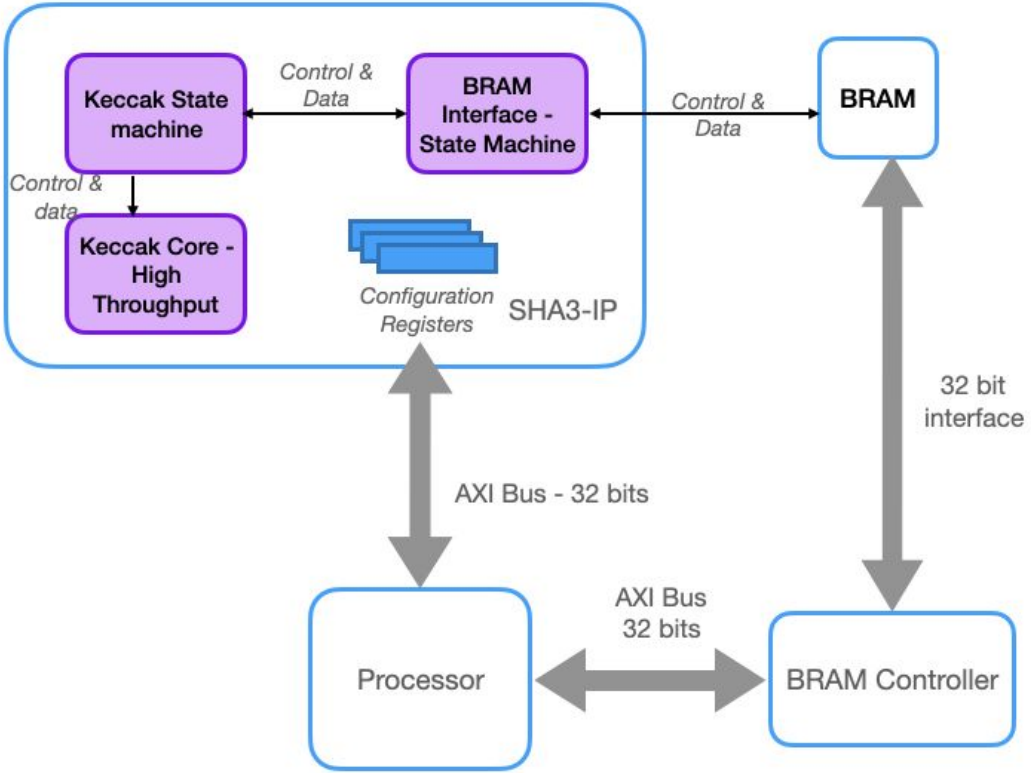
Optimizing SHA-3

Malav, Dinesh and Mugdha

Background

- Cryptographic hash: arbitrary input -> fixed sized hash value
- SHA-3 developed as an alternative for attacks on SHA-1 and SHA-2
- Keccak is a winning SHA-3 algorithm
    - First hash that National Institute of Standards and Technology (NIST) adopted
    - Sponge construction (rounds of padding, permutation, and logical operations)
    - 224, 256, 384, and **512 bits (highest security)**
- Two starting baselines:
    - One with a lot of bugs (don’t think they verified it)

