

# Revisiting Hardware Priority Queue Architectures

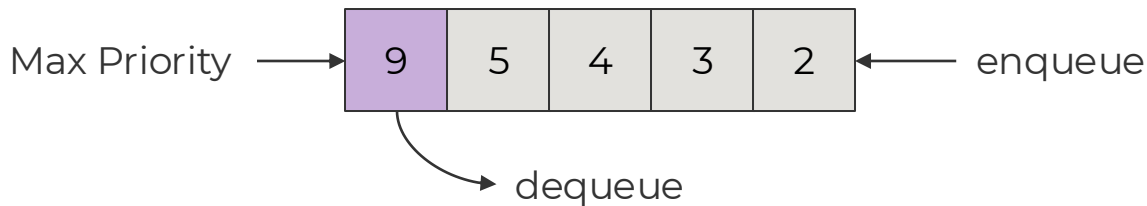
**Building an Open-Source Library for Evaluation and Design Exploration**

Qihang (Charlie) Wu and Austin Rovinski

# What is Priority Queue

Queue **sorts** elements based on **priority values**

- Elements sorted by **priority** first and insertion order second



# Priority Queue Applications

- **Task scheduling in operating systems**
  - CPU scheduling algorithms
- **Graph pathfinding algorithms**
  - Dijkstra's shortest path
  - A\* Pathfinding algorithm [3]
- **Event management and simulation**
  - Network Package Management [2]
  - Discrete Event Simulations



A\* Pathfinding Algorithm

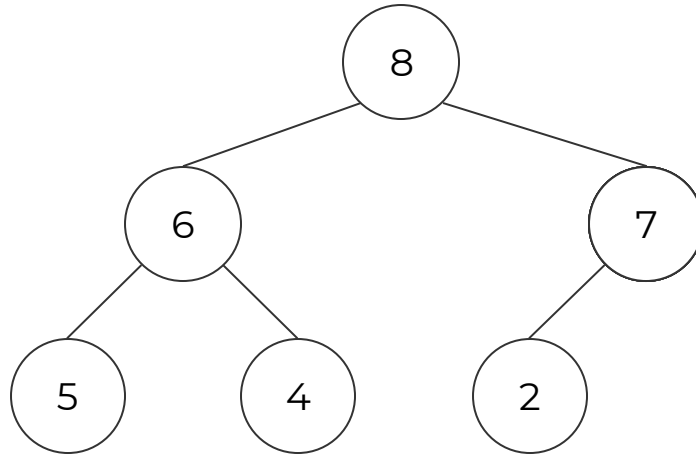
# Overview

- Priority Queue Overview
- Current Approaches and Limitations
- HWPQ Library Introduction
- Experimental Methodology
- Results and Insights
- Conclusion

# Software Implementation

- **Operations**

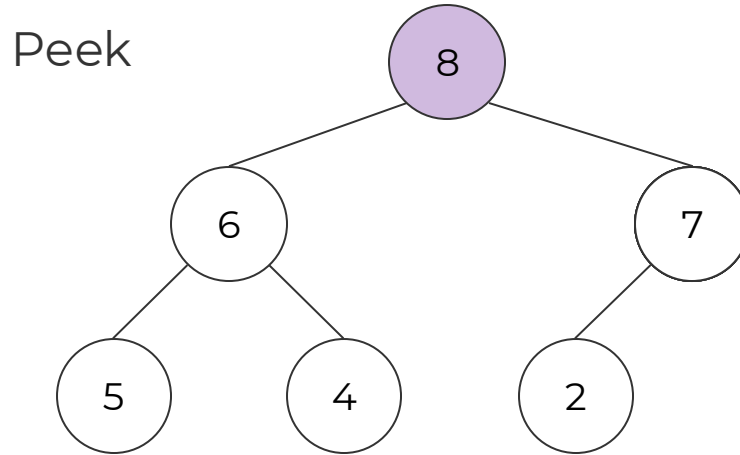
- Peek
- Enqueue
- Dequeue



# Software Implementation

- **Operations**

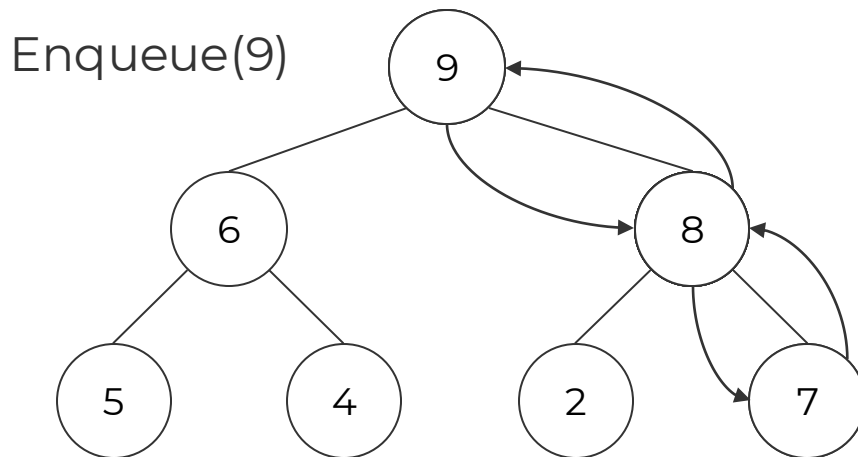
- Peek
- Enqueue
- Dequeue



# Software Implementation

- **Operations**

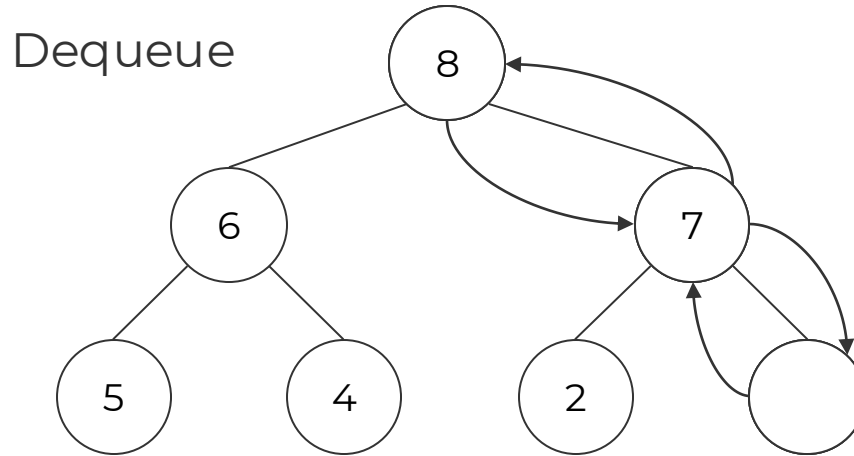
- Peek
- Enqueue
- Dequeue



# Software Implementation

- **Operations**

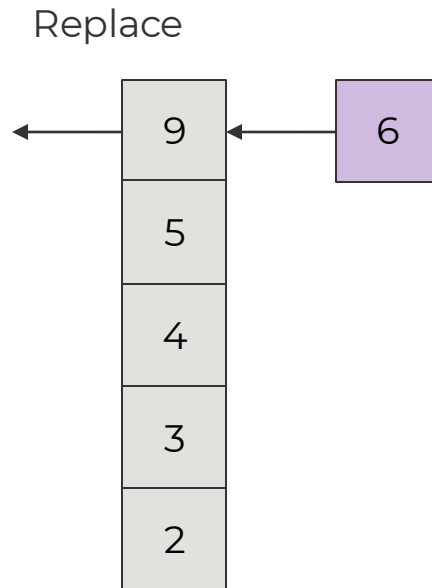
- Peek
- Enqueue
- Dequeue





# Hardware Priority Queue

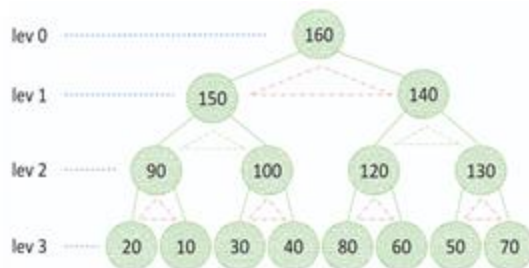
- **Support Operations**
  - Replace[1], Enqueue, Dequeue
- **Key Metrics**
  - Performance
  - Resource Efficiency



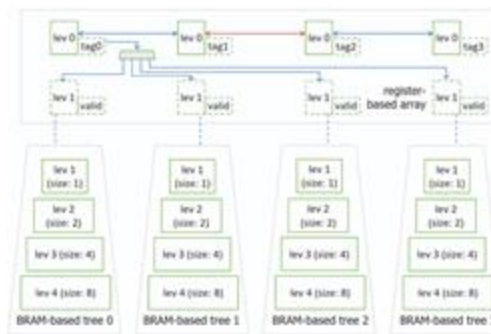
# Past Works and Motivation

- Register Tree [1]
- Register Array [1]
- Systolic Array [3]
- BRAM Tree [1]
- Hybrid Tree [1]
- Shift Register [2]
- ...

