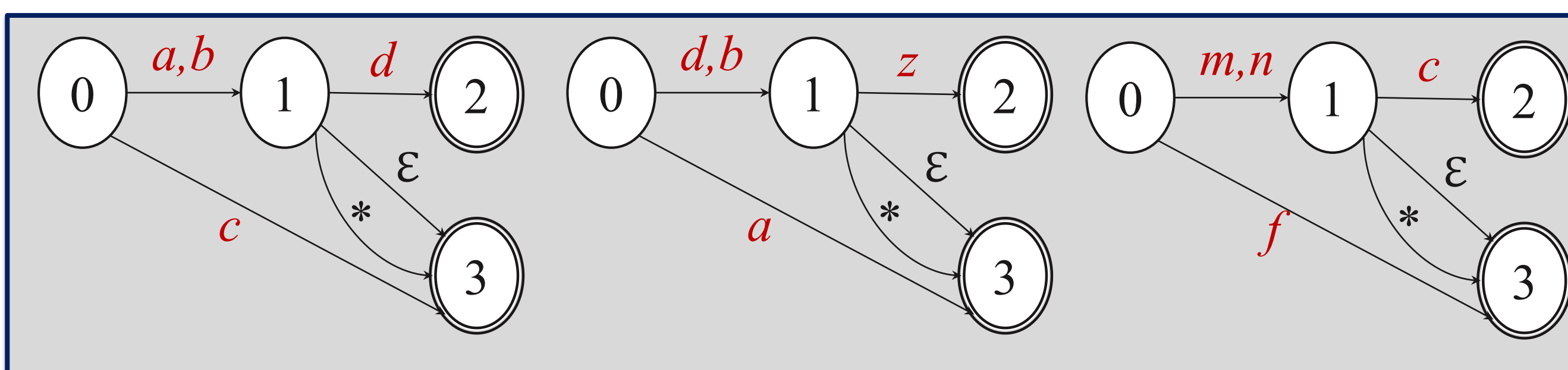


Introduction

Pattern matching can be performed using **non-deterministic finite automata (NFAs)** by traversing the NFA guided by the symbols in the input string. NFA processing acceleration has been studied on various parallel platforms. **Many existing acceleration methods store the finite automata states and transitions in memory.** As a result, for these designs, memory size and bandwidth are the main limiting factors to performance. Many applications, require processing several fixed-topology automata that differ only in the symbols associated to the transitions: this property enables the design of alternative, memory-efficient solutions.

