Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2016, Article ID 5718124, 18 pages
http://dx.doi.org/10.1155/2016/5718124

Research Article

An FPGA-Based Quantum Computing Emulation Framework Based on Serial-Parallel Architecture

Y. H. Lee, M. Khalil-Hani, and M. N. Marsono

VeCAD Research Laboratory, Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), 81310 Skudai, Johor Bahru, Malaysia

Correspondence should be addressed to M. Khalil-Hani; khalil@fke.utm.my

Received 13 October 2015; Revised 19 February 2016; Accepted 14 March 2016

Academic Editor: João Cardoso

Hardware emulation of quantum systems can mimic more efficiently the parallel behaviour of quantum computations, thus allowing higher processing speed-up than software simulations. In this paper, an efficient hardware emulation method that employs a serial-parallel hardware architecture targeted for field programmable gate array (FPGA) is proposed. Quantum Fourier transform and Grover's search are chosen as case studies in this work since they are the core of many useful quantum algorithms. Experimental work shows that, with the proposed emulation architecture, a linear reduction in resource utilization is attained against the pipeline implementations proposed in prior works. The proposed work contributes to the formulation of a proof-of-concept baseline FPGA emulation framework with optimization on datapath designs that can be extended to emulate practical large-scale quantum circuits.

1. Introduction

Quantum computing is based on the properties of quantum mechanics, namely, superposition and entanglement. Superposition allows a quantum state to be in more than one basis state simultaneously, whereas entanglement is the strong correlation between multiqubit (quantum bit) basis states in a quantum system. Superposition and entanglement facilitate massive parallelism which enables exponential speed-ups to be achieved in the well-known integer factoring and discrete logarithms algorithms [1] and quadratic speed-ups in solving classically intractable brute-force searching and optimization problems [2, 3].

Similar to classical computing, quantum algorithms are developed long before any large-scale practical quantum computer is physically available. In 1994, Shor proposed the integer factoring and discrete logarithms algorithms [1] that brought the world's attention to the enormous potential of quantum computing. An example of this is the Rivest-Shamir-Adleman (RSA) security scheme [4] which is widely applied in current public key cryptosystem. It is based on the assumption that integer factoring of large number is intractable in classical computing. Shor's proposal, which, in contrast, factors integer in polynomial time, would make such security scheme no longer secure. In [5], Grover proposed a quantum search algorithm that is capable of identifying a specific element in an unordered m elements database in $(\Pi/4)\sqrt{m}$ attempts. This algorithm achieves a quadratic speed-up over the corresponding classical method that requires $m/2$ queries on average, to retrieve the desired data. Although the solution is only polynomially faster than the classical approach, Grover's quantum algorithm is an important one as it can be generalized to be applied in many intractable computer science problems. Recently, quantum equivalents for random walks [6], genetic algorithms [3], and NAND tree evaluation [7] have been developed.

Shor in [8] categorized quantum algorithms known to provide substantial speed-up over the classical approach into three types: (a) algorithms that achieve notable speed-up by applying quantum Fourier transform (QFT) in periodicity finding; examples of this type of algorithm include integer factoring and discrete logarithms algorithms [1], Simon's periodicity algorithm [9], Hallgren's algorithms for Pell's equation [10], and the quantum algorithms for solving hidden subgroup problems [11, 12]; (b) Grover's search algorithm and its extensions [2, 13] which in general offer square root

speed improvements over their classical counterparts; and (c) algorithms for simulating or solving problems in quantum mechanics [14].

Physical realization of a quantum computer is proving to be extremely challenging [15]. With research into viable large-scale quantum computers still ongoing, various technologies, namely, ion trap [16], nuclear magnetic resonance [17], and superconductor [18], were attempted. Nevertheless, only small-scale quantum computation implementations have been achieved [19, 20]. Instead of focusing on the realization of quantum gates, a different approach known as quantum annealing which solves optimization problems by finding the minimum point is used in the 128-qubit D-Wave One, 512-qubit D-Wave Two, and 1000-qubit D-Wave 2X systems [21, 22]. However, based on the research report presented in [23], the expected quantum speed-ups were not found in the D-Wave systems.

In parallel to efforts to develop physical quantum computers, there is also much effort in the theoretical research of quantum algorithms. Until large-scale practical quantum computers become prevalent, quantum algorithms are currently developed using the classical computing platform. However, due to their inherent sequential behaviour, classical computers that are based on Von Neumann architecture cannot simulate the inherent parallelism in quantum systems efficiently. On the other hand, the technology of field programmable gate array (FPGA) offers the potential of massive parallelism through hardware emulation. Consequently, significant improvement in speed performance over the equivalent software simulation can be achieved. However, FPGA is still a form of classical digital computing, and resource utilization on such a classical computing platform grows exponentially as the number of qubits increases. The problem is further compounded with the fact that accurate modelling of quantum circuit in FPGA technology is nonintuitive and therefore difficult, providing the research motivation for this paper.

This paper presents an efficient FPGA emulation framework for quantum computing. In the proposed emulation model, quantum computations are mapped to a serial-parallel architecture that facilitates scalability by managing the exponential growth of resource requirement against number of qubits. Quantum Fourier transform and Grover's search are chosen as case studies in this work since they are the core of many useful quantum algorithms, and in addition, they have been used as benchmarking models in prior works on FPGA emulation. Experimental results on the efficiencies of different FPGA emulation architectures and fixed point formats are presented, which will sufficiently demonstrate the feasibility of proposed framework.

The rest of this paper is organized as follows: Section 2 discusses prior works on FPGA-based quantum computing emulation, emphasizing issues of hardware architecture and modelling of quantum system on FPGA platform. In Section 3, the theoretical background on quantum computing and related quantum algorithms is provided. Section 4 presents the design of the proposed FPGA emulation models for QFT and Grover's search algorithms. Experimental results and analysis are given in Section 5. Finally, concluding remarks are made in Section 6.

2. Related Work

Modelling of a quantum system on classical computing platform is a challenging task. Hence, it is even more difficult to map quantum algorithms for emulation on classical computing environment based on FPGA, which is highly resource-constrained. Many attempts have been made in the last decade in FPGA emulation of quantum algorithms, and these works include [24–27]. However, details of the critical design processes such as mapping of the quantum algorithms into the FPGA emulation models and the verification of the implementations are not revealed in these prior works.

For software-based simulation using classical computer, various types of quantum simulators have been proposed. An open source C library, *libquantum*, for simulation of quantum computing is presented in [28] where pure quantum computer simulation as well as general quantum simulation is supported by the tool. In 2007, a variant of binary decision diagram named quantum information decision diagram (QuIDD) for compact state vector storage was introduced in [29] for efficient quantum circuit simulation. García and Markov [30] proposed a compact data structure based on stabilizer formalism called stabilizer frames.

Most of the previous FPGA emulation works are based on the quantum circuit model, which is essentially an interconnection of quantum gates. A different approach was taken by Goto and Fujishima [24] where a general purpose quantum processor was developed instead of applying the quantum circuit model. However, Fujishima's quantum processor assumed that the amplitudes of a quantum state can be either all zeros or with evenly distributed probability. In its emulation of Shor's integer factoring algorithm, details of the implementation are inadequate for its results to be verified as claimed. For instance, it is stated in [24] that a 64-bit factorization was demonstrated using their emulator with only 40 Kbits of classical memory instead of 320 qubits as required with Shor's algorithm in a quantum computer. This statement was not supported by design and implementation details on how factorization of such a large integer can be done with only 40 Kbits memory, where typically it would require at least 2^{320} bytes to represent a quantum state of such a scale on the classical platform.

In [25], FPGA emulation of 3-qubit QFT and Grover's search are proposed. In this work, which is based on the quantum circuit model, qubit expansion is performed prior to the application of multiqubit quantum gate transformations. This leads to an inaccurate modelling of a quantum algorithm, since, according to [31], the input quantum state to QFT circuit should first be placed in superposition of basis states, where signal samples are encoded as sequence of amplitudes. In the work by [26], hardware emulation of QFT restricts its input quantum state to the computational basis state, implying that superposition is not included in the modelling. Rivera-Miranda et al. in [26] claims 16-qubit QFT emulation is achieved. However, the emulator can only process up to 32

input signal samples in one evaluation, which is equivalent to a 5-qubit QFT emulation if effects of superposition and entanglement are included.

From the above discussion it should be noted then that the critical quantum properties of superposition and entanglement were not considered in these previous works, resulting in inaccurate modelling of quantum algorithms. Without the superposition and entanglement effects, the power of quantum parallelism cannot fully be exploited. Previous works reported in [25–27] applied pipeline architecture in their FPGA emulation implementations so as to obtain high throughput and low critical path delay. However, a pipeline design imposes high resource utilization (due to the requirement of additional pipeline registers and associated logic), thus limiting FPGA emulation to be deployed in more practical quantum computing applications that typically require high qubit sizes. In these pipeline implementations proposed in prior works, resource growth was exponential to the increase in qubit sizes.

In this paper, the issues outline above is addressed. The efficiencies of different hardware architectural designs for FPGA emulation purposes are evaluated based on the chosen case studies of QFT and Grover's search. We propose an accurate modelling of quantum system for FPGA emulation, targeting efficient resource utilization while maintaining significant speed-up over the equivalent simulation approach. Since our proposed FPGA emulation framework applies the state vector approach, simulation models based on the *libquantum* library are selected in this work for benchmarking purposes.

3. Theoretical Background

In general, quantum algorithms obey the basic process flow structure. The computation process begins with a system set in a specific quantum state, which is then converted into superposition of multiple basis states. Unitary transformations are performed on the quantum state according to the required operations of the algorithm. Finally, measurement is carried out, resulting in the qubits collapsing into classical bits.

3.1. Quantum Bit (Qubit). In classical computing, the smallest unit of information is the *bit*. A bit can be in either state 0 or state 1, and the state of a *bit* can be represented in matrix form as

$$\begin{aligned} \text{state } 0 &= \begin{matrix} \mathbf{0} \\ \mathbf{1} \end{matrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ \text{state } 1 &= \begin{matrix} \mathbf{0} \\ \mathbf{1} \end{matrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \end{aligned} \tag{1}$$

On the other hand, in quantum computing, the smallest unit of information is the *quantum bit* or a *qubit*. To distinguish the classical bit with the quantum qubit, Dirac *ket* notation is used. Using the *ket* notation, the quantum computational basis state is represented by $|0\rangle$ and $|1\rangle$. A qubit can be in state $|0\rangle$, or in state $|1\rangle$, or in superposition of both basis states. The state of a qubit can be represented as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \equiv \begin{matrix} \mathbf{0} \\ \mathbf{1} \end{matrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{2}$$

where both α and β are complex numbers and $|\alpha|^2 + |\beta|^2 = 1$. $|\alpha|^2$ is the probability where the qubit is in state $|0\rangle$ and $|\beta|^2$ is the probability where the qubit is in state $|1\rangle$ upon measurement. An n-qubit quantum state vector contains 2^n complex numbers which represents the measurement probability of each basis state. However, on measurement, the superposition is destroyed and the qubits return to the classical state of bits depending on the probability derived from the complex-valued state vector.

