# Combining Perfect Shuffle and Bitonic Networks for Efficient Quantum Sorting

Naveed Mahmud, Bailey K. Srimoungchanh, Bennett Haase-Divine,
Nolan Blankenau, Annika Kuhnke, and Esam El-Araby

University of Kansas (KU)

Fifth International Workshop on
Heterogeneous High-performance Reconfigurable Computing ($H^2RC$'19)

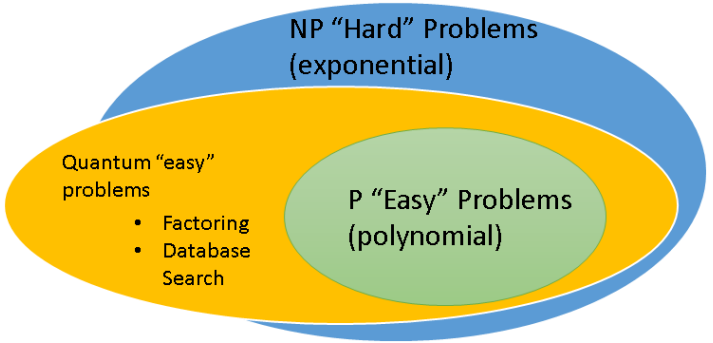
November 17-22, 2019
Denver, Colorado

# Outline

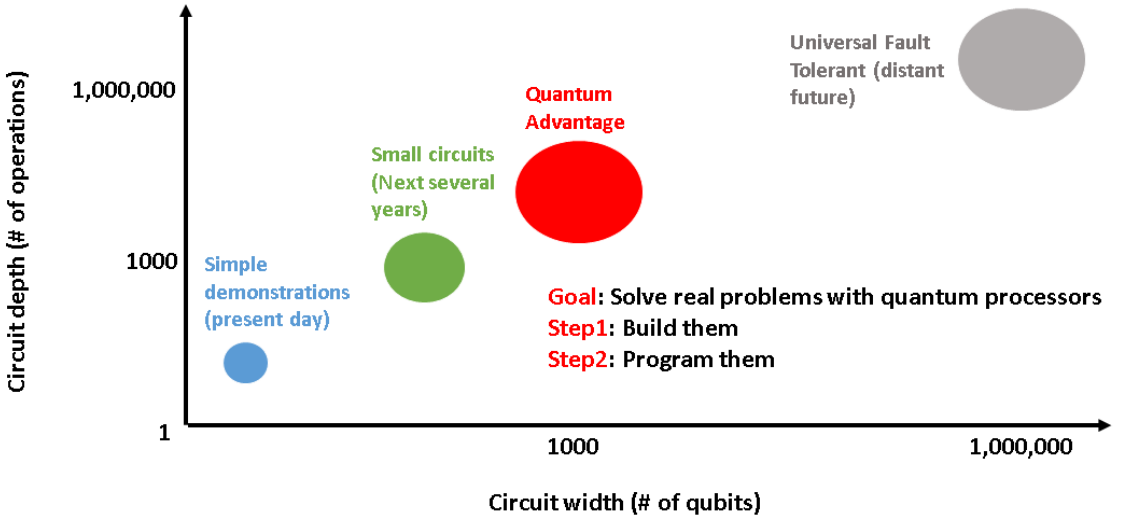
- **Introduction and Motivation**
- Background and Related Work
- Proposed Work
- Experimental Results
- Conclusions and Future Work

# Introduction and Motivation

- **Why Quantum?**
  - Efficient quantum algorithms
  - Solving NP-hard problems
  - Speedup over classical
  - Quantum supremacy
  - Quantum Ready NISQ devices
- **Need for Quantum Emulation**
  - Difficult to control QC experiments
  - Verification and benchmarking
  - High-cost of accessing QCs
    - E.g., academic hourly rate of $1,250 up to 499 annual hours
- **Emulation using FPGAs**
  - Greater speedup vs. SW
  - Dynamic (reconfigurable) vs. fixed architectures
  - Exploiting parallelism
  - Limitation → Scalability

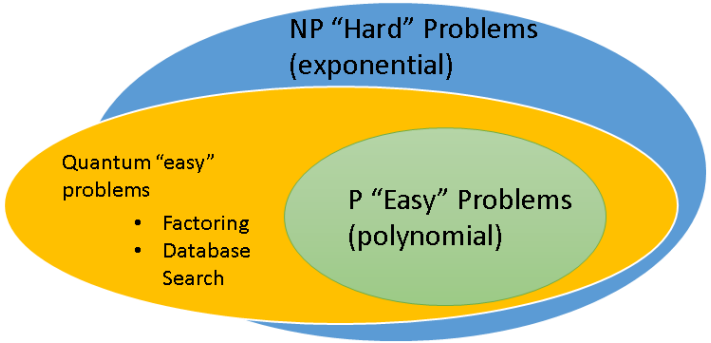



source: https://learning.acm.org/techtalks/qiskit

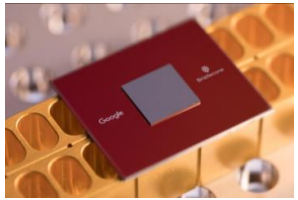


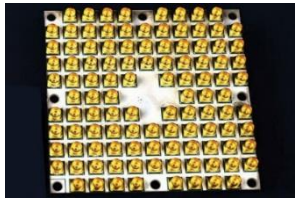
# Introduction and Motivation

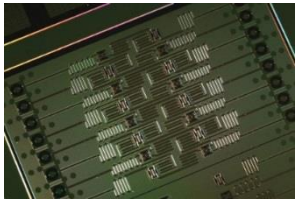
- **Why Quantum?**
  - Efficient quantum algorithms
  - Solving NP-hard problems
  - Speedup over classical
  - Quantum supremacy
  - Quantum Ready NISQ devices
- **Need for Quantum Emulation**
  - Difficult to control QC experiments
  - Verification and benchmarking
  - High-cost of accessing QCs
    - E.g., academic hourly rate of $1,250 up to 499 annual hours
- **Emulation using FPGAs**
  - Greater speedup vs. SW
  - Dynamic (reconfigurable) vs. fixed architectures
  - Exploiting parallelism
  - Limitation → Scalability