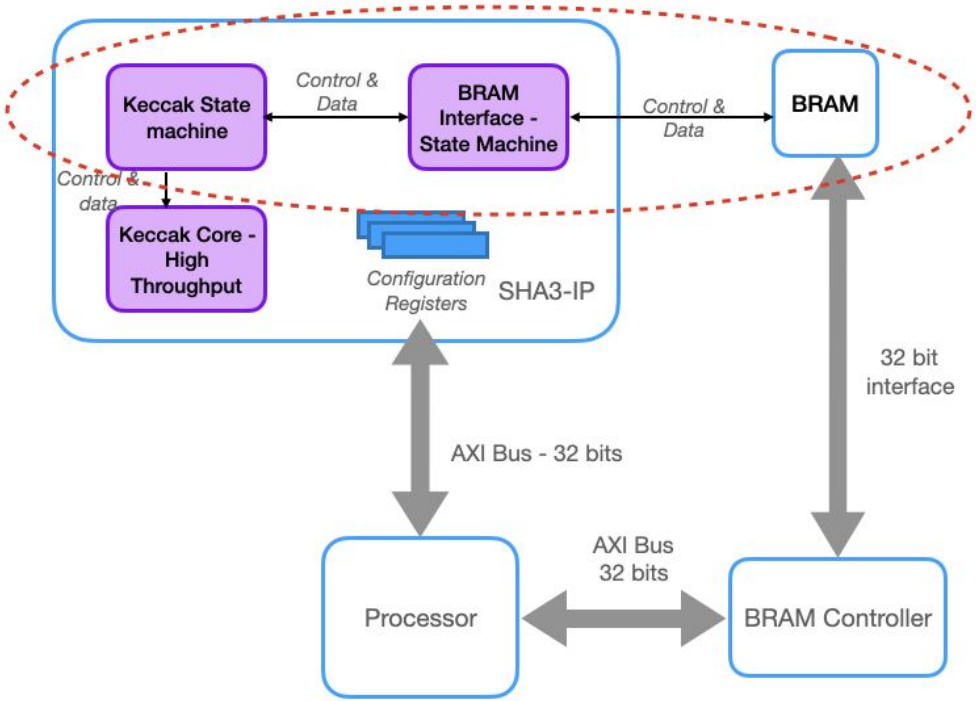
Baseline: Architecture



- Write Data to BRAM
- SHA IP reads from BRAM through BRAM Interface
- Read hashed value in SHA IP registers

Baseline: Areas to Improve

- Keccak core buffers 8 bytes at once
  - BRAM reads 4 bytes (32 bits)
  - 2 BRAM reads per one keccak input
- Improve BRAM interface for faster or parallel reading

