QASM-to-HLS: A Framework for Accelerating Quantum Circuit Emulation on High-Performance Reconfigurable Computers

Anshul Maurya
amaurya2022@my.fit.edu
Florida Institute of Technology
Melbourne, Florida, USA

Naveed Mahmud
nmahmud@fit.edu
Florida Institute of Technology
Melbourne, Florida, USA

ABSTRACT

High-performance reconfigurable computers (HPRCs) make use of Field-Programmable Gate Arrays (FPGAs) for efficient emulation of quantum algorithms. Generally, algorithm-specific architectures are implemented on the FPGAs and there is very little flexibility. Moreover, mapping a quantum algorithm onto its equivalent FPGA emulation architecture is challenging. In this work, we present an automation framework for converting quantum algorithms/circuits to their equivalent FPGA emulation architectures. The framework processes quantum circuits represented in Quantum Assembly Language (QASM) and derives high-level descriptions of the hardware emulation architectures for High-Level Synthesis (HLS) on HPRCs. Experimental results show that the framework-generated architectures deployed on an HPRC perform faster than a state-of-the-art software simulator.

KEYWORDS

Quantum Computing, Field-Programmable Gate Arrays

ACM Reference Format:
Anshul Maurya and Naveed Mahmud. 2023. QASM-to-HLS: A Framework for Accelerating Quantum Circuit Emulation on High-Performance Reconfigurable Computers. In *Proceedings of ACM Conference (SC'23).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The simulation of quantum circuits on classical platforms has been made necessary due to the noisy nature and intermediate scale of current quantum devices. The use of Field-Programmable Gate Arrays (FPGAs) to accelerate the simulation of quantum circuits (also known as FPGA-based emulation) has been investigated for some time. Existing FPGA-based emulation methods[1][2] exhibit limited flexibility in emulating a variety of quantum algorithms due to their fixed algorithm-specific architectures. Additionally, the computational load of generating matrix representations of quantum circuits generally fall on the CPU, thereby restricting the extent to which FPGAs can be leveraged for accelerating the emulation of quantum circuits/algorithms. Mapping a quantum algorithm to an FPGA hardware architecture for emulation is also challenging, particularly for quantum algorithm developers with limited FPGA design experience. In this work, we propose an automation framework to interface between front-end quantum algorithm design and backend FPGA-based emulation. We develop a methodology for converting quantum circuits represented in Quantum Assembly Language (QASM) [3] into corresponding digital hardware architectures for emulation on FPGAs. The proposed framework automatically derives high-level descriptions of the hardware emulation architectures for High-Level Synthesis (HLS) [4] on FPGA backends.

SC'23, Nov. 2023, Denver, CO, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 PROPOSED FRAMEWORK

We present a framework that automatically derives efficient hardware emulation architectures described in high-level languages for HLS. HLS is a fast and convenient entry point into the hardware design process. This work implements an efficient interface between the QASM representation of quantum circuits and the HLS design process for facilitating emulation of quantum circuits on FPGA hardware. The proposed methodology is shown in Fig. 1.

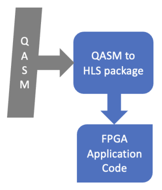

Figure 1: Work flow of proposed framework.

2.1 QASM Parsing

In order to derive the FPGA application code from QASM, we have developed a software package, hereby named *QASM-to-HLS*. This package takes QASM code as input and generates a high-level language based application code for FPGA, as depicted in Figure 1. The *QASM-to-HLS* parses the QASM code and produces HLS application code, consisting of host (PC) code and kernel (FPGA) code. During QASM parsing, information about gate types and their positions within the circuit are extracted. Next, gates are organized into circuit layers, see Figure 2, where the width of each layer equals the number of qubits. With the exception of non-entangling gates, each gate's inclusion within the list of layers is determined by its sequential position within the circuit. An absence of a gate in the sequence gets populated with identity gates. Entangling gates such as controlled NOT[5] are processed as single layers without forming pairs with other gates. After circuit layer formation, tensor product of respective gate matrices are performed, resulting in a combined matrix for each layer.

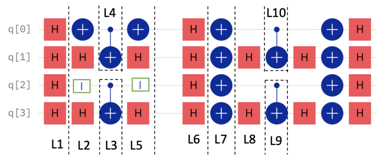

Figure 2: Circuit Layering

2.2 FPGA Architecture

After parsing the QASM code, *QASM-to-HLS* generates the corresponding HLS code describing the FPGA hardware architecture. The derived FPGA architectures support 64-bit floating-point precision to perform complex number arithmetic. *QASM-to-HLS* can derive three types of architectures that explore space-time trade-offs.