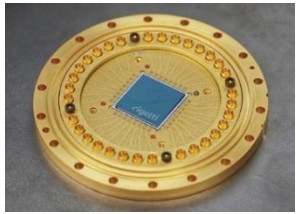


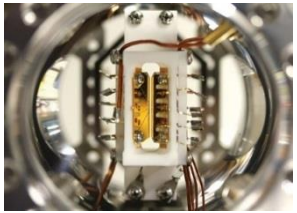

source: https://learning.acm.org/techtalks/qiskit


Google's 72-qubit "Bristlecone"


Intel's 49-qubit "Tangle Lake"


IBM-Q 53-qubit computer


Rigetti's 16-qubit ASPEN-4


IonQ's 79-qubit computer


D-Wave 2000Q

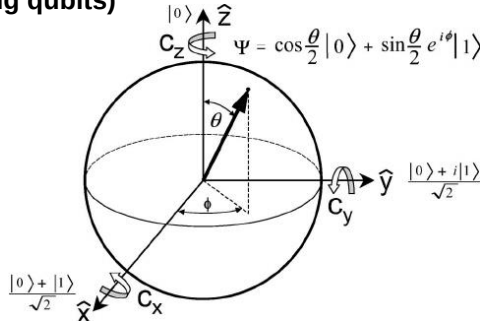
# Outline

- Introduction and Motivation
- **Background and Related Work**
- Proposed Work
- Experimental Results
- Conclusions and Future Work

# Background (Quantum Computing)

- **Qubits**
  - **Physical** implementations
    - Electron (spin)
    - Nucleus (spin through NMR)
    - Photon (polarization encoding)
    - Josephson junction (superconducting qubits)
    - Trapped ions
    - Anions
  - **Theoretical** representation
    - **Bloch sphere**
      - » Basis states → $|0\rangle$, $|1\rangle$
      - » Pure states → $|\psi\rangle$
    - **Vector** of complex **coefficients**

NMR ≡ Nuclear Magnetic Resonance




- **Superposition**
  - **Linear sum** of distinct basis states
  - Converts to **classical logic** when measured
  - Applies to state with ***n*-qubits**

- **Entanglement**
  - Strong **correlation** between qubits
  - **Measuring** a qubit gives information about other qubits
  - Entangled state cannot be **factored** into a tensor product

Single-Qubit Superposition: $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle \equiv \begin{bmatrix}\alpha\\ \beta\end{bmatrix}$

*Born Rule*: $p\left(|\psi_1\rangle \to |0\rangle\right) = |\alpha|^2, \quad p\left(|\psi_1\rangle \to |1\rangle\right) = |\beta|^2$

Multi-Qubit Superposition:

$$|\psi_3\rangle = |q_2 q_1 q_0\rangle = |q_2\rangle \otimes |q_1\rangle \otimes |q_0\rangle = \begin{bmatrix}\alpha_2\\ \beta_2\end{bmatrix} \otimes \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_0\\ \beta_0\end{bmatrix}$$

$$|\psi_3\rangle = \alpha_2\alpha_1\alpha_0|000\rangle + \alpha_2\alpha_1\beta_0|001\rangle + \ldots + \beta_2\beta_1\beta_0|111\rangle$$

$$|\psi_3\rangle = c_0|0\rangle + c_1|1\rangle + \ldots + c_7|7\rangle \Rightarrow |\psi_n\rangle = \sum_{q=0}^{2^n-1} c_q|q\rangle$$

*Born Rule*: $p\left(|\psi_n\rangle \to |q\rangle\right) = |c_q|^2 \Rightarrow \||\psi_n\rangle\|^2 = \sum_{q=0}^{2^n-1}|c_q|^2 = 1$

Multi-Qubit Entanglement:

$$\left(|\psi_n\rangle_{\text{entangled}} = |q_{n-1}\ldots q_1 q_0\rangle_{\text{entangled}}\right) \neq \left(|\psi_n\rangle_{\text{un-entangled}} = |q_{n-1}\rangle \otimes \ldots |q_1\rangle \otimes |q_0\rangle\right)$$

*For Example*: $\left(|\psi_2\rangle_{\text{entangled}} = |q_1 q_0\rangle_{\text{entangled}}\right) \neq \left(|q_1\rangle \otimes |q_0\rangle = \begin{bmatrix}\alpha_1\\ \beta_1\end{bmatrix} \otimes \begin{bmatrix}\alpha_0\\ \beta_0\end{bmatrix}\right)$

$$|\psi_2\rangle_{\text{entangled}} = c_0|00\rangle + c_3|11\rangle \neq \alpha_1\alpha_0|00\rangle + \alpha_1\beta_0|01\rangle + \beta_1\alpha_0|10\rangle + \beta_1\beta_0|11\rangle$$

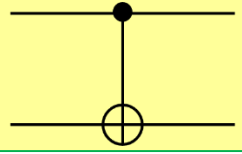
# Background (Quantum Gates)

- **X Gate (NOT) gate**
  - 1-qubit gate
  - Inverts the magnitude of the qubit

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

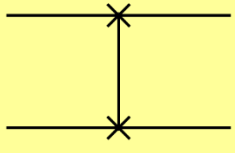
- **cX (Controlled NOT) Gate**
  - 2-qubit gate
  - Control qubit and a target qubit
  - Inverts target qubit based on value of control

$$cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



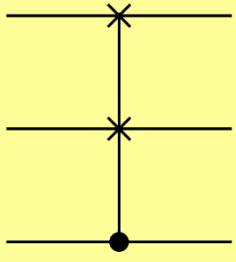
- **SWAP Gate**
  - 2-qubit gate
  - Exchanges positions of the two qubits