3.2. Tensor/Kronecker Product. Tensor product or Kronecker product is the basic operation that is applied in the formation of a larger quantum system as well as multiqubit quantum transformations. A quantum state vector that can be written as the tensor of two vectors is separable, whereas a state vector that cannot be expressed as the tensor of two vectors is entangled [15]. The tensor operation on any arbitrary two 1-qubit transformations is shown below:

$$\begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 & b_1 \\ b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\ a_0b_2 & a_0b_3 & a_1b_2 & a_1b_3 \\ a_2b_0 & a_2b_1 & a_3b_0 & a_3b_1 \\ a_2b_2 & a_2b_3 & a_3b_2 & a_3b_3 \end{bmatrix}. \tag{3}$$

3.3. Quantum Circuit Model. A quantum algorithm is a description of a sequence of quantum operations (or transformations) applied upon qubits to generate new quantum states. The model most widely used in describing the evolution of a quantum system is the quantum circuit model, first proposed in [32]. A quantum circuit is the interconnection of quantum gates with quantum wires, and gate operations are represented by unitary matrices.

All unitary matrices are invertible and the products of unitary matrices as well as the inverse of unitary matrix are unitary. An N-by-N matrix U is unitary if $UU^\dagger = U^\dagger U = I_N$, where $U^\dagger$ is the adjoint (conjugate transpose) of U. Since all quantum transformations are reversible, quantum gate operations can always be undone. Fundamental quantum gates include the Hadamard gate, phase-shift gate, and swap gate, and these gates are described as follows.

Hadamard gate H is one of the most useful single qubit quantum gates. It operates by placing the computational basis state into superposition of basis states with equal probability. The Hadamard transform can be represented by the following unitary matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4}$$

The following example illustrates the application of Hadamard gates in mapping a 2-qubit basis state $|00\rangle$ to a superposition of basis states with equal probability:

$$|00\rangle \xrightarrow{H\otimes H} \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right),$$
$$\left(\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \otimes \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\right)\begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}. \quad (5)$$

Controlled phase-shift gate ${}^{C}R_k$ operates on 2 qubits, one of which is the control qubit and the other is the target qubit. If the control qubit is true, a phase-shift operation is performed on the target qubit; otherwise, there is no operation. The operation is represented by the following matrix:

$${}^{C}R_k = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & e^{2\pi i/2^k}\end{bmatrix}. \quad (6)$$

Quantum SWAP gate is used for swapping two qubits. It switches the amplitudes of a quantum state vector. The operation of a 2-qubit SWAP gate is represented by matrix in

$$\text{SWAP} = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}. \quad (7)$$

3.4. Quantum Fourier Transform (QFT). The Fourier transform is deployed in wide range of engineering and physics applications such as signal processing, image processing, and quantum mechanics. It is a reversible transformation that converts signals from time/spatial domain to frequency domain and vice versa. The Fourier transform is defined in (8) for continuous signals and in (9) for discrete signals:

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi ft}\, dt, \quad (8)$$
$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi(kn/N)}. \quad (9)$$

The quantum Fourier transform (QFT) is a transformation on qubits and is the quantum equivalent of the discrete Fourier transform. It should be noted that a quantum computer performs QFT with exponentially less number of operations than the classical Fourier transform. However, QFT does not reduce the execution time of the algorithm when classical data is used. This is due to the characteristic of the quantum computer that does not allow parallel readout of all quantum state amplitudes. In addition, there is no known method that can effectively instantiate the desired input state amplitudes to be Fourier-transformed [33].

In order to harness the power of quantum computing on Fourier transform, QFT has to be deployed within other practical applications. QFT is pivotal in quantum computing since it is part of many quantum algorithms. These algorithms include integer factorization and discrete logarithms algorithms [1], Simon's periodicity algorithm [9], and Hallgren's algorithms [10]. They offer significant speed-up over their classical counterparts. QFT has also found applications in many real-world problems such as image watermarking [34] and template matching [35].

To compute Fourier transform in quantum domain, discrete signal samples are encoded as the amplitude sequences of a quantum state vector which is in superposition of basis states [31]. An n-qubit QFT operation which transforms an arbitrary superposition of computational basis states is expressed in

$$|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{j=0}^{2^n-1} f(j\Delta t)\,|j\rangle \xrightarrow{\text{QFT}} \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^n-1} f(j\Delta t)\, e^{2\pi i(jk/2^n)}\,|k\rangle. \quad (10)$$

As the requirement for a valid quantum state, $|\psi\rangle$ must be normalized such that it fulfils (11). If the original signal inputs do not comply with this requirement, the amplitudes of the signal samples have to be divided by the normalization factor, $\sqrt{\sum_{l=0}^{2^n-1}|f(l\Delta t)|^2}$. In most cases, the input states formed by the normalized signal samples are entangled:

$$\sum_{j=0}^{2^n-1}\left|f(j\Delta t)\right|^2 = 1. \quad (11)$$

From (10), it can be observed that the term $j/2^n$ in QFT equation is a rational number in the range of $0 \le j/2^n < 1$. As qubit representation is typically used in computations, the j in base-10 integer is redefined in base-2 notation as individual bit such that the binary fraction form as expressed in (12) can be conveniently adopted:

$$\begin{aligned}(j)_{10} &\equiv (j_1 j_2 \cdots j_n)_2\\ &= \left(2^{n-1}j_1 + 2^{n-2}j_2 + \cdots + 2^0 j_n\right)_{10}\\ &= 2^n\left(2^{-1}j_1 + 2^{-2}j_2 + \cdots + 2^{-n}j_n\right)_{10}\\ &= 2^n\left(0\cdot j_1 j_2 \cdots j_n\right)_2.\end{aligned} \quad (12)$$

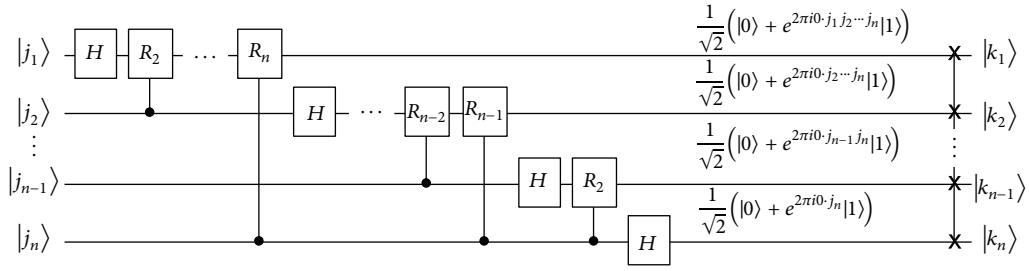


Figure 1: Quantum circuit model for n-qubit QFT.

With some algebraic manipulations, the QFT equation can be derived from (13) to form (14) [33]:

$$\mathrm{QFT}_{2^n}\left|j\right\rangle = \frac{1}{\sqrt{2^n}} \sum_{k=0}^{2^n-1} e^{2\pi i (jk/2^n)} \left|k\right\rangle \tag{13}$$

$$\begin{aligned} &= \frac{1}{\sqrt{2^n}} \left(|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle\right) \\ &\quad \cdot \left(|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \cdots j_n} |1\rangle\right). \end{aligned} \tag{14}$$

Since the term $e^{2\pi i 0 \cdot j_1}$ produces either -1 if $j_1 = 1$ or $+1$ otherwise, Hadamard computation on the first qubit results in $(1/\sqrt{2})(|0\rangle + e^{2\pi i 0 \cdot j_1}|1\rangle)$. Computations of the consecutive bits in the binary fraction are obtained using controlled phase-shift gates according to (14). QFT circuit consists of three types of elementary gates which are Hadamard gate, H, controlled phase-shift gate, ${}^{C}R_k$, and SWAP gate. The circuit model of an n-qubit QFT is depicted in Figure 1.

The size of a QFT circuit grows exponentially as the number of input qubits increases. An n-qubit QFT involves $\sum_{k=1}^{n} k+1$ unitary transformations and could process up to 2^n input samples in one evaluation (provided the input samples are encoded as the amplitude sequences of a superposition of computational basis states).

3.5. Grover's Search Algorithm. In computer science area, a typical search problem is to identify the desired element from an unordered array. For many computing applications, it is critical that the search technique is efficient. In terms of a function, the search problem $f : \{0,1\}^n \rightarrow \{0,1\}$ is given with assurance that there exists one binary string x_0 where $f(x) = 1$ if $x = x_0$; else, $f(x) = 0$.

In classical computing, $m/2$ queries on average are required to search for a particular element in an unordered array with m elements. In quantum computing, Grover's search algorithm can complete the job in $(\Pi/4)\sqrt{m}$ queries (for the rest of the text and figures in Section 3.5, the required Grover iterations are abbreviated as $\sqrt{2^n}$ times). Although the speed-up achieved is only quadratic, Grover's algorithm and its extensions are extremely useful in enhancing current methods in solving database searching and optimization problems, which include 3-satisfiability [36], global optimization [37], minimum point searching [38], and pattern matching [39]. The core operations of Grover's algorithm are phase inversion and inversion about mean. Phase inversion inverts the phase of the state-of-interest, and its quantum circuit model is given in Figure 2.

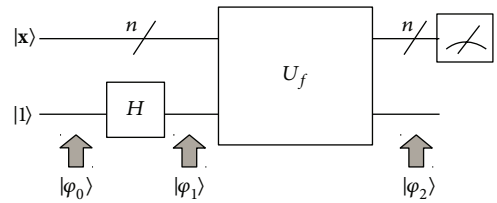


Figure 2: Quantum circuit for phase inversion [15].

In Figure 2, the top n-qubit, $|\mathbf{x}\rangle$, is the target qubit, and the bottom qubit is called the ancilla qubit. The function of U_f is to pick out the desired binary string. To apply phase inversion on target qubits, Hadamard gate operation is performed on the ancilla qubit, which is initialized as $|1\rangle$. This is to complement the effect of U_f which takes $|\mathbf{x}, y\rangle$ to $|\mathbf{x}, f(\mathbf{x}) \oplus y\rangle$.