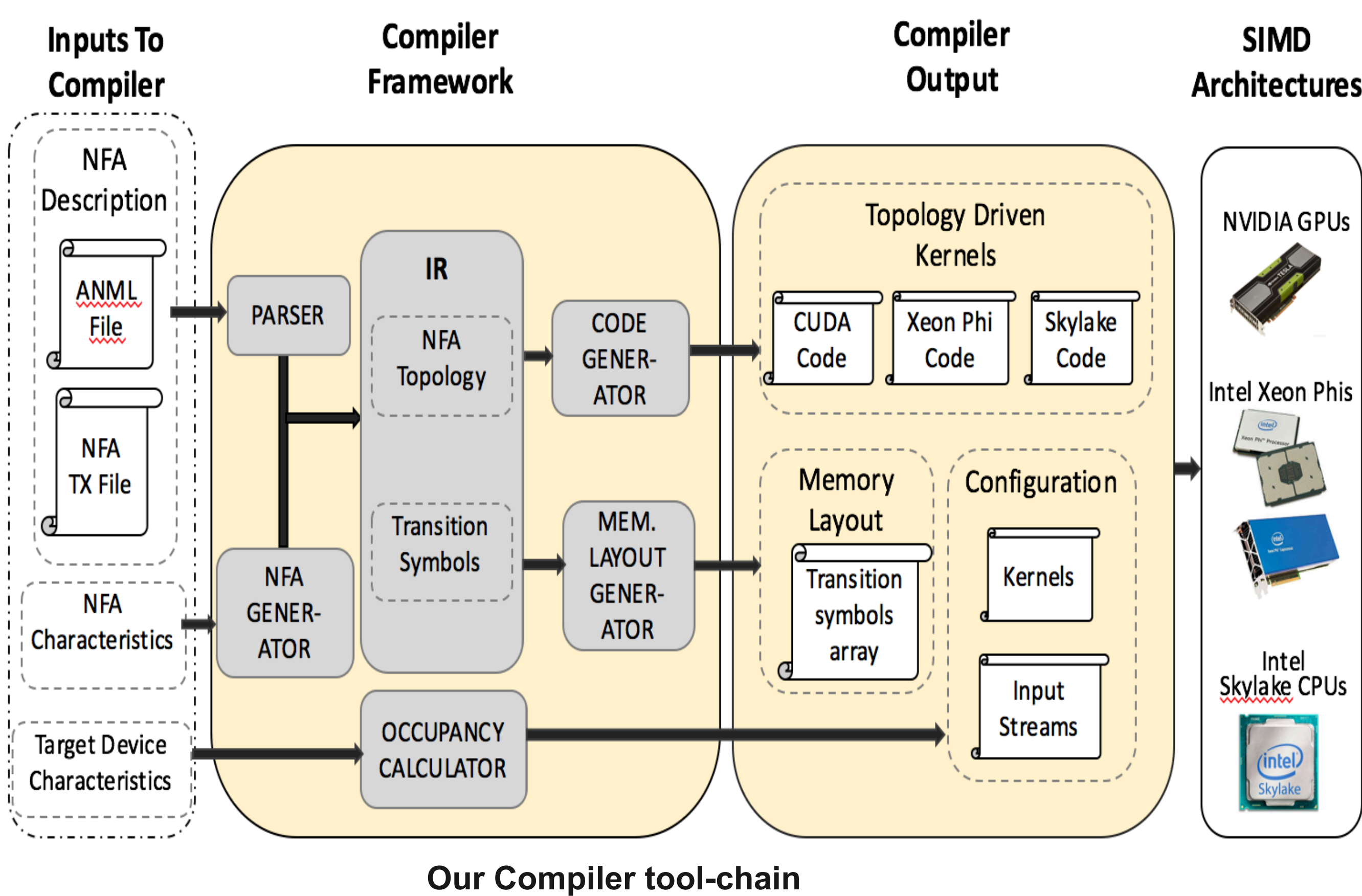


Fixed-topology NFAs

Only transition symbols differ (shown in red)

We target fixed-topology NFAs and propose a SIMD-friendly design, called *SIMD_NFA*, that **embeds the automata topology in code and stores only the transition symbols in memory.** We design a compiler that automates deployment of this design on SIMD platforms for a set of fixed-topology NFAs. Our compiler framework performs a combination of platform-agnostic and platform-specific design decisions and optimizations. We showcase our compiler framework on GPU, Intel Xeon Phi and Intel Xeon Skylake platforms.

