Impacts of Power, Area, and Performance of Various Branch Predictors

Thomas Gaul, Jackson Hafele, Gregory Ling
CPR E 581 Final Project Fall 2023

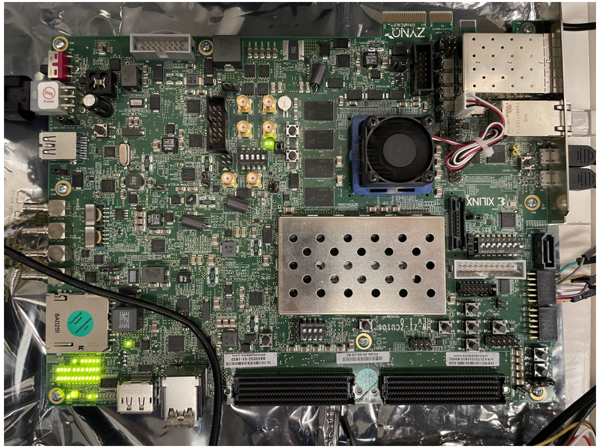
IOWA STATE UNIVERSITY

Motivation

- Explored performance of many branch predictors in class
  - Extend this knowledge to have hardware cost context
  - Cost effectiveness of branch prediction schemes
- With the decline of Moore's Law and Dennard Scaling
  - Power and area become a limited commodity to be budgeted

Main Idea

1. Explore using Chipyard to generate an out-of-order core
2. Compare branch predictors in terms of power, area, and performance
    a. TAGE, Tournament, GShare - Provided
    b. Global, Local, Null - Custom
3. Explore running SPEC benchmarks on a Chipyard softcore on an FPGA
    a. Utilize Firemarshal for Linux distribution



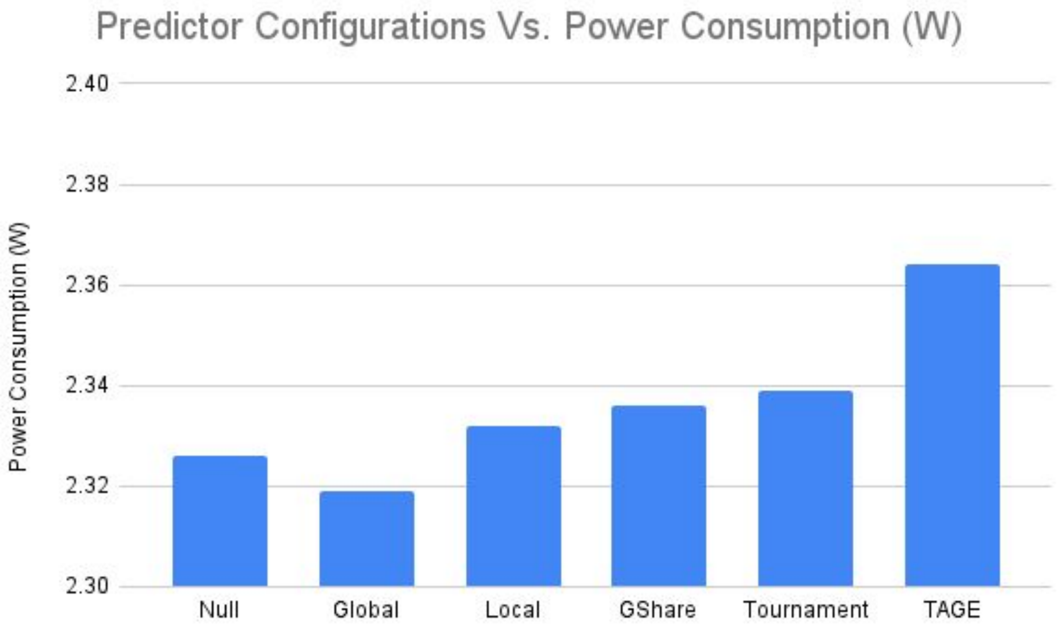
Methodology

- Use Chipyard to generate OOO cores with each branch predictor
- Use the ZCU106 configuration provided by Jordan's group
- Use Vivado to synthesize, generate the bitstream, program the ZCU106, and measure power/area usage
- Use SPEC benchmarks to provide representative example programs to measure performance

