KU SCHOOL OF ENGINEERING
The University of Kansas

ASAP 2019

Information and Telecommunication Technology Center

Improving Emulation of Quantum Algorithms using Space-Efficient Hardware Architectures

Naveed Mahmud, and Esam El-Araby

University of Kansas (KU)

30th IEEE International Conference on Application-specific Systems, Architectures and Processors

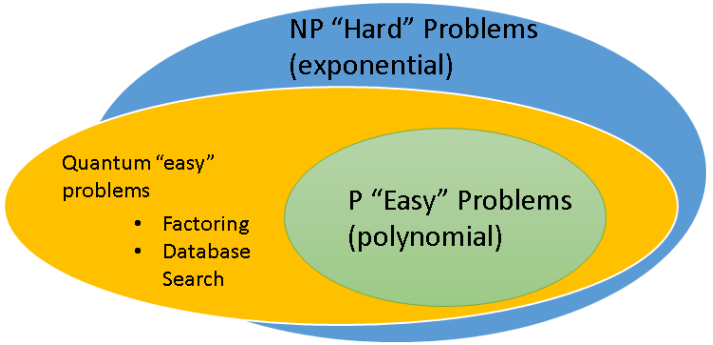
July 15-17, 2019
Cornell Tech, New York

Outline

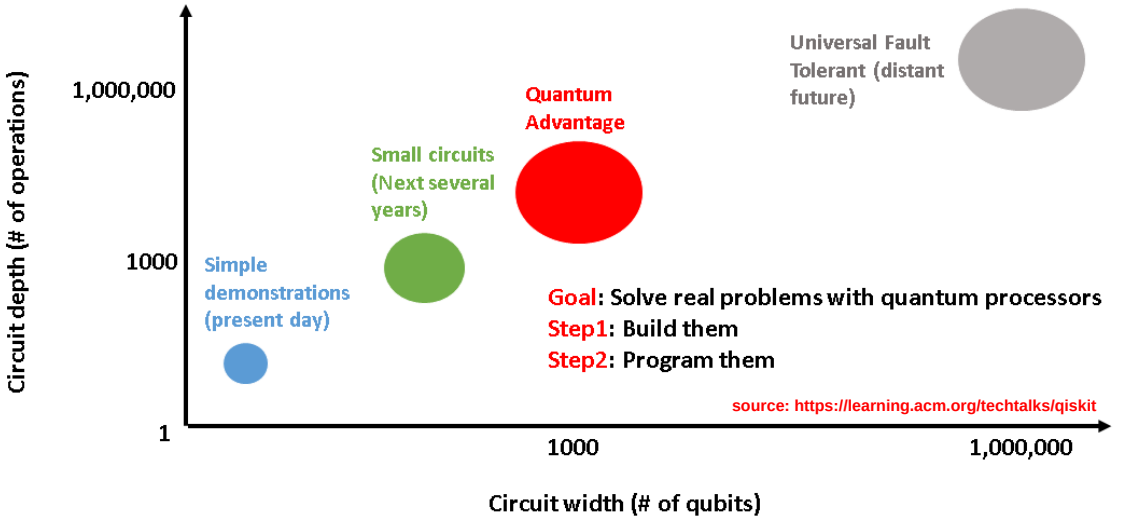
- **Introduction and Motivation**
- Related Work and Background
- Proposed Work
- Experimental Results
- Conclusions and Future Work

Introduction and Motivation

- **Why Quantum Computing?**
  - Efficient quantum algorithms
  - Solving NP-hard problems
  - Speedup over classical
- **Need for Quantum Emulation**
  - Difficulty of maintenance & control
  - High-cost of access
    - E.g., academic hourly rate of $1,250 up to 499 annual hours
  - Verification and benchmarking
  - Analysis of quantum algorithms
  - Improving classical computing paradigms
- **Emulation using FPGAs**
  - Greater speedup vs. SW
  - Dynamic (reconfigurable) vs. fixed architectures
  - Exploiting parallelism
  - Limitation → Scalability

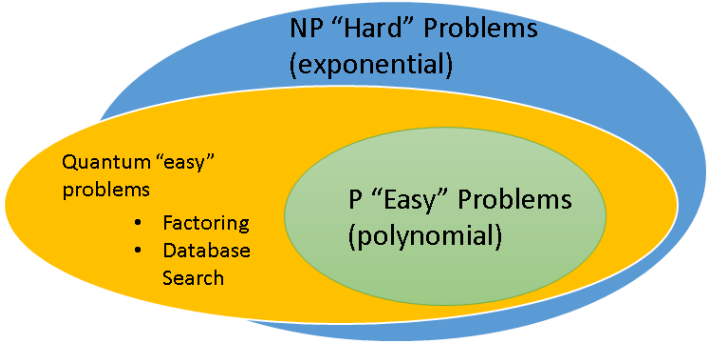


source: https://learning.acm.org/techtalks/qiskit

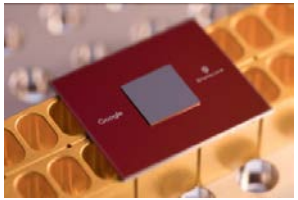


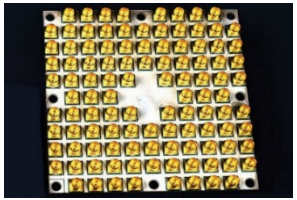
Introduction and Motivation

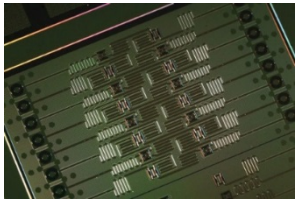
- **Why Quantum Computing?**
  - Efficient quantum algorithms
  - Solving NP-hard problems
  - Speedup over classical
- **Need for Quantum Emulation**
  - Difficulty of maintenance & control
  - High-cost of access
    - E.g., academic hourly rate of $1,250 up to 499 annual hours
  - Verification and benchmarking
  - Analysis of quantum algorithms
  - Improving classical computing paradigms
- **Emulation using FPGAs**
  - Greater speedup vs. SW
  - Dynamic (reconfigurable) vs. fixed architectures
  - Exploiting parallelism
  - Limitation → Scalability