Proposed Framework

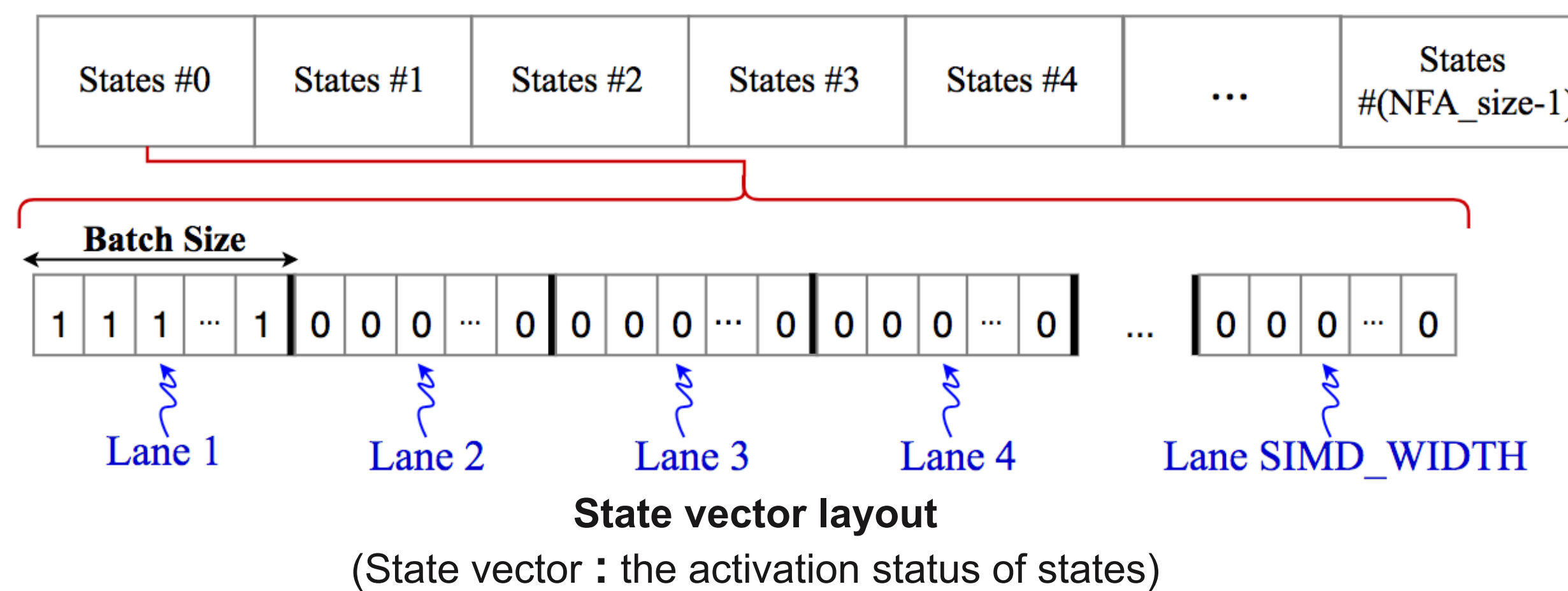


Our Compiler tool-chain

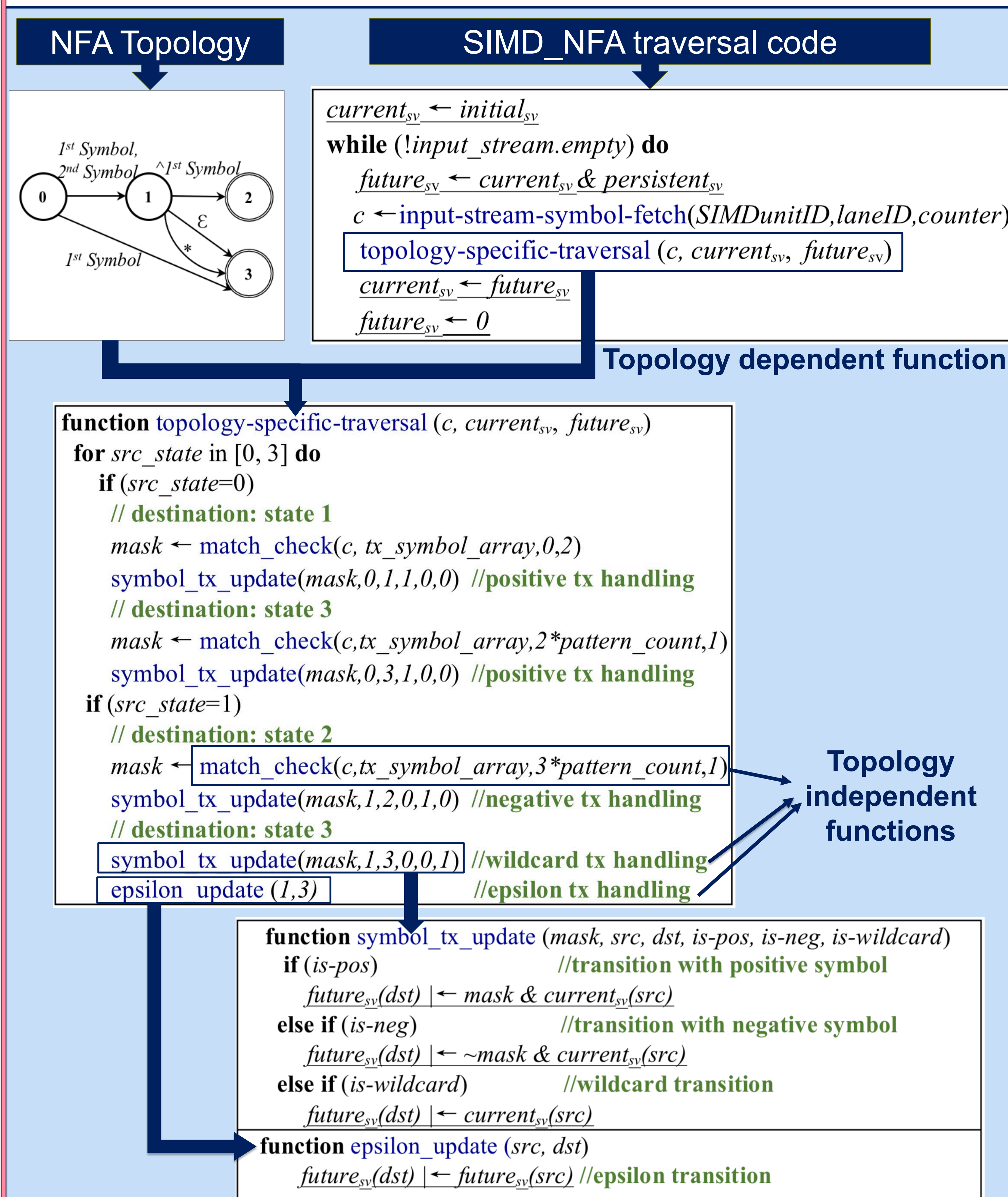
- Two compiler input files:
 - One that encodes the set of NFAs (i.e., patterns) to be deployed
 - One that describes the hardware configuration of the target device
- Compiler generates the memory-layout of the transition symbols using pattern-specific information
- Compiler encodes a parallel traversal kernel based on the NFA topology
- Compiler distributes input streams to the hardware resources to optimize the hardware utilization and maximize the throughput