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



- **cSWAP (Controlled SWAP) Gate**
  - 3-qubit gate
  - Exchanges positions of the two qubits based on the control qubit

$$c\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



# Background (Sorting)

- **Classical Sorting**
  - Quicksort
  - Merge sort
  - Insertion sort
  - Bitonic sort with perfect shuffle

| Complexity | Quicksort | Merge sort | Insertion sort | Bitonic sort with perfect shuffle |
|---|---|---|---|---|
| Time | $N \log N$ | $N \log N$ | $N^2$ | $\log^2 N$ |
| Space | $\log N$ | $N$ | $1$ | $N$ |

source: https://www.bigocheatsheet.com/

# Background (Sorting)

## Classical Sorting

- Quicksort
- Merge sort
- Insertion sort
- Bitonic sort with perfect shuffle

| Complexity | Quicksort | Merge sort | Insertion sort | Bitonic sort with perfect shuffle |
|---|---|---|---|---|
| Time | $N \log N$ | $N \log N$ | $N^2$ | $\log^2 N$ |
| Space | $\log N$ | $N$ | $1$ | $N$ |

source: https://www.bigocheatsheet.com/

## Quantum Sorting

- Relatively **new** realm of research
- Based on **encoding** of data as **coefficients** of a superimposed quantum state ($N=2^n$)
- **Parallel** architecture
- **Speedup** compared to classical sorters

***N*** ≡ number of **states**
***n*** ≡ number of **qubits**

# Background (Sorting)

## Classical Sorting

- Quicksort
- Merge sort
- Insertion sort
- Bitonic sort with perfect shuffle

| Complexity | Quicksort | Merge sort | Insertion sort | Bitonic sort with perfect shuffle |
|---|---|---|---|---|
| Time | $N \log N$ | $N \log N$ | $N^2$ | $\log^2 N$ |
| Space | $\log N$ | $N$ | $1$ | $N$ |

source: https://www.bigocheatsheet.com/

## Quantum Sorting

- Relatively **new** realm of research
- Based on **encoding** of data as **coefficients** of a superimposed quantum state ($N=2^n$)
- **Parallel** architecture
- **Speedup** compared to classical sorters

| Complexity | Quantum merge sorting [Chen, et al] | Quantum bitonic sort with perfect shuffle |
|---|---|---|
| Time | $\log^2 n$ | $\log^2 n$ |
| Space | $n$ | $n$ |

***N*** ≡ number of **states**
***n*** ≡ number of **qubits**

# Related Work (Quantum Sorting)

- **Chen, et al., "Quantum switching and quantum merge sorting," February 2006**
  - Bitonic merge sorting with a divide-and-conquer approach
  - $O(log^2 n)$ time complexity to sort $n$ qubits
  - Not enough details about 'quantum comparator'
  - No experimental evaluation

- **Hoyer, et al., "Quantum complexities of ordered searching, sorting, and element distinctness," November 2002**
  - Proof showing lower bound of general quantum sorting is $\Omega(N\ log\ N)$
  - Based on comparison matrix given as input oracle
  - No circuit realizations or implementations

# Related Work (Parallel SW Simulators)

- **Villalonga, et al., "Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation," May 2019**
  - Simulation of 7x7 and 11x11 random quantum circuits (RQCs) of depth 42 and 26 respectively.
  - Summit supercomputer (ORNL, USA) with 4550 nodes
  - 1.6 TB of non-volatile memory per node
  - Power consumption of 7.3 MW

List of quantum SW simulators
https://quantiki.org/wiki/list-qc-simulators

- **Li et al., "Quantum Supremacy Circuit Simulation on Sunway TaihuLight," August 2018**
  - Simulation of 49-qubit random quantum circuits of depth of 55
  - Sunway supercomputer (NSC, China) with 131,072 nodes (32,768 CPUs)
  - 1 PB total main memory

- **J. Chen, et al., "Classical Simulation of Intermediate-Size Quantum Circuits," May 2018**
  - Simulation of up to 144-qubit random quantum circuits of depth 27
  - Supercomputing cluster (Alibaba Group, China) with 131,072 nodes
  - 8 GB memory per node

- **De Raedt et al., "Massively parallel quantum computer simulator eleven years later," May 2018**
  - Simulation of Shor's algorithm using 48-qubits
  - Various supercomputing platforms: IBM Blue Gene/Q (decommissioned), JURECA (Germany), K computer (Japan), Sunway TaihuLight (China)
  - Up to 16-128 GB memory/node utilized

- **T. Jones, et al., "QuEST and High Performance Simulation of Quantum Computers," May 2018**
  - Simulation of random quantum circuits up to 38 qubits
  - ARCUS supercomputer (ARCHER, UK) with 2048 nodes
  - Up to 256 GB memory per node

# Related Work (FPGA-based Quantum Emulators)

- J. Pilch, and J. Dlugopolski, "An FPGA-based real quantum computer emulator," December 2018
  - Results for up to 2-qubit Deutsch's algorithm
  - Details of precision used not presented
  - Limited scalability
- A. Silva, and O.G. Zabaleta, "FPGA quantum computing emulator using high level design tools," August 2017
  - Results for up to 6-qubit QFT
  - Details of precision used not presented
  - No approach to improve scalability
- Y.H. Lee, M. Khalil-Hani, and M.N. Marsono, "An FPGA-based quantum computing emulation framework based on serial-parallel architecture," March 2016
  - Results of 5-qubit QFT and 7-qubit Grover's reported
  - Up to 24-bit fixed-point precision
  - No optimizations to make designs scalable
- A.U. Khalid, Z. Zilic, and K. Radecka, "FPGA emulation of quantum circuits," October 2004
  - 3-qubit QFT and Grover's search implemented
  - Fixed-point precision (16 bits)
  - Low operating frequency
- M. Fujishima, "FPGA-based high-speed emulator of quantum computing," December 2003
  - Logic quantum processor that abstracts quantum circuit operations into binary logic
  - Coefficients of qubit states modeled as binary, not complex
  - No resource utilization reported

# Outline

- Introduction and Motivation
- Background and Related Work
- **Proposed Work**
- Experimental Results
- Conclusions and Future Work