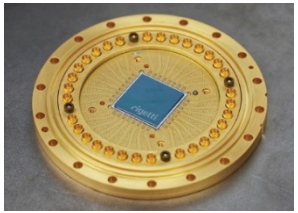


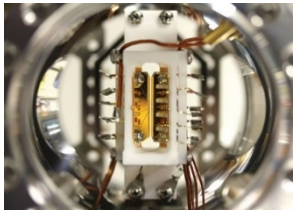
source: https://learning.acm.org/techtalks/qiskit


Google's 72-qubit "Bristlecone"


Intel's 49-qubit "Tangle Lake"


IBM-Q 50-qubit computer


Rigetti's 16-qubit ASPEN-4


IonQ's 79-qubit computer


D-Wave 2000Q

Outline

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

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," Aug. 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

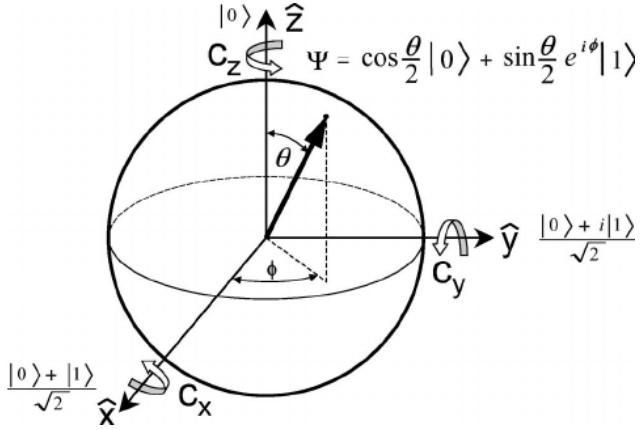
- **T. Jones, et al., "QuEST and High Performance Simulation of Quantum Computers," May 2018**
  - Simulation of random quantum circuits up to 38 qubits
  - ARCUS supercomputer (ARCHER, UK) with 2048 nodes
  - Up to 256 GB memory per node

Related Work (FPGA 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

Background (Quantum Computing)

- **Qubits**
  - **Physical implementations**
    - Electron (spin)
    - Nucleus (spin through NMR)
    - Photon (polarization encoding)
    - Josephson junction (superconducting qubits)
  - **Theoretical representation**
    - **Bloch sphere**
      - Basis states → $|0\rangle$, $|1\rangle$
      - Pure states → $|\psi\rangle$
    - **Vector of complex coefficients**
- **Superposition**
  - **Linear sum of distinct basis states**
  - **Converts to classical logic when measured**
  - **Applies to state with *n*-qubits**
- **Entanglement**
  - **Strong correlation between qubits**
  - **Entangled state cannot be factored**
  - **Tensor (Kronecker) product representation**
    - ***N* = 2^n basis states, where, *n* is number of qubits**



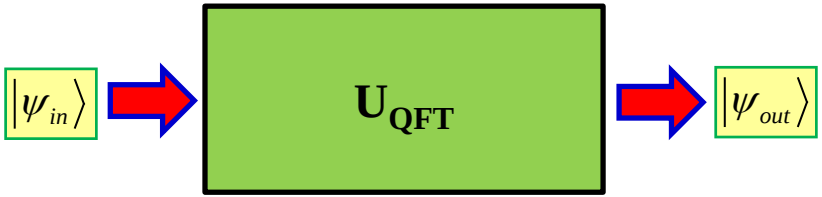
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \equiv \begin{bmatrix}\alpha\\ \beta\end{bmatrix}, and$$
$$p(\psi \to |0\rangle) = |\alpha|^2 ; p(\psi \to |1\rangle) = |\beta|^2$$

$$|q1q2q3\rangle = |q1\rangle \otimes |q2\rangle \otimes |q3\rangle$$
$$|\psi\rangle = \alpha_1\alpha_2\alpha_3|000\rangle + \alpha_1\alpha_2\beta_3|001\rangle + \alpha_1\beta_2\alpha_3|010\rangle +$$
$$\ldots + \beta_1\,\beta_2\beta_3|111\rangle$$

NMR ≡ Nuclear Magnetic Resonance

Background (Quantum Fourier Transform)

- **Quantum Fourier Transform (QFT)**
  - **Fundamental** quantum algorithm
  - **Quantum equivalent** of classical Discrete Fourier Transform (DFT)
  - **Quadratic** speedup over DFT
  - **Input**
    - Coefficients of a quantum superimposed state
  - **Output**
    - Coefficients of transformed superimposed state



QFT

$$|\psi_{in}\rangle = \frac{1}{\sqrt{N}} \sum_{q=0}^{N-1} f(q\Delta t)|q\rangle = \sum_{q=0}^{N-1} C_q^{in}|q\rangle$$

$$\xrightarrow{U_{QFT}} |\psi_{out}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{q=0}^{N-1} f(q\Delta t) e^{2\pi i \left(\frac{qk}{N}\right)} |k\rangle = \sum_{k=0}^{N-1} C_k^{out}|k\rangle$$

QFT matrix

$$U_{QFT} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \dots & \omega_n^{N-1} \\ 1 & \omega_n^2 & \omega_n^4 & \dots & \omega_n^{2(N-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \dots & \omega_n^{3(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{N-1} & \omega_n^{2(N-1)} & \dots & \omega_n^{(N-1)(N-1)} \end{bmatrix}$$