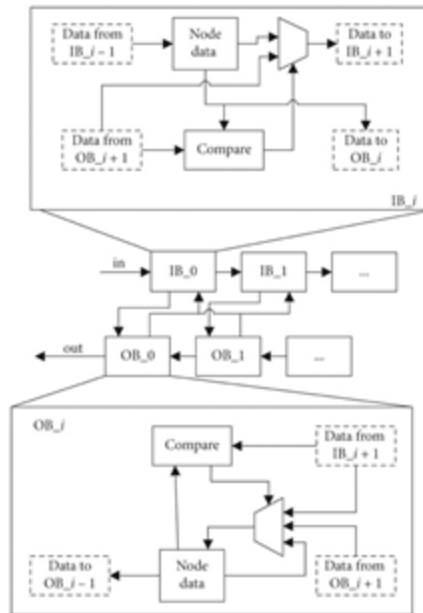
Register Tree [1]



Hybrid Tree [1]



Systolic Array [3]



# Past Works and Motivation

- Register Tree [1]
- Register Array [1]
- Systolic Array [3]
- BRAM Tree [1]
- Hybrid Tree [1]
- Shift Register [2]
- ...

- **Outdated testing environment**
- **Lack of standardized and comprehensive comparisons**

# Past Works and Motivation

- Register Tree [1]
- Register Array [1]
- Systolic Array [3]
- BRAM Tree [1]
- Hybrid Tree [1]
- Shift Register [2]
- ...

**Which architecture is the best  
for my application?**

# HWPQ

- Reimplement the architectures on a baseline platform
  - Perform head to head comparison

# HWPQ

- Reimplement the architectures on a baseline platform
  - Perform head to head comparison
- Architectures implemented in SystemVerilog
  - Parameterization

# HWPQ

- Reimplement the architectures on a baseline platform
  - Perform head to head comparison
- Architectures implemented in SystemVerilog
  - Parameterization
  - Standard Interface

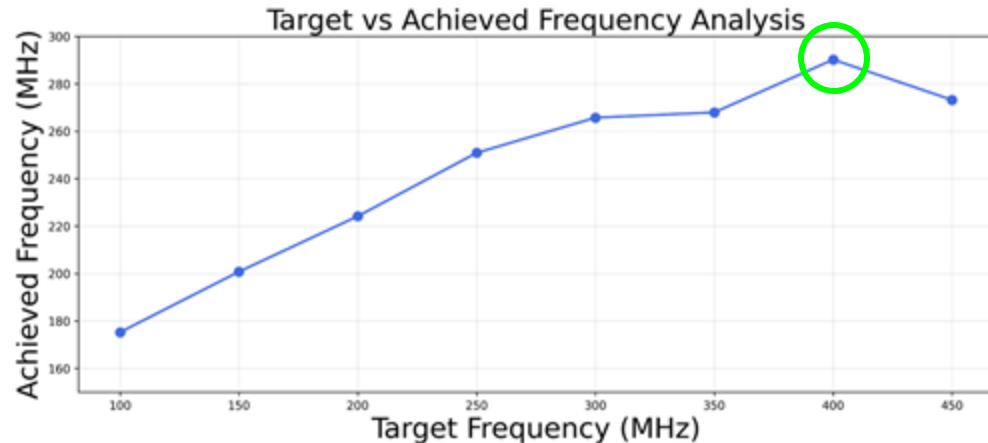
# HWPQ

- Reimplement the architectures on a baseline platform
  - Perform head to head comparison
- Architectures implemented in SystemVerilog
  - Parameterization
  - Standard Interface
  - Full-Verification



# Methodology

- Artix UltraScale+ FPGA (XCAU25P) with AMD Vivado 2024.2
- Sweep through parameters to find the maximum achievable frequency
  - 16-bit data width, sweep queue size
  - Sweep a range of target frequencies



# Methodology

- Artix UltraScale+ FPGA (XCAU25P) with AMD Vivado 2024.2
- Sweep through parameters to find the maximum achievable frequency
  - 16-bit data width, sweep queue size
  - Sweep a range of target frequencies
- Record and calculate metrics for comparison
  - Max Archived Frequency for certain queue size
  - Performance (MOPS)
  - Resource Efficiency (Performance / Resource Utilization)
- Enqueue enabled/disabled for applicable architectures

# Architectures

	Enqueue Switch	Enqueue Latency (cycle) *N means queue size	Replace/Dequeue Latency (cycle)
<b>Register Array [1]</b>	Yes	1	1
<b>Register Array (Pipelined)</b>	Yes	2	2
<b>Register Tree [1]</b>	Yes	Log N	1
<b>Register Tree (Pipelined)</b>	Yes	Log N + 1	2
<b>Systolic Array [3]</b>	Yes	1	1
<b>BRAM Tree [1]</b>	No	N/A	8
<b>BRAM Tree (Pipelined)</b>	No	N/A	4
<b>Hybrid Tree [1]</b>	W.I.P.	1 (expected)	1

# Register Array


9	8	5	2
---	---	---	---

replace(3)