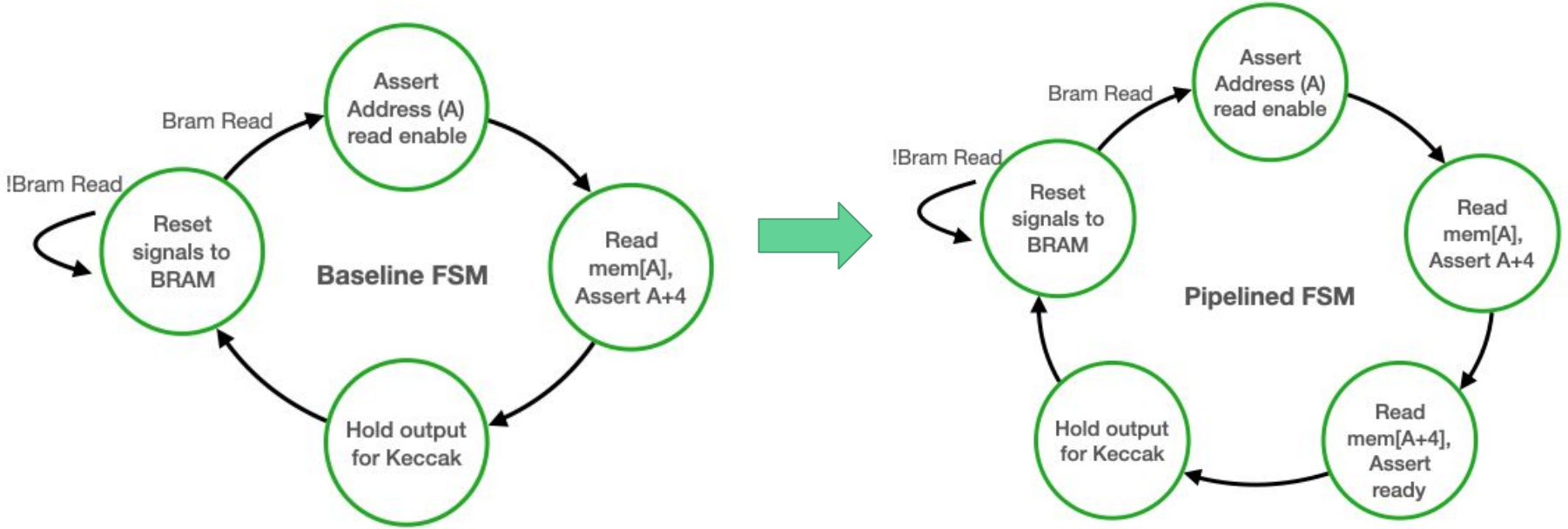


Approaches

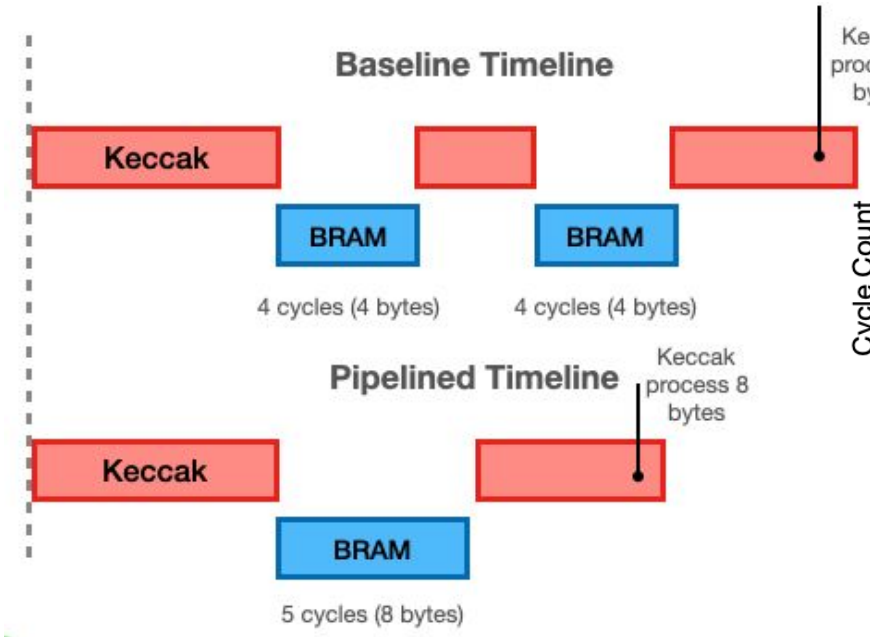
1. Two Pipelined Reads (Malav)
2. Increase Bus Widths (Mugdha)
3. Parallel Reads (Dinesh)
4. Pipeline + Parallelism with FIFO (Malav)