$\omega = e^{\frac{2\pi i}{N}}$, $N = 2^n$, and $n =$ number of qubits

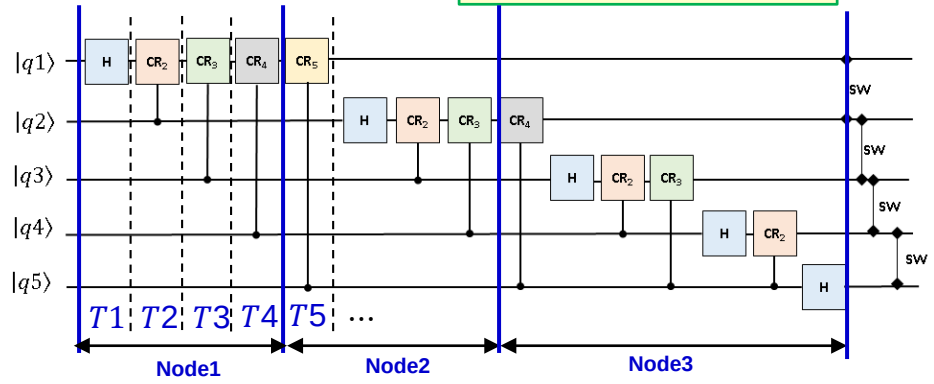
Outline

- Introduction and Motivation
- Background and Related Work
- **Proposed Work**
  - **Emulation Approaches**
  - Hardware Architectures
- Experimental Work
- Conclusions and Future Work

Emulation Approaches – Direct circuit/gate approach

- **QFT Emulation**

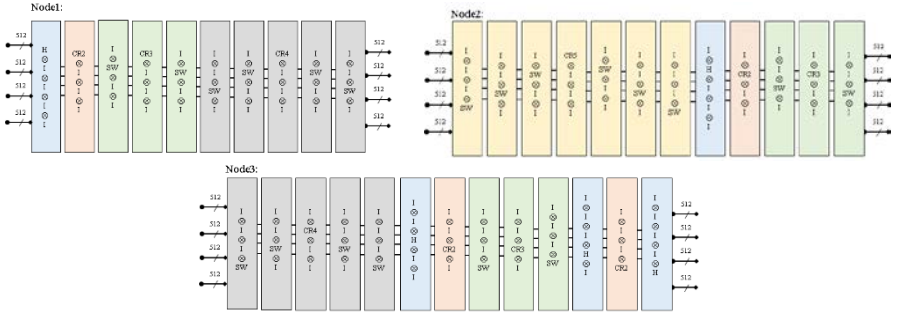
QFT ≡ Quantum Fourier Transform



5-qubit QFT algorithm/circuit

$$T1 = H \otimes I \otimes I \otimes I \otimes I$$
$$T2 = CR_2 \otimes I \otimes I \otimes I$$
$$T3 = (I \otimes SW \otimes I \otimes I).(CR_3 \otimes I \otimes I \otimes I).(I \otimes SW \otimes I \otimes I)$$
$$T4 = (I \otimes I \otimes SW \otimes I).(I \otimes SW \otimes I \otimes I).(CR_4 \otimes I \otimes I \otimes I).(I \otimes SW \otimes I \otimes I).(I \otimes I \otimes SW \otimes I)$$
$$T5 = (I \otimes I \otimes I \otimes SW).(I \otimes I \otimes SW \otimes I).(I \otimes SW \otimes I \otimes I).(CR_5 \otimes I \otimes I \otimes I).(I \otimes SW \otimes I \otimes I).(I \otimes I \otimes I \otimes SW)$$
$$\vdots$$

Modeling the quantum circuit as a series of transformations

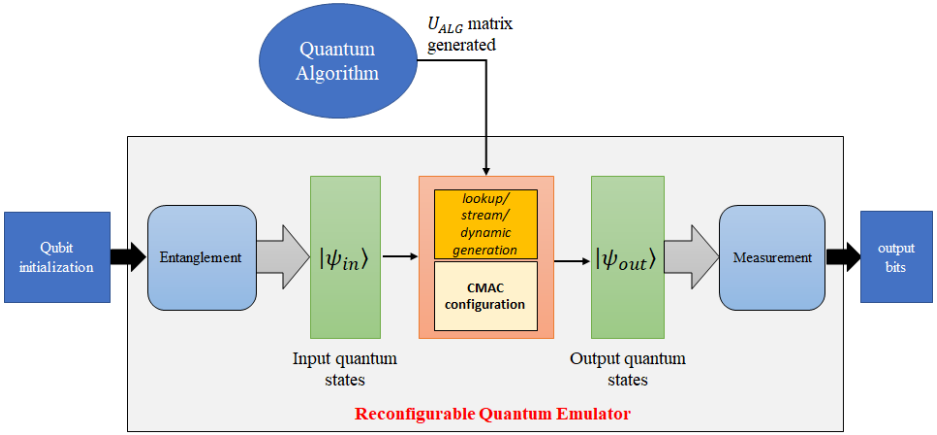


Hardware node models for 5-qubit QFT

- **Advantages**
  - Modular
  - Generalized framework
- **Disadvantages**
  - High resource utilization
  - Longer latency
  - Poor scalability

Emulation Approaches – CMAC approach

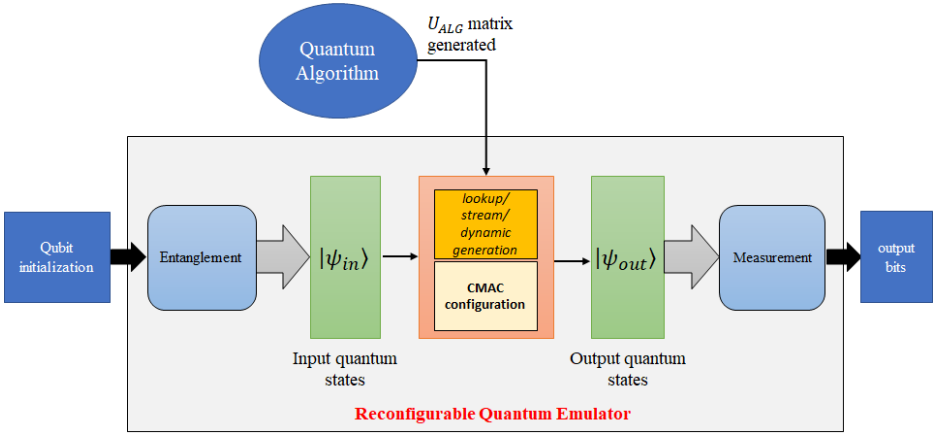
- **Methodology**
  - Vector-matrix multiplication
    - Complex multiply-and-accumulate (CMAC)
  - Input quantum state vector, $|\psi_{in}\rangle$
  - Algorithm reduced to a unitary matrix, U_{ALG}
    - *lookup / dynamic generation / stream*
  - Output quantum state vector, $|\psi_{out}\rangle$
- **Advantages**
  - Generalized approach for any quantum algorithm
  - Independent of circuit depth
  - Lower resource utilization
  - Lower latency
  - Higher scalability
  - Parallelizable hardware architectures
- **Disadvantages**
  - Calculating U_{ALG} could be challenging, but doable