In terms of matrices, the phase inversion operation can be expressed as $U_f(I_n \oplus H)|\mathbf{x}, 1\rangle$, and the corresponding quantum states are described as follows:

$$\begin{aligned} \left|\varphi_0\right\rangle &= \left|\mathbf{x}, 1\right\rangle, \\ \left|\varphi_1\right\rangle &= \left|\mathbf{x}\right\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] = \left[\frac{|\mathbf{x}, 0\rangle - |\mathbf{x}, 1\rangle}{\sqrt{2}}\right], \\ \left|\varphi_2\right\rangle &= \left|\mathbf{x}\right\rangle \left[\frac{\left|f(\mathbf{x}) \oplus 0\right\rangle - \left|f(\mathbf{x}) \oplus 1\right\rangle}{\sqrt{2}}\right] \\ &= \left|\mathbf{x}\right\rangle \left[\frac{\left|f(\mathbf{x})\right\rangle - \left|\overline{f(\mathbf{x})}\right\rangle}{\sqrt{2}}\right] \\ &= (-1)^{f(\mathbf{x})} \left|\mathbf{x}\right\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \\ &= \begin{cases} -1 \left|\mathbf{x}\right\rangle \left[\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} = \mathbf{x_0}, \\ +1 \left|\mathbf{x}\right\rangle \left[\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right], & \text{if } \mathbf{x} \neq \mathbf{x_0}. \end{cases} \end{aligned} \tag{15}$$

Inversion about mean boosts the phase separation between the element-of-interest and other elements in the unordered arrays (after phase inversion operation is applied to invert the phase of the target element). The mean of

(1) Start with $|\mathbf{0}\rangle$ as the target input qubits
(2) Apply n-qubit Hadamard gate on target qubits, $H^{\otimes n}$
(3) **for** $\sqrt{2^n}$ times **do**
(4) Apply phase inversion operation, $U_f(I \otimes H)$
(5) Apply inversion about mean operation on the target qubits, $-I + 2A$
(6) **end for**
(7) Measure the target qubits

Algorithm 1: Grover's search algorithm.

all elements is computed and inversions are made about the mean. The overall mean remains unchanged after the inversion process. This is because the distance between one element and the mean is the same before and after inversion. The only change is if the original sequence is above the mean, during the inversion it is flipped about the mean to the same distance below the mean and vice versa. In general, the inversion about mean operation can be expressed as

$$v' = -v + 2a, \tag{16}$$

where a is the mean, v is the value of an element in the array, and v' is the new value of that element after inversion.

In terms of matrices, the mean of a 2^n elements vector V is obtained by the product of matrices A and V where all the elements in the 2^n-by-2^n matrix A are set to $1/2^n$. Hence, inversion about mean in matrix form becomes

$$V' = -V + 2AV = (-I + 2A)\,V. \tag{17}$$

In order to achieve high confidence of getting the desired element, the amplitude amplification process (amplitude amplification in Grover's search algorithm involves phase inversion and inversion about mean operations) has to be repeated for $\sqrt{2^n}$ times. This is because the probability of success changes sinusoidally by the number of amplitude amplification iterations (as illustrated in Figure 3) and the highest probability of success first happened after the required iterations. Pseudocode for a generic Grover's search algorithm is given in Algorithm 1.

There are two approaches to model Grover's search quantum algorithm. The first approach, which is based on quantum circuit model, is discussed next. The second method is modelling using arithmetic functions, and this is presented in Section 4.2 since this approach is applied in this paper.

As shown in Figure 4, Grover's search circuit given in [15] is constructed with assumption that black box modules U_f and $-I + 2A$ are available. Descriptions of the U_f and $-I + 2A$ modules have been given earlier.

On the other hand, the circuit model for n-qubit Grover's algorithm presented in [33] is shown in Figure 5(a). In this figure, H is the Hadamard gate, and G is the Grover iteration circuit, which is illustrated in Figure 5(b).

The function of Grover iteration circuit is equivalent to the phase inversion and inversion about mean. In order to achieve high probability of successful search, G is concatenated for $\sqrt{2^n}$ times. In Figure 5(b), the role of oracle module is to recognize the solution to a particular search

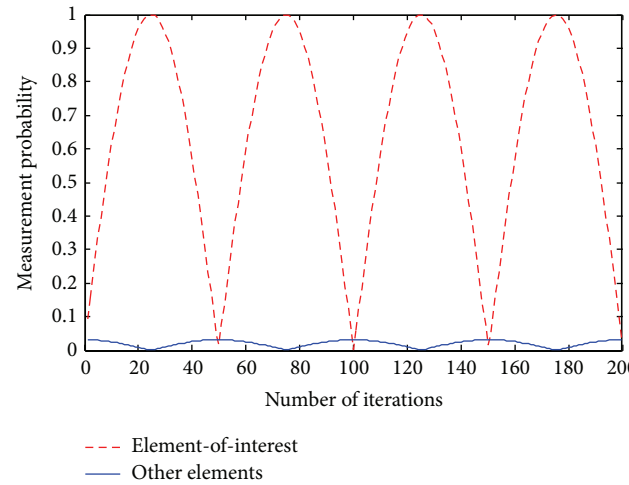


Figure 3: Probability of success by the number of amplitude amplification iterations amongst 2^{10} probabilities. For 10-qubit search, first highest probability of success happens at 25th iteration.

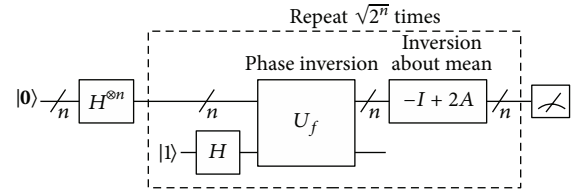
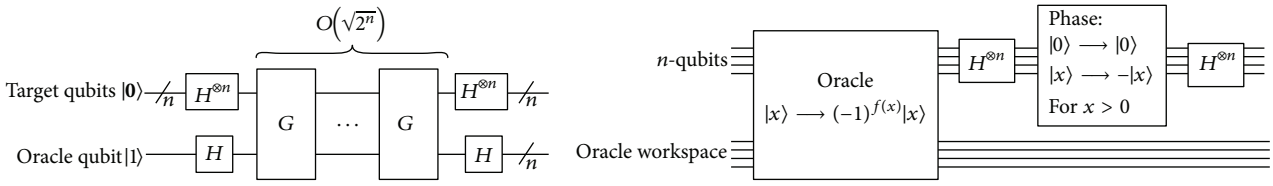


Figure 4: Modelling of Grover's search based on quantum circuit model [15].

problem in the phase inversion operation. By monitoring the oracle qubit, a solution to the search problem can be detected through the changes of the oracle qubit. The design of oracle module varies with different search applications, and an example of the oracle circuit for a simple 3-bit search task is shown in Figure 6. In Figure 6, the X symbol represents Pauli-X matrix. The open circle notation indicates conditioning on the qubit being set to zero, whereas the closed circle indicates conditioning on the qubit being set to one.

Deriving from the circuits in Figure 5, the corresponding 3-qubit Grover's search circuit is provided in Figure 7. This circuit model is made up of Hadamard, oracle, quantum NOT, and multiqubit controlled-NOT gates.



(a) n-qubit Grover's search

(b) Grover iteration circuit, G

Figure 5: Quantum circuit model for Grover's search algorithm [33].

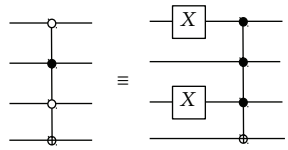


Figure 6: Oracle circuit for recognizing binary string "010".

4. Proposed FPGA-Based Hardware Emulation

This section presents our approach in modelling quantum Fourier transform and Grover's search algorithms for FPGA emulation. The proposed techniques can be generalized to FPGA emulation of more complex quantum algorithms that apply QFT or Grover's algorithm. This paper extends our earlier work presented in [40, 41]. In these previous works, the hardware architecture proposed was restricted to the serial design with resource sharing facilitated at the register level. In this paper, we enable resource sharing at the operator (or computational) level that allows for more efficient emulation of the quantum algorithms. Furthermore, additional case study on Grover's search algorithm is included for generalization of the proposed framework. The choice of hardware architecture varies based on the need of different applications. Based on the selected case studies, the efficiencies of different hardware architectures for quantum computing emulation purposes are discussed and analysed in this work. The goal is to achieve scalability and also efficient resource utilization for emulating practical larger qubit size quantum systems.

4.1. Modelling QFT for FPGA Emulation. The derivation of quantum circuit model for n-qubit QFT was discussed earlier is Section 3.4. Here, we present the modelling of QFT for FPGA emulation based on a 3-qubit example. According to (14), the mathematical expression for 3-qubit QFT is derived as shown in

$$\begin{aligned}\mathrm{QFT}_{2^3}\left|j\right\rangle = {} & \frac{1}{\sqrt{2^3}}\left(|0\rangle + e^{2\pi i 0\cdot j_3}\,|1\rangle\right) \\ & \cdot\left(|0\rangle + e^{2\pi i 0\cdot j_2 j_3}\,|1\rangle\right) \\ & \cdot\left(|0\rangle + e^{2\pi i 0\cdot j_1 j_2 j_3}\,|1\rangle\right).\end{aligned} \tag{18}$$

Deriving from the general n-qubit QFT circuit provided in Figure 1, the corresponding quantum circuit for 3-qubit QFT is obtained as shown in Figure 8.

The circuit consists of Hadamard gates, H, controlled phase-shift gates, ${}^{C}R_2$ and ${}^{C}R_3$, and also the SWAP gate. Referring to the functional block diagram given in Figure 9, this quantum circuit model corresponds to a sequence of unitary transformations, Ui, i = 1 to 7, defined by

$$U1 = H \otimes I \otimes I, \tag{19}$$

$$U2 = {}^{C}R_2 \otimes I, \tag{20}$$

$$U3 = (I \otimes \mathrm{SWAP}) \cdot \left({}^{C}R_3 \otimes I\right) \cdot (I \otimes \mathrm{SWAP}), \tag{21}$$

$$U4 = I \otimes H \otimes I, \tag{22}$$

$$U5 = I \otimes {}^{C}R_2, \tag{23}$$

$$U6 = I \otimes I \otimes H, \tag{24}$$

$$U7 = \mathrm{SWAP}_{2^3}. \tag{25}$$

Note that, in Figure 9, the modelling of n-qubit quantum system with superposition and entanglement properties resulted in a circuit with 2^n signals. Since the input samples to QFT circuit are encoded as sequence of amplitudes in an entangled superposition of basis states (discussed in Section 3.4), modelling based on individual qubit with separate quantum gate operations is unable to reflect the effects of applying a quantum gate on entangled qubits correctly.

In order to model the effect of superposition and entanglement, derivation of each unitary transformation is made through the tensor product of individual quantum gate and identity matrix to form unitary matrix of equal dimension with the quantum state vector. Detailed derivations of the quantum unitary matrices for the 3-qubit QFT have been presented in our previous paper [41].

Since these quantum unitary matrices are mostly sparse matrices, we extract minimal number of useful arithmetic operations (due to nonzero elements in the matrices), resulting in an optimal realization of the model that can be mapped to an efficient FPGA emulation architecture. Incidentally, a software program has been developed to automate this mapping, hence easily scaling up the circuit model to larger qubit sizes. From these arithmetic functions, the corresponding data-flow graph for the 3-qubit QFT is derived as shown in Figure 10.