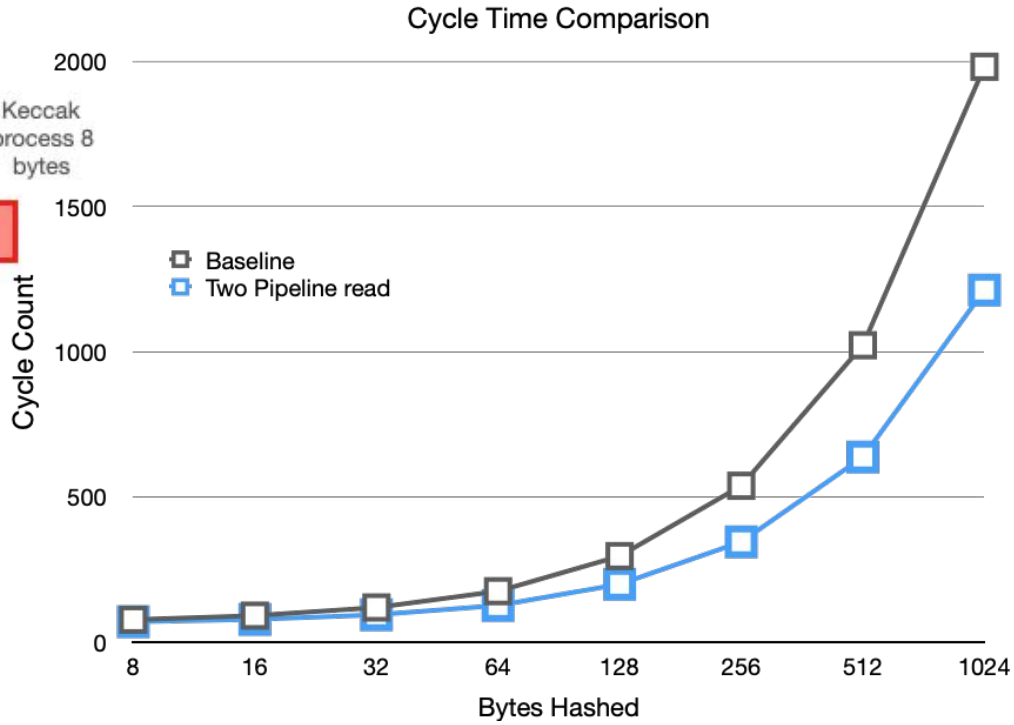
A1. Pipelined Read: Architecture

- Take advantage of spatial locality with extra state



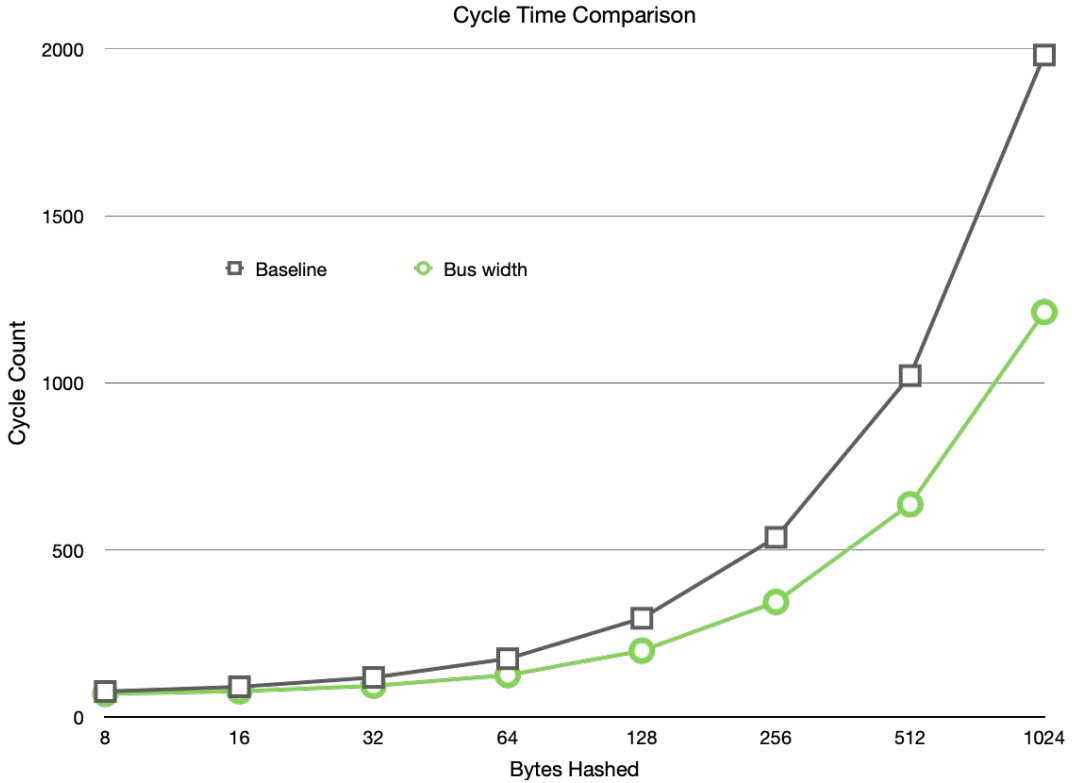
A1. Pipelined Read: Data



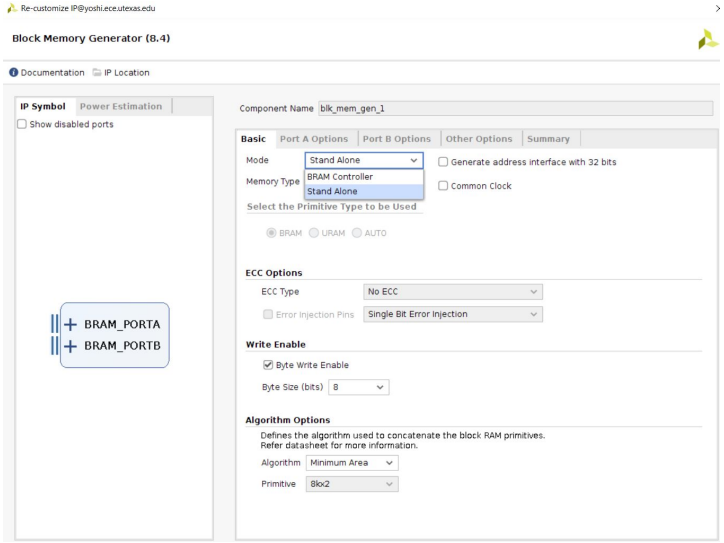


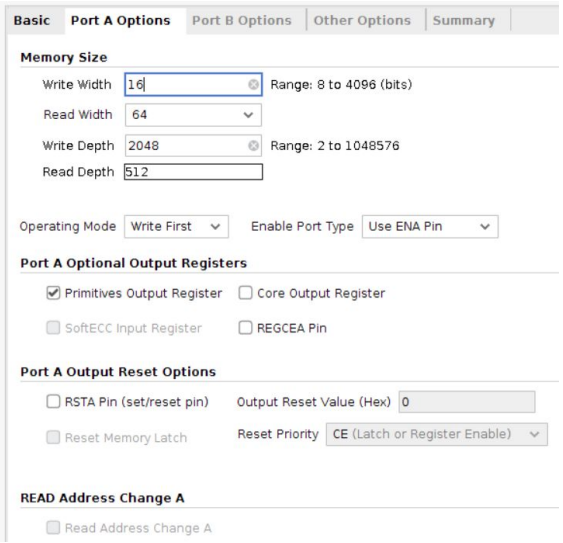
A2. Increasing Bus Widths: Data

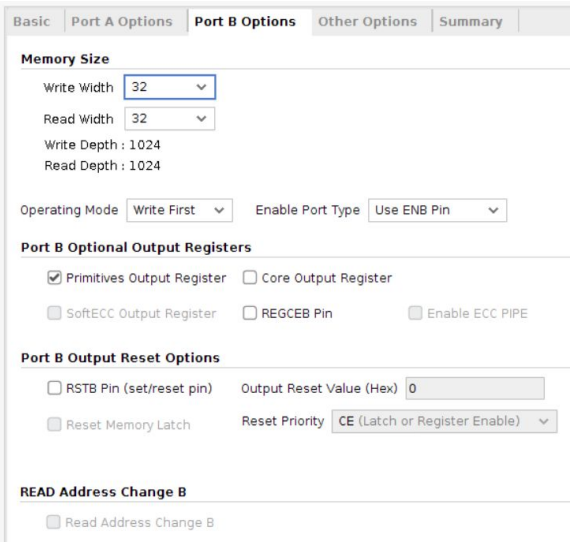
- 32 bits to 64 bit bus width
  - Reduce # of reads from 2 to 1 BRAM read for 8 bytes
- Reduced the states in Keccak state machine.



