Behavioral Model of Integrated Qubit Gates for Quantum Reversible Logic Design

Matthew Lewandowski, Nagarajan Ranganathan, and Matthew Morrison
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida
{mamorris, mlewandow, nranganat}@cse.usf.edu

Abstract—Reversible logic is gaining significant consideration as the potential logic design style for implementation in modern nanotechnology and quantum computing with minimal impact on physical entropy. Recent advances in reversible logic allow schemes for computer architectures using improved quantum computer algorithms. We present a VHDL behavioral model for the design and simulation of the quantum interactions of qubits in theoretical reversible logic structures. Modeling IQ gates, as opposed to only Control-V gates or Toffoli gates, allows for a more robust model that more accurately reflects a theoretical reversible computing structure. This method is an extension to existing programming language and modeling method that allows for reversible logic structures to be designed, simulated, and verified. To the best of our knowledge, this is the first work in the behavioral model of integrated qubit gates.

Index Terms—Emerging Technologies; Low Power; Nanotechnology; Quantum Computing; Reversible Logic.

I. INTRODUCTION

Quantum computing devices dissipate energy due to bit erasure within their interconnected primitive structures, which is an important consideration as transistor density increases. Entropy gain in these environments is directly related to the probability of a quantum particle occupying any of its states [1]. Conventional computers are not sufficient for simulating both general quantum systems as well as quantum computers, because the problem complexity needed to describe the system grows exponentially as the number of quantum corpuscles increases.

In order to design an ideal universal computer that dissipates arbitrarily-low energy, reversible logic must be implemented, since the laws of physics indicate toward reversibility in time [2]. Additionally, quantum computing allows for a different view of Moore’s law. Reversible quantum computers would be able to keep pace with Moore’s law provided that a single qubit is added to the computer every two years.

The basic principle of reversible computing is that a bijective device with an identical number of input and output lines will have no heat dissipation, since the electrodynamics of the system allow for prediction of all future states based on known past states, and the system reaches every possible state [3]. Therefore, the primary motivation for the study of reversible computing is the desire to reduce energy dissipation in computing machinery, which would permit higher transistor density and computing speed.

We present a robust behavioral model for the fundamental Integrated Qubit (IQ) gates towards the design of locally reversible logic structures. In Section II, we review the fundamental quantum principles behind reversible logic, the mathematical modeling of Bloch spheres, Control-V gates and previous work in the modeling of those structures. In Section II, we present the behavioral modeling system, based in VHDL-93, and simulation results of reversible logic structures using the IQ library are shown. In Section IV, the advantages of modeling quantum systems in this language are discussed and demonstrated.

II. REVERSIBLE LOGIC DESIGN

A. Reversible Quantum Computing

In [5], Feynman asked two very important questions which govern our approach towards reversible logic design: Can physics be simulated with a universal computer, and what kind of physics are we going to imitate? The first question brings about the issue of local interconnections. The major issue with interconnections is that wires are a major source of energy dissipation with computing structures, due to changes in the wire’s voltage and internal resistance [6]. The second involves consideration of how computing is different from physical law. In a computing structure, there are a finite number of input and output combinations, as well as a finite number of logical calculations. Even the size of the computer is finite. This is different from physics, where space can be measured in infinitesimal distances and wavelengths, and may be summed to an infinite size of the universe. Since physical knowledge is always incomplete, and the laws of physics indicate toward reversibility in time, the goal, to any mathematical model of a reversible logic structure, is to design a theoretical structure that beats the experiment at the present time.
B. Reversible Design Goals

The three major design goals of reversible logic are as follows. First, minimization of the quantum cost - the number of 1*1 and 2*2 reversible calculations necessary to generate the logical output - will reduce the device’s computational complexity. Second, minimization of the delay - the logical depth of the device – will improve the throughput of the device. Third, reduction of the ancillary inputs and garbage outputs - inputs and outputs not implemented in the design of the gate and only serve to maintain reversibility of the device – will improve the design space required to implement the logic.

C. Unitary Matrix Representation of Quantum Interaction

We may now introduce the matrix operations necessary to fully realize the locally reversible Turing machine. In [7], DiVincenzo provided a proof that a universal quantum circuit may be completely constructed using only 2x2 reversible gates. This became an important endeavor after Shor presented his algorithms for finding discrete logarithms and factoring integers [8]. Since the laws of physics only permit unitary transformations, deterministic computation is performable on a quantum computer if and only if it is reversible.