In Figure 10, each IN and OUT signal is a fixed point complex number register for an element of the quantum state vector. The operation of module F corresponds to out_r = −in_i and out_i = in_r, where F is the multiplication

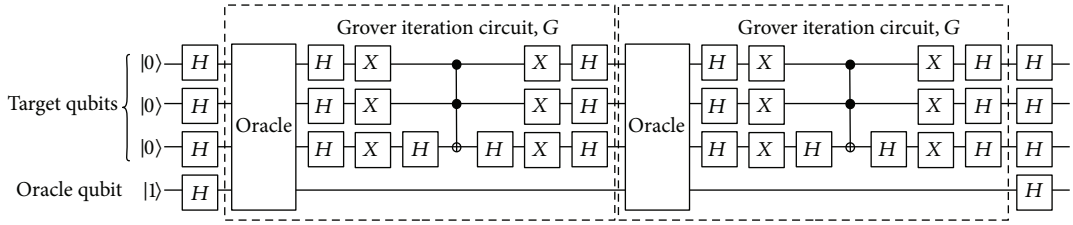


Figure 7: Modelling of 3-qubit Grover's search based on quantum gates.

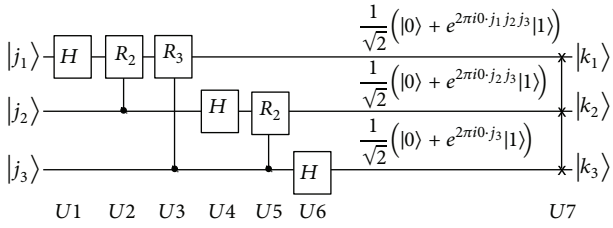


Figure 8: Quantum circuit for 3-qubit QFT.

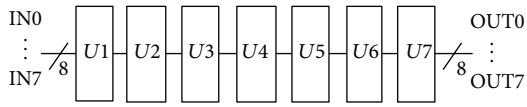


Figure 9: The 3-qubit QFT in terms of a sequence of unitary transformations.

of the input complex number with imaginary number i. This is applied in unitary transformations $U2$ and $U5$. The operation of module CMULT in Figure 10 is described in (26). The function of CMULT is the multiplication of the input complex number with a constant complex number which is derived based on the controlled phase-shift gate. As expressed previously in (21), it is used in unitary transformation $U3$:

$$\begin{aligned} \text{out\_}r &= (\text{in\_}r \times \text{constant\_}r) - (\text{in\_}i \times \text{constant\_}i)\,, \\ \text{out\_}i &= (\text{in\_}i \times \text{constant\_}r) + (\text{in\_}r \times \text{constant\_}i)\,. \end{aligned} \tag{26}$$

4.2. Modelling Grover's Search for FPGA Emulation. As mentioned earlier, the second approach of modelling Grover's quantum search is modelling using arithmetic functions (based on mathematical model). In this work, we have chosen this approach of modelling Grover's algorithm for FPGA emulation. In contrast to the quantum circuit model approach that involves complex large dimensional matrix operations (i.e., matrix multiplication and tensor product), the chosen method can utilize the computational resources available on FPGA such as comparators, adders/subtracters, and multiplexers for efficient emulation of Grover's search algorithm.

This technique is based on phase inversion and inversion about mean as described in Section 3.5. As shown in Figure 11, mathematically, Grover's search for a database with 2^n elements mainly involves the processes of inverting the phase of target element, function F, and performing inversion about mean on all elements, function $-I + 2A$, for $\sqrt{2^n}$ times. Initialization of the quantum state vector with equal probability, function INIT, is carried out once in the beginning of the process.

For the case study of a 3-bit search problem, we derive the data-flow graphs and obtain the result of the required arithmetic functions as shown in Figure 12. For experimental purposes, the oracle module is developed by comparing all elements in database with targeted element using comparators, COMP modules. The arithmetic functions of the inversion about mean are derived through straightforward computations that involve summation, bit shift, and subtraction.

4.3. Architecture of Proposed FPGA Emulation Model. It is clear that if implemented on classical computing platforms, the resource utilization for a quantum system would grow exponentially. Hence, the choice of suitable architecture is critical for FPGA-based quantum computing emulation. Here, we discuss the efficiencies of different architectural choices in datapath: concurrent, pipeline, serial, and serial-parallel. The block diagram of various architectures is constructed based on the example of 3-qubit QFT.

4.3.1. Concurrent Processing. In concurrent processing of an algorithm, all computations are completed within a clock cycle. In the case of 3-qubit QFT, computation blocks between the input and output registers, through the functional blocks, $U1$ to $U7$ are performed in one clock cycle (refer to Figure 13(a)). However, such an architecture consumes enormous resources, such that the number of registers required to emulate an n-qubit QFT is 2^{n+1}. In addition, the critical path delay is very high which results in unrealistic low operating frequency.

4.3.2. Pipeline Architecture. Most of the prior works on FPGA-based quantum circuit emulation [25–27] are developed based on pipeline architecture. The pipeline architecture has the advantages of high throughput and much shorter critical path delay. Figure 13(b) shows the proposed pipeline architecture of the 3-qubit QFT. However, the main issue of this approach is that resource utilization grows drastically by the number of qubits, due to the circuit augmentation of pipeline registers. $2^n(\sum_{k=1}^{n} k{+}2)$ pipeline registers are required to emulate n-qubit QFT. Consequently, hardware emulation scalability is highly constrained by the available resources in FPGA.

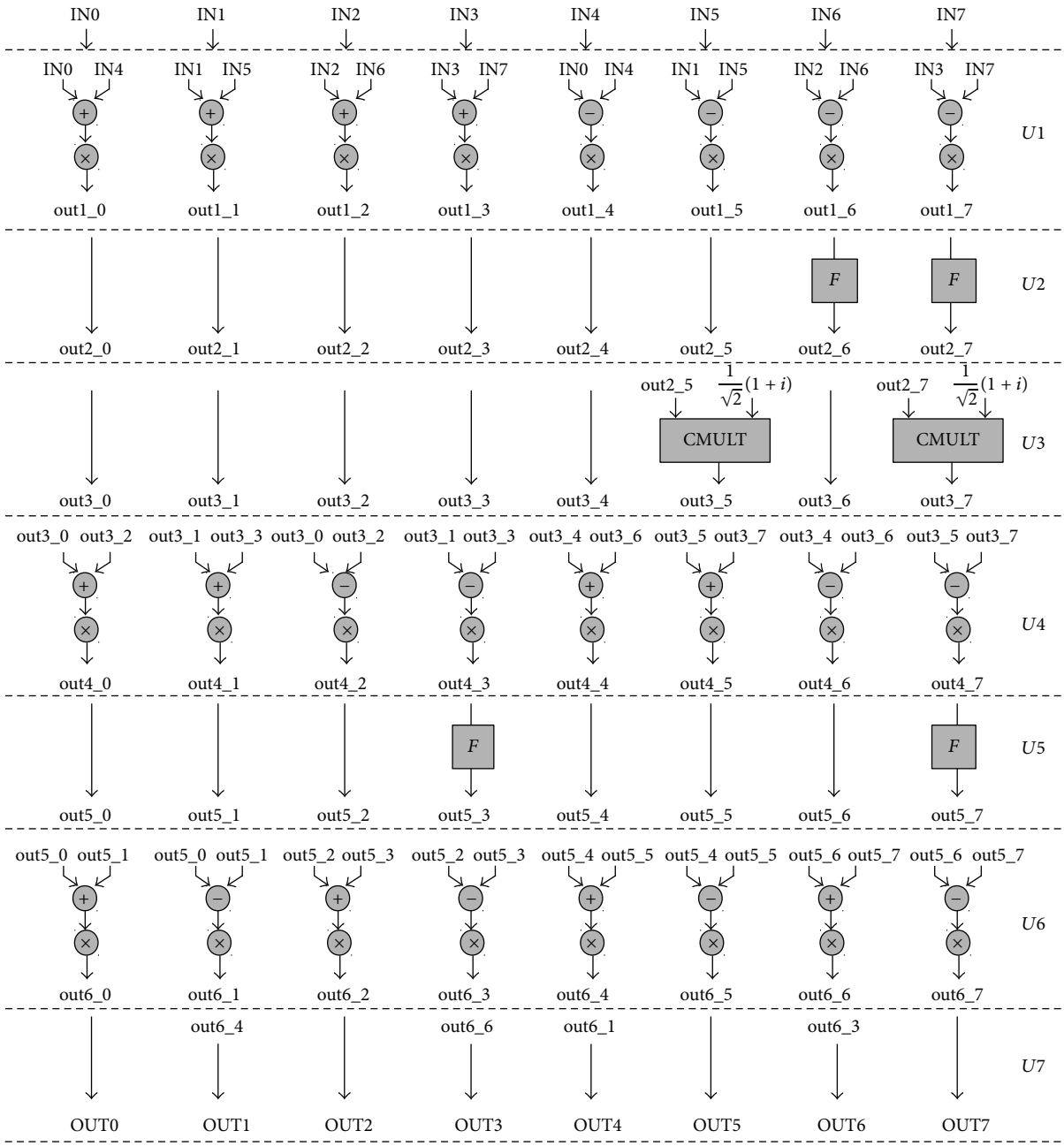


Figure 10: Data-flow graph for 3-qubit QFT. The multipliers shown in this diagram represent the multiplication of input complex number with constant $1/\sqrt{2}$.

4.3.3. Serial Processing. Although serial design requires multiple iterations to perform a complex computation, it opens up the opportunity for resource sharing. Serial processing is suitable for applications where resource utilization is a critical design constraint. Figure 13(c) depicts the serial form of the 3-qubit QFT circuit that consists of a control unit and a datapath unit. As resources can be reused between transformations, a serial-based n-qubit QFT consumes 2^n registers, a register utilization that is much lower than in concurrent or pipeline architectures. However, pure serial approach forfeits the purpose of conducting FPGA emulation whose aim is to exploit the parallelism inherent in a quantum system, as it would still suffer from slow sequential behaviour as exhibited in simulation on classical computer.

4.3.4. Proposed Serial-Parallel Architecture. In this paper, we propose a hybrid serial-parallel architecture for FPGA emulation of quantum algorithms. The proposed approach takes advantage of both serial and parallel design techniques.

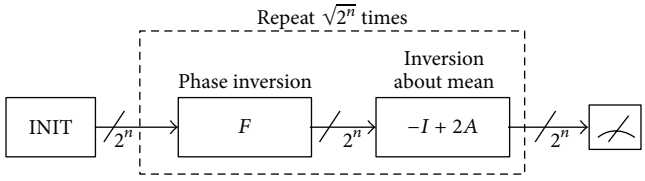


Figure 11: Mathematical modelling of Grover's search algorithm.

Applying the concepts of quantum parallelism and quantum dynamics modelled by sequential transformations on a quantum state vector, it is found that the proposed serial-parallel architecture is suitable for efficient and accurate quantum computing emulation on FPGA platform.

Figure 14 shows the functional block diagram of the proposed serial-parallel FPGA emulation architecture of the 3-qubit QFT. The serial-parallel design of the datapath unit involves a number of quantum computation units that can perform parallel computations for each stage of unitary transformation whereby the same computational resources can be reused for the following stage of transformations.

For data storage and synchronization purposes, 2^n registers are shared between unitary transformations. As compared to the pipeline design, our proposed serial-parallel approach achieves linear reduction on the usage of registers to emulate the same quantum system. The arithmetic logic unit (ALU) in the datapath unit contains multiple custom processing elements and the allocation of resources in ALU varies based on the target application. The number of processing elements is basically determined based on the desired 2^n parallelism in the n-qubit quantum system.