Methodology

- Sequential processing of each state of NFA topology
- Parallel processing of different NFAs
- Each parallel lane in a SIMD unit is assigned a batch of NFAs (ensures regular memory access and limits divergence)



(State vector : the activation status of states)

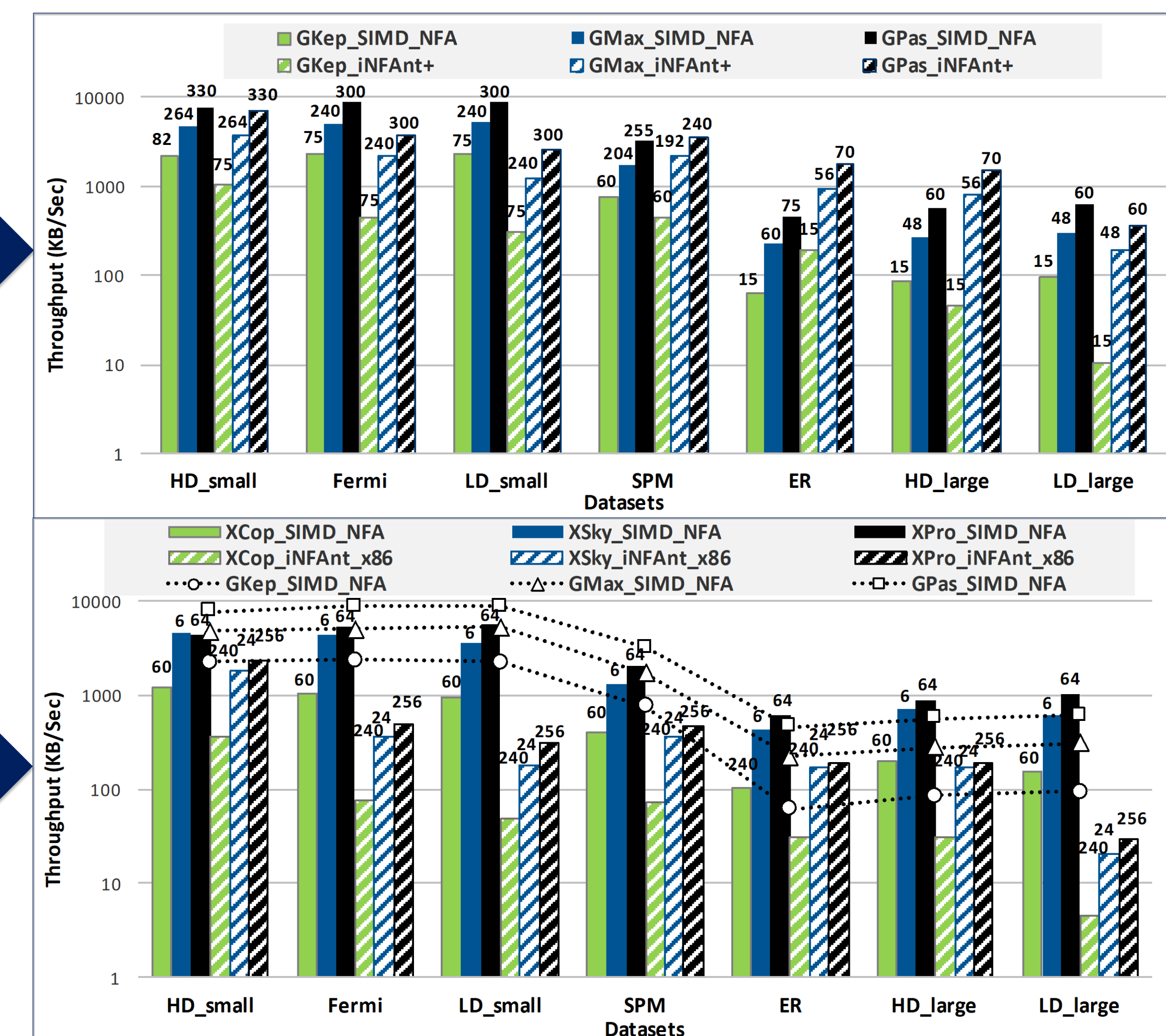


- $current_{sv}$ and $future_{sv}$: state vectors before and after processing each input symbol
- $symbol_tx_update$ or $epsilon_update$: encode (src,dst) transitions as a bitwise operation updating $future_{sv}$
- $Mask$: indicates the match between the input and transition symbols (generated by $match_check$ function, imposing the only control divergence case)

Results and Discussion

- Target NVIDIA GPUs** : Tesla K40c ,TITAN X and TITAN XP with Kepler (GKep), Maxwell (GMax), Pascal (GPas) architectures
- Target Intel SIMD devices**: Xeon Phi coprocessor (XCop), a Xeon Phi processor (XPro) and a Xeon Skylake processor (XSky)
- Datasets** (taken from [1]): Hamming Distance (HD) , Levenshtein Distance (LD), Fermi, Sequential Pattern Mining (SPM), Entity Resolution (ER)
- Baseline for evaluation**: an extended version of iNFAnt [2] (iNFAnt+) and a traversal engine using iNFAnt memory layout that only leverages stream-level parallelism (iNFAnt^{x86})
- Input streams**: counts indicated on top of bars in graph (each input stream:15KB)
- Results**:
 - Characteristics of the NFA topology affect the performance of *SIMD_NFA* and baseline implementations differently. This characteristics include : NFA size, transitions types, and activation ratio (affecting the baseline performance)
 - Older architectures with lower memory bandwidth, benefit more from *SIMD_NFA*
 - Intel SIMD devices outperform GPUs on large datasets, due to the efficient caching mechanism on Intel SIMD devices that can favor datasets with large memory utilization.

Throughput on GPUs



Throughput on Intel platforms

Conclusions

- We propose a memory-efficient pattern matching engine for fixed-topology NFAs suitable on SIMD architecture and we design a compiler toolchain that automates mapping of NFAs onto target SIMD devices.
- Testing on GPU and Intel platforms show promising speedups specifically on Intel SIMD devices (from 1.9X to 47.4X) and indicates the pros and cons of this design with respect to the topology characteristics and target platform's architecture.

Acknowledgements

This work is supported by NSF awards CNS-1724934 and CCF-1740583, by a gift from NEC Laboratories America, and by equipment donation from Nvidia Corporation.

References

- J. Wadden *et al.*, "ANMLzoo: a benchmark suite for exploring bottlenecks in automata processing engines and architectures," presented at the IISWC 2016.
- N. Cascarano *et al.*, "iNFAnt: NFA pattern matching on GPGPU devices," *SIGCOMM Comput. Commun. Rev.*, 2010.