3	8	5	2
---	---	---	---


1st compare swap

8	3	5	2
---	---	---	---



2nd compare swap

8	5	3	2
---	---	---	---

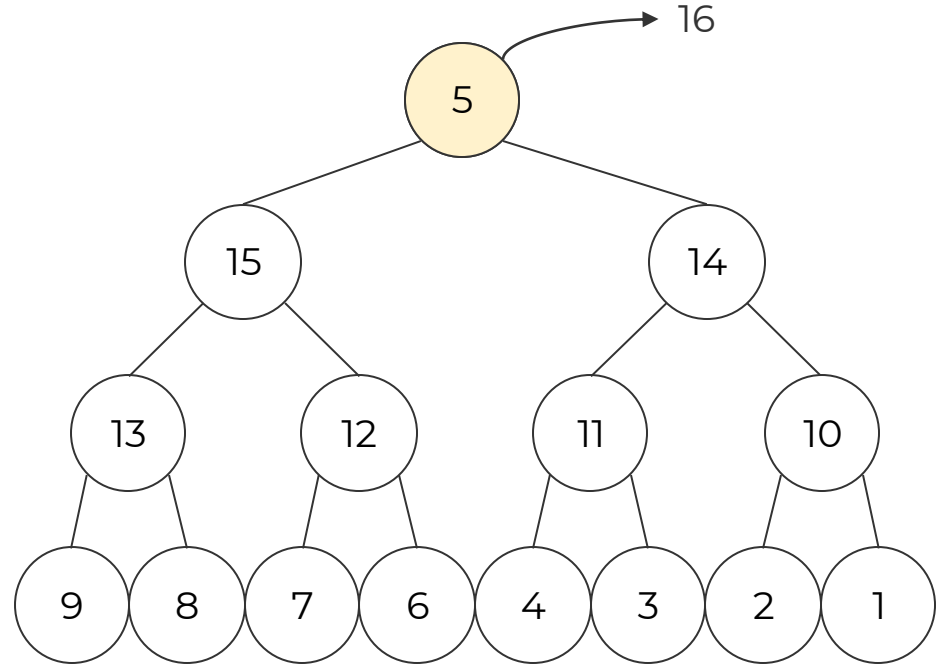


# Architectures

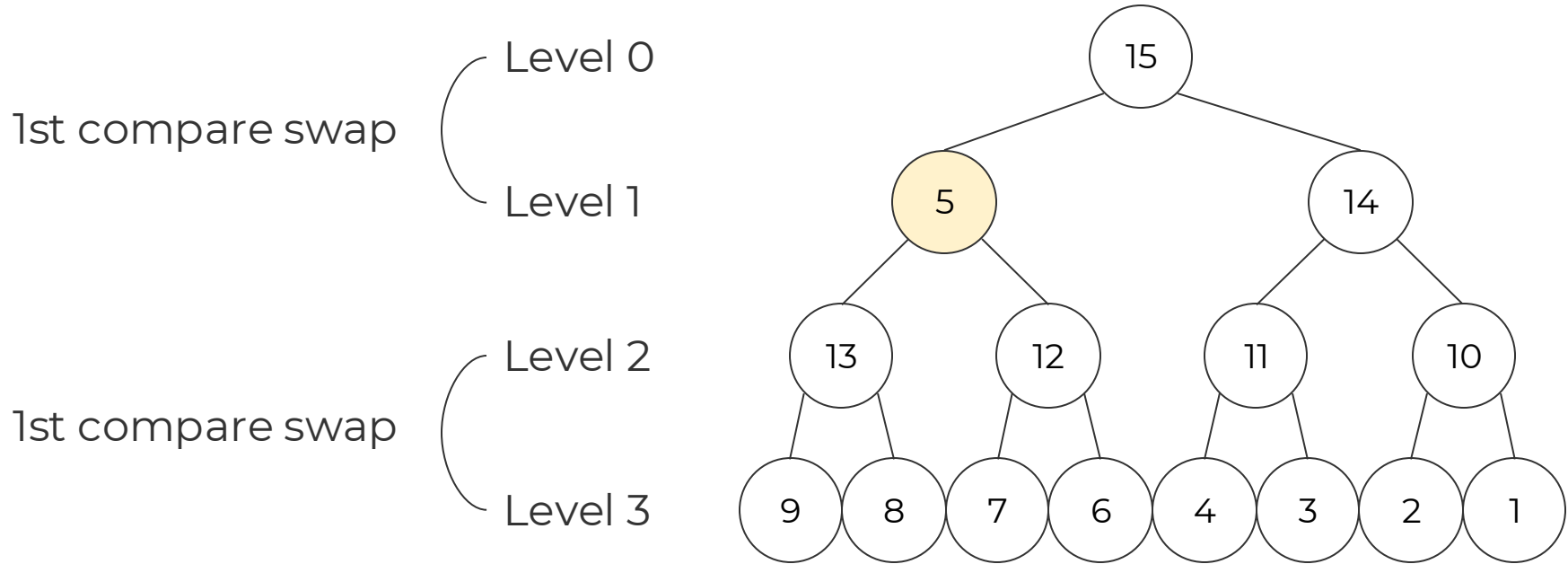
	Enqueue Switch	Enqueue Latency (cycle) *N means queue size	Replace/Dequeue Latency (cycle)
Register Array [1]	Yes	1	1
Register Array (Pipelined)	Yes	2	2
Register Tree [1]	Yes	Log N	1
Register Tree (Pipelined)	Yes	Log N + 1	2
Systolic Array [3]	Yes	1	1
BRAM Tree [1]	No	N/A	8
BRAM Tree (Pipelined)	No	N/A	4
Hybrid Tree [1]	W.I.P.	1 (expected)	1

# Register Tree / BRAM Tree

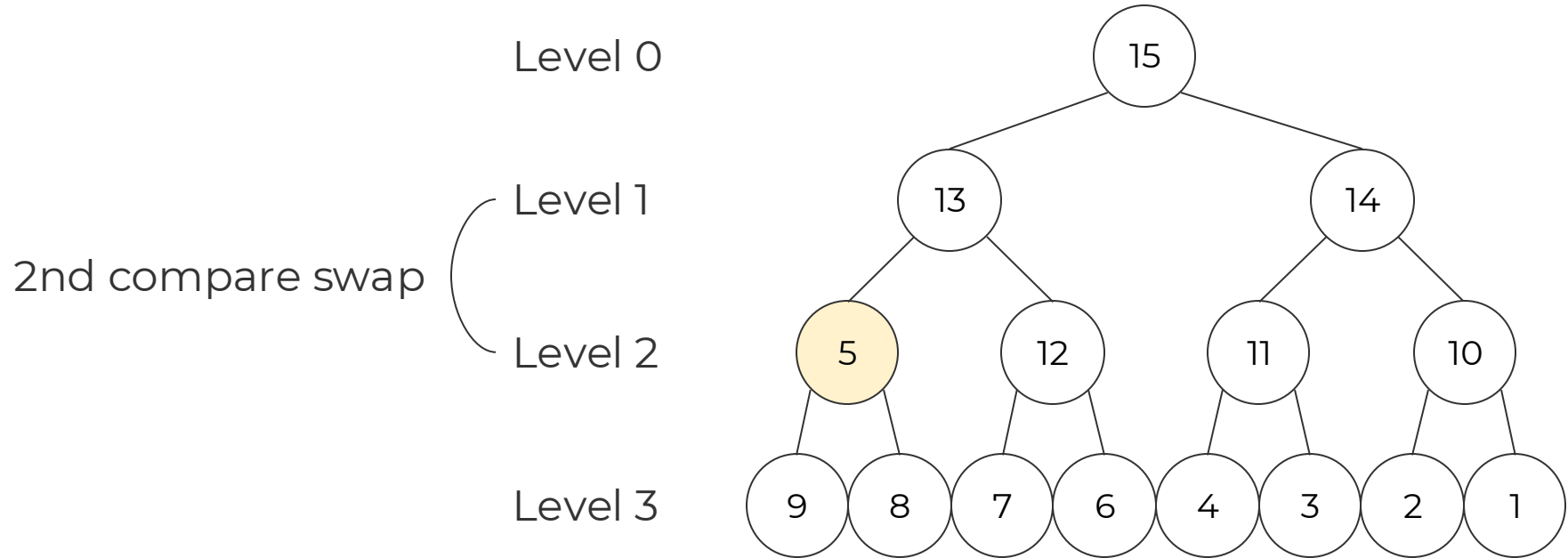
Replace(5)



# Register Tree / BRAM Tree



# Register Tree / BRAM Tree





# Architectures

	Enqueue Switch	Enqueue Latency (cycle) *N means queue size	Replace/Dequeue Latency (cycle)
Register Array [1]	Yes	1	1
Register Array (Pipelined)	Yes	2	2
Register Tree [1]	Yes	Log N	1
Register Tree (Pipelined)	Yes	Log N + 1	2
Systolic Array [3]	Yes	1	1
BRAM Tree [1]	No	N/A	8
BRAM Tree (Pipelined)	No	N/A	4
Hybrid Tree [1]	W.I.P.	1 (expected)	1

# Architectures

	Enqueue Switch	Enqueue Latency (cycle) *N means queue size	Replace/Dequeue Latency (cycle)
Register Array [1]	Yes	1	1
Register Array (Pipelined)	Yes	2	2
Register Tree [1]	Yes	Log N	1
Register Tree (Pipelined)	Yes	Log N + 1	2
Systolic Array [3]	Yes	1	1
BRAM Tree [1]	No	N/A	8
BRAM Tree (Pipelined)	No	N/A	4
Hybrid Tree [1]	W.I.P.	1 (expected)	1

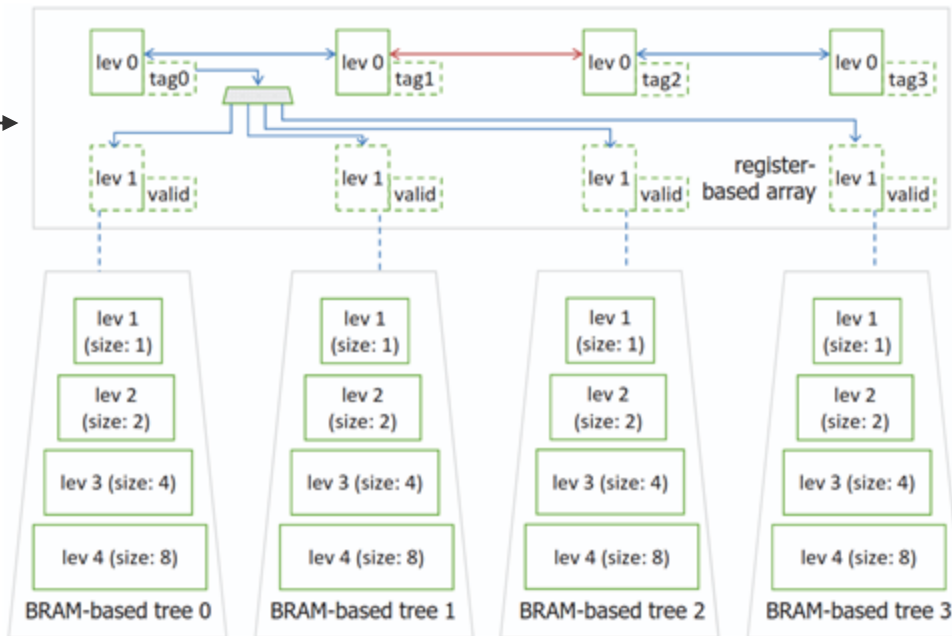
# Architectures

	Enqueue Switch	Enqueue Latency (cycle) *N means queue size	Replace/Dequeue Latency (cycle)
Register Array [1]	Yes	1	1
Register Array (Pipelined)	Yes	2	2
Register Tree [1]	Yes	Log N	1
Register Tree (Pipelined)	Yes	Log N + 1	2
Systolic Array [3]	Yes	1	1
BRAM Tree [1]	No	N/A	8
BRAM Tree (Pipelined)	No	N/A	4
Hybrid Tree [1]	W.I.P.	1 (expected)	1