As illustrated in Figures 10 and 12, the data-flow graph of QFT and Grover's search algorithm exhibit similar repetitive pattern between unitary transformations. This implies that the proposed serial-parallel approach can be generalized for the two case studies to achieve balance in both resource utilization and speed performance. For the case of Grover's search, the ALU would be the Grover iteration module shown in Figure 12, whereas the control unit is designed to keep track on the number of Grover iterations required by the target search problem.

5. Experimental Results

The proposed emulation designs are modelled in SystemVerilog HDL, synthesized using Altera Quartus II software, and implemented into target emulation platform which is based on Altera Stratix IV EP4SGX530KF43C4 FPGA. In Section 5.1, we discuss the verification of the hardware emulation designs for the QFT and Grover's search case studies. In addition, the automated process for scaling up the design to larger qubit size is described. In Section 5.2, investigation is conducted to study the effects of the number of mantissa bits used in our fixed point representation format on resource utilization and precision error. In the section that follows, we analyse, for different emulation architectures, how the increase in qubit size impacts on resource growth and maximum operating frequency allowed in the designs. Finally, the runtime speeds in simulation and emulation for quantum algorithms with different qubit sizes are compared.

5.1. Design Verification. To show that the quantum algorithms have been modelled accurately, design verification of the proposed emulation hardware is performed. Golden references based on the software simulation models (in C) are developed and their outputs are compared with the emulation hardware under test (which are described in SystemVerilog HDL).

In our QFT case study, FFTW3 [42], a widely applied fast Fourier transform library in C, is used to perform Fourier transform computations on the signal samples that are used in this work. The outputs of the classical Fourier transform then serve as the golden reference model in verification of the proposed emulation model. Furthermore, since the discrete Fourier transform (DFT) is a linear transformation that can be defined in unitary matrix form, the functional correctness of our QFT hardware emulation model can be conveniently verified against the DFT matrix. The expression of an n-qubit DFT matrix is shown in

$$\mathrm{DFT} = \frac{1}{\sqrt{2^n}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^1 & \omega^2 & \dots & \omega^{2^n-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(2^n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{2^n-1} & \omega^{2(2^n-1)} & \dots & \omega^{(2^n-1)(2^n-1)} \end{bmatrix}, \quad (27)$$

where ω is the 2^nth root of unity; that is, $\omega = e^{2\pi i/2^n}$. The choice of $e^{2\pi i/2^n}$ or $e^{-2\pi i/2^n}$ is purely a matter of convention as both the term $e^{2\pi i/2^n}$ and the term $e^{-2\pi i/2^n}$ to the power of 2^n are equal to 1.

On the other hand, the design of Grover's search FPGA emulation model is verified against the mathematical model provided in literature. The simulation model of Grover's search algorithm also serves as the golden reference model for verification purposes.

For the development of FPGA emulation models for practical quantum computing applications, it is important that the emulation hardware can be scaled up to larger qubit size architectures. In this work, the designs are scaled up with the aid of software program developed in-house. HDL codes of the two case studies are autogenerated by the software program based on the proposed modelling techniques (as discussed in Section 4). The generated HDL code produces efficient hardware emulation model based on the proposed serial-parallel architecture.

5.2. Fixed Point Representation. As defined in (2), a quantum state vector is represented by complex floating point numbers. To ensure effective resource utilization in our FPGA emulation hardware, floating point numbers are replaced by fixed point representations. In this work, a fixed point format with 1 sign bit, 1 integer bit, and N mantissa bits (as shown in Figure 15) is used. Since the amplitudes of a quantum state, that is, the probabilities of collapsing into computational basis

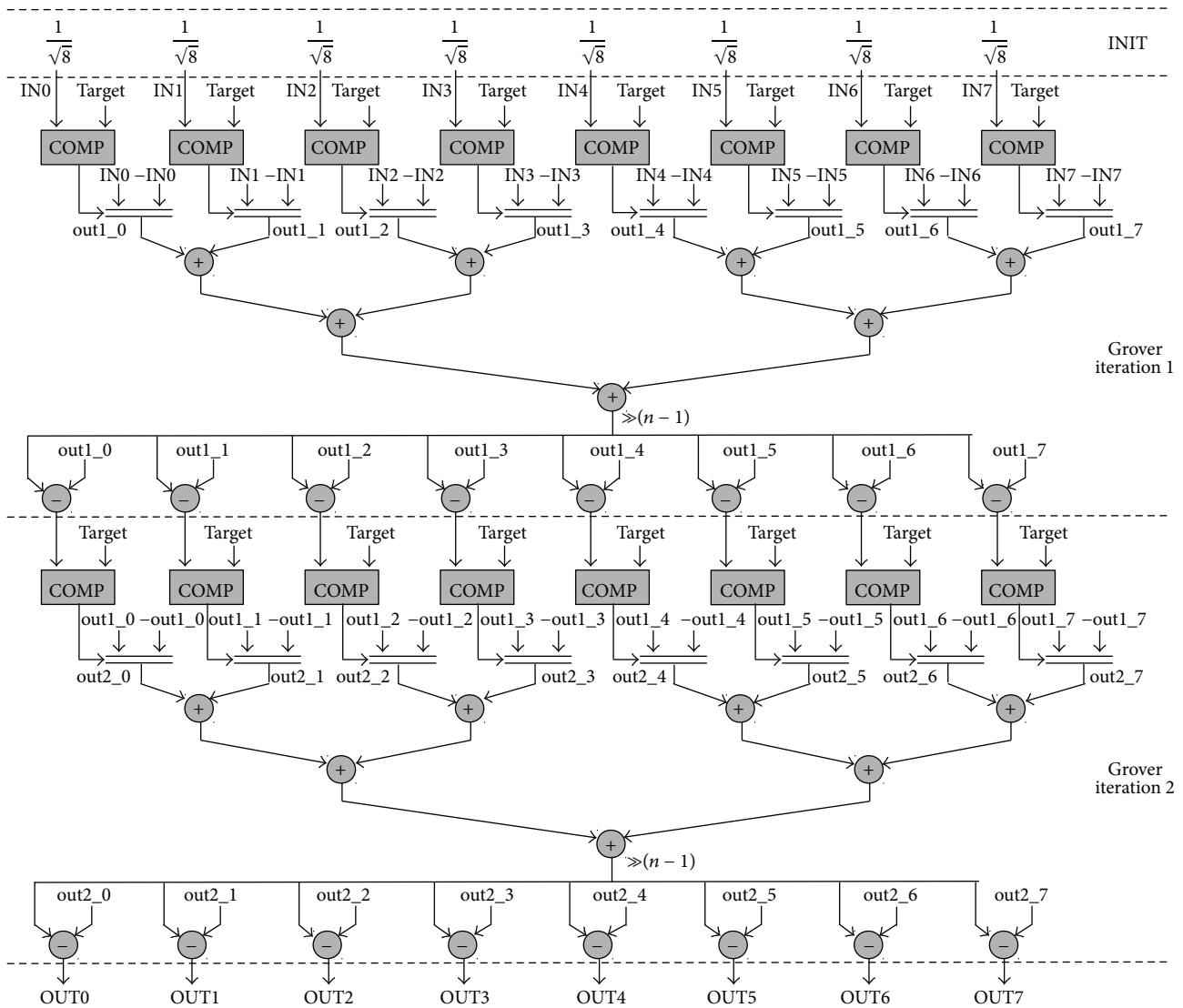
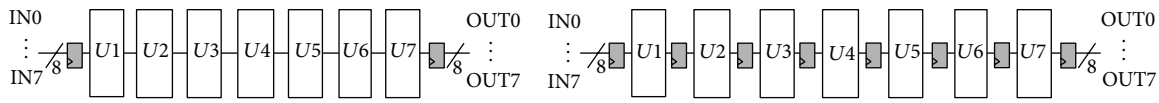
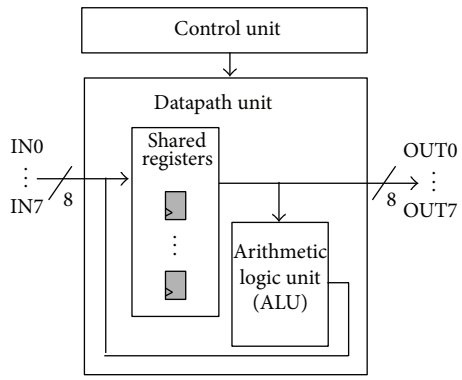


Figure 12: Data-flow graph of Grover's circuit for 3-bit search.



(a) Concurrent processing (b) Pipeline architecture



(c) Serial architecture

Figure 13: Three-qubit QFT implemented using different hardware architectures.

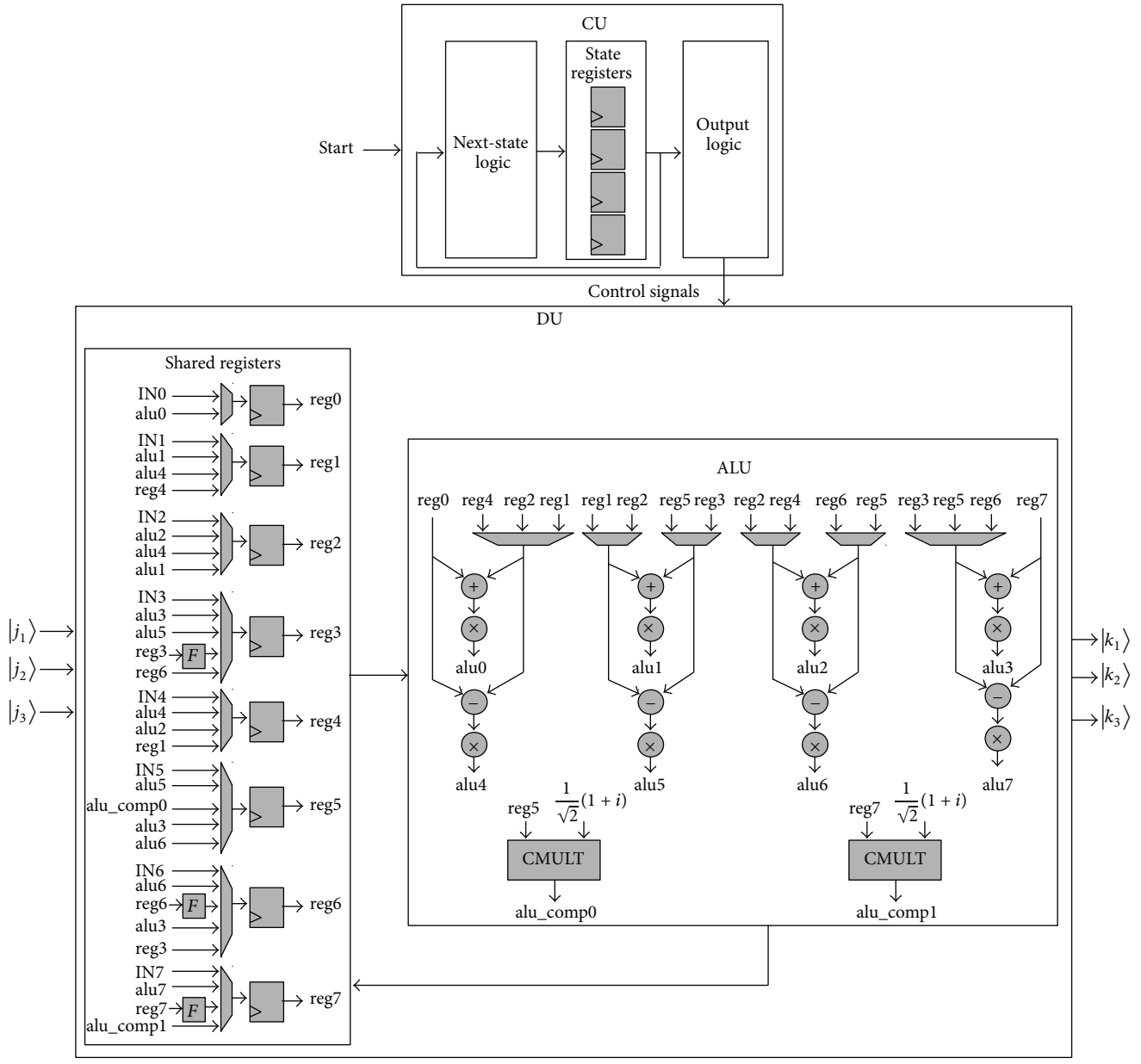


Figure 14: Proposed serial-parallel architecture for 3-qubit QFT. The multipliers shown in this diagram represent the multiplication of input complex number with constant $1/\sqrt{2}$.