The representation of the states in the two-dimensional Hilbert space is found by using a complex projective line, which is a geometrical sphere known as the Bloch sphere, which contains points on the edge of the sphere that correspond to mutually orthogonal state vectors. The points on the sphere represent the system states, and the poles of the sphere represent the “spin up” and “spin down” electron states. It is important to note that the boundary condition on the Hilbert space is that only states on the surface of the sphere are “pure” states, whereas any state not on the surface is a mixed state [9]. Therefore, any reversible computing design is constrained to operations that achieve states on the surface of the Bloch sphere, or else there will be information loss. The state representation of the spin of the electron on a Bloch sphere is given by \( |\psi\rangle = \cos(\theta/2)|0\rangle + e^{-i\phi}\sin(\theta/2)|1\rangle \). This may further be represented with rotation due to \( x \) and \( y \) as well as the negation and identity matrices already presented.

The NOT matrix, which may be applied to a one-qubit signal to invert the result, such that

\[
\begin{pmatrix}
0 & 1 \\
1 & 0 \\
\end{pmatrix}
\begin{pmatrix}
a_0 \\
a_1 \\
\end{pmatrix} =
\begin{pmatrix}
a_1 \\
a_0 \\
\end{pmatrix}.
\]

Using the unitary operators first presented in [10] by Deutsch, we may accurately represent a NOT operator in the Hilbert space using two qubit operators in series to represent their half spins, where \( \varphi = \pi/2 \). Therefore, the NOT matrix is properly represented by its square root matrix, \( \frac{1+i}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \).

\[ R_y(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ -\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}, \quad R_z(\varphi) = \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & e^{-i\varphi} \end{pmatrix} \]

Figure 1: Graphical Representation Qubit Representation of the Bloch Sphere

This operator is known as a V gate – or Square Root of Not - and performing two V gates in series will result in the NOT matrix \( \frac{1+i}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \) * \( \frac{1+i}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \).

Similarly, if we obtain the Hermitian conjugate of the V gate, we obtain the V+ gate, \( \frac{1+i}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix} \). Just like the V gate, performing two V+ operations will produce the NOT matrix.

Since the V and V+ gates are Hermitian conjugates, the result that they will produce is the unity matrix when performed in series. The quantum representation of these gates is shown in Fig. 5.

This allows for the consideration of 2x2 Control-V and Control-V+ gates for use in reversible quantum computation. When the control signal is zero, the input value is matched on the output. When the control signal is ‘1’, then the unitary V operation is performed, just like with the V and V+ gates previously shown. When the resulting matrix for the Control-V gate is multiplied by itself, the result is that it produces the CNOT matrix. Therefore, two Control-V or Control-V+ gates in series are equivalent to a Feynman gate. Like the V and V+ gates, when the VV+ operation occurs, we obtain the unity matrix. Therefore, two-bit quantum gates have been demonstrated to be sufficient to synthesize any unitary operation in any size Hilbert space [9].

Figure 2: Quantum Representation of the Control-V and Control-V+ gates

C. Previous Work in Control-V Simulation

The unitary matrix approach using Control-V and Control-V+ algorithms was implemented in [13] using a genetic algorithm based approach [14][15], which were limited to a 4*4 implementation due to high computational complexity. They defined the set of intermediate values for the control-V and control-V+ matrix operations, and then use
the mutation, crossover and selection operators of genetic algorithms in order to derive the circuit. This was meshed in [16] with the transformation based method in [17] to produce a more efficient implementation. In [18], the control-V gates follow a set of transformation rules in order to minimize the synthesized circuit using K-maps. The K-maps are used to extract the circuit to determine which nodes may be deleted to preserve the output calculations and reversibility.

D. Integrated Qubit Gates

The next stage in reversible logic design is developing a library of two-bit quantum gates that allow for minimized construction of locally reversible logic structures. However, these operations may be combined into integrated Qubit gates, since their unitary evolution is a correct representation of space-time quantum mechanical operation [11], since a set of quantum acyclic gates can simulate quantum Turing machines in this manner [12]. This gate is implemented with a Feynman gate along with either a Controlled-V or Controlled V+ gate. The XOR output of the Feynman gate is used as the control signal for the Controlled-V or V+ gate that it is coupled with. The quantum cost of the integrated qubit gate is 1 and its worst-case delay is 1. The quantum configurations of these gates are shown below in Fig. 3. Figure 4 shows that the Fredkin Gate [3] may be realized as a cost of 5 using IQ gates [7].