$$|\psi_{out}\rangle = U_{ALG} \cdot |\psi_{in}\rangle$$

Emulation Approaches – CMAC approach optimizations

- ***Lookup***
  - Computation elements (U_{ALG}) stored in memory
  - Scalability limited by available memory
  - Speed limited by memory bandwidth
- ***Dynamic generation***
  - Additional hardware units required for generating U_{ALG}
  - Improved memory utilization
  - Improved scalability
  - Speed limited by complexity of algorithm generating U_{ALG}
- ***Stream***
  - No hardware required for U_{ALG}
  - Improved memory utilization
  - Improved scalability
  - Speed limited by I/O bandwidth



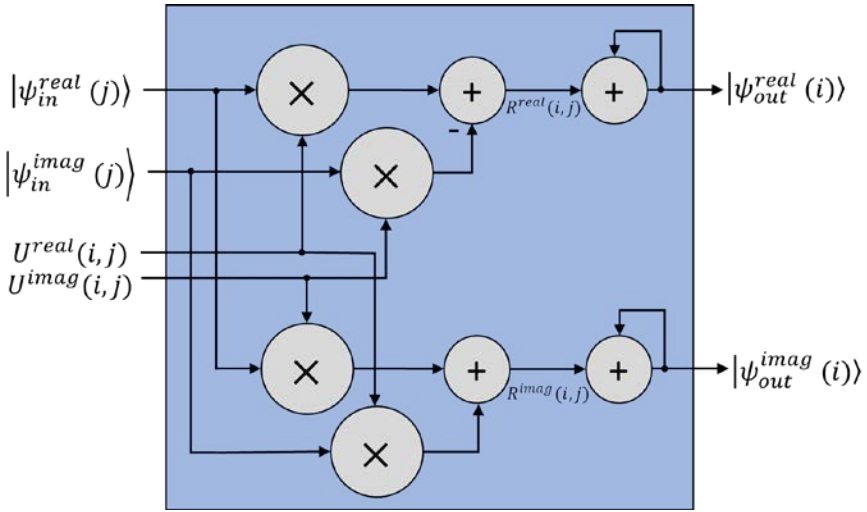
$$|\psi_{out}\rangle = U_{ALG} \cdot |\psi_{in}\rangle$$

Outline

- Introduction and Motivation
- Background and Related Work
- **Proposed Work**
  - Emulation Approaches
  - **Hardware Architectures**
- Experimental Work
- Conclusions and Future Work

Hardware Architectures – CMAC approach

- **Complex Multiply-and-Accumulate (CMAC) unit**
  - Implements **vector-matrix** multiplication on hardware
  - **Complex** valued inputs
  - **Single/double-precision** floating-point



$$\psi_{out}^{real}(i) = \sum_{j=0}^{N-1} \boldsymbol{R^{real}(i,j)}$$

$$\psi_{out}^{imag}(i) = \sum_{j=0}^{N-1} \boldsymbol{R^{imag}(i,j)}$$

where,

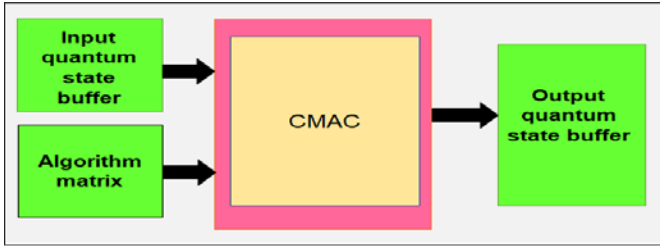
$$i = 0, 1, 2, \ldots, (N-1)$$

$$\boldsymbol{R^{real}(i,j)} = \psi_{in}^{real}(j) \times \boldsymbol{U^{real}(i,j)} - \psi_{in}^{imag}(j) \times \boldsymbol{U^{imag}(i,j)}$$

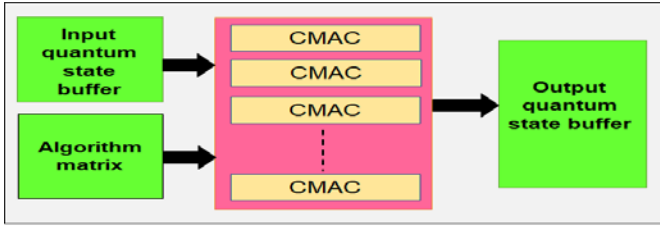
$$\boldsymbol{R^{imag}(i,j)} = \psi_{in}^{imag}(j) \times \boldsymbol{U^{real}(i,j)} + \psi_{in}^{real}(j) \times \boldsymbol{U^{imag}(i,j)}$$

Hardware Architectures – CMAC approach

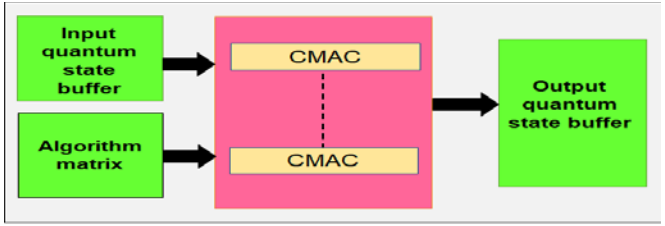
- **Single CMAC**
  - Fully optimized for area
  - One CMAC instance
  - N cycles to store input state vector elements
  - N^2 cycles to compute for all algorithm matrix elements
- ***N*-concurrent CMAC**
  - Fully optimized for speed
  - N parallel CMAC instances
  - N cycles to store input state vector elements
  - N cycles to compute for all algorithm matrix elements
- **Dual-sequential CMAC**
  - Two CMAC instances connected serially
  - Storage overlapped with computations
  - Improved execution time