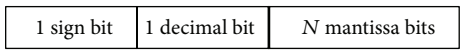


Figure 15: Fixed point representation format.

states after measurement, are constrained in the range of 0 to 1, only one bit is required to represent the integer part.

Due to the limitations of the classical digital computing platform, representation of qubit amplitudes with infinite precision is infeasible. In the context of quantum computer modelling, particularly for quantum systems with large qubit sizes, minimising precision loss is critical to preserve the consistency of the quantum state during the modelling process [43]. Here, we investigate how precision error is affected by the number of mantissa bits used in our fixed point representation format. The corresponding experimental results are given in Figure 16.

Precision error shown in Figure 16 is computed based on the following equations:

$$\begin{aligned}
\text{error_}r &= \sum_{m=0}^{2^n-1} \frac{\left|\text{emulate_}r_m - \text{simulate_}r_m\right|}{\left|\text{simulate_}r_m\right|}, \\
\text{error_}i &= \sum_{m=0}^{2^n-1} \frac{\left|\text{emulate_}i_m - \text{simulate_}i_m\right|}{\left|\text{simulate_}i_m\right|}, \\
\text{precision_error} &= \frac{1}{2^{n+1}} \left(\text{error_}r + \text{error_}i\right),
\end{aligned} \tag{28}$$

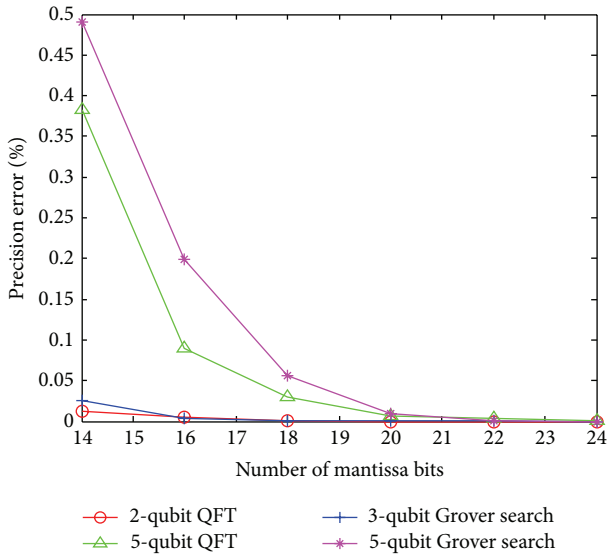


FIGURE 16: Precision error against different fixed point formats used.

where simulate_r and simulate_i are the floating point real and imaginary amplitudes of the reference output state generated from simulation, whereas emulate_r and emulate_i are the amplitudes extracted from the output state of proposed FPGA emulator (converted from original fixed point format to floating point for verification purposes).

Figure 16 shows that 16-bit fixed point format (14 mantissa bits) incurs significant precision error for 5-qubit emulations in both case studies. However, the error produced by 2-qubit emulations is insignificant. This behaviour is because the amplification of the fixed point truncation errors grows with the size of quantum system. For FPGA emulation purposes, precision error due to fixed point representation can be reduced by increasing the number of mantissa bits with trade-off on resource utilization. By expanding the number of mantissa bits up to 24-bit (which results in 26, total number of bits), negligible precision error for 5-qubit emulations is attained. It is important to note that the proposed FPGA emulator is parameterizable in terms of the number of mantissa bits for fixed point representation. This is crucial to ensure different fixed point formats can be applied to emulate quantum circuit of various sizes based on the demanded precision error tolerance and resource constraint.

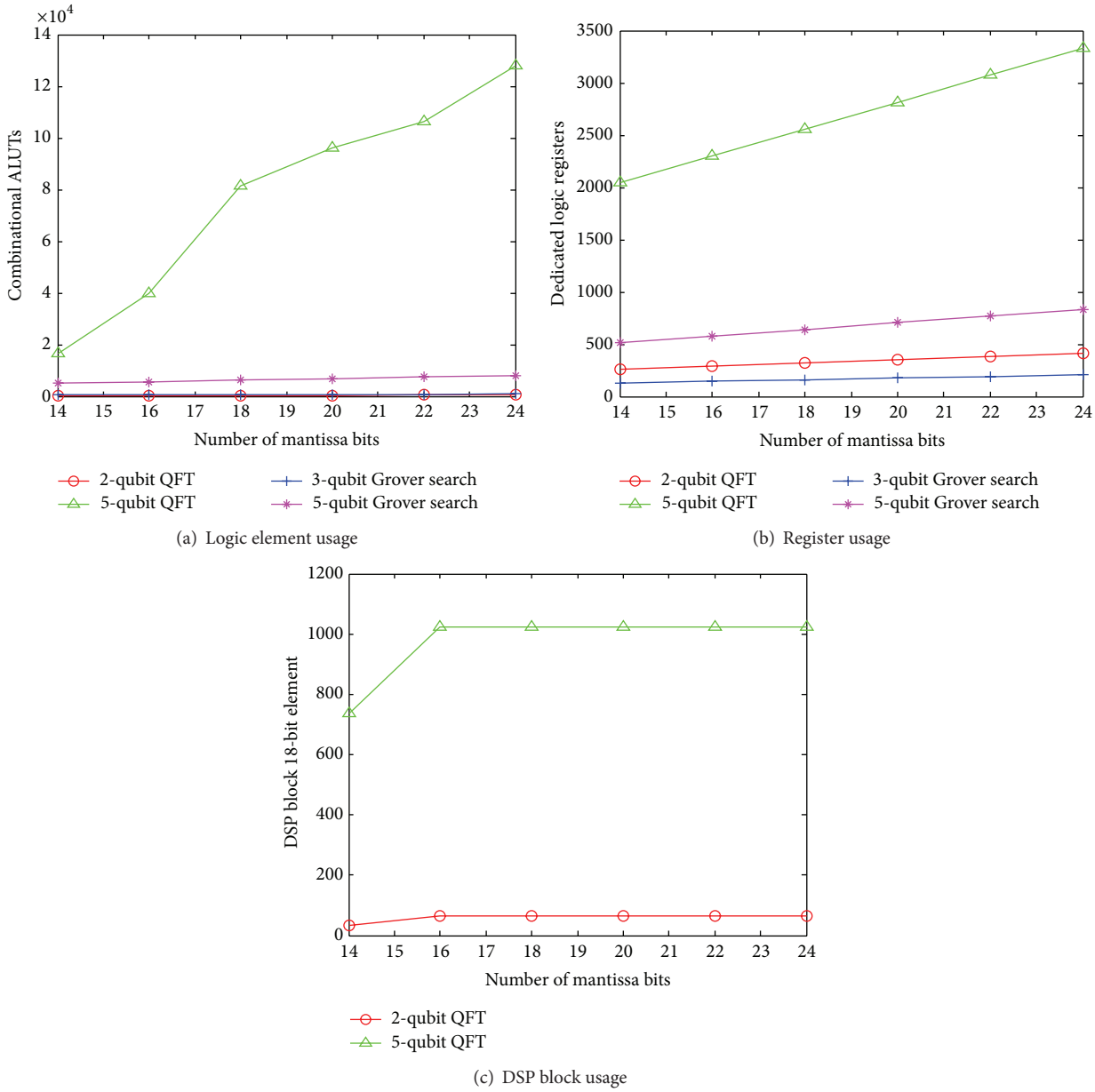
The size of our fixed point number format also affects resource utilization. The corresponding experimental results for QFT case study are shown in Figure 17. Since the resources available on FPGA device are mostly in blocks or multiples of 8 bits, the choice of 26-bit fixed point format is not suitable. Hence, we apply 22-mantissa-bit size (i.e., total number of bits is 24) in our fixed point representation formats. Note that, for Grover's search emulations, the experiment on resource utilization of DSP blocks is not relevant because FPGA emulation model developed here (as depicted in Figure 12) does not involve multiplication.

5.3. Efficiency of Proposed FPGA Emulation Architecture. In this subsection, we investigate how the increase in qubit size impacts resource growth and maximum allowable operating frequency in different emulation architectures (i.e., concurrent, pipeline, and serial-parallel). In the case of QFT emulation model, we have two versions of serial-parallel architectures. Type 1 serial-parallel uses DSP blocks to perform multiplications, whereas type 2 replaces the multiplications (required in Hadamard gate operations, $U1$, $U4$, and $U6$, in the 3-qubit QFT case) with shift-add operations. Although conventional hardware design methods encourage the usage of shift-add operation instead of multiplication to reduce resource utilization, the case is now different with FPGA devices containing efficient built-in DSP blocks. Figures 18 and 19 show the results of the experiments conducted on QFT and Grover's search algorithms, respectively.

Based on the experimental results shown in Figure 18, when comparing to type 1, type 2 method consumes less DSP blocks but more logic elements due to the construction of adders needed in shift-add operations. As the number of qubits increases, resource utilization of logic elements for type 1 emulation grows rapidly when available DSP blocks are used up and it ends up with similar resource utilization as the type 2 approach. Hence, we can conclude that, for large-scale FPGA emulations, both methods would lead to similar resource utilization. Thus, the first approach is preferred due to the ease of design process where DSP blocks are utilized by default when implementing the design with the FPGA design automation tool.