# Hybrid Tree

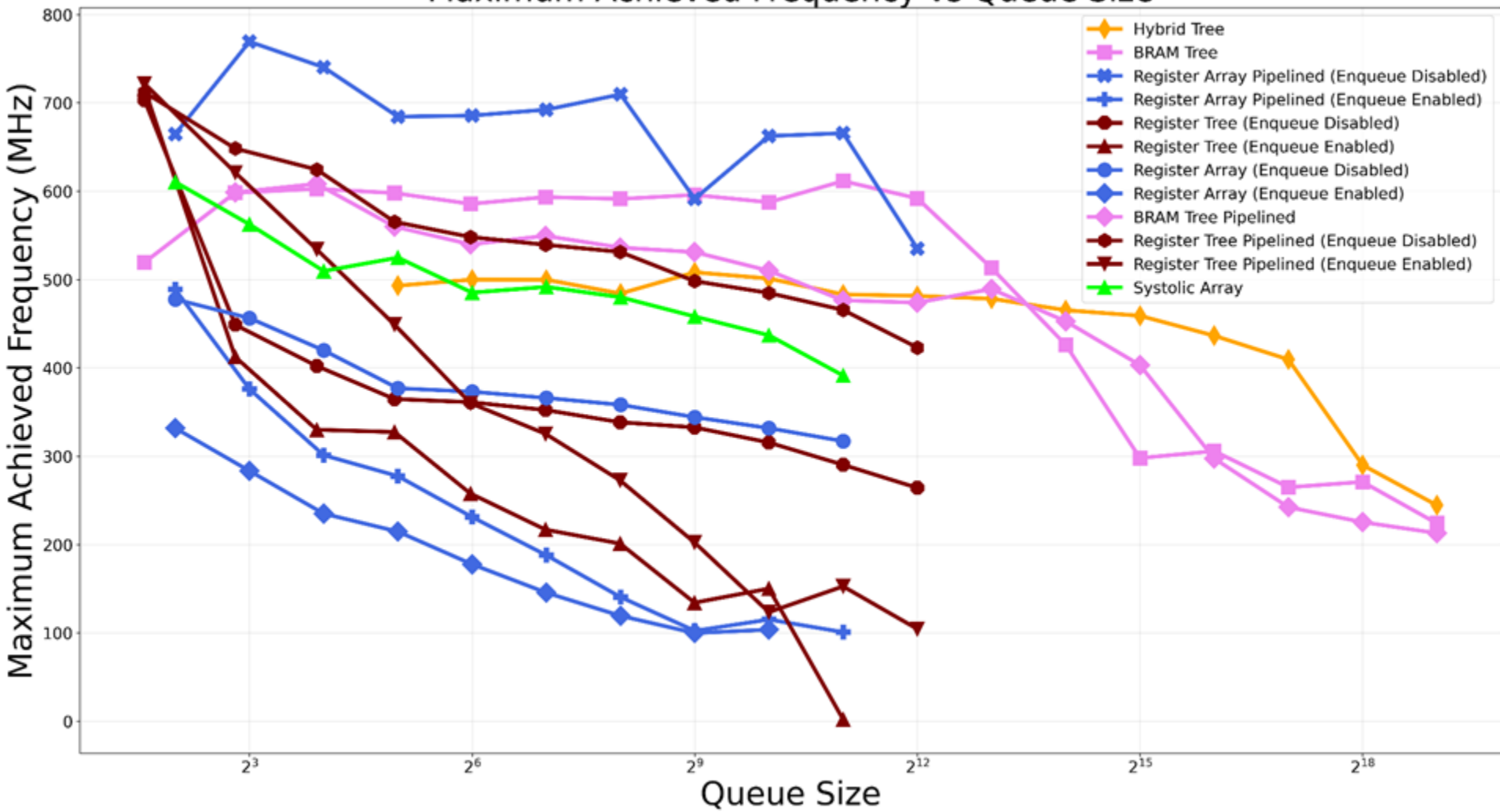
Register Array

BRAM Tree

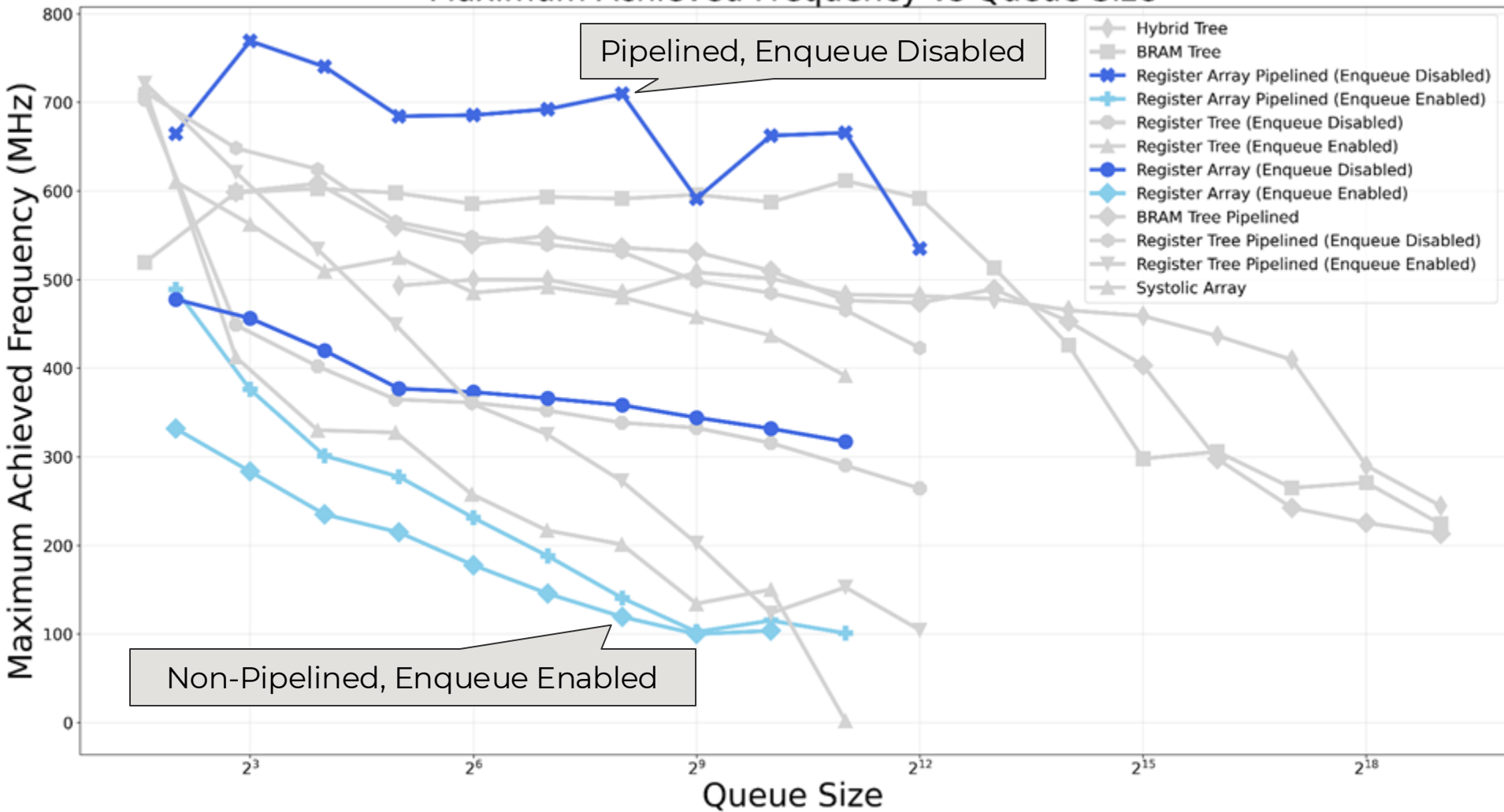


# Experimental Results

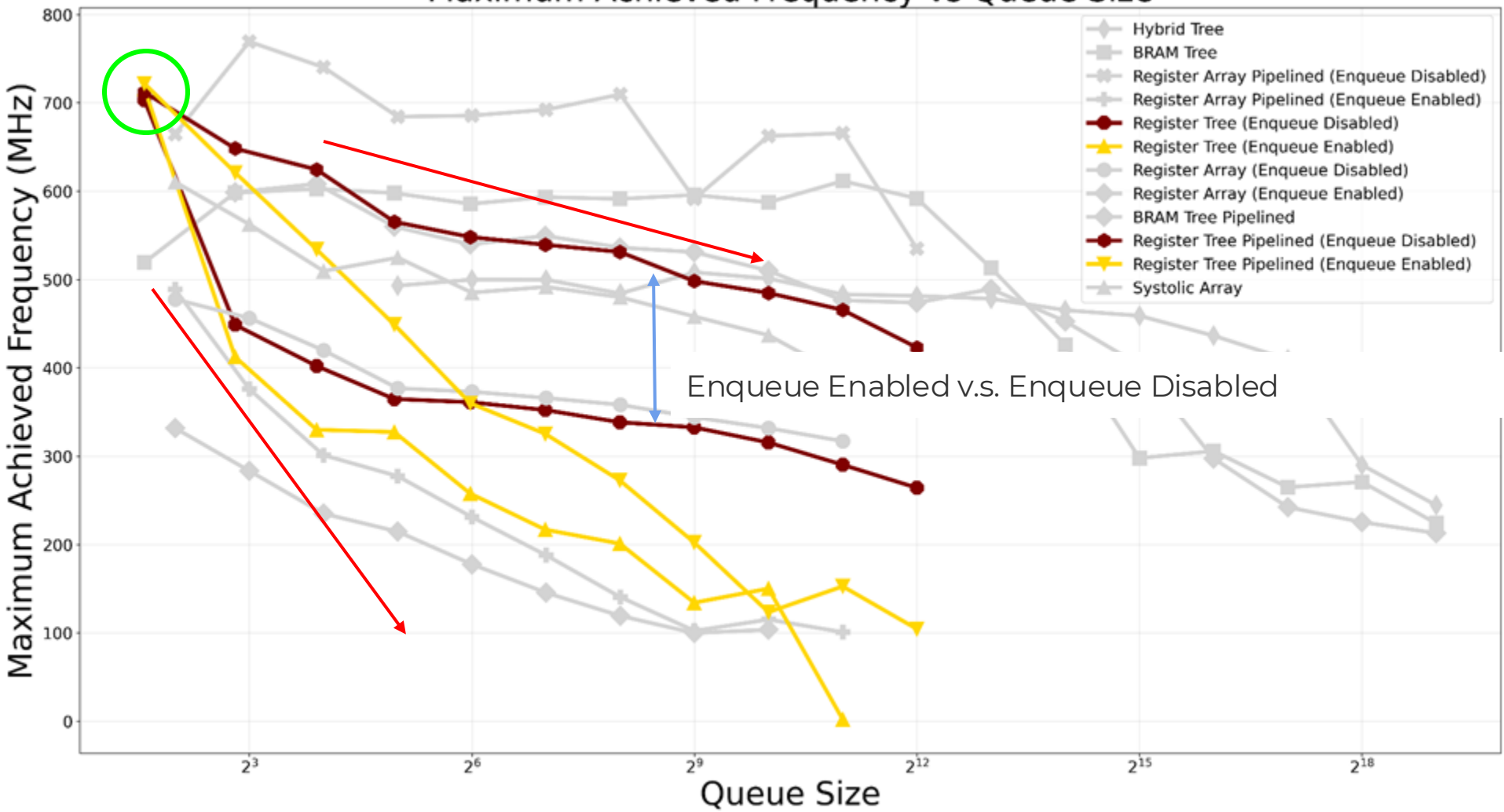
Maximum Achieved Frequency vs Queue Size