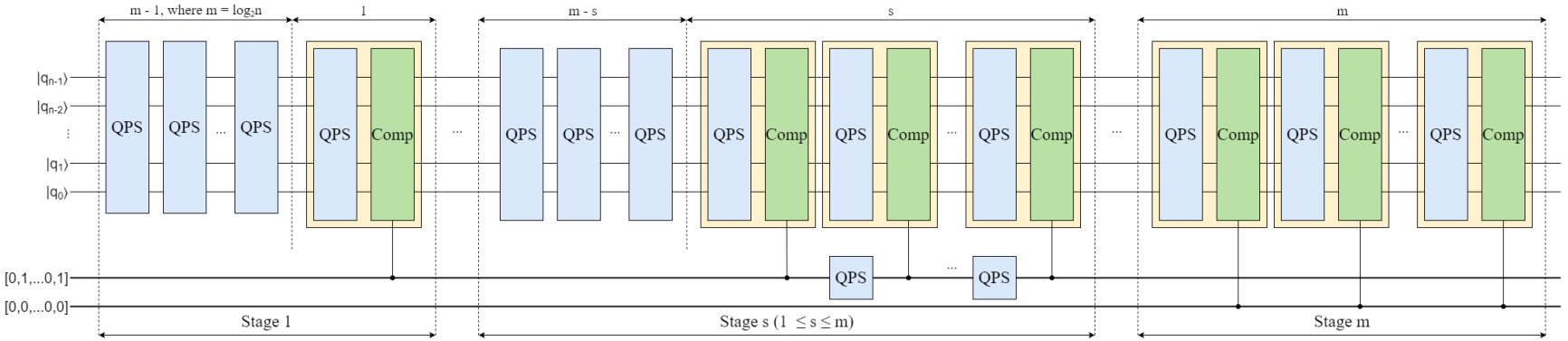
# Proposed Work

## Quantum algorithm for sorting

- For $n$ qubits, $m$ stages where $m = log_2 n$
- For each stage $s$, $1 \leq s \leq m$
  - $m - s$ quantum perfect shuffle (QPS) operations
  - Followed by $s$ QPS-Comparator pairs

**Algorithm: Bitonic sort with perfect shuffle**

```
for s=1 to m do
  for i=1 to m do
    QPS(qubits)
  end
  for i=m-s+1 to m do
    QPS(qubits)
    comp(qubits, mode)
    QPS(mode)
  end
end
```




Generic perfect shuffle based quantum sorter

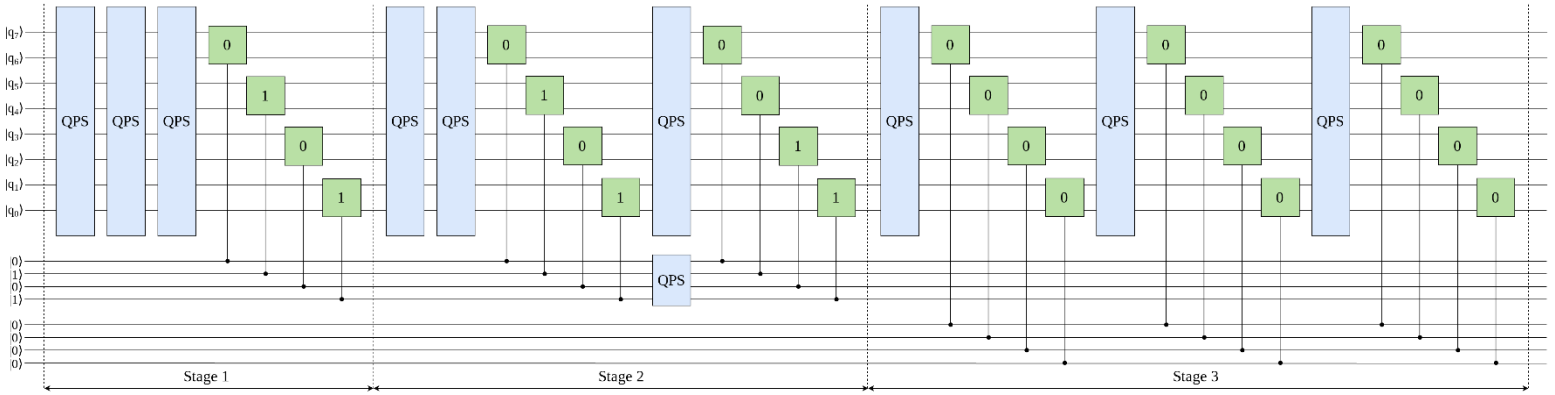
# Proposed Work

## Quantum algorithm for sorting

- For $n$ qubits, $m$ stages where $m = log_2 n$
- For each stage $s$, $1 \leq s \leq m$
  - $m - s$ quantum perfect shuffle (QPS) operations
  - Followed by $s$ QPS-Comparator pairs

**Algorithm: Bitonic sort with perfect shuffle**

```
for s=1 to m do
  for i=1 to m do
    QPS(qubits)
  end
  for i=m-s+1 to m do
    QPS(qubits)
    comp(qubits, mode)
    QPS(mode)
  end
end
```

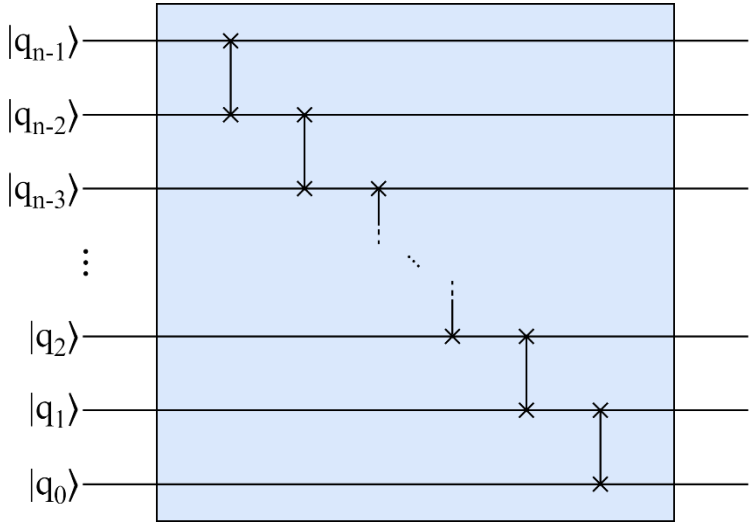



8-qubit perfect shuffle based quantum sorter

# Proposed Work

## Quantum perfect shuffle

- **Rotate left operation on coefficient indices**
- **Quantum gate utilized: *SWAP***