In contrast to the concurrent and pipeline designs, the experimental results for QFT and Grover's search show that the proposed serial-parallel architecture achieves balance on both resource utilization and operating frequency. The proposed architecture has significantly reduced resource growth in the application of logic elements, dedicated logic registers, and DSP block, yet maintaining reasonable operating frequency. With the concurrent and pipeline architectures, 5-qubit QFT emulation completely used up the resources available in the Altera Stratix IV FPGA device used in this work. However, the same device can support up to 7-qubit QFT emulation with the proposed serial-parallel architecture. It is important to note that the processing power of 7-qubit QFT is far higher than the 5-qubit implementation as an n-qubit QFT can process up to 2^n samples in one evaluation.

For scalability, software simulation would depend on resources available on the computer servers. The scalability of the FPGA emulation framework would depend on the resources available on target FPGA devices. As there have been rapid advances in FPGA technology in recent years, by designing an efficient architecture that is implemented in a high-density FPGA device (such as the Altera Stratix 10 that contains up to 5.5 million logic elements), one can actually emulate large qubit size quantum circuit on a single FPGA chip. Furthermore, new approach to FPGA emulation may be made through the exploration of efficient data structures and modelling methods. Thus, the proposed work contributes to the formulation of a proof-of-concept baseline FPGA emulation framework with optimization on datapath designs that can be extended to emulate practical large-scale quantum circuits.



(a) Logic element usage

(b) Register usage

(c) DSP block usage

Figure 17: Resource utilization for different fixed point formats used.

5.4. Benchmarking between Simulation and Emulation. Here, our emulation models are benchmarked against the equivalent software simulations. The simulation models used are based on an open source quantum library, *libquantum* [28]. Software simulation is performed on an Intel Core i7-4790 eight-core processor with 3.6 GHz clock rate running on a Linux-based Ubuntu 14.04 kernel, whereas hardware emulation is based on the Altera Stratix IV EP4SGX530KF43C4 FPGA. Table 1 shows the runtime comparison between simulation and our emulation.

Figure 20 illustrates the runtime speed-up (simulation/emulation) of Grover's search case study. It is important to note that the proposed hardware emulation is implemented based on 24-bit fixed point format (which is determined based on the experimental work presented in Section 5.2) whereas 32-bit single precision float is used in the software

Table 1: Comparison of runtime between simulation and emulation.

	QFT runtime (s)		Grover's search runtime (s)	
	Simulation	Emulation	Simulation	Emulation
2-qubit	15.5×10^{-6}	35.8×10^{-9}	29.6×10^{-6}	4.6×10^{-9}
3-qubit	46.6×10^{-6}	80.5×10^{-9}	74.0×10^{-6}	12.0×10^{-9}
4-qubit	51.0×10^{-6}	134.4×10^{-9}	220.7×10^{-6}	21.4×10^{-9}
5-qubit	56.5×10^{-6}	219.3×10^{-9}	398.8×10^{-6}	36.7×10^{-9}
6-qubit	—	—	1069.6×10^{-6}	62.7×10^{-9}
7-qubit	—	—	2771.4×10^{-6}	96.8×10^{-9}

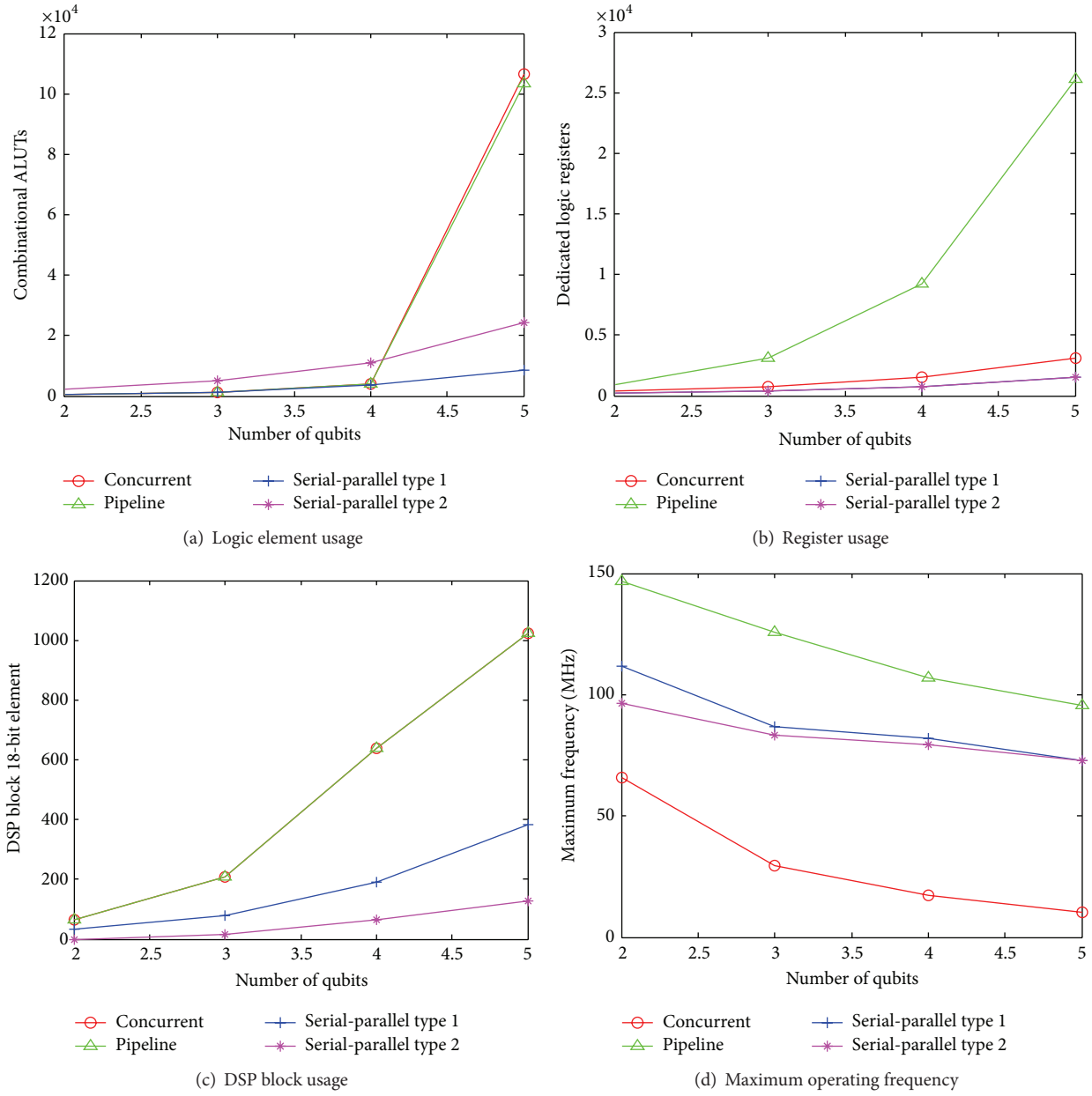


FIGURE 18: Resource utilization and maximum operating frequency for different emulation architectures of QFT (based on 24-bit fixed point format).

simulation to describe the quantum state. For a larger qubit size quantum circuit, the number of mantissa bits for fixed point representation has to be increased accordingly to ensure similar precision as in software simulation can be achieved with trade-offs on resource utilization and execution speed [44].

From Figure 20, it can be observed that the proposed hardware emulation provides significant speed-up over software simulation using *libquantum*. It is important to note that the achieved speed-up increases drastically as the number of qubits increases. This result further supports the notion that hardware emulation has significant potential in the modelling of a large-scale quantum system on the classical computing platform based on FPGA.

As the number of required I/O pins for emulating QFT and Grover's search algorithms with parallel read-outs is too much to fit in the existing FPGA devices, board-level verification is infeasible. Although the usage of multiplexers reduces the number of output pins, resource consumption rises significantly with the increase in the number of qubits, and this affects the analysis of the overall experiment. Thus, the estimated runtime of the proposed FPGA emulation architecture is obtained based on the hardware clock cycle and the operating frequency that is acquired from the FPGA development tool. This is not a critical issue in future practical deployment as the selected case studies are meant to work as core modules within other quantum applications that might not require parallel read-out.

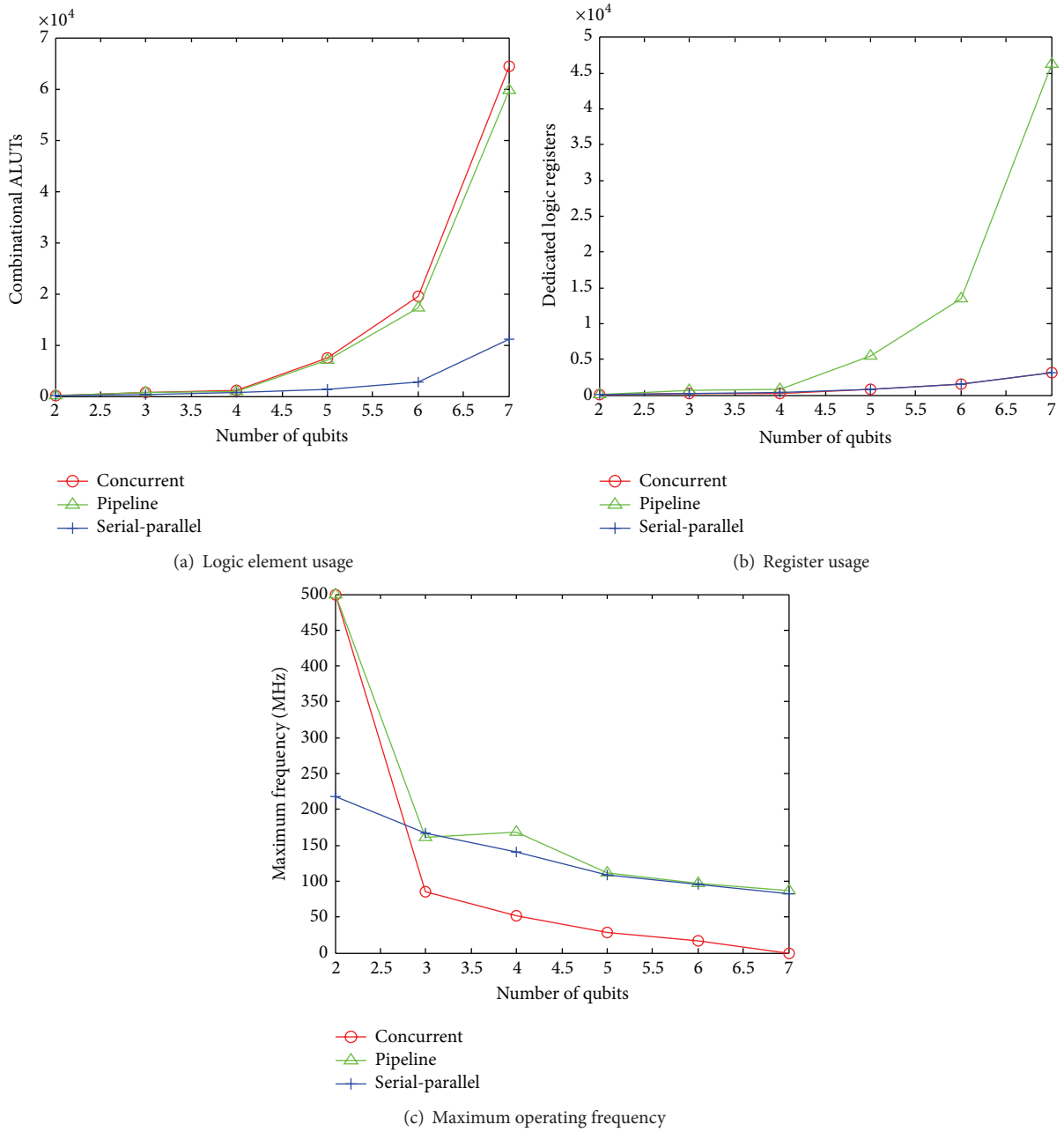


FIGURE 19: Resource utilization and maximum operating frequency for different emulation architectures of Grover's search (based on 24-bit fixed point format).

