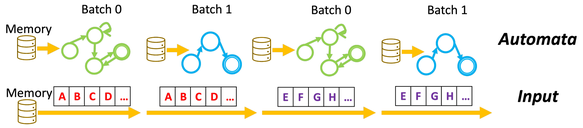
# Exploiting Sparsity to Design An Accelerator for Automata Processing

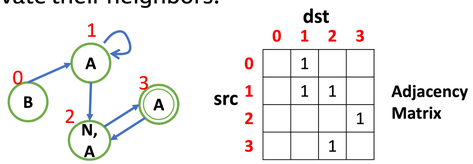
Xingran (Maggie) Du, Joel S. Emer, Daniel Sanchez {xrdu, emer, sanchez}@csail.mit.edu

## Introduction and Background

Input symbols are streamed from the host system while automata are stored on-chip. Reducing automata storage can reduce the number of batches needed for processing the whole workload, improving throughput.




Each automaton is represented as a graph. Graph nodes match against the input symbol, and activate their neighbors.




dst

| src | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | 1 | | |
| 1 | | 1 | 1 | |
| 2 | | | | 1 |
| 3 | | | 1 | |

Adjacency Matrix

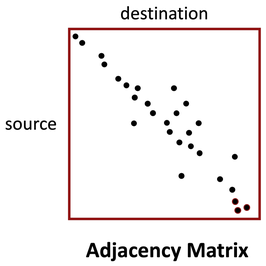
Input: B, A, N, A, N, A, ...

| Input | Frontier | Matched |
|---|---|---|
| B | 0 | 0 |
| A | 1 | 1 |
| N | 1,2 | 2 |
| A | 3 | 3 |

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| A | | 1 | 1 | 1 |
| B | 1 | | | |
| N | | | 1 | |
| ... | | | | |

Match Set

The adjacency matrix is ***sparse***, but prior work inefficiently store it ***uncompressed.***



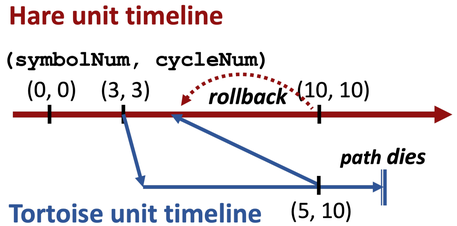

- Each graph node only has **1 to 2 neighbors** (1 to 2 non-zeros (black dots) per row)
- 200 nodes take **200x200 bits** to store uncompressed, while only having **400 non-zero elements**, causing **storage inefficiency**

## Our Approach

We use **compressed** format in addition to uncompressed format.

We design a hybrid system to **make the common case fast, and the uncommon case cheap**.

- *Hare units* use uncompressed format to provide high-throughput for frequently accessed nodes
- *Tortoise units* use compressed format to store the remaining nodes efficiently, with lower processing throughput
- Tortoise units *rarely* interrupt Hare units and slowdown execution




## Key Contributions

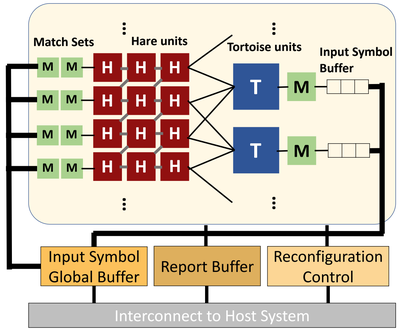
- We frame automata processing as sparse computation, and analyze its optimization opportunities
- We design an automata processing accelerator that extensively **exploits sparsity** in automata graphs
- We propose a novel approach that combines uncompressed and compressed representations to achieve both **high throughput** and **high space efficiency**
- We evaluate these techniques in depth, showing their performance gains

## Architecture

### System Diagram

Only local communication is needed within and between Hare and Tortoise units.

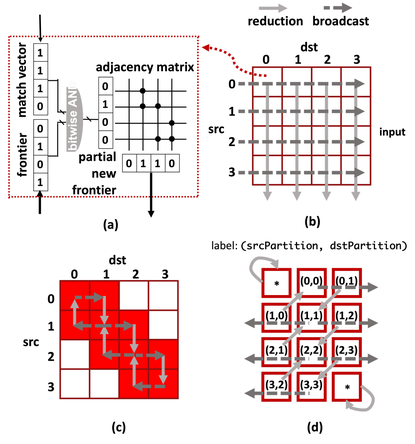
Units are laid out in columns and replicated in the vertical dimension.




### Hare Unit Design

Only storing frequent nodes allows Hare units to be more efficient even with uncompressed format.

Small uncompressed blocks are composed diagonally to store a full adjacency matrix.




## Evaluation

We evaluate our system on the AutomataZoo benchmark suite and achieve 2.5x and 2.2x speedup over prior accelerators.