Single CMAC



N-concurrent CMAC



Dual sequential CMAC

$N = 2^n$ basis states, and n = number of qubits

Hardware Architectures – CMAC approach

Complexity analysis of CMAC Architectures

- **Single CMAC**

$$O_{time} = (L_1 + N + N^2) \times T_{clock} = O(N^2)$$
$$O_{space} = 1 \times CMAC = O(1)$$

- ***N*-concurrent CMACs**

$$O_{time} = (L_2 + 2N) \times T_{clock} = O(N)$$
$$O_{space} = N \times CMACs = O(N)$$

- **Dual-sequential CMACs**

$$O_{time} = (L_3 + N^2) \times T_{clock} = O(N^2)$$
$$O_{space} = 2 \times CMACs = O(1)$$

Space and Time complexities of proposed architectures

CMAC Architecture	Complexity	
	Space (O_{space})	Time (O_{time})
Single	$O(1)$	$O(N^2)$
N-concurrent	$O(N)$	$O(N)$
Dual sequential	$O(1)$	$O(N^2)$

L_1, L_2, L_3 ≡ initial pipeline latencies
T_{clock} ≡ system clock period

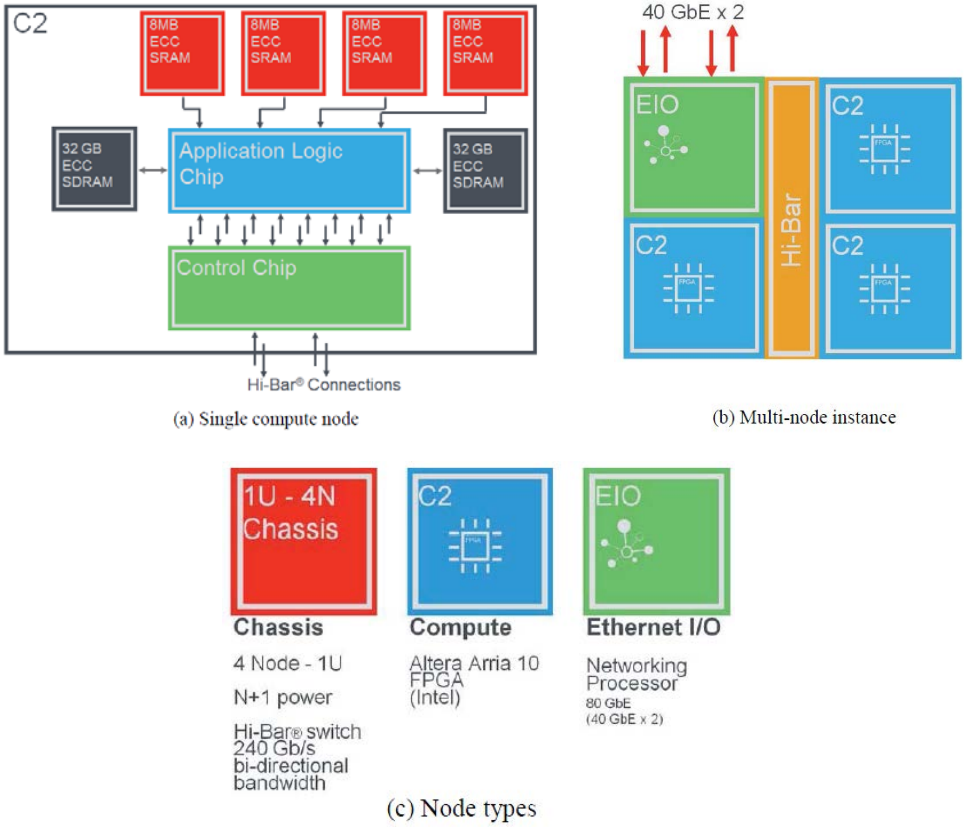
Outline

- Introduction and Motivation
- Background and Related Work
- Proposed Work
- **Experimental Work**
- 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
    - On-chip memory (OCM)
    - On-board memory (OBM)
    - Removes interconnection bottlenecks
    - Reduces resource contention and energy use
  - Highly productive development environment
    - Parallel High-Level Language
    - C++-to-HW (previously Carte-C) compiler
    - Smooth learning curve + fast development time

DirectStream (DS8) system



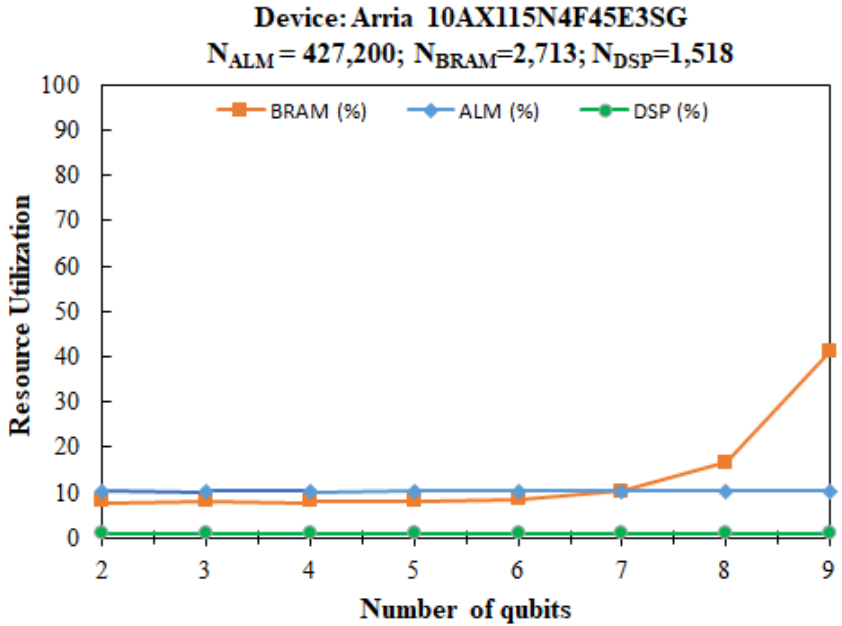
Experimental Results

- **QFT emulation using on-chip memory (OCM) architectures**

Single CMAC (*lookup*)

