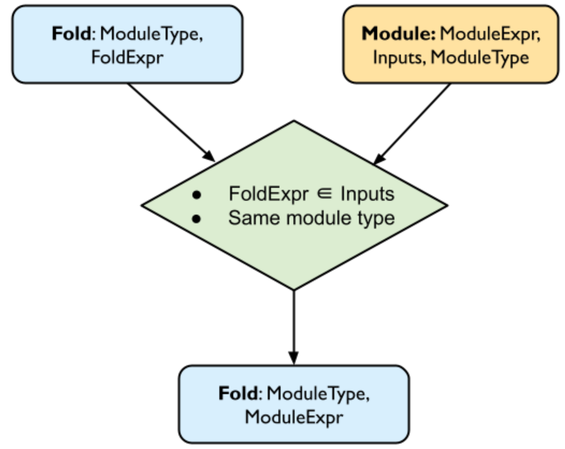
We converted PyRTL designs to Egglog format while preserving structural relationships between components for pattern analysis.



Fold Pattern Detection

1. We defined rules in Egglog to identify fold structures in the netlist, focusing on operations and rules that reduced collection.
2. We also added Boolean algebra rules for this, and this ruleset applied identities and simplifications that recognize higher-order functions.

Recursion



Output

1. Module Classification
2. Boolean Verification

Results: Fold Pattern Detection

Experiment 1: Folds & Types of Modules in Standard Designs

Description	# of Folds Identified	Type of Module Found
And Fold	1	AndModule
Add Fold	8	AddModule, AndModule, XorModule
Mux Fold	2	AndModule, OrModule, MuxModule

Experiment 2: Verification of Design Boolean Simplifications

Description	Size before	Size after
And Fold	9	1
Add Fold	34	4
Mux Fold	9	4

Future Work

In the future, we would extend the implementation to also encompass maps, as they are another form of higher-order functions that the code can generalize to. Furthermore, adding a larger database of test cases and reduction rules would help track performance better.

Citations

[1] Zachary D. Sisco, Jonathan Balkind, Timothy Sherwood, and Ben Hardekopf. 2023. Loop Rerolling for Hardware Decompilation. Proc. ACM Program. Lang. 7, PLDI, Article 123 (June 2023), 23 pages. https://doi.org/10.1145/3591237
[2] Smith, G. H., Knizek, C., Petrisko, D., Tatlock, Z., Balkind, J., Bernstein, G. L., ... & Nandi, C. (2024). Scaling Program Synthesis Based Technology Mapping with Equality Saturation. *arXiv preprint arXiv:2411.11036*.
[3] Smith, G. H., Sisco, Z. D., Techaumnuaiwit, T., Xia, J., Canumalla, V., Cheung, A., ... & Balkind, J. (2024). There and back again: A netlist's tale with much egraphin'. *arXiv preprint arXiv:2404.00786*.

Acknowledgements

We would like to thank Professor Matni, Pranjali Jain, and Jingtao Xia!