III. PRESENTED INTEGRATED QUBIT BEHAVIORAL MODEL
A. Abstraction

We created an IEEE Standard 1076-1993 [19] package (often referred to as VHDL-93) to define, describe and efficiently model the reversible transformations that could be used within a reversible logic structure using integrated qubit gates. This library, called QUBIT, is a robust multiple-valued logic system which defines the given deterministic quantum states and generalizes the non-deterministic states, which allows the logic system to generalize a characterization of a Qubit that enters into a state other than the known deterministic logic values that exist in the system.

Table I shows the behavioral modeling system used to represent the spins of the qubits on the Bloch sphere when the Control-V receives binary inputs. Table II shows the expected results when the gate receives inputs A0 and A1. The differentiation between the control line and the second input allows for real-time control checks on each gate that ensures proper logic is being simulated, since the control lines are the qubit basis states. Passing a control line anything other than a ‘0’ or ‘1’ will cause simulation of the reversible logic structure to fail. Therefore, this differentiation, and introduction of variable ‘U’ allows for a robust system, as well as error checking. Additionally, any structure designed that has a ‘v’, ‘V’, ‘p’ or ‘P’ on the output has not been properly designed, since the final design should produce a binary output to match the binary input.

<table>
<thead>
<tr>
<th>Table I - IQ Spin Operation Lookup Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qubit State</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>v</td>
</tr>
<tr>
<td>V</td>
</tr>
<tr>
<td>p</td>
</tr>
<tr>
<td>P</td>
</tr>
<tr>
<td>U</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table II - V AND V+ Behavioral Modeling System</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0(Control)</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3: Quantum Representation of Integrated Qubit Gates

Figure 4: Realization of Fredkin Gate with IQ Gates
Fig. 5 shows the syntax of the LIQP gate procedure in the presented QUBIT library. The gate uses the lookup table for both the Control-V and the QXOR, which has the input/output lookup for the Feynman (CNOT) gate. As previously mentioned, this is a proper quantum mechanical representation of one two-qubit operation. Fig. 6 shows the resulting VHDL simulation of the LIQP gate.

```vhdl
PROCEDURE LIQP ( SIGNAL $k_1$ : IN QUBIT;
    SIGNAL $b_1$ : IN QUBIT;
    SIGNAL $p_1$ : OUT QUBIT;
    SIGNAL $q_1$ : OUT QUBIT ) IS
  VARIABLE delay : TIME := 1 ns;
  VARIABLE $q_1$ : QUBIT := '0';
  VARIABLE $p_1$ : QUBIT := '0';
BEGIN
  $q_1$ <= qxor_table($k_1$, $b_1$);
  $p_1$ <= cnot_table($k_1$, $q_1$);
  $p_1$ <= $p_1$ AFTER delay;
  $q_1$ <= $q_1$ AFTER delay;
END PROCEDURE;
```

Fig. 1: Presented VHDL-93 QUBIT Syntax

**B. Reversible Logic Design Rules for VHDL**

In order to successfully model the quantum interactions as well as develop a system that performs the desired reversible computation, the rules for reversible logic must be adapted into the behavioral model. In this section, we present a set of design rules and show how they were implemented.

First, we define each IQ gate as having two input signals and two output signals. This allows for the creation of intermediate signal values that are tied to the input/output signals of a fundamental gate. This permits easier debugging, and to maintain reversibility through the designed circuit.

This rule is realized in the syntax by declaring \( x \) signals for a line, where \( x \) is the value returned from the equation below. This works since the number of gates interacting with a particular line is equal to one then only a single new signal must be generated to interact with intermediate values.

\[
f(x) = \begin{cases} \sum \text{Gates}_{\text{line}}, & \text{Gates}_{\text{line}} > 1 \\ 1, & \text{Gates}_{\text{line}} = 1 \end{cases}
\]

Fig. 7 shows an example of how these signals are maintained in a designed reversible logic structure. The 3\( \times \)3 Toffoli gate incurs a cost of 5, since it requires two Control-V, one Control-V+ and two CNOT gates. Line \( A \) encounters 3 gates, so it requires 3 intermediate signals, satisfying our equations. Similarly, line \( B \) had 4 intermediate signals and line \( C \) has 3 intermediate signals. The total number of intermediate signals is 10, which means that each 2\( \times \)2 gate produces 2 intermediate signals, satisfying our constraint.