A2. Increasing Bus Widths: BRAM Configurations





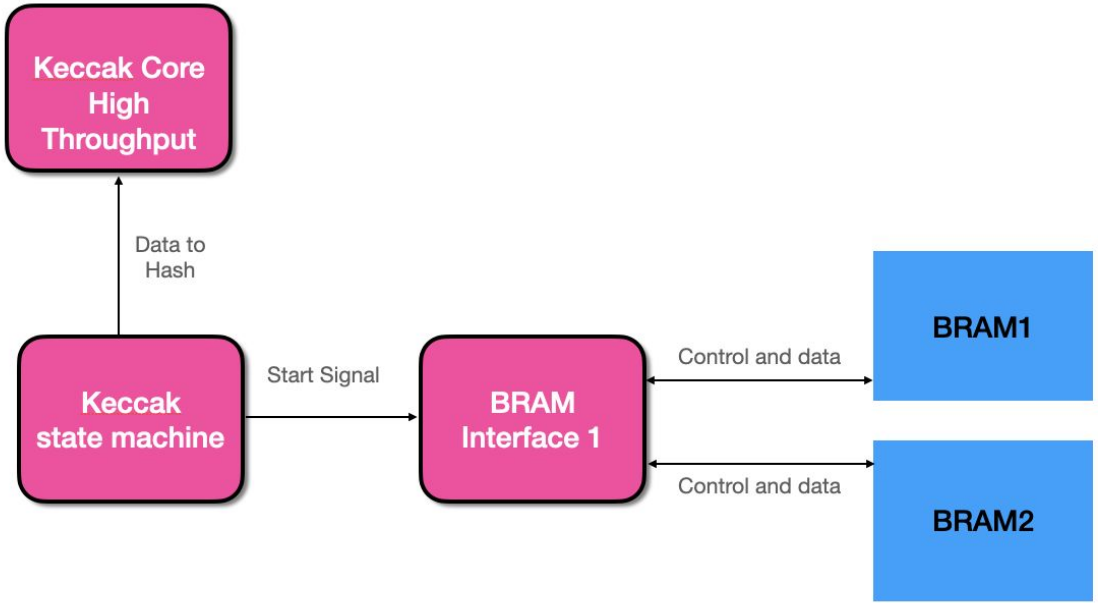


References: Block Memory Generator v8.4 LogiCORE IP Product Guide - Xilinx

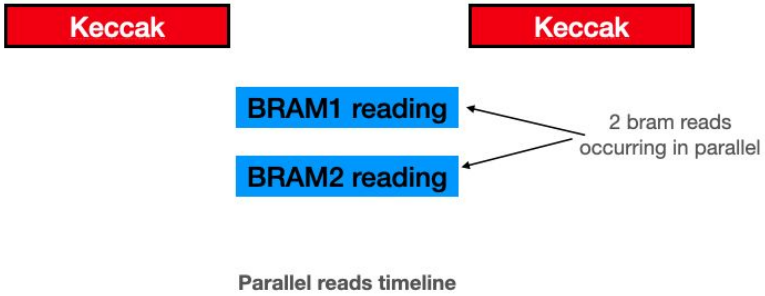
A2. Increasing Bus Widths: BRAM Configurations (Contd.)