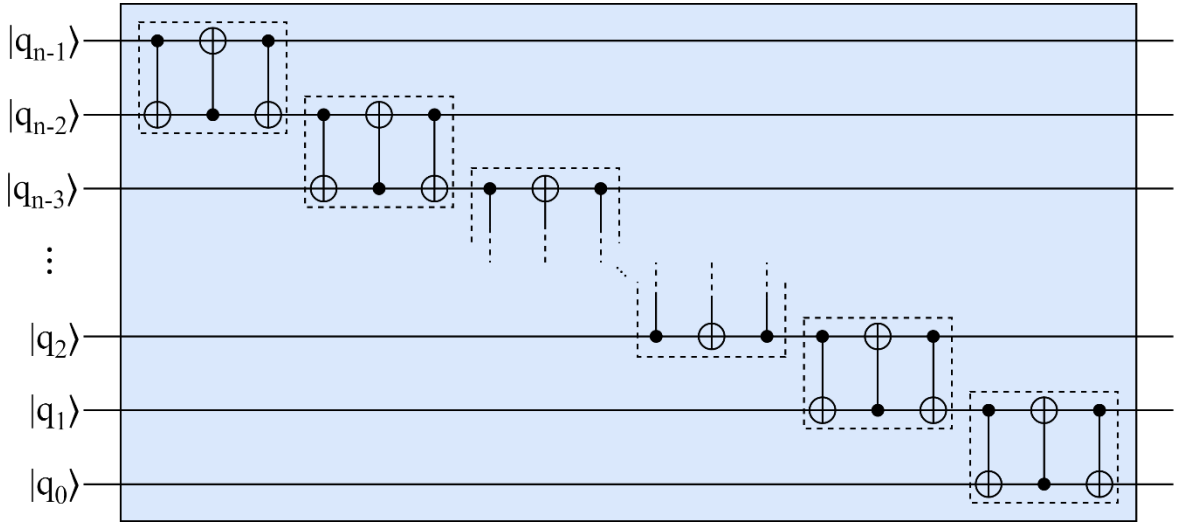



**Quantum perfect shuffle (QPS) circuit**

# Proposed Work

## Quantum perfect shuffle

- **Rotate left operation on coefficient indices**
- **Quantum gate utilized: *SWAP***

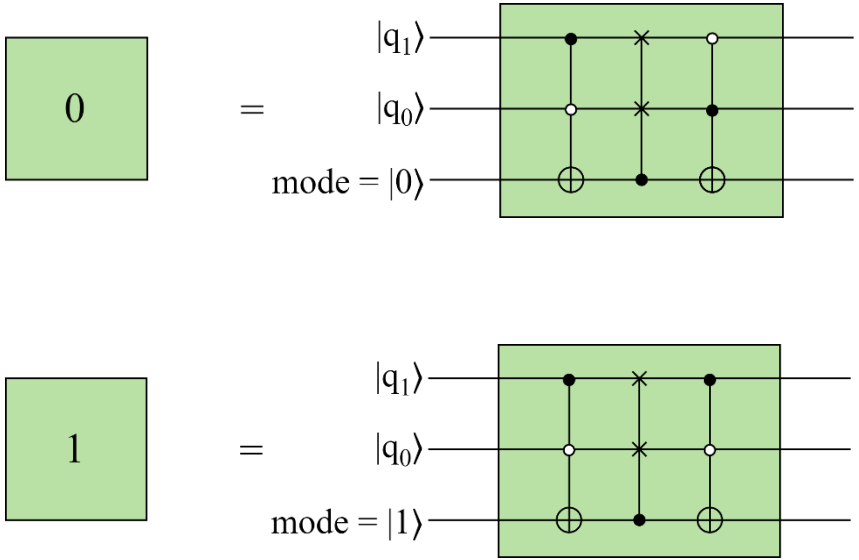



**Quantum perfect shuffle (QPS) circuit**

# Proposed Work

## Quantum comparator

- Two modes: *min-max* and *max-min*
- Mode control: ancilla qubit
- Mode = 0 (*min-max*)
  - $q1 = min(q1, q0)$
  - $q0 = max(q1, q0)$
- Mode = 1 (*max-min*)
  - $q1 = max(q1, q0)$
  - $q0 = min(q1, q0)$
- Quantum gates
  - *cSWAP*
  - *ccX*

