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
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

◆ 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

  ▪ 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. Dlugopolksi, “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

  - Results of 5-qubit QFT and 7-qubit Grover’s reported
  - Up to 24-bit fixed-point precision
  - No optimizations to make designs scalable

  - 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⟩, |1⟩
    - Pure states → |ψ⟩
  - 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

\[
|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \equiv \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \text{and}
\]

\[
p(\psi \rightarrow |0\rangle) = |\alpha|^2; p(\psi \rightarrow |1\rangle) = |\beta|^2
\]

\[
|q_1q_2q_3\rangle = |q_1\rangle \otimes |q_2\rangle \otimes |q_3\rangle
\]

\[
|\psi\rangle = \alpha_1 \alpha_2 \alpha_3 |000\rangle + \alpha_1 \alpha_2 \beta_3 |001\rangle + \alpha_1 \beta_2 \alpha_3 |010\rangle + \ldots + \beta_1 \beta_2 \beta_3 |111\rangle
\]
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**

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

\[
U_{QFT} \rightarrow |\psi_{out}\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{q=0}^{N-1} f(q\Delta t)e^{2\pi i qk/N}|k\rangle = \sum_{k=0}^{N-1} C_k^{out}|k\rangle
\]

- **QFT matrix**

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

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

\[ T_1 = H \otimes I \otimes I \otimes I \otimes I \]
\[ T_2 = CR_2 \otimes I \otimes I \otimes I \]
\[ T_3 = (I \otimes SW \otimes I \otimes I). (CR_3 \otimes I \otimes I \otimes I). \\
   (I \otimes SW \otimes I \otimes I) \]
\[ T_4 = (I \otimes I \otimes SW \otimes I). (I \otimes SW \otimes I \otimes I). \\
   (CR_4 \otimes I \otimes I \otimes I). (I \otimes SW \otimes I \otimes I). \\
   (I \otimes I \otimes SW \otimes I) \]
\[ T_5 = (I \otimes I \otimes I \otimes SW). (I \otimes I \otimes SW \otimes I). \\
   (I \otimes SW \otimes I \otimes I). (CR_5 \otimes I \otimes I \otimes I). \\
   (I \otimes SW \otimes I \otimes I). (I \otimes I \otimes I \otimes SW) \]

Modeling the quantum circuit as a series of transformations

**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

\[
\psi_{out} = U_{ALG} \cdot \psi_{in}
\]
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

\[
\begin{align*}
\psi_{out}^{\text{real}}(i) &= \sum_{j=0}^{N-1} R^{\text{real}}(i,j) \\
\psi_{out}^{\text{imag}}(i) &= \sum_{j=0}^{N-1} R^{\text{imag}}(i,j)
\end{align*}
\]

where,
\[
\begin{align*}
R^{\text{real}}(i,j) &= \psi_{in}^{\text{real}}(j) \times U^{\text{real}}(i,j) - \psi_{in}^{\text{imag}}(j) \times U^{\text{imag}}(i,j) \\
R^{\text{imag}}(i,j) &= \psi_{in}^{\text{imag}}(j) \times U^{\text{real}}(i,j) + \psi_{in}^{\text{real}}(j) \times U^{\text{imag}}(i,j)
\end{align*}
\]
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

\[ N = 2^n \text{ basis states, and } n = \text{number of qubits} \]
Hardware Architectures – CMAC approach

**Complexity analysis of CMAC Architectures**

- **Single CMAC**
  \[
  O_{\text{time}} = (L_1 + N + N^2) \times T_{\text{clock}} = O(N^2) \\
  O_{\text{space}} = 1 \times \text{CMAC} = O(1)
  \]

- **N-concurrent CMACs**
  \[
  O_{\text{time}} = (L_2 + 2N) \times T_{\text{clock}} = O(N) \\
  O_{\text{space}} = N \times \text{CMACs} = O(N)
  \]

- **Dual-sequential CMACs**
  \[
  O_{\text{time}} = (L_3 + N^2) \times T_{\text{clock}} = O(N^2) \\
  O_{\text{space}} = 2 \times \text{CMACs} = O(1)
  \]

---

Space and **Time** complexities of proposed architectures

<table>
<thead>
<tr>
<th>CMAC Architecture</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Space</strong> $O_{\text{space}}$</td>
</tr>
<tr>
<td>Single</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>N-concurrent</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>Dual sequential</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

$L_1, L_2, L_3 \equiv$ initial pipeline latencies

$T_{\text{clock}} \equiv$ 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
Experimental Results