Number of qubits	On-chip resource* utilization (%)			Emulation time (sec)**
	ALMs	BRAMs	DSPs	
2	10.30	8.04	1.05	1.4E-6
3	10.24	8.12	1.05	1.15E-6
4	10.24	8.11	1.05	2.01E-6
5	10.27	8.18	1.05	5.37E-6
6	10.26	8.55	1.05	1.87E-5
7	10.26	10.25	1.05	7.17E-5
8	10.29	16.73	1.05	3.19E-4
9	10.31	41.28	1.05	0.0013

*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Operating frequency: 233 MHz



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

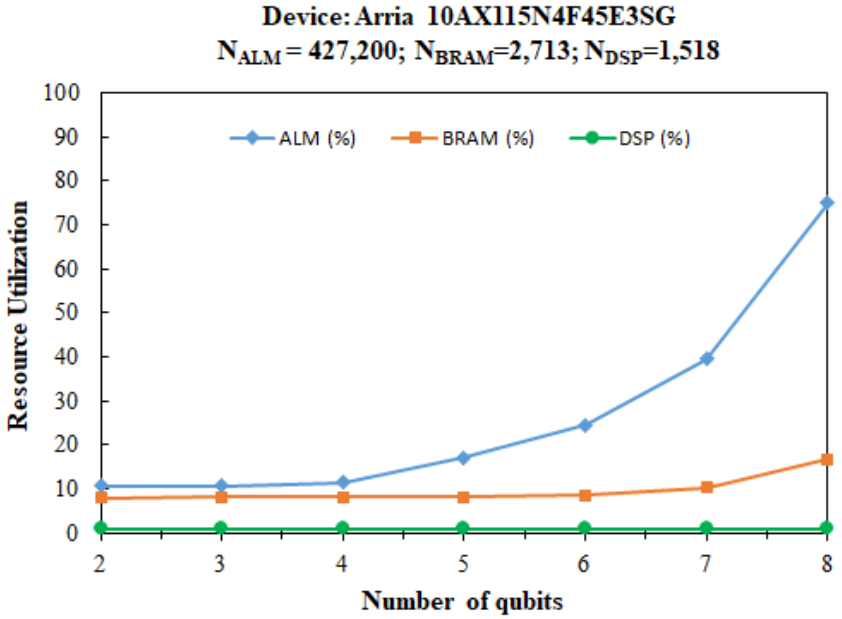
Experimental Results

- **QFT emulation using on-chip memory (OCM) architectures**

***N*-concurrent CMAC (*lookup*)**

Number of qubits	On-chip resource* utilization (%)			Emulation time (sec)**
	ALMs	BRAMs	DSPs	
2	10.70	8.04	1.05	6.78E-7
3	10.74	8.12	1.05	7.64E-7
4	11.53	8.11	1.05	9.36E-7
5	17.10	8.18	1.05	1.28E-6
6	24.5	8.55	1.05	1.97E-6
7	39.5	10.25	1.05	3.34E-6
8	74.88	16.73	1.05	6.09E-6

*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Operating frequency: 233 MHz



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

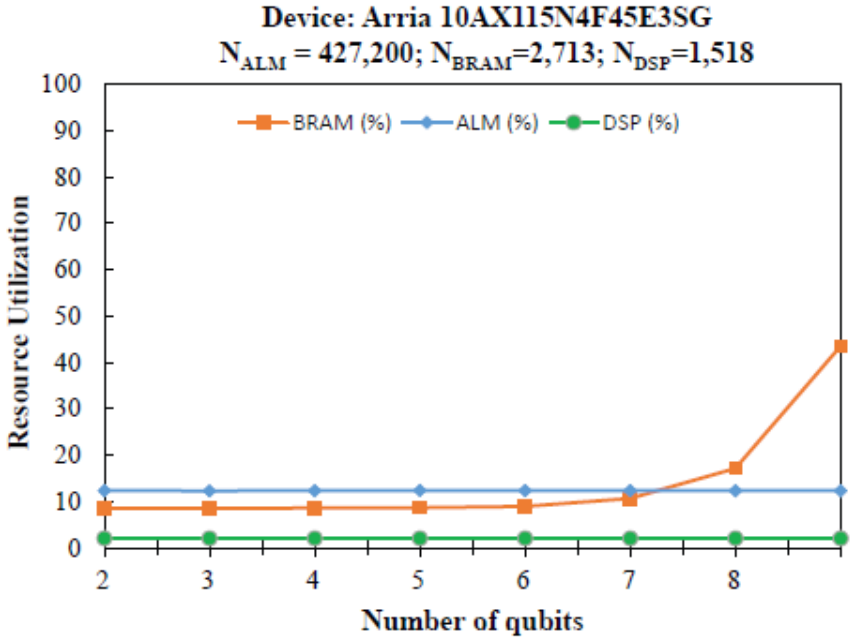
Experimental Results

- **QFT emulation using on-chip memory (OCM) architectures**

Dual sequential CMAC (*lookup*)

Number of qubits	On-chip resource* utilization (%)			Emulation time (sec)**
	ALMs	BRAMs	DSPs	
2	12.39	8.55	2.11	7.55E-7
3	12.34	8.55	2.11	9.61E-7
4	12.36	8.63	2.11	1.79E-6
5	12.43	8.70	2.11	5.08E-6
6	12.38	8.99	2.11	1.83E-5
7	12.39	10.69	2.11	7.1E-5
8	12.37	17.18	2.11	0.0003
9	12.37	43.54	2.11	0.0011