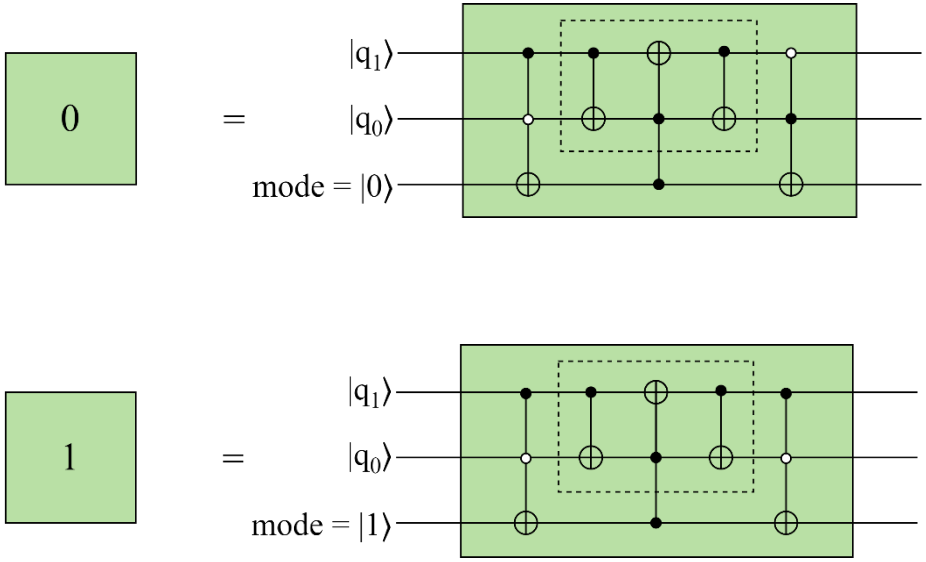



3-qubit, 2-mode quantum comparator circuit

# Proposed Work

## Quantum comparator

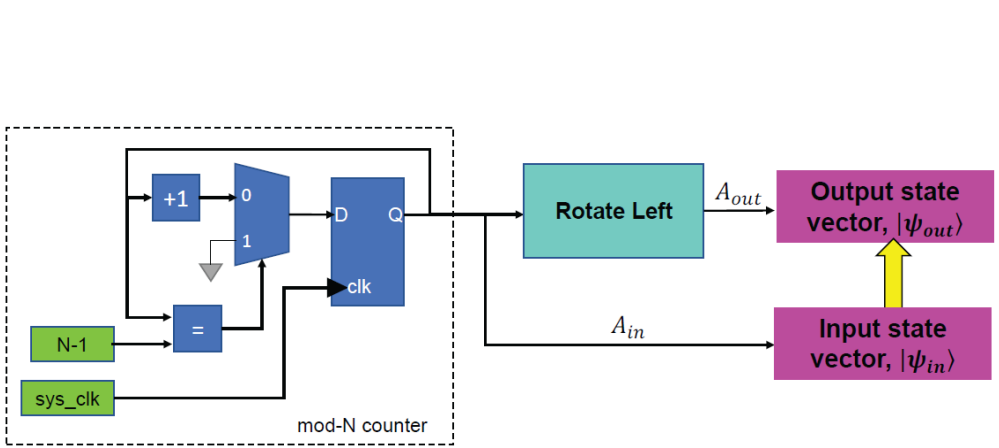
- Two modes: *min-max* and *max-min*
- Mode control: ancilla qubit
- Mode = 0 (*min-max*)
  - $q1 = min(q1, q0)$
  - $q0 = max(q1, q0)$
- Mode = 1 (*max-min*)
  - $q1 = max(q1, q0)$
  - $q0 = min(q1, q0)$
- Quantum gates
  - *cSWAP*
  - *ccX*



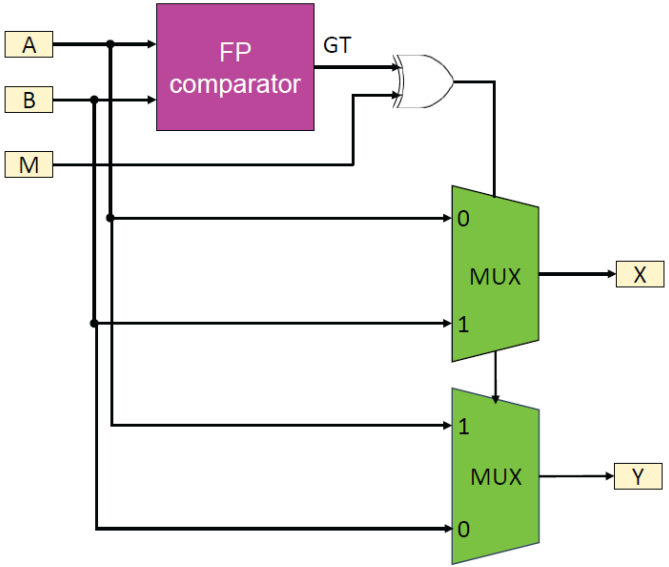

3-qubit, 2-mode quantum comparator circuit

# Proposed Work

## Emulation Hardware Architectures




Emulation architecture for quantum perfect shuffle




Emulation architecture for quantum comparator

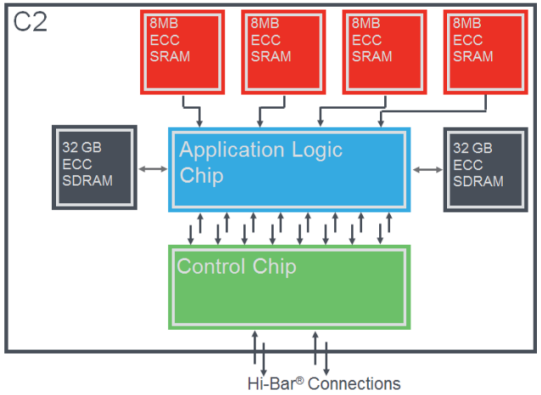
# Outline

- Introduction and Motivation
- Related Work and Background
- Proposed Work
- **Experimental Results**
- Conclusions and Future Work

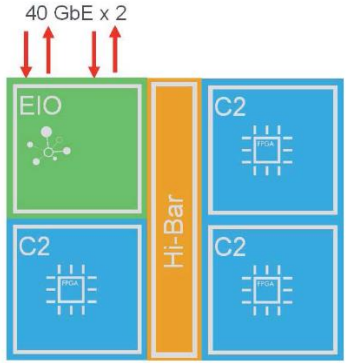
# Experimental Setup

- **Testbed Platform**
  - High-performance reconfigurable computing (HPRC) system from DirectStream
  - Single compute node to warehouse scale multi-node deployments
  - OS-less, FPGA-only (Arria 10) architecture
  - Single node on-chip resources (OCR)
    - 427,200 Adaptive Logic Modules (ALMs)
    - 1,518 Digital signal Processors (DSPs)
    - 2,713 Block RAMs (BRAMs)
  - Single node on-board memory (OBM)
    - $2 \times 32$ GB SDRAM modules
    - $4 \times 8$ MB SRAM modules
  - Highly productive development environment
    - Parallel High-Level Language
    - C++-to-HW (previously Carte-C) compiler
    - Quartus Prime 17.0.2