QFT emulation using on-chip memory (OCM) architectures

Single CMAC (lookup)

<table>
<thead>
<tr>
<th>Number of qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>Emulation time (sec)**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
</tr>
<tr>
<td>2</td>
<td>10.30</td>
<td>8.04</td>
</tr>
<tr>
<td>3</td>
<td>10.24</td>
<td>8.12</td>
</tr>
<tr>
<td>4</td>
<td>10.24</td>
<td>8.11</td>
</tr>
<tr>
<td>5</td>
<td>10.27</td>
<td>8.18</td>
</tr>
<tr>
<td>6</td>
<td>10.26</td>
<td>8.55</td>
</tr>
<tr>
<td>7</td>
<td>10.26</td>
<td>10.25</td>
</tr>
<tr>
<td>8</td>
<td>10.29</td>
<td>16.73</td>
</tr>
<tr>
<td>9</td>
<td>10.31</td>
<td>41.28</td>
</tr>
</tbody>
</table>

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

Device: Arria 10AXI15N4F45E3SG
N_{ALM} = 427,200; N_{BRAM} = 2,713; N_{DSP} = 1,518

ALM ≡ Adaptive Logic Modules
BRAM ≡ Block Random Access Memory
DSP ≡ Digital Signal Processing block
### Experimental Results

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

#### N-concurrent CMAC (lookup)

<table>
<thead>
<tr>
<th>Number of qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>Emulation time (sec)**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
</tr>
<tr>
<td>2</td>
<td>10.70</td>
<td>8.04</td>
</tr>
<tr>
<td>3</td>
<td>10.74</td>
<td>8.12</td>
</tr>
<tr>
<td>4</td>
<td>11.53</td>
<td>8.11</td>
</tr>
<tr>
<td>5</td>
<td>17.10</td>
<td>8.18</td>
</tr>
<tr>
<td>6</td>
<td>24.5</td>
<td>8.55</td>
</tr>
<tr>
<td>7</td>
<td>39.5</td>
<td>10.25</td>
</tr>
<tr>
<td>8</td>
<td>74.88</td>
<td>16.73</td>
</tr>
</tbody>
</table>

*Total on-chip resources: \(N_{\text{ALM}}=427,000\), \(N_{\text{BRAM}}=2,713\), \(N_{\text{DSP}}=1,518\)

**Operating frequency: 233 MHz**

---

**Diagram:**

Device: Arria 10AX115N4F45E3SG

\(N_{\text{ALM}} = 427,200\); \(N_{\text{BRAM}} = 2,713\); \(N_{\text{DSP}} = 1,518\)

**Legend:**
- ALM: Adaptive Logic Modules
- BRAM: Block Random Access Memory
- DSP: Digital Signal Processing block
### Experimental Results

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

#### Dual sequential CMAC (lookup)

<table>
<thead>
<tr>
<th>Number of qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>Emulation time (sec)**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
</tr>
<tr>
<td>2</td>
<td>12.39</td>
<td>8.55</td>
</tr>
<tr>
<td>3</td>
<td>12.34</td>
<td>8.55</td>
</tr>
<tr>
<td>4</td>
<td>12.36</td>
<td>8.63</td>
</tr>
<tr>
<td>5</td>
<td>12.43</td>
<td>8.70</td>
</tr>
<tr>
<td>6</td>
<td>12.38</td>
<td>8.99</td>
</tr>
<tr>
<td>7</td>
<td>12.39</td>
<td>10.69</td>
</tr>
<tr>
<td>8</td>
<td>12.37</td>
<td>17.18</td>
</tr>
<tr>
<td>9</td>
<td>12.37</td>
<td>43.54</td>
</tr>
</tbody>
</table>

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

**Operating frequency: 233 MHz

---

**Notes:**
- $ALM \equiv$ Adaptive Logic Modules
- $BRAM \equiv$ Block Random Access Memory
- $DSP \equiv$ Digital Signal Processing block
Comparison of execution times for on-chip memory (OCM) architectures
## Experimental Results

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

### Single CMAC (lookup)

<table>
<thead>
<tr>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAM</td>
<td>DSPs</td>
</tr>
<tr>
<td>2</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>3</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>4</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>5</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>6</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>7</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>8</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>9</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>10</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>11</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>12</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>13</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>14</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>15</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
<tr>
<td>16</td>
<td>10.71</td>
<td>8.44</td>
<td>1.05</td>
</tr>
</tbody>
</table>

*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

### Dual-sequential CMAC (lookup)

<table>
<thead>
<tr>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAM</td>
<td>DSPs</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>7</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>8.63</td>
<td>2.11</td>
</tr>
</tbody>
</table>
Experimental Results