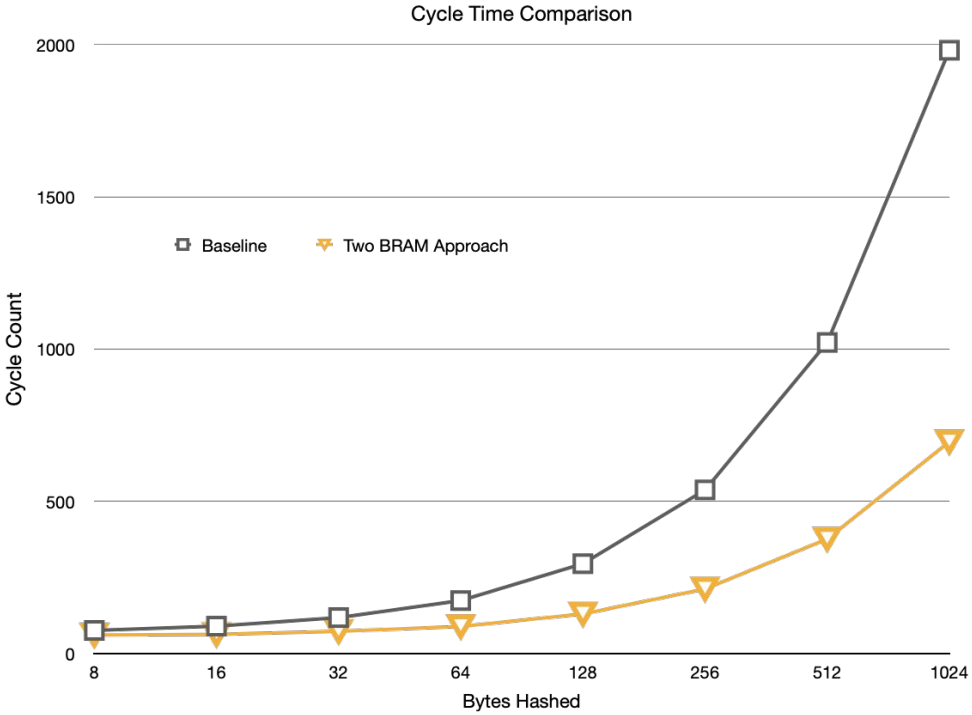
Native Interface Core	AXI Interface Core
- Generates Single-Port RAM, Simple Dual-Port RAM, True Dual-Port RAM, Single-Port ROM, or Dual-Port ROM - Performance up to 450 MHz - Data widths from 1 to 4096 bits - Memory depths from 2 to 128k - Variable Read-to-Write aspect ratios - Option to optimize for resource or power - Ability to initialize the memories with predefined values - Supports individual Write enable per byte with or without parity - Selectable per-port operating mode - Support for hard and soft ECC	- Generates Dual Port RAM - Performance up to 300 MHz - Data widths ranging from 8 to 64 bits