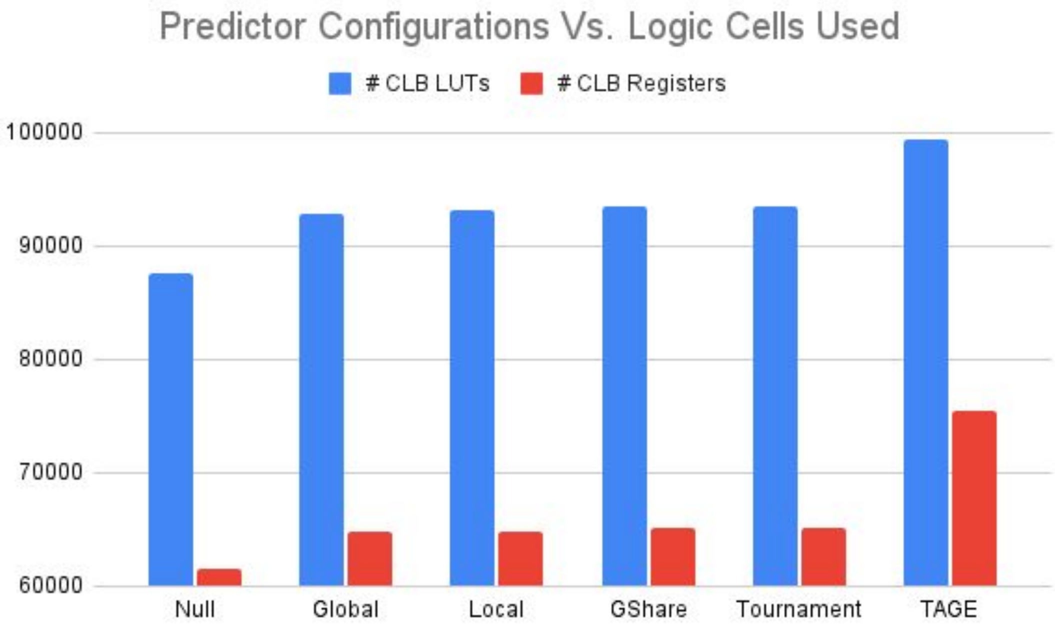
Method	Global Len	Local Len	Local Sets
TAGE	64	1	0
Tournament	32	32	128
GShare	16	16	1
Local	0	32	128
Global	32	0	0
Null	0	0	0