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

![Graph showing comparison of execution times for single and dual sequential OBM architectures. The x-axis represents the number of qubits, and the y-axis represents emulation time in seconds. The graph compares performance for different numbers of qubits, with single and dual sequential architectures.]
Experimental Results

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

### Dual-sequential CMAC (dynamic generation)

<table>
<thead>
<tr>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
<td>DSPs</td>
</tr>
<tr>
<td>2</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>4</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>6</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>8</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>10</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>12</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>14</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>16</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>18</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>20</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>22</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>24</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>26</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>28</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>30</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
<tr>
<td>32</td>
<td>13.16</td>
<td>9.58</td>
<td>3.23</td>
</tr>
</tbody>
</table>

*Total on-chip resources: \(N_{ALMs}=427,000, N_{BRAMs}=2,713, N_{DSPs}=1,518\)

**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

### Dual-sequential CMAC (stream)

<table>
<thead>
<tr>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
<td>DSPs</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

ALM ≡ Adaptive Logic Modules
BRAM ≡ Block Random Access Memory
DSP ≡ Digital Signal Processing block

ASAP 2019 – July 16th, 2019
## Experimental Results

### Grover's search using OBM architectures

**Single CMAC (stream)**

<table>
<thead>
<tr>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
<td>DSPs</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>11</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

*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

### Quantum Haar transform (QHT) using OBM architectures

**Kernel approach**

<table>
<thead>
<tr>
<th>Number of pixels</th>
<th>Qubits</th>
<th>On-chip resource* utilization (%)</th>
<th>OBM** Utilization (bytes)</th>
<th>Emulation time (sec)***</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALMs</td>
<td>BRAMs</td>
<td>DSPs</td>
<td>SDRAM</td>
</tr>
<tr>
<td>16x16</td>
<td>8</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>32x32</td>
<td>10</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>64x64</td>
<td>12</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>128x128</td>
<td>14</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>256x256</td>
<td>16</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>512x512</td>
<td>18</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>1024x1024</td>
<td>20</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>2048x2048</td>
<td>22</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>4Kx4K</td>
<td>24</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>8Kx8K</td>
<td>26</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>16Kx16K</td>
<td>28</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>32Kx32K</td>
<td>30</td>
<td>14</td>
<td>9</td>
<td>2</td>
</tr>
</tbody>
</table>

*ALMs ≡ Adaptive Logic Modules
BRAM ≡ Block Random Access Memory
DSP ≡ Digital Signal Processing block

**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
## Experimental Results

### Comparison with related work (FPGA emulation)

<table>
<thead>
<tr>
<th>Reported Work</th>
<th>Algorithm</th>
<th>Number of qubits</th>
<th>Precision</th>
<th>Operating frequency (MHz)</th>
<th>Emulation time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fujishima (2003)</td>
<td>Shor’s factoring</td>
<td>-</td>
<td>-</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>Khalid et al (2004)</td>
<td>QFT</td>
<td>3</td>
<td>16-bit fixed pt.</td>
<td>82.1</td>
<td>61E-9</td>
</tr>
<tr>
<td></td>
<td>Grover’s search</td>
<td>3</td>
<td>16-bit fixed pt.</td>
<td>84E-9</td>
<td></td>
</tr>
<tr>
<td>Lee et al (2016)</td>
<td>QFT</td>
<td>5</td>
<td>24-bit fixed pt.</td>
<td>90</td>
<td>219E-9</td>
</tr>
<tr>
<td></td>
<td>Grover’s search</td>
<td>7</td>
<td>24-bit fixed pt.</td>
<td>85</td>
<td>96.8E-9</td>
</tr>
<tr>
<td>Silva and Zabaleta (2017)</td>
<td>QFT</td>
<td>4</td>
<td>32-bit floating pt.</td>
<td>-</td>
<td>4E-6</td>
</tr>
<tr>
<td>Pilch and Dlugopolski (2018)</td>
<td>Deutsch</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Proposed work</strong></td>
<td>QFT</td>
<td>20</td>
<td>32-bit floating pt.</td>
<td>233</td>
<td>18.4</td>
</tr>
<tr>
<td></td>
<td>QHT</td>
<td>30</td>
<td></td>
<td></td>
<td>13.8</td>
</tr>
<tr>
<td></td>
<td>Grover’s search</td>
<td>20</td>
<td></td>
<td></td>
<td>18.4</td>
</tr>
</tbody>
</table>
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