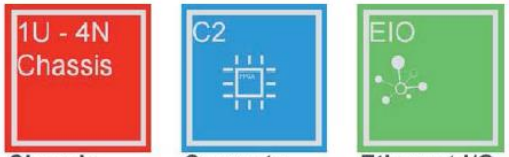
**DirectStream (DS8) system**




Single compute node




Multi-node instance




**Chassis**
4 Node-1U
N+1 power
Hi-bar switch
240 Gb/s
bi-directional
bandwidth

**Compute**
Compute
Altera Arria 10
FPGA
(Intel)

**Ethernet I/O**
Ethernet I/O
Networking
Processor
80 GbE
(40 GbE x 2)

Node types

# Experimental Results

## Quantum sorting emulation results using on-chip resources

| Number of qubits, *n* | Number of states, *N* | On-chip resource* utilization | | Emulation time (sec)*** |
|---|---|---|---|---|
| | | ALMs | BRAMs | |
| 2 | 4 | 47,571 | 230 | 7.74E-06 |
| 3 | 8 | 49,036 | 237 | 2.40E-05 |
| 4 | 16 | 49,460 | 237 | 6.15E-05 |
| 5 | 32 | 49,302 | 237 | 1.54E-04 |
| 6 | 64 | 49,594 | 239 | 3.91E-04 |
| 7 | 128 | 49,253 | 241 | 1.01E-03 |
| 8 | 256 | 49,733 | 243 | 2.85E-03 |
| 9 | 512 | 49,681 | 243 | 8.96E-03 |
| 10 | 1024 | 49,640 | 247 | 3.09E-02 |
| 11 | 2048 | 52,400 | 226 | 1.14E-01 |
| 12 | 4096 | 52,567 | 242 | 4.35E-01 |
| 13 | 8192 | 50,066 | 315 | 1.70E+00 |
| 14 | 16,384 | 50,078 | 391 | 6.72E+00 |
| 15 | 32,768 | 50,331 | 555 | 2.67E+01 |
| 16 | 65,536 | 50,571 | 875 | 1.07E+02 |
| 17 | 131,072 | 50,768 | 1,515 | 4.26E+02 |

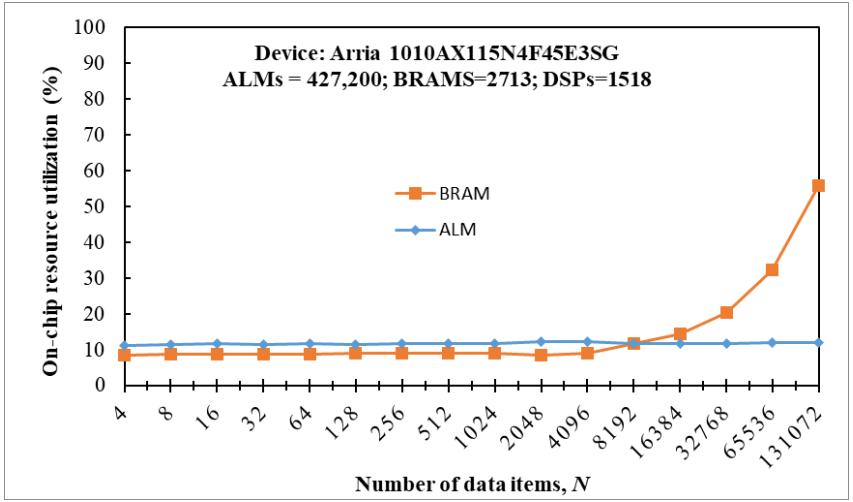
## Quantum sorting emulation results using on-board memory

| Number of qubits, *n* | Number of states, *N* | On-chip resource* utilization | | On-board memory | | Emulation time (sec)*** |
|---|---|---|---|---|---|---|
| | | ALMs | BRAMs | SDRAM 1 | SDRAM 2 | |
| 18 | $2^{18}$ | 55,684 | 261 | 2M | 2M | 1.70E+03 |
| 19 | $2^{19}$ | 55,862 | 261 | 4M | 4M | 6.80E+03 |
| 20 | $2^{20}$ | 56,557 | 261 | 8M | 8M | 2.72E+04 |
| 30 | $2^{30}$ | 56,641 | 261 | 8G | 8G | 2.85E+10† |
| 31 | $2^{31}$ | 56,684 | 261 | 16G | 16G | 1.14E+11† |

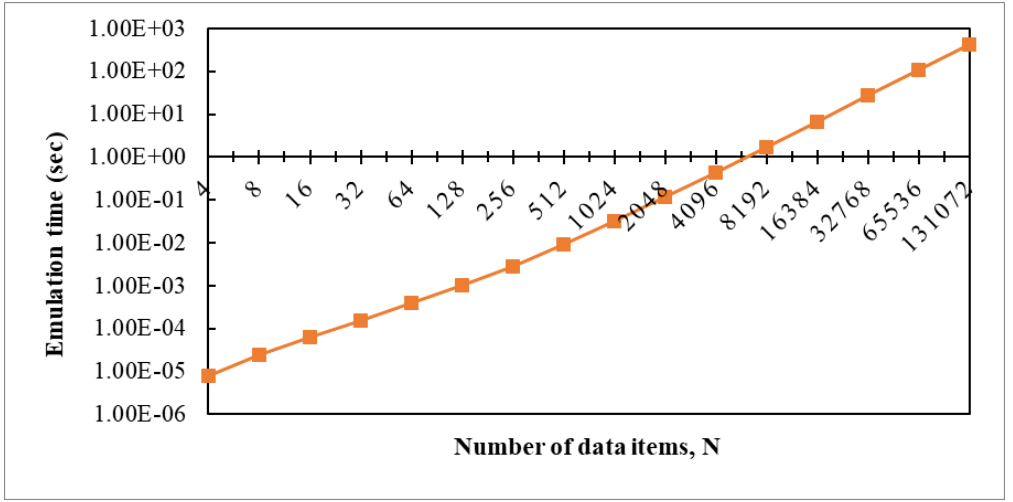
SDRAM banks of 32GB each
***Operating frequency: 233 MHz
† Results projected using regression

**ALM** ≡ **A**daptive **L**ogic **M**odules
**BRAM** ≡ **B**lock **R**andom **A**ccess **M**emory
**DSP** ≡ **D**igital **S**ignal **P**rocessing block