Table I: Branch Predictor History and set sizes

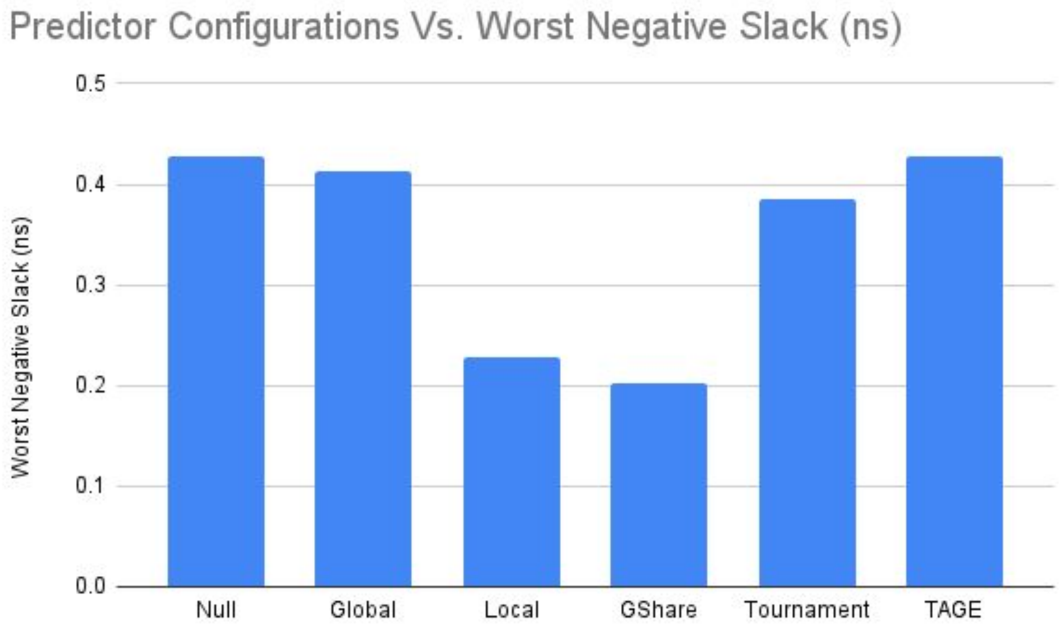
Results - Power



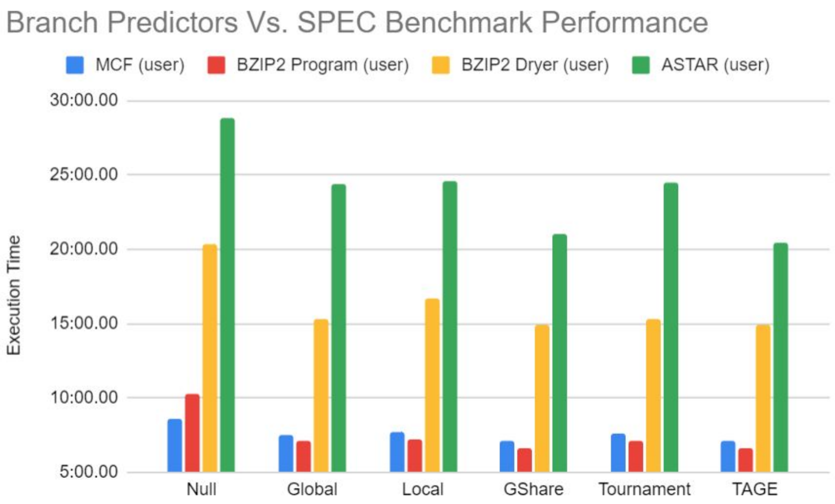
Results - Utilization

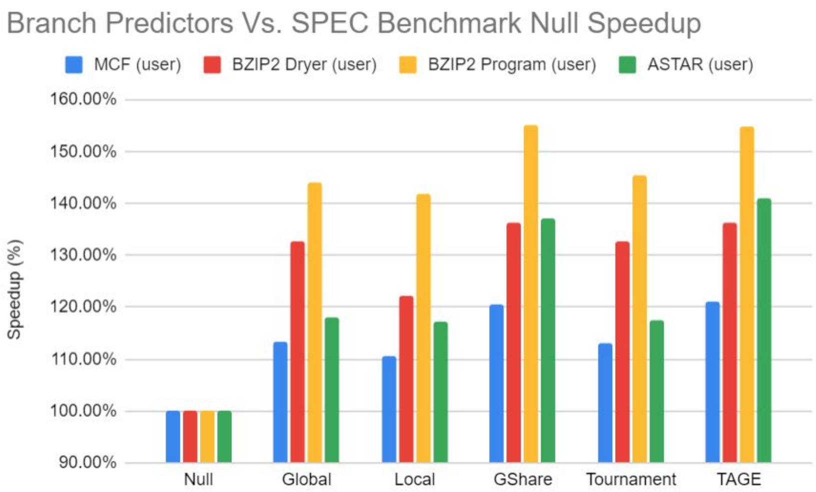


Results - Timing



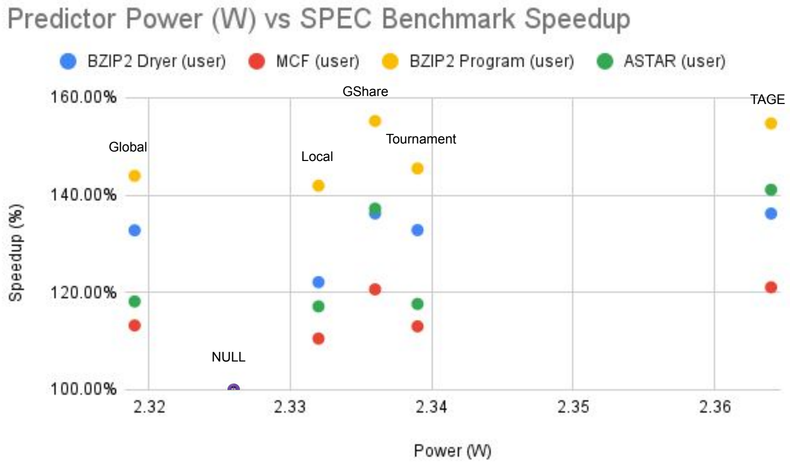
Results - Execution Time, Speedup





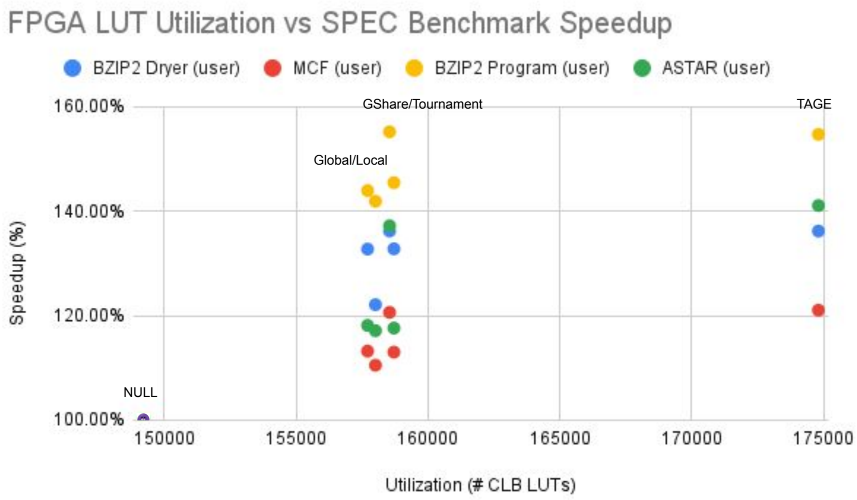
Analysis

- All predictors more performant than Null
- Global consumes less power than Null
- Tournament less performant than GShare despite more power
- TAGE consumes much more power relative



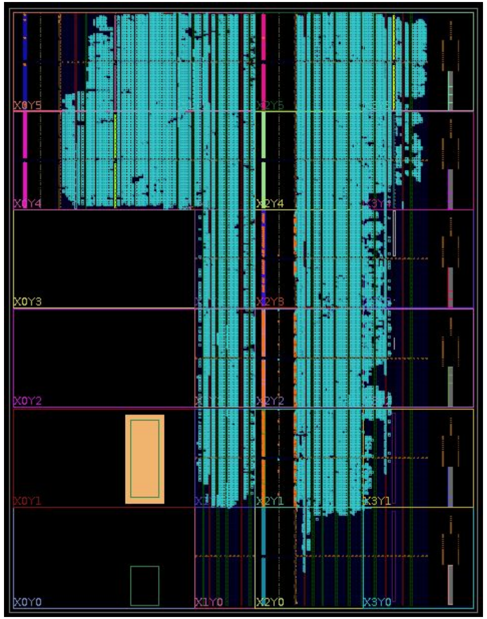
Analysis

- NULL consumed marginally less LUTs
- Utilization of Global, Local, and GShare very similar
- TAGE consumed far more LUTs, little performance benefit
- How much could changing global/local sizes affect each predictor?



Future Work

- Additional Benchmarks
- Run full benchmarks
- Multiple configurations for one predictor type
- Hardware Security



Conclusion

- Successfully generated RTL for 6 branch prediction schemes with BOOM Core
- Attained Vivado implementation results for area, power, and timing reports
- Compared the performance impacts of all 6 branch predictors for BZIP2, ASTER, and MCF

References

A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanovi ́c, and B. Nikoli ́c, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," IEEE Micro, vol. 40, no. 4, pp. 10–21, 2020.

Questions

Learning Achieved through the project

- Learned more about Open-Source tools (Chipyard BOOM)
- Applied multiple branch prediction schemes covered in class to real hardware
- Utilized and expanded on previous work with Chipyard and BOOM Core
- Compared the impacts of different predictors for power, area, and performance