![VHDL-93 QUBIT Syntax](image)

The second rule we implement is that *control lines are Qubit Basis States*. In order for the reversible logic structure to properly operate, any control line must receive a ‘0’ or a ‘1’. If any other value in the lookup table is reached, our model throws an error, since the design rule has been violated. Such an error would typically occur from the improper use of multiple reversible gate transformations; creating a reversible logic structure consisting of a single RIQ or RIQP gate is an example of what would cause a Qubit to enter a state other than one from the presented multiple-valued logic system. Figs. 8 and 9 show an example of a CNOT gate’s design and simulation.

![Signal Generation for CNOT gate](image)

**C. Simulation of Fundamental Reversible Logic Structures**

Using this package an extensive library of fundamental reversible logic gates and reversible logic structures was designed and verified. In this section, we use our model to simulate the operations of two fundamental reversible logic structures. First, we simulate a Swap gate using two integrated qubit gates. This design, presented in [20], uses the IQ design paradigm to reduce the quantum cost from 3 to 2. Our behavioral model is advantageous in that it accurately
simulates the quantum computation of a SWAP operation, with a smaller quantum cost and delay than any other behavioral model. Fig. 10 shows the IQ configuration, and Fig. 11 shows our simulation results.

Next, we simulate a Fredkin gate as shown in Fig. 4. This design was produced using only the CNOT gates, Control-V structure and IQ gates as proposed in [7]. Without the IQ gates, the minimum quantum cost and delay of the Fredkin gate would be seven. By using our behavioral modeling system, we are able to simulate the minimal quantum operations needed to perform the Fredkin’s logical calculations. The resulting waveform is shown in Fig. 12. By getting the correct output waveforms without specifying the output functions of the Fredkin gate, the robustness of our behavioral model is demonstrated.

**IV. VDHL QUBIT IMPLEMENTATION AND ADVANTAGES**

Implementing our behavioral model in VHDL allows us several advantages than if we created our tool in another behavioral language, such as Ada. First, VHDL allows for operator overloading, and the redefining of a function, procedure, and operator. This work uses a variety of functions and procedures for simulation and modeling each with purposes that are task specific. Simulation log file generation and file writing require that a string is used for being written to a file. Thus, function calls to convert a Qubit type to string to be written to a file during test mode were used while procedures were used to write to the simulation log as nothing need be returned. Procedures are also beneficial because a procedure allows the use of signals, local variables, and the use of delay statements to model delay. This allows for quantum simulation not just for logical calculations, but to determine the delay of the propagation of the signal through the designed reversiblelogic structure.

Third, the user may use the VHDL Qubit library to create reversible logic structure layouts that may be used in the design of larger reversible logic structures. For this tool, we used Xilinx ISE 13.2 to create schematic symbols of gates that can be used in a visual design layout. With this schematic layout VHDL code can be automatically generated by Xilinx and require minimal changes to the code to become functional. Fig. 13 shows a Xilinx schematic symbol of a Fredkin gate composed of the Integrated Qubits in our QUBIT library.

**E. Verification**

Using the automatic testbench generation tool previously mentioned and ISIM M.81d, we were able to provide error-checking that ensures the user maintains the reversibility of their design. This follows the one-to-one mapping of inputs and outputs when creating reversible logic structures.

**VI. EXAMPLE MODEL**

In [20], a tree-based comparator was presented, which produces the initial comparator values for each bit, incurring a cost of 256. When the comparisons are ar ranged in tree-based, the 32-bit comparator requires 31 of these UPG configurations, incurring a cost of 12 each. We used our VHDL behavioral model to verify the logical correctness of the design. The design of a 4-bit tree-based reversible comparator is shown in Fig. 14, and was verified in VHDL in Xilinx 12.4.
V. CONCLUSIONS

Simulation results show that our VHDL quantum behavioral model based on Integrated Qubit gates is a more robust and efficient method for the design, simulation, and verification of reversible logic structures than previously existing models. It was also shown the use of VHDL in this method allows for the use of a natively robust programming language that allows for concurrency from the built in capabilities of this language.

REFERENCES