Goran Negovetic
ECE572: Advanced Logic Synthesis
Project Proposal
November 4, 2001

Hardware Implementation of Quantum Circuits

1. Introduction

Since there are no means to make a true quantum computer at present, we have to settle on different software and hardware simulations. Quantum calculations obey the laws of quantum mechanics. Quantum computation requires exponential amount of calculation in a polynomial amount of space and time. This is the reason why even limited size-circuits require exponential amount of memory and processing time and recourses. However, quantum circuit operation exhibits a large amount of regularity and parallelism. This led to an idea of using FPGA based hardware for quantum circuit simulation. It is important to realize that this circuit can be only of a finite size to meet the hardware resources. However, today's FPGA offer very large amount of memory and flip-flops, as well as hard-wired arithmetic operations such as multipliers.

2. Quantum Circuits

Quantum circuits are composed of quantum logic gates. There are number of elementary quantum gates; however, only a subset of these are required to implement an arbitrary size quantum circuit. Penalty for smaller set of gates is paid with a larger over-all circuit.

The basic storage unit in a quantum computer is a Qubit. A Qubit can take value of zero, one, and a superposition of 0 and 1. In the superposition state, complex amplitudes are used to represent probabilities of the Qubit being in one of the familiar logic states, namely 0 and 1. For a register of N bits, there are 2^N states, each having a complex probability in the superposition state. This is the root of parallelism, since all possible states are calculated in parallel. When the state of a Qubit is observed, its probability collapses to either 0 or 1, hence the superposition is destroyed. Also, since only finite precision can be employed for either software or hardware quantum circuit simulation, there is accumulated error that propagates through the system.

Quantum gate operations can be expressed as matrix operations. Each gate operation is expressed as multiplying 2^N-dimensional vector by the 2^N x 2^N transformation matrix. Going from M x 2^N-dimensional Quantum vectors to one 2^(M+N) vector, we employ the Kronecker product matrix operation. Also, since I will use only gates that operate on a small number of Qubit (N = 1 or 2), these matrices are 2x2 or 4x4 matrices replicated many times, as on Figure 1.

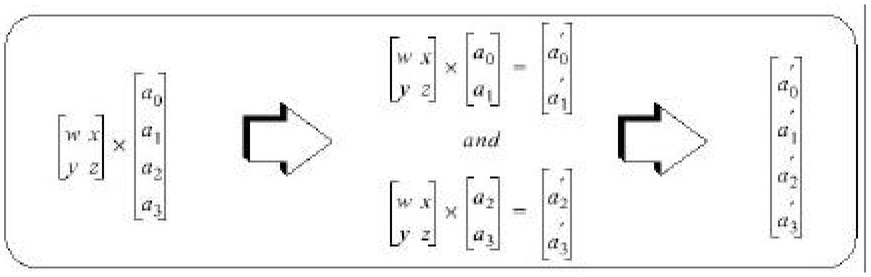

Figure 1

To visualize the quantum circuit operation, I created elementary quantum gates as functions on matrices in Matlab. The quantum circuit is than the input vector passed through these functions. Figure 2 shows a small circuit in Simulink, using those Matlab functions.