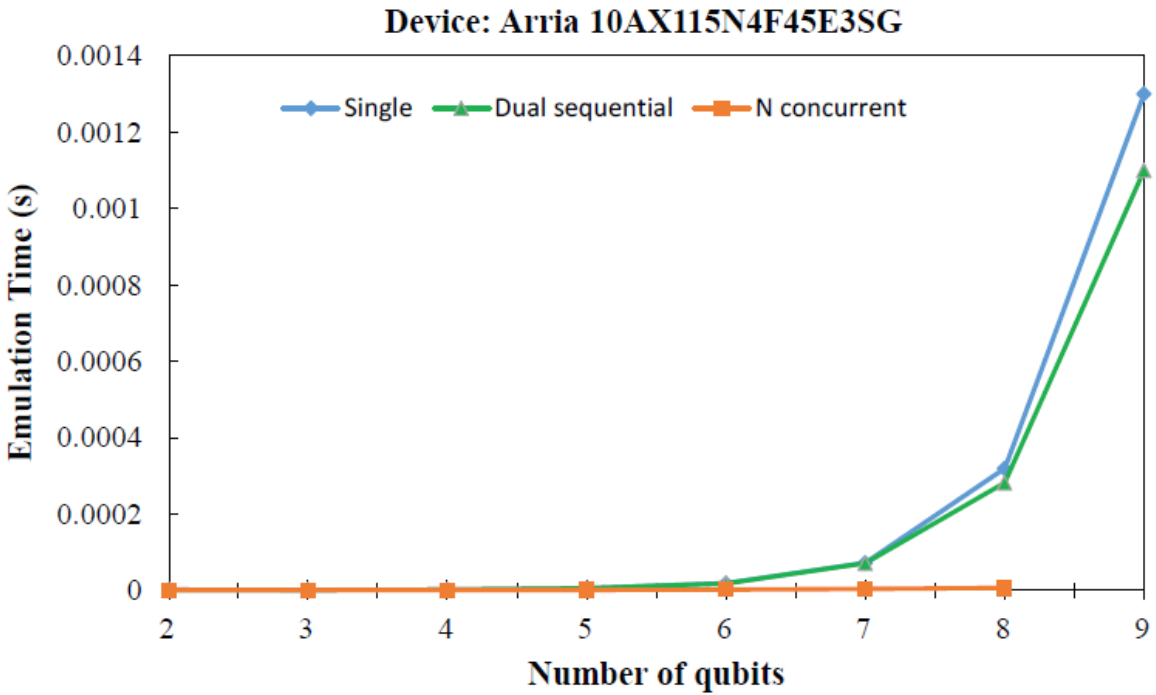
*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Operating frequency: 233 MHz



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 of execution times for on-chip memory (OCM) architectures



Experimental Results

- **QFT emulation using on-board memory (OBM) architectures**

Single CMAC (*lookup*)

Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)		Emulation time (sec)***
	ALMs	BRAM	DSPs	SRAM	SDRAM	
2	10.71	8.44	1.05	32	128	1.7E-6
3	10.71	8.44	1.05	64	512	2.0E-6
4	10.71	8.44	1.05	128	2K	3.9E-6
5	10.71	8.44	1.05	256	8K	1.1E-5
6	10.71	8.44	1.05	512	32K	3.9E-5
7	10.71	8.44	1.05	1K	128K	0.00015
8	10.71	8.44	1.05	2K	512K	0.00061
9	10.71	8.44	1.05	4K	2M	0.00241
10	10.71	8.44	1.05	8K	8M	0.00963
11	10.71	8.44	1.05	16K	32M	0.03851
12	10.71	8.44	1.05	32K	128M	0.15399
13	10.71	8.44	1.05	64K	512M	0.61586
14	10.71	8.44	1.05	128K	2G	2.36324
15	10.71	8.44	1.05	256K	8G	9.853
16	10.71	8.44	1.05	512K	32G	39.4209

Dual-sequential CMAC (*lookup*)

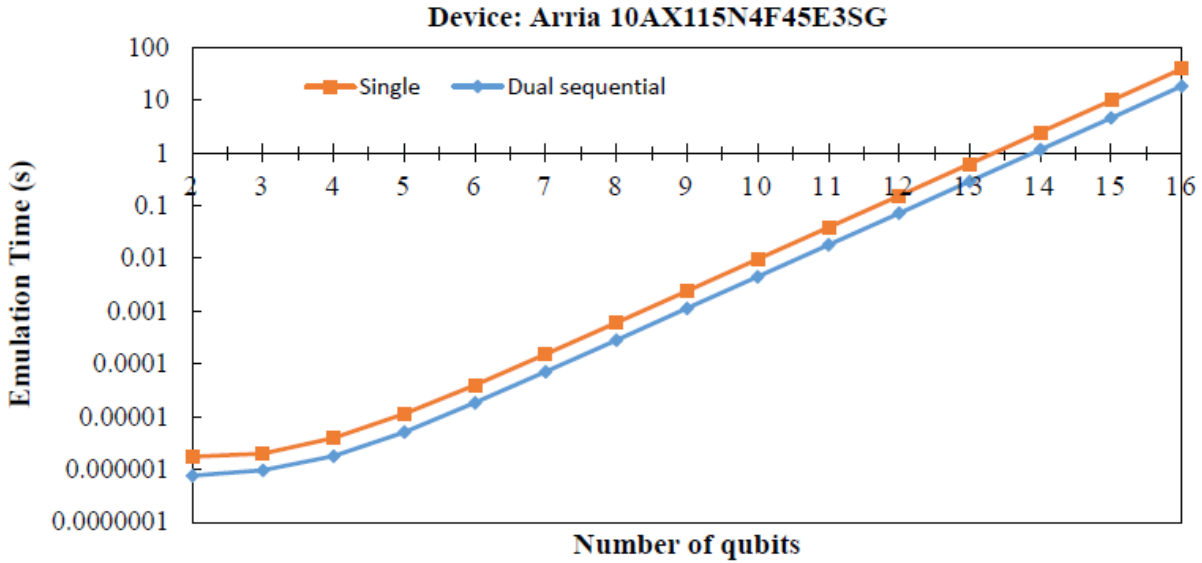
Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)		Emulation time (sec)***
	ALMs	BRAM	DSPs	SRAM	SDRAM	
2	12	8.63	2.11	32	128	7.55E-7
3	12	8.63	2.11	64	512	9.61E-7
4	12	8.63	2.11	128	2K	1.79E-6
5	12	8.63	2.11	256	8K	5.08E-6
6	12	8.63	2.11	512	32K	1.83E-5
7	12	8.63	2.11	1K	128K	7.10E-5
8	12	8.63	2.11	2K	512K	0.00028
9	12	8.63	2.11	4K	2M	0.00113
10	12	8.63	2.11	8K	8M	0.00451
11	12	8.63	2.11	16K	32M	0.018002
12	12	8.63	2.11	32K	128M	0.072006
13	12	8.63	2.11	64K	512M	0.2888021
14	12	8.63	2.11	128K	2G	1.152083
15	12	8.63	2.11	256K	8G	4.608329
16	12	8.63	2.11	512K	32G	18.4331