A3. Parallel Reads: Architecture



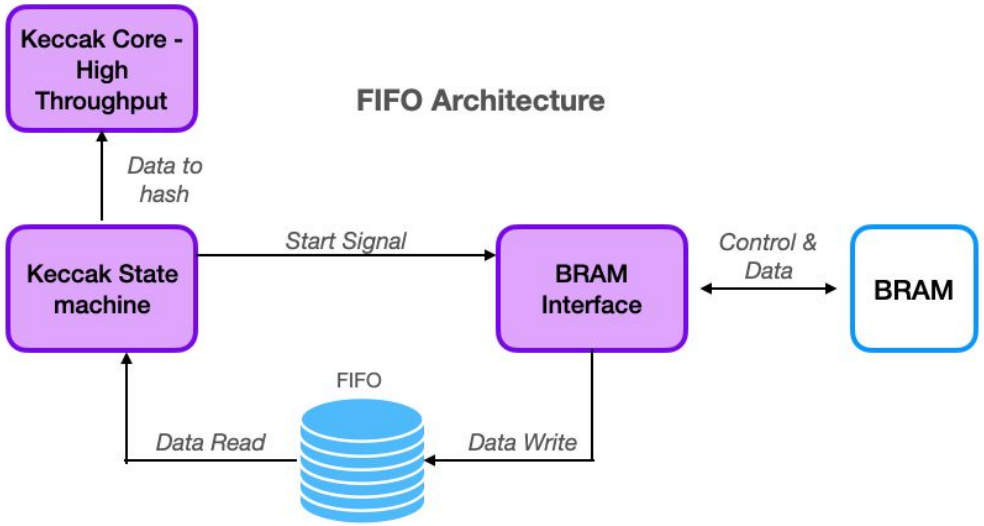
A3. Parallel Reads: Data



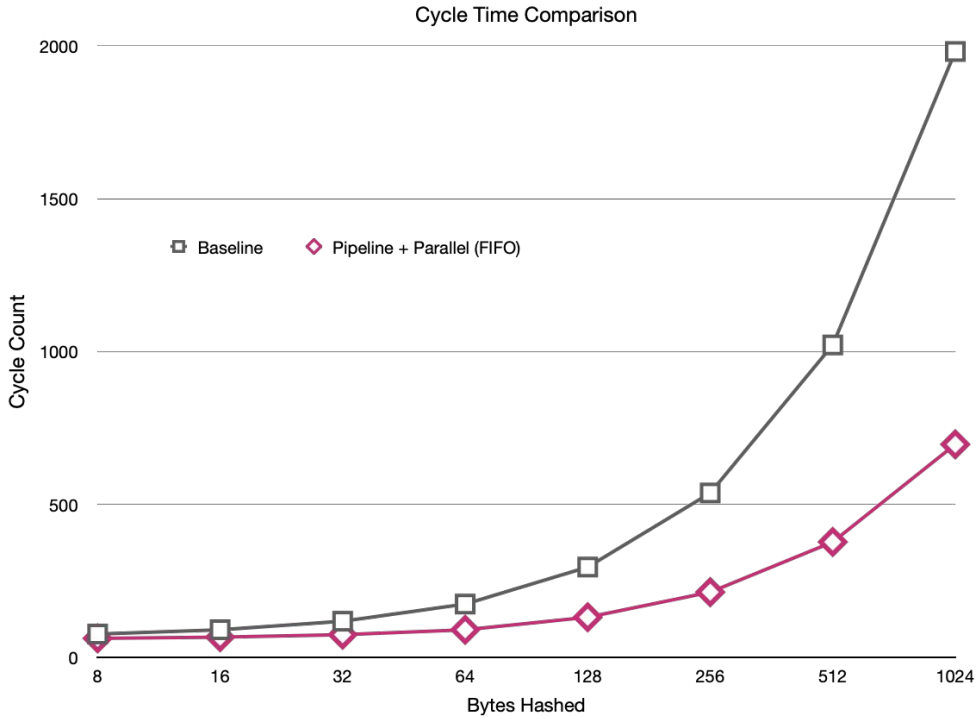


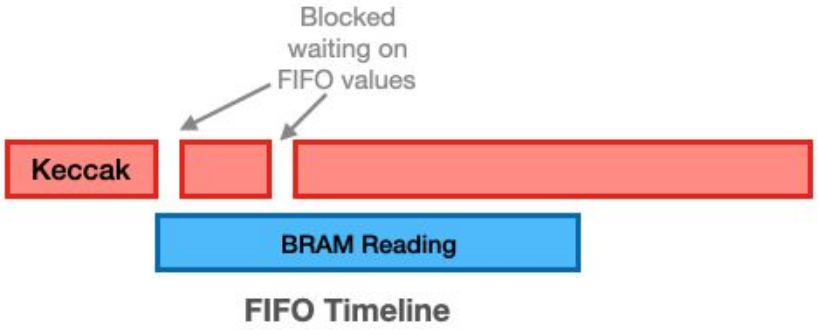
A4. Parallel + Pipeline (FIFO): Architecture

- Decouple BRAM Interface and Keccak State Machine
- FIFO large to remove wait states in BRAM interface