# Maximum Achieved Frequency vs Queue Size

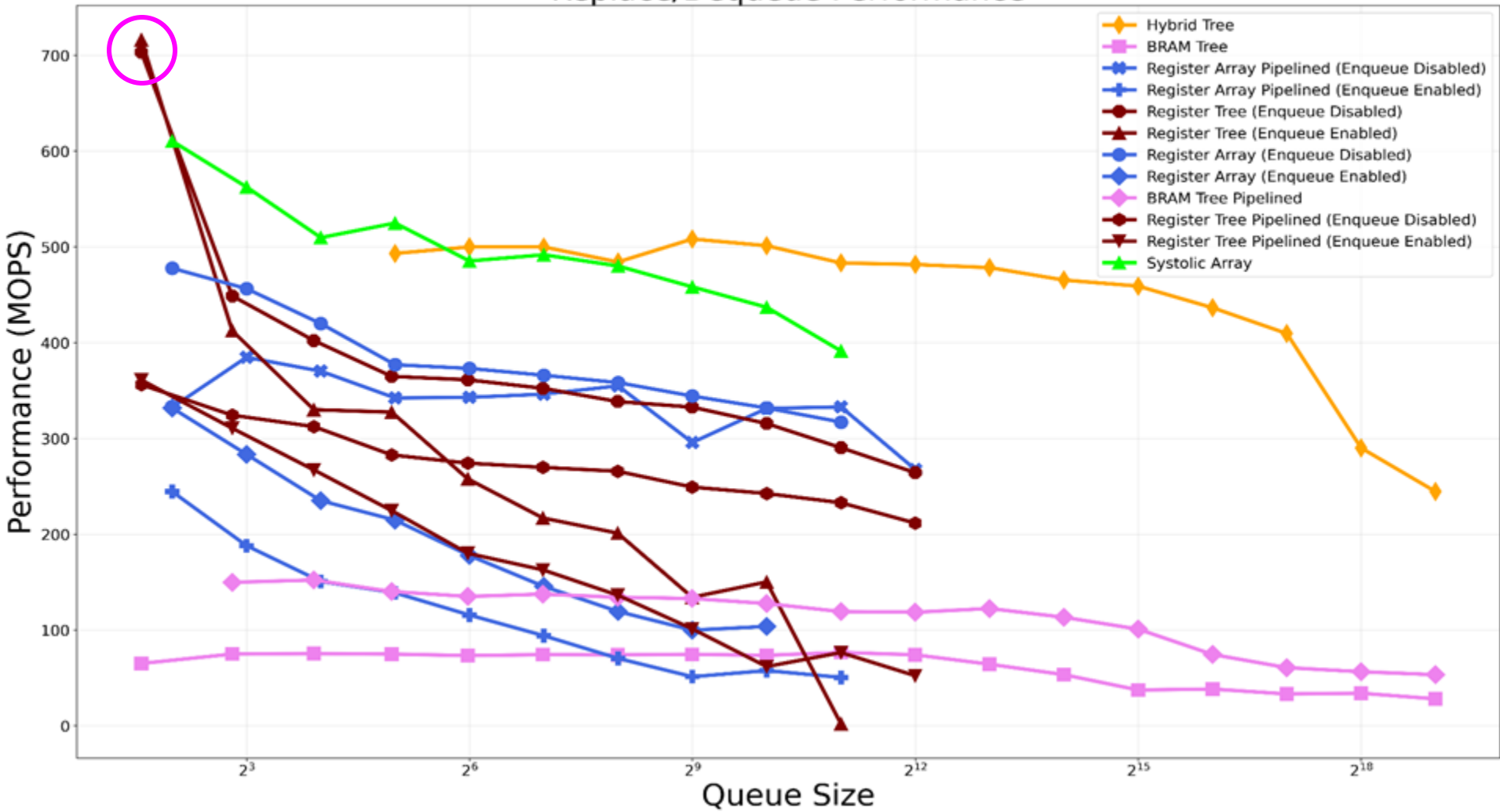


# Maximum Achieved Frequency vs Queue Size

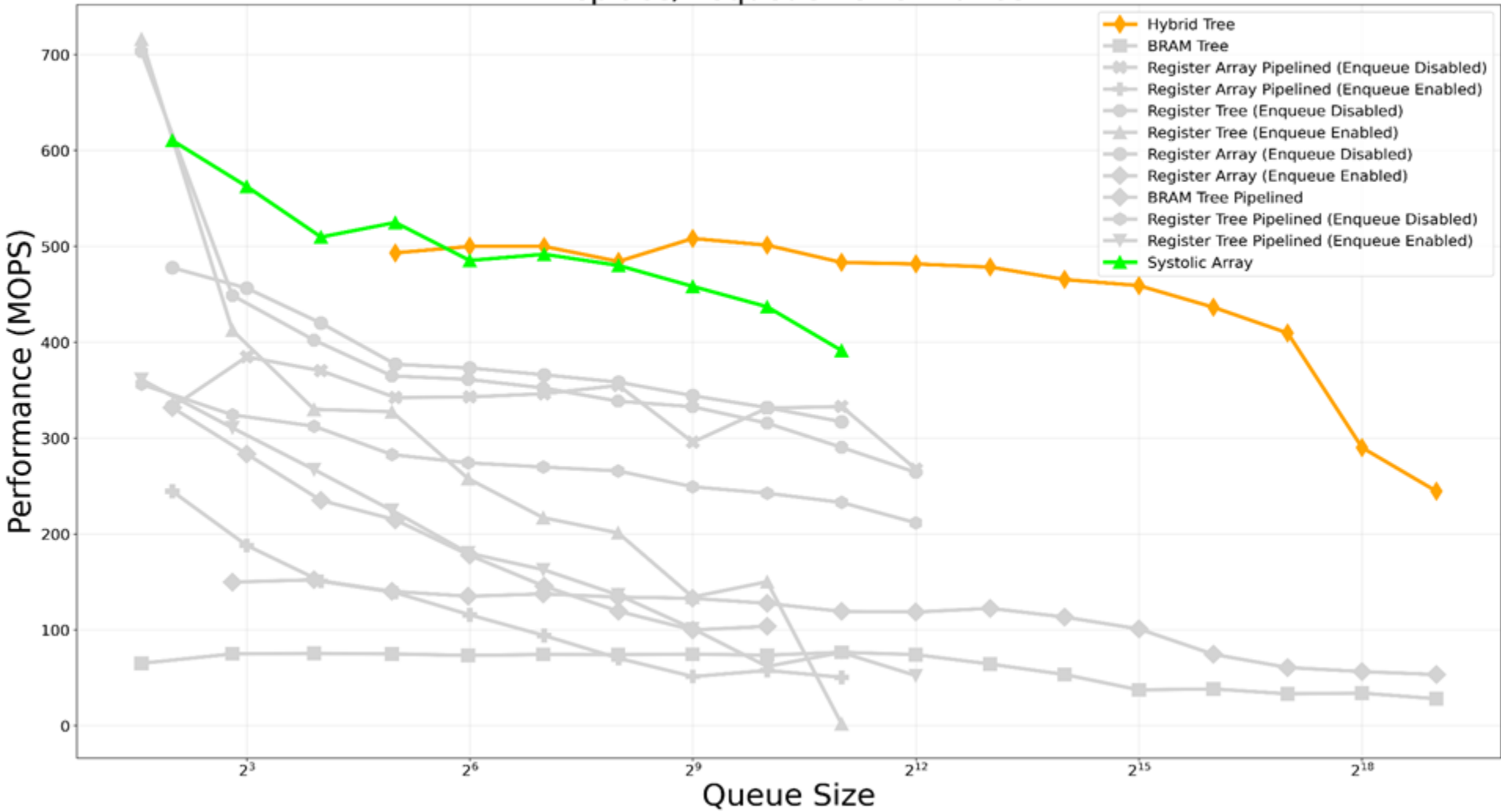




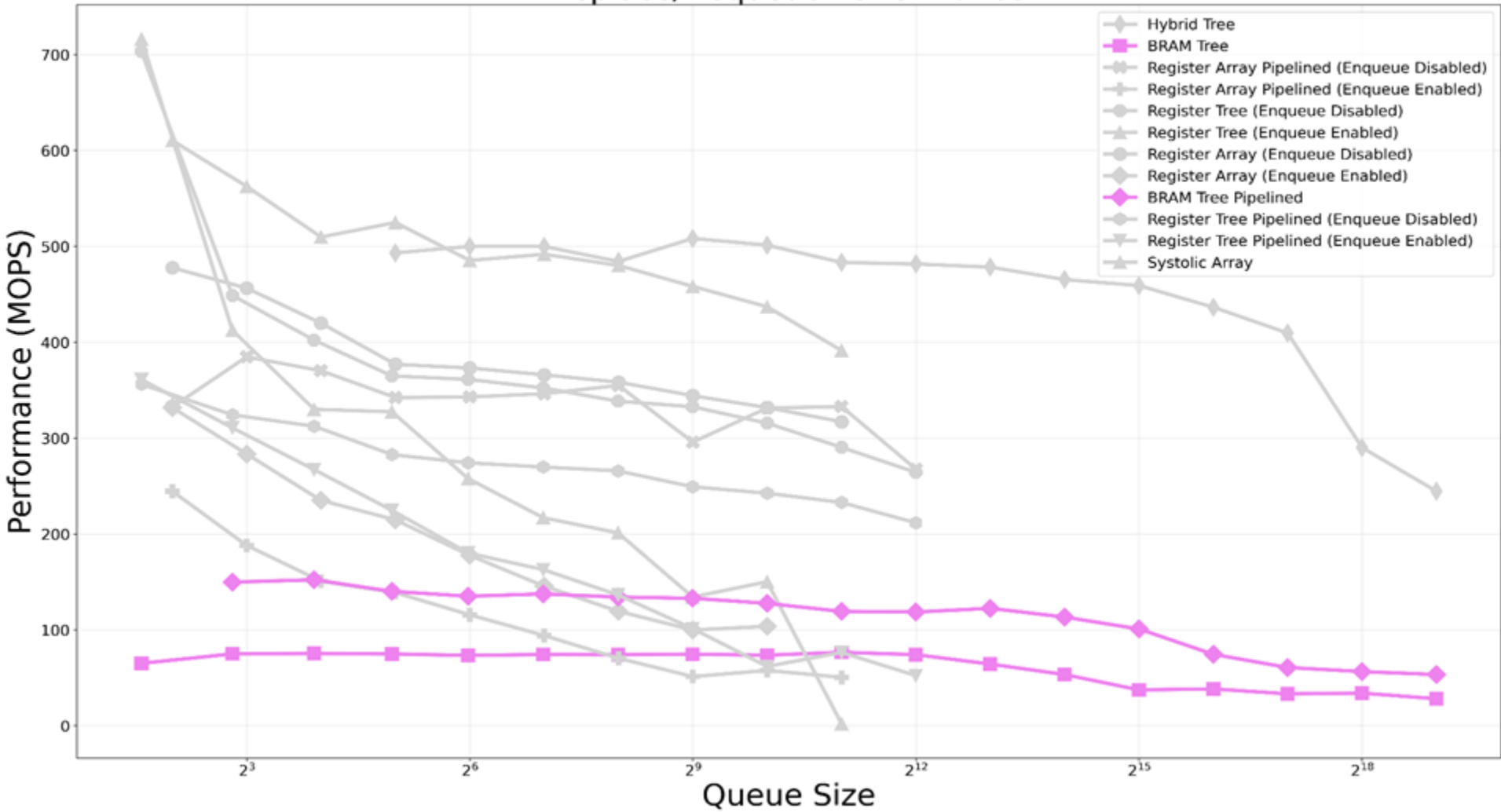
# Replace/Dequeue Performance



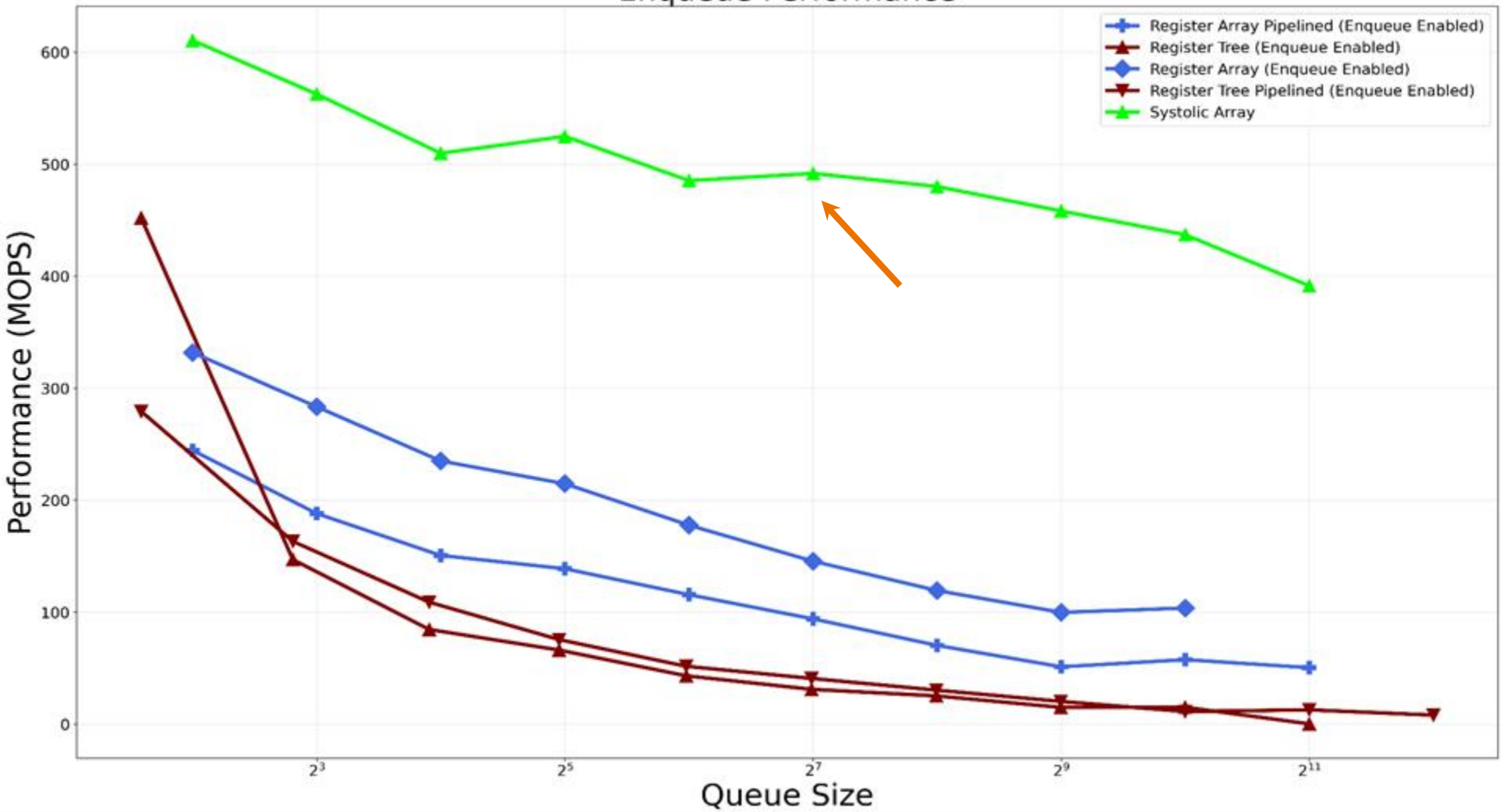
# Replace/Dequeue Performance



# Replace/Dequeue Performance

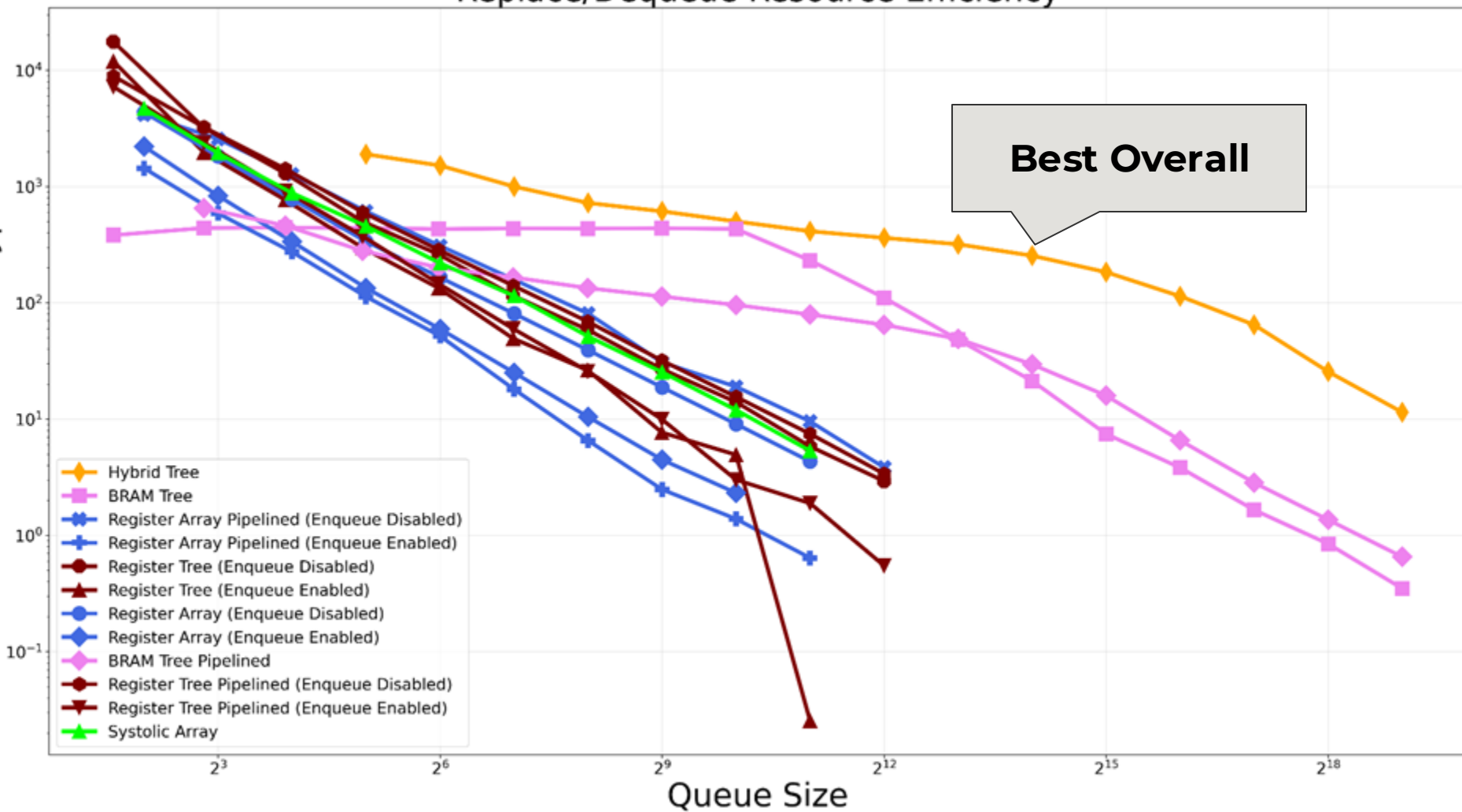


# Enqueue Performance



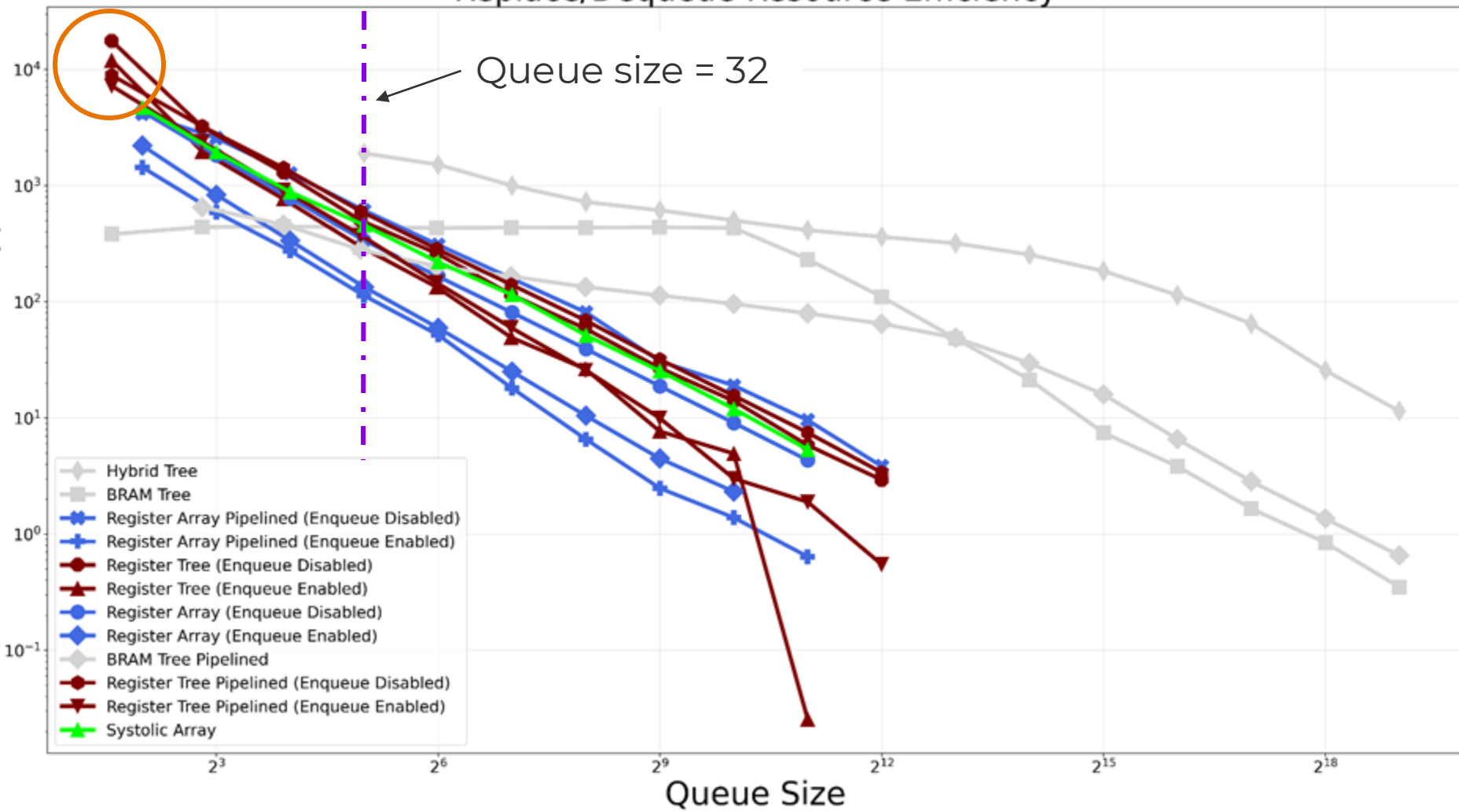
# Replace/Dequeue Resource Efficiency

Performance / Max Resource Type Utilization

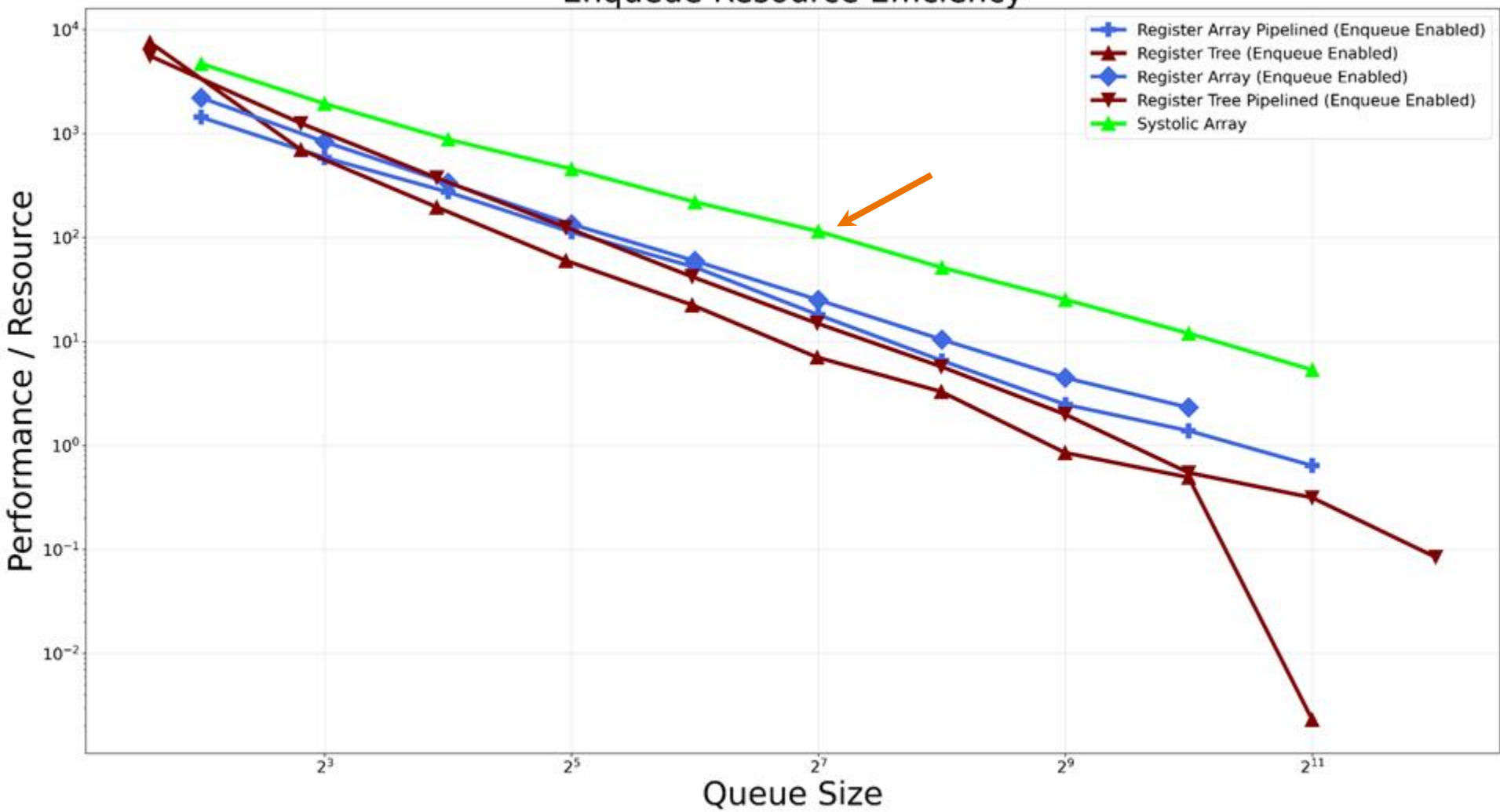


# Replace/Dequeue Resource Efficiency

Performance / Max Resource Type Utilization