2.2.1 TYPE-1 Design. The design involves consecutive multiplication of the initial quantum state vector $S_{i/p}$ with each circuit layer matrix M_{Li}, to produce the final output state vector, $S_{o/p}$. This is performed by a single kernel that takes a layer matrix and state vector as inputs and performs complex matrix-vector computation, see Figure 3, where $K = 1$. The expression for total execution time of this architecture is $(t_{avg}+t^c_{avg})\times L$, where t_{avg} is the average time of data transfer between host and kernel, t^c_{avg} is the average computation time of kernel, and L is the total number of circuit layers. The space complexity of the architecture using 64-bit floating-point precision is $2^{n+5} + 2^{2n+4}$, where n is the number of qubits.

2.2.2 TYPE-2 Design. The implemented kernel in this design, see Figure 3, takes K matrices as input, reducing the number of data transfers from host to FPGA. This design is optimal for emulation of small circuits and utilizes the FPGA resources more efficiently. The data transfer time and overall application time is also reduced compared to Type-1 design. For K parallel matrix inputs, the number kernel calls will be $r = L/K$. The time expression of this design is $r \times t_{avg} + L \times t^c_{avg}$ and space complexity will be $2^{n+5} + K \cdot 2^{2n+4}$.

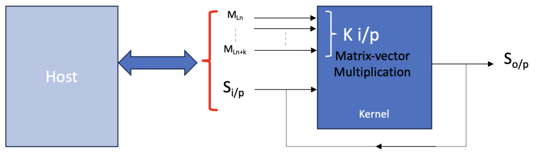

Figure 3: Type-2 Design

2.2.3 TYPE-3 Design. The approach involves passing K layer matrices into kernel buffers. This architecture exploits the fact that each layer is independent in a quantum circuit and can be multiplied in concurrence. Matrix-Matrix multiplication results are stored in intermediate buffers and then multiplied later to produce a final matrix representing the full quantum circuit. The resultant final matrix is used in matrix-vector computation to obtain the output state vector. Figure 4 shows the total architecture design with two kernels, one for matrix-matrix multiplication with time t^c, and another for matrix-vector multiplication with time $t^{c'}$. The total time and space complexity, T and S respectively, of this design is given by (1).

$$T = \left(t^c_{avg} \cdot \log_2 K + t_{avg}\right) \times r + t^{c'}$$
$$S = K \cdot \left(\frac{k}{2} + 1\right) \cdot 2^{2n+4} \tag{1}$$

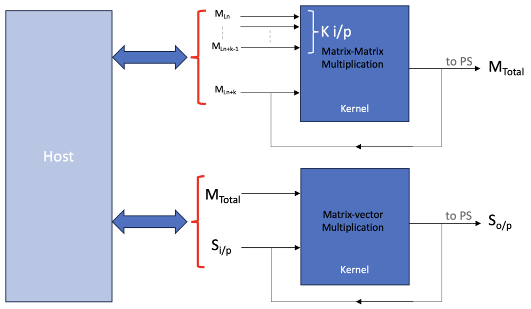

Figure 4: Type-3 Design

3 EXPERIMENTAL RESULTS

We present our preliminary results in Table 1 where a *QASM-to-HLS* generated Type-1 architecture with $K = 1$ was deployed on a Xilinx Alveo U-200 accelerator [6] for randomized circuits with constant depth. Compared to IBM's software-based statevector simulator[7], the Type-1 architecture shows up to around ×100 speedup.

Table 1: Experimental offshoots for Type-1 design.

Number of Qubits	FPGA Kernel Execution Time (ms)	LUT%	Register%	BRAM%	DSP%	Software Simulation Time (ms)
3	0.012	0.52	0.46	1.06	0.16	7.16
5	0.131	0.51	0.47	1.25	0.16	12
7	1.923	0.52	0.46	3.89	0.16	37.1

4 CONCLUSION AND FUTURE WORK

FPGAs can be used for efficient emulation of quantum algorithms, however mapping quantum circuits to FPGA emulation architectures is challenging. The proposed automation framework facilitates automatic mapping of quantum circuits to efficient FPGA emulation architectures. Future work will include implementations of all architecture types and further optimizations.

REFERENCES

[1] Yee Hui Lee, Mohamed Khalil-Hani, Muhammad Nadzir Marsono, et al. An fpga-based quantum computing emulation framework based on serial-parallel architecture. *International Journal of Reconfigurable Computing*, 2016, 2016.
[2] Yunpyo Hong, Seokhun Jeon, Sihyeong Park, and Byung-Soo Kim. Quantum circuit simulator based on fpga. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1909–1911, 2022.
[3] Andrew W Cross, Lev S Bishop, John A Smolin, and Jay M Gambetta. Open quantum assembly language. *arXiv preprint arXiv:1707.03429*, 2017.
[4] Philippe Coussy and Adam Morawiec. *High-level synthesis*, volume 1. Springer, 2010.
[5] Marc Bataille. Quantum circuits of cnot gates. *arXiv preprint arXiv:2009.13247*, 2020.
[6] Alveo U200 Data Center Accelerator Card. *https://www.xilinx.com/products/boards-and-kits/alveo/u200.htmloverview.*
[7] IBM Simulators overview. *https://quantum-computing.ibm.com/lab/docs/iql/manage/simulator.*