6. Conclusion and Future Work

Efficient resource utilization is critical for FPGA-based implementations especially for emulating quantum computing applications as they typically exhibit exponential resource requirement with increasing number of qubits. In this paper, we proposed a baseline FPGA emulation framework with focus on the datapath design optimization based on the conventional state vector model as well as an effective methodology that facilitates accurate modelling of quantum algorithms for FPGA emulation. A serial-parallel architecture with efficient resource utilization for FPGA-based emulation of quantum computing is presented. The proposed emulation architecture achieves linear reduction in resource utilization compared to pipeline implementations as found in previous works. This work has also demonstrated the advantage of FPGA emulation over software simulation where hardware emulation of 7-qubit Grover's search is about 3×10^4 times faster than the software simulation performed on Intel Core i7-4790 eight-core processor running at 3.6 GHz clock rate.

However, experimental results obtained in this work show that it is difficult to realize a scalable and flexible

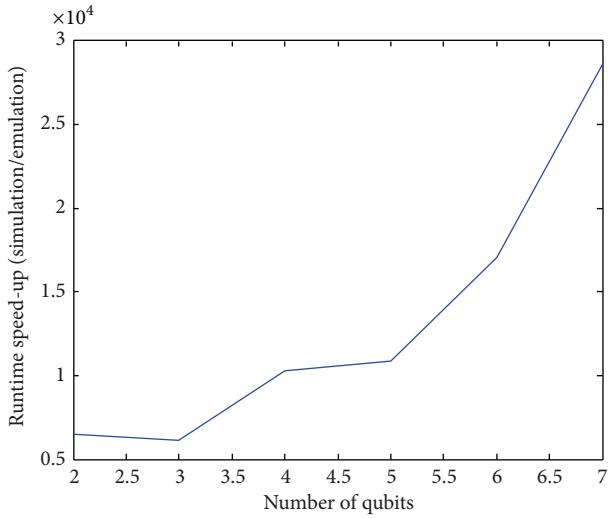


Figure 20: Runtime speed-up against number of qubits in Grover's search emulation.

emulation platform for large qubit size real-world quantum system using the approach that applies existing state vector quantum models. This concurs with [45] that states that the practical limit on the size of the quantum system that can be modelled on classical computing platform can hardly be overcome due to exponentially large memory requirements for storing the entire state vector. Hence, this suggests that a model with a more effective data structure to represent quantum systems is required. Recently, the work on stabilizer frames [30] has shown promise in providing a more suitable data structure for quantum models targeted for FPGA emulation. This is the subject of future work in our research in applying FPGA emulation in modelling of large-scale quantum systems. In addition, the error-correction structure available with stabilizer frames will be considered for application in practical quantum computations. With a more efficient modelling technique, FPGA can represent a more efficient emulation strategy of quantum systems.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work is supported by the Ministry of Higher Education (MOHE) and Universiti Teknologi Malaysia (UTM) under Fundamental Research Grant Scheme (FRGS) Vote number 4F422.

References

[1] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science (SFCS '94)*, pp. 124–134, Santa Fe, NM, USA, November 1994.

[2] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical Review Letters*, vol. 79, no. 2, pp. 325–328, 1997.

[3] A. Malossini and T. Calarco, "Quantum genetic optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 231–241, 2008.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[5] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 212–219, ACM, 1996.

[6] N. Shenvi, J. Kempe, and K. B. Whaley, "Quantum random-walk search algorithm," *Physical Review A: Atomic, Molecular, and Optical Physics*, vol. 67, no. 5, Article ID 052307, 2003.

[7] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum algorithm for the Hamiltonian NAND tree," *Theory of Computing*, vol. 4, pp. 169–190, 2008.

[8] P. W. Shor, "Why haven't more quantum algorithms been found?" *Journal of the ACM*, vol. 50, no. 1, pp. 87–90, 2003.

[9] D. R. Simon, "On the power of quantum computation," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1474–1483, 1997.

[10] S. Hallgren, "Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem," *Journal of the ACM*, vol. 54, article 4, 2007.

[11] M. Mosca and A. Ekert, "The hidden subgroup problem and eigenvalue estimation on a quantum computer," in *Quantum Computing and Quantum Communications*, pp. 174–188, Springer, Berlin, Germany, 1999.

[12] D. Bacon, A. M. Childs, and W. Van Dam, "From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups," in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS '05)*, pp. 469–478, IEEE, October 2005.

[13] L. K. Grover and A. M. Sengupta, "From coupled pendulums to quantum search," in *Mathematics of Quantum Computation*, pp. 119–134, Chapman and Hall/CRC, 2002.

[14] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6-7, pp. 467–488, 1982.

[15] N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*, vol. 20, Cambridge University Press, Cambridge, UK, 2008.

[16] C. Monroe, D. M. Meekhof, B. E. King, W. M. Itano, and D. J. Wineland, "Demonstration of a fundamental quantum logic gate," *Physical Review Letters*, vol. 75, no. 25, pp. 4714–4717, 1995.

[17] N. A. Gershenfeld and I. L. Chuang, "Bulk spin-resonance quantum computation," *Science*, vol. 275, no. 5298, pp. 350–356, 1997.

[18] J. E. Mooij, T. P. Orlando, L. Levitov, L. Tian, C. H. Van der Wal, and S. Lloyd, "Josephson persistent-current qubit," *Science*, vol. 285, no. 5430, pp. 1036–1039, 1999.

[19] I. L. Chuang, N. Gershenfeld, and M. Kubinec, "Experimental implementation of fast quantum searching," *Physical Review Letters*, vol. 80, no. 15, article 3408, 1998.

[20] R. Barends, J. Kelly, A. Megrant et al., "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, 2014.

[21] M. H. Amin, N. G. Dickson, and P. Smith, "Adiabatic quantum optimization with qudits," *Quantum Information Processing*, vol. 12, no. 4, pp. 1819–1829, 2013.

[22] A. D. King, E. Hoskinson, T. Lanting, E. Andriyash, and M. H. Amin, "Degeneracy, degree, and heavy tails in quantumannealing," http://arxiv.org/abs/1512.07325.

[23] T. F. Rønnow, Z. Wang, J. Job et al., "Defining and detecting quantum speedup," *Science*, vol. 345, no. 6195, pp. 420–424, 2014.

[24] Y. Goto and M. Fujishima, "Efficient quantum computing emulation system with unitary macro-operations," *Japanese Journal of Applied Physics*, vol. 46, no. 4, pp. 2278–2282, 2007.

[25] A. U. Khalid, Z. Zilic, and K. Radecka, "FPGA emulation of quantum circuits," in *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '04)*, pp. 310–315, IEEE, October 2004.

[26] J. F. Rivera-Miranda, Á. J. Caicedo-Beltrán, J. D. Valencia-Payán, J. M. Espinosa-Duran, and J. Velasco-Medina, "Hardware emulation of quantum Fourier transform," in *Proceedings of the IEEE 2nd Latin American Symposium on Circuits and Systems (LASCAS '11)*, pp. 1–4, IEEE, Bogata, Colombia, February 2011.

[27] M. Aminian, M. Saeedi, M. S. Zamani, and M. Sedighi, "FPGA-based circuit model emulation of quantum algorithms," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI '08)*, pp. 399–404, IEEE, Montpellier, France, April 2008.

[28] H. Weimer, M. Müller, I. Lesanovsky, P. Zoller, and H. P. Büchler, "A Rydberg quantum simulator," *Nature Physics*, vol. 6, no. 5, pp. 382–388, 2010.

[29] G. F. Viamontes, *Efficient quantum circuit simulation [Ph.D. thesis]*, The University of Michigan, Ann Arbor, Mich, USA, 2007.

[30] H. J. García and I. L. Markov, "Simulation of quantum circuits via stabilizer frames," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2323–2336, 2015.

[31] C. P. Williams and S. H. Clearwater, *Explorations in Quantum Computing*, Springer, Berlin, Germany, 1998.

[32] A. Barenco, D. Deutsch, A. Ekert, and R. Jozsa, "Conditional quantum dynamics and logic gates," *Physical Review Letters*, vol. 74, no. 20, pp. 4083–4086, 1995.

[33] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.

[34] W.-W. Zhang, F. Gao, B. Liu, Q.-Y. Wen, and H. Chen, "A watermark strategy for quantum images based on quantum Fourier transform," *Quantum Information Processing*, vol. 12, no. 2, pp. 793–803, 2013.

[35] D. Curtis and D. A. Meyer, "Towards quantum template matching," in *Proceedings of the SPIE 48th Annual Meeting in Quantum Communications and Quantum Imaging*, pp. 134–141, International Society for Optics and Photonics, August 2003.

[36] R. Schützhold and G. Schaller, "Adiabatic quantum algorithms as quantum phase transitions: first versus second order," *Physical Review A*, vol. 74, no. 6, Article ID 060304, 2006.

[37] W. P. Baritompa, D. W. Bulger, and G. R. Wood, "Grover's quantum algorithm applied to global optimization," *SIAM Journal on Optimization*, vol. 15, no. 4, pp. 1170–1184, 2005.

[38] C. Durr and P. Hoyer, "A quantum algorithm for finding the minimum," http://arxiv.org/abs/quant-ph/9607014.

[39] A. Montanaro, "Quantum pattern matching fast on average," *Algorithmica*, 2015.

[40] Y. H. Lee, M. Khalil-Hani, and M. N. Marsono, "FPGA-based quantum circuit emulation: a case study on Quantum Fourier transform," in *Proceedings of the 14th International Symposium on Integrated Circuits (ISIC '14)*, pp. 512–515, IEEE, Singapore, December 2014.

[41] M. Khalil-Hani, Y. H. Lee, and M. N. Marsono, "An accurate FPGA-based hardware emulation on quantum fourier transform," in *Proceedings of the Australasian Symposium on Parallel and Distributed Computing (AusPDC '15)*, vol. 1, Sydney, Australia, 2015, a1b3.

[42] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[43] D. Wecker and K. M. Svore, "LIQUi|>: a software design architecture and domain-specific language for quantum computing," http://arxiv.org/abs/1402.4467.

[44] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization*, John Wiley & Sons, New York, NY, USA, 2007.

[45] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik, "qHiPSTER: the quantum high performance software testing environment," http://arxiv.org/abs/1601.07195.