# Experimental Results



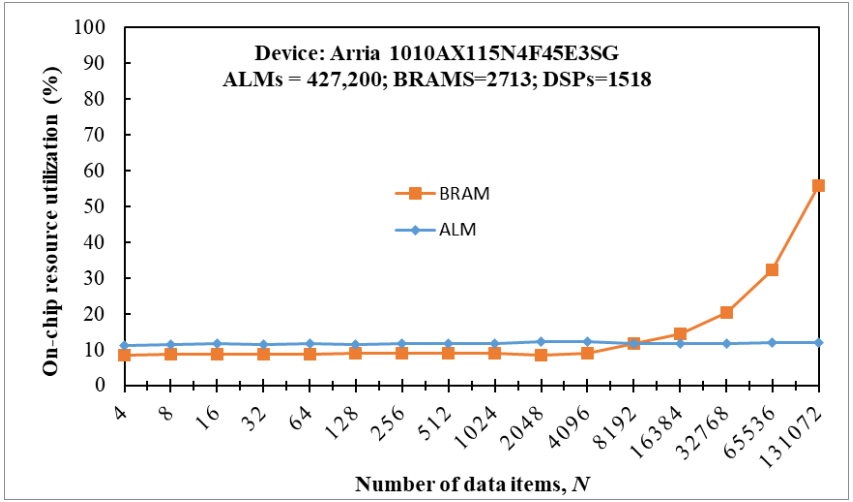

On-chip resource utilization vs number of states, *N*

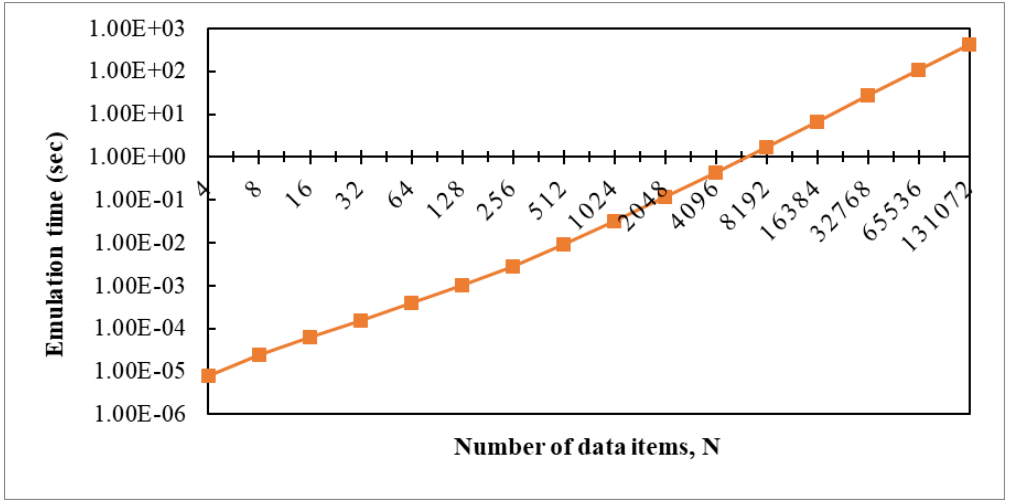



On-chip emulation time vs number of states, *N*

**ALM** ≡ **A**daptive **L**ogic **M**odules
**BRAM** ≡ **B**lock **R**andom **A**ccess **M**emory
**DSP** ≡ **D**igital **S**ignal **P**rocessing block

# Experimental Results




**On-chip resource utilization vs number of states, *N***

| Resource | ALM | BRAM |
|---|---|---|
| Space complexity | *O(1)* | *O(N)* |

**On-chip emulation time vs number of states, *N***

| Task | I/O | Compute (sort) |
|---|---|---|
| Time complexity | *O(N)* | *O(log² N)* |

**ALM** ≡ **A**daptive **L**ogic **M**odules
**BRAM** ≡ **B**lock **R**andom **A**ccess **M**emory
**DSP** ≡ **D**igital **S**ignal **P**rocessing block

# Experimental Results

## Comparison with related work (FPGA-based emulation)

| Reported Work | Algorithm | Number of qubits | Precision | Operating frequency (MHz) | Emulation time (sec) |
|---|---|---|---|---|---|
| Fujishima (2003) | Shor's factoring | - | - | 80 | 10 |
| Khalid et al (2004) | QFT | 3 | 16-bit fixed pt. | 82.1 | 61E-9 |
| | Grover's search | 3 | 16-bit fixed pt. | | 84E-9 |
| Aminian et al (2008) | QFT | 3 | 16-bit fixed pt. | 131.3 | 46E-9 |
| Lee et al (2016) | QFT | 5 | 24-bit fixed pt. | 90 | 219E-9 |
| | Grover's search | 7 | 24-bit fixed pt. | 85 | 96.8E-9 |
| Silva and Zabaleta (2017) | QFT | 4 | 32-bit floating pt. | - | 4E-6 |
| Pilch and Dlugopolski (2018) | Deutsch | 2 | - | - | - |
| **Proposed work** | **QFT** | **32** | **32-bit floating pt.** | **233** | **7.92E10†** |
| | **QHT** | **30** | | | **13.825** |
| | **Grover's search** | **32** | | | **7.92E10†** |
| | **QHT + Grover's** | **32** | | | **7.92E10†** |
| | **Quantum sorting** | **31** | | | **1.14E+11†** |

† Results projected using regression

# Conclusions

- **Supremacy of Quantum Computing**
- **Need for Quantum Emulation**
  - Emulation using FPGAs
- **Case study**
  - Quantum sorting algorithm
- **Proposed Methodology**
  - Combining bitonic merge sorting with perfect shuffle
- **Testbed Platform**
  - State-of-the-art HPRC system from DirectStream
  - C++ to hardware compiler

# Future Work

- **Design Optimizations**
  - **Dynamic Partial Run-time Reconfiguration (PRTR)**
- **More algorithms/applications**
  - **Data dimensionality reduction using QHT**
  - **Quantum multi-pattern search using QHT and Grover's algorithm**
  - **Quantum machine learning**
  - **Quantum cybersecurity**
- **Quantum error correction (QEC)**
  - **More accurate emulation of quantum computers**
- **Power efficiency**
  - **Comparison with GPU/CPU simulations**


THANK YOU