# Enqueue Resource Efficiency



# Takeaways

**The “right” architecture always depends on the system requirements and applications’ unique constraints.**

- Architecture selection may depend heavily on the top level design



# Takeaways

**The “right” architecture always depends on the system requirements and applications’ unique constraints.**

- Architecture selection may depend heavily on the top level design
- Small queue (<4 elements)
  - Register Tree

# Takeaways

**The “right” architecture always depends on the system requirements and applications’ unique constraints.**

- Architecture selection may depend heavily on the top level design
- Small queue (<4 elements)
  - Register Tree
- Medium queue (>4, <256 elements)
  - Systolic Array

# Takeaways

**The “right” architecture always depends on the system requirements and applications’ unique constraints.**

- Architecture selection may depend heavily on the top level design
- Small queue (<4 elements)
  - Register Tree
- Medium queue (>4, <256 elements)
  - Systolic Array
- Large queue (>256 elements)
  - Hybrid Tree

# GitHub Library

- File Structure
  - RTL & Simulation
  - Vivado Synthesis Script
  - Python Analysis Script



**HWPQ Repository**

# Reference

- [1] Muhuan Huang, Kevin Lim, and Jason Cong. 2014. A scalable, high-performance customized priority queue. In 2014 24th International Conference on Field Programmable Logic and Applications (FPL). 1–4. <https://doi.org/10.1109/FPL.2014.6927413>
- [2] Sung-Whan Moon, J. Rexford, and K.G. Shin. 2000. Scalable hardware priority queue architectures for high-speed packet switches. IEEE Trans. Comput. 49, 11 (2000), 1215–1227. <https://doi.org/10.1109/12.895938>
- [3] Yuzhi Zhou, Xi Jin, Tianqi Wang, and Jose A. Boluda. 2020. FPGA Implementation of A Star Algorithm for Real-Time Path Planning. Int. J. Reconfig. Comput. 2020 (Jan. 2020), 11 pages. <https://doi.org/10.1155/2020/889638>

# Q&A



**HWPQ Repository**