*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Total on-board memory: 4 parallel SRAM banks of 8MB each and 2 parallel SDRAM banks of 32GB each
***Operating frequency: 233 MHz

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 of execution times for on-board memory (OBM) architectures



Experimental Results

- **QFT emulation using on-board memory (OBM) architectures**

Dual-sequential CMAC (*dynamic generation*)

Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)	Emulation time (sec)***
	ALMs	BRAMs	DSPs	SDRAM	
2	13.16	9.58	3.23	32	1.99E-6
4	13.16	9.58	3.23	128	3.02E-6
6	13.16	9.58	3.23	512	1.95E-5
8	13.16	9.58	3.23	2K	0.0003
10	13.16	9.58	3.23	8K	0.0045
12	13.16	9.58	3.23	32K	0.0720
14	13.16	9.58	3.23	128K	1.1521
16	13.16	9.58	3.23	512K	18.433
18	13.16	9.58	3.23	2M	294.93
20	13.16	9.58	3.23	8M	4718.934
22	13.16	9.58	3.23	32M	18876†
24	13.16	9.58	3.23	128M	302012†
26	13.16	9.58	3.23	512M	4832188†
28	13.16	9.58	3.23	2G	7.73E+7 †
30	13.16	9.58	3.23	8G	1.23E+9 †
32	13.16	9.58	3.23	32G	1.979E+10 †

Dual-sequential CMAC (*stream*)

Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)	Emulation time (sec)***
	ALMs	BRAMs	DSPs	SDRAM	
2	11	8	1	32	2.3E-6
4	11	8	1	128	3.4E-6
6	11	8	1	512	2.0E-5
8	11	8	1	2K	2.8E-4
10	11	8	1	8K	4.5E-3
12	11	8	1	32K	7.2E-2
14	11	8	1	128K	1.15E0
16	11	8	1	512K	1.84E+1
18	11	8	1	2M	2.95E+2
20	11	8	1	8M	4.72E+3
22	11	8	1	32M	7.5E+4†
24	11	8	1	128M	1.2E+6†
26	11	8	1	512M	1.93E+7†
28	11	8	1	2G	3.09E+8†
30	11	8	1	8G	4.95E+9†
32	11	8	1	32G	7.92E+10†

*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Total on-board memory: 4 parallel SRAM banks of 8MB each and 2 parallel 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

Grover's search using OBM architectures
Single CMAC (*stream*)

Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)	Emulation time (sec)***
	ALMs	BRAMs	DSPs	SDRAM	
2	11	8	1	32	2.3E-6
4	11	8	1	128	3.4E-6
6	11	8	1	512	2.0E-5
8	11	8	1	2K	2.8E-4
10	11	8	1	8K	4.5E-3
12	11	8	1	32K	7.2E-2
14	11	8	1	128K	1.15E0
16	11	8	1	512K	1.84E+1
18	11	8	1	2M	2.95E+2
20	11	8	1	8M	4.72E+3
22	11	8	1	32M	7.5E+4†
24	11	8	1	128M	1.2E+6†
26	11	8	1	512M	1.93E+7†
28	11	8	1	2G	3.09E+8†
30	11	8	1	8G	4.95E+9†
32	11	8	1	32G	7.92E+10†

Quantum Haar transform (QHT) using OBM architectures
Kernel approach

Number of pixels	Qubits	On-chip resource* utilization (%)			OBM** Utilization (bytes)	Emulation time (sec)***
		ALMs	BRAMs	DSPs	SDRAM	
16x16	8	14	9	2	4K	6.73E-6
32x32	10	14	9	2	16K	1.66E-5
64x64	12	14	9	2	64K	5.62E-5
128x128	14	14	9	2	256K	0.0002
256x256	16	14	9	2	1M	0.0008
512x512	18	14	9	2	4M	0.0034
1024x1024	20	14	9	2	16M	0.0135
2048x2048	22	14	9	2	64M	0.0540
4Kx4K	24	14	9	2	256M	0.2160
8Kx8K	26	14	9	2	1G	0.8641
16Kx16K	28	14	9	2	4G	3.4563
32Kx32K	30	14	9	2	16G	13.825

*Total on-chip resources: N_{ALM}=427,000, N_{BRAM}=2,713, N_{DSP}=1,518
**Total on-board memory: 4 parallel SRAM banks of 8MB each and 2 parallel 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

- **Comparison with related work (FPGA 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**	**20**	**32-bit floating pt.**	**233**	**18.4**
	QHT	**30**			**13.8**
	Grover's search	**20**			**18.4**

Conclusions

- **Supremacy of Quantum Computing**
- **Need for Quantum Emulation**
  - Emulation using FPGAs
- **Proposed Approaches and Methods**
  - Direct circuit/gate approach
  - CMAC approach + *lookup / dynamic generation / stream*
  - Kernel approach
- **Case studies**
  - Quantum Fourier Transform (QFT)
  - Multi-dimensional Quantum Haar Transform (QHT)
  - Single-pattern / multi-pattern Grover's search algorithm
- **Testbed Platform**
  - State-of-the-art HPRC system from DirectStream
  - C++ to hardware compiler

Future Work

- **More algorithms**
  - Integer factoring using Shor's algorithm
  - Dimension reduction using QHT
  - Image pattern recognition using QHT and Grover's search
- **Design Optimizations**
  - Combining / sharing resources, e.g., multipliers
- **Accuracy trade-off study**
  - Fixed-point vs. floating-point implementations for higher scalability
- **Quantum error correction (QEC)**
- **Power efficiency**


THANK YOU