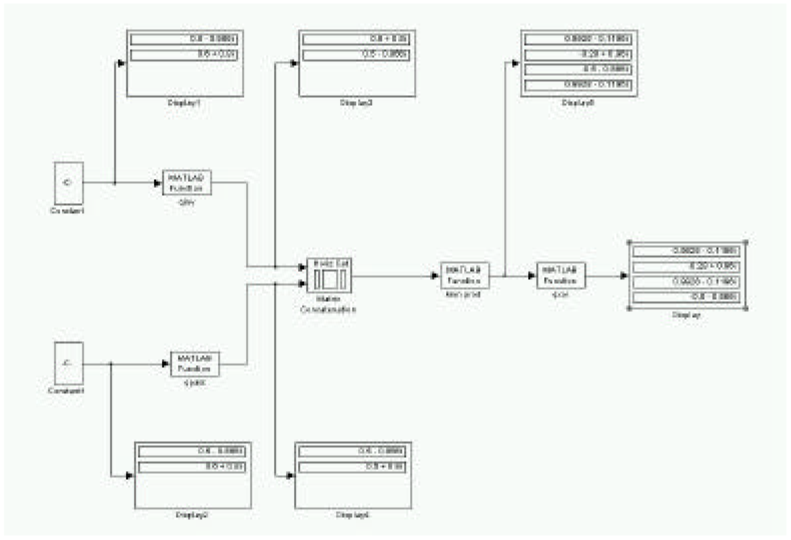

Figure 2

Most quantum operations can be implemented as register content transfer. For example this is so-called controlled NOT gate:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathrm{X} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_3 \\ a_2 \end{bmatrix} \longleftarrow$$

Swap Amplitudes
a_2* and *a_3

Figure 2

So the function of the gate is to swap a_2 and a_3.

3. Implementation

The following diagram describes the flow of proposed Quantum circuit implementation.

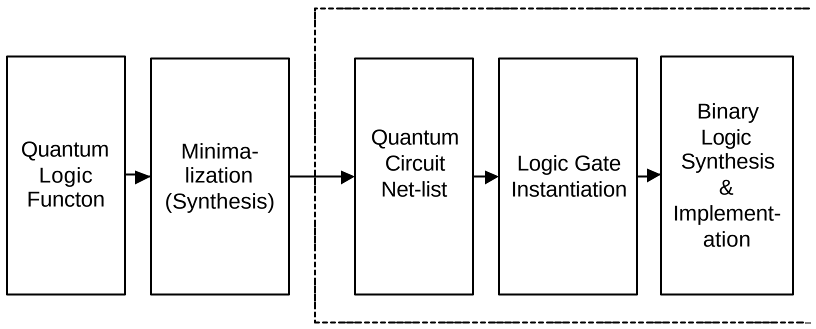

Figure 3

My goal is to implement a library of Quantum gate primitives, together with the gates for matrix manipulation necessary for the circuit implementation. There will be a utility program to transform the Quantum circuit net-list to the instantiation of these library primitives. From that point on, the circuit will be synthesized and implemented using existing tools for a chosen FPGA architecture.

3.1 Hardware resources and data representation

As per previous discussion, each Qubit is represented as two complex numbers. Each complex number requires two registers, each storing a floating-point number. I assumed that amplitudes of Qubits are normalized. IEEE floating number format offers excessive dynamic range, hence I decided to use simplified format:

S	Fraction – 15-bit

The largest number less than one is 0.99998 and smallest number larger than zero is 0.0000305.

I will build a complex number multiplier using this number representation. It will consist of four unsigned integer multipliers, two adder/subtractions and the logic for sign decision.

As shown earlier, most quantum operations consist of swapping the register contents. Even though these are counterparts of combinatorial binary logic, in this implementation they will be clocked. This naturally leads to the pipelined architecture.

3.2 FPGA

I chose to use Xilix Virtex-II architecture FPGA for the Quantum circuit implementation. It offers up to 8 million system gates, up to 1.5Mb SRAM memory, dedicated 18 x 18-bit multiplier blocks and fast look-ahead carry logic chains.

4. Conclusion

The performance and feasibility of proposed implementation is limited by the size of the circuit and available hardware resources. Exponentially increasing size of the parallel processing is the limiting factor. However, similarity of quantum circuit operations with unitary transforms often employed for image and other signal processing, leaves possibilities of reusing and modifying the procedure for applications in digital signal processing

5. Acknowledgements

1. A Parallel Quantum Computer Simulator, Kevin M. Obanland and Alvin M. Despain
2. Quantum Computation: Theory and Implementation, Edward S. Boyden
3. Efficient Quantum Transforms, Peter Hoyer
4. Elementary Gate for Quantum Computation, Adriano Barenco et al.