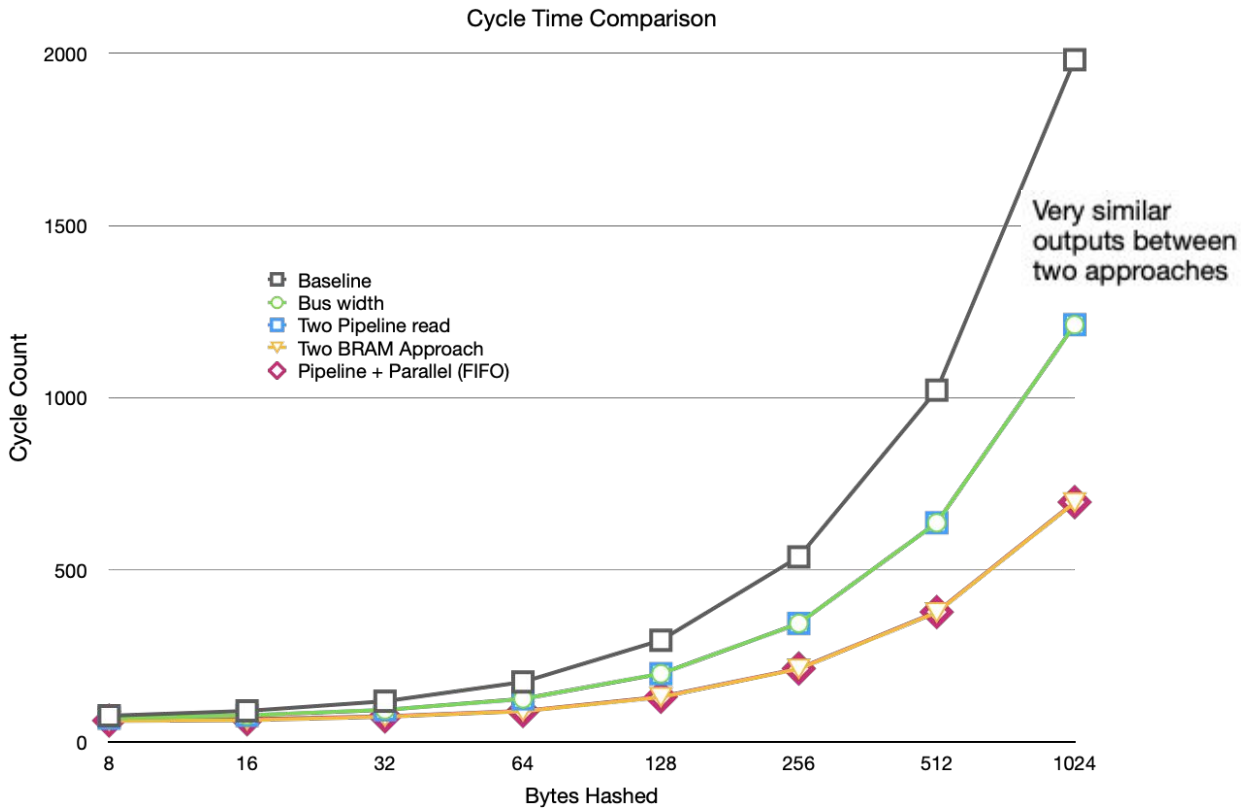


A4. Parallel + Pipeline (FIFO): Data





Data Comparisons Across Approaches



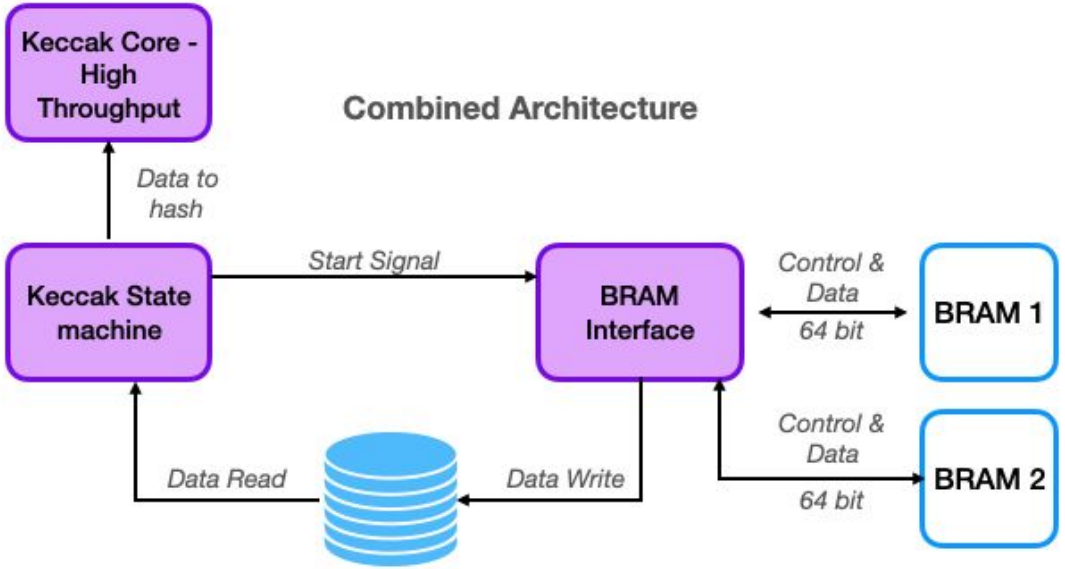
Bytes Hashed	Two Pipelined Read	Bus Width	Two BRAM Approach	FIFO Approach
8	9%	9%	19%	18%
16	14%	14%	30%	26%
32	21%	21%	38%	37%
64	28%	28%	49%	48%
128	33%	33%	56%	55%
256	36%	36%	60%	60%
512	38%	38%	63%	63%
1024	39%	39%	65%	65%

Observations

- Similar cycles between FIFO and Two BRAM
  - Reaching Keccak as bottleneck
  - Or more blocked states in FIFO approach
- Similar cycles Parallel Read and 64 bit bus
  - State machines are similar

Combining Approaches (Theoretical)

- Combine all approaches
  - Two BRAMs with 64 bit and FIFO
  - 4x data in FIFO to reduce blocked times



Key Learnings

- Optimization comes at the expense of memory
- Debugging: More data the better
  - Register Readings
  - Sticky Bits
  - Counters
  - State machines
- Keccak Verilog Code usage
  - The core sometimes does not properly reset when reset asserted

Demo

Questions?

Raw Data Captures - For back up, NOTE: ALL data captured at 150 PL Frequency

Bytes Hashed	Baseline	Two Pipeline read	Bus width	Parallel Read	Pipeline + Parallel (FIFO)
8	77	70	70	62	63
16	91	78	78	64	67
32	119	94	94	74	75
64	175	126	126	90	91
128	296	199	199	131	132
256	538	345	345	213	214
512	1022	637	637	377	378
1024	1981	1212	1212	696	697