Prefix Parallel Adder Virtual Implementation in Reversible Logic

David Y. Feinstein, Mitchell A. Thornton and V.S.S. Nair
Department of Computer Science and Engineering, Southern Methodist University
{dfeinste, mitch, nair}@engr.smu.edu

Abstract **- This paper demonstrates a simplified approach for reversible logic synthesis based on direct translation of the circuit VHDL description into virtual Fredkin gates. We investigate the size and speed of such a reversible logic implementation of the Brent-Kung Parallel Prefix Adder (PPA) in comparison to a standard logic implementation. Using the Altera Corporation's Quartus II synthesis and simulation tool, we show that our virtual reversible logic implementation follows the $O(\log_2 n)$ delay and $O(n)$ cost of the standard logic implementation.**

I. INTRODUCTION

Pioneering work by Bennet, Feynman, Fredkin, and Toffoli investigated the potential of reversible logic to create circuits that theoretically achieve zero internal power dissipation [16]. Accordingly, futuristic logic design based on quantum devices must be based on such reversible logic where the circuit has the same number of inputs and outputs, all with fan-in and fan-out of 1. A variety of basic reversible gates have been proposed in the past three decades. Such gates are combined in a cascade fashion using various complex synthesis methods that aim to minimize garbage outputs and ancillary inputs [15]. The synthesis of quantum reversible circuits is generally regarded as a hard research problem. The synthesis tools that have been developed so far are able to work only with relatively small (< 25 inputs) reversible circuits.

This paper presents a simplified synthesis and simulation approach for reversible circuits of moderate complexity. We use direct translation of the circuit VHDL description into virtual Fredkin gates. While our approach is not suitable (nor is it intended) to replace the gate cascade based synthesis techniques for quantum circuits, it enables us to obtain some preliminary insights into synthesis issues for future reversible circuits of meaningful sizes.

Our goal is to investigate the cost and delay of addition circuits as they are implemented in our virtual reversible logic and to compare the resulting reversible logic circuits with their classic logic implementation. We use the QuartusII tool (from the Altera Corporation [1]) to synthesize and simulate the VHDL files representing reversible circuits.

In this work, we are particularly interested in the *Parallel Prefix Adder* (PPA) architecture that was pioneered by Ladner and Fischer [11]. This approach was later improved by Brent and Kung [3] for reduced fan-out, thus making it potentially suitable for future use in reversible logic. Therefore, our benchmark for this research is the Brent-Kung parallel prefix adder with operand sizes up to 32 bits.

Zimmerman [19] developed an extensive VHDL library that conveniently implements the Ladner and Fischer, the Brent and Kung, and many other PPA architectures using standard logic. The Zimmerman library is parameterized, allowing the user to create PPAs of arbitrary size. We have modified the Zimmerman library to produce a VHDL translation into reversible logic, which is then used as input to the Quartus II tool for virtual implementation and simulation.

This paper is organized as follows. Section II discusses the basic concepts of reversible logic and introduces the fundamental architecture of PPAs. Section III describes our approach in modeling the reversible Fredkin gate based on CMOS transmission gates and discusses the tools we use. Section IV provides our experimental results and we conclude in Section V.

II. PRELIMINARIES

A. *Reversible Logic*

Classical logic circuits provide outputs that result in a loss of information and hence, corresponding power dissipation. For example, an n-bit input decision circuit obtains a one-bit yes or no result from which there is no way to reconstruct the original input set. Landauer's principle states that the erasure of a single bit of information dissipates at least $K_B T \ln 2$ energy, where K_B is Boltzmann's constant and T is the temperature [6, 20]. This places a firm theoretical lower bound on the power consumption of computation based on information entropy. It prompted the development of reversible logic circuits by Bennet [2], Fredkin, Toffoli [7], and others. Reversible logic circuits allow the full reconstruction of the circuit's input information at any time from the output since such circuits can be modeled as bijective functions.

All quantum logic circuits are necessarily reversible since the fan-in and fan-out of 1 characteristic occurs due to the principle of conservation of matter and thus also obeys the theoretical elimination of energy loss. We note that the first computers based on molecular level quantum devices are just recently beginning to be realized and there exists an intense research interest for the future of computing [16]. Adiabatic circuits are based on traditional CMOS technology that achieve ultra low power loss due to reversibility. Lim et al. [13] produced a 16-bit Carry-Lookahead adder employing *Reversible Energy Recovery Logic* (RERL), which is a dual-rail adiabatic logic style. While the RERL device is not a full implementation of reversible logic, it uses partial reversible features which contribute to the achievement of low power loss.

A variety of basic reversible gates have been proposed that are realizable using quantum devices including the Toffoli, Fredkin, Controlled-NOT, Controlled-V, and controlled V^+. The use of Toffoli and Fredkin gates shown in Fig. 1 illustrate how common classical functions (NAND, AND, NOT, and fan-out duplication) may be implemented. It is clear that once the NAND gate or the equivalent set of AND and NOT gates are achievable, any arbitrary logic function can be implemented. Note that these implementations require the extra cost of auxiliary inputs (also called ancilla inputs), which are crucial to the operation of the gates. For example, the Fredkin NOT and Fan-out implementation uses only one input for the input variable *x*, while the remaining gate inputs become ancillary inputs, initialized to either 0 or 1.

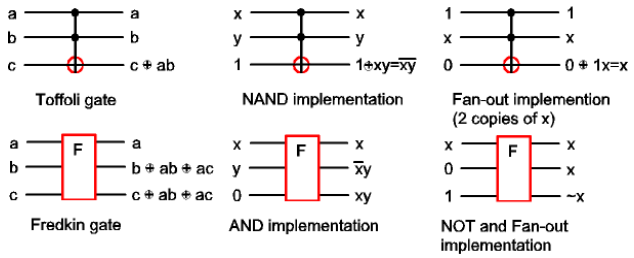


Fig. 1. Classical Logic Functions using the Toffoli and Fredkin Reversible gates

Since the number of outputs for reversible logic gates must equal the number of inputs, we find that reversible circuits can create a large number of unused or "garbage" outputs. Therefore, a major constraint in the synthesis, specification, and simulation of quantum circuits is the need to minimize the garbage outputs and the ancillary inputs.

Research efforts attempt to develop techniques to synthesize reversible quantum circuits using a cascade of gates like the example shown in Fig. 2. This circuit is the 5-variable hidden weighted bit "hwb5tc.tfc" benchmark from Maslov's benchmarks web page and was captured using Maslov's Reversible Circuit Viewer [14]. The circuit is comprised of generalized Tofolli gates with one to four control lines.

Reversible logic synthesis techniques that create circuits such as the one shown in Fig. 2 are still at an infancy stage and can currently tackle only small circuits. Interested readers should refer to recent work by Shende et al. [17], Kerntopf [10], Iwama et al. [9], and the "MMD" template based synthesis by Miller et al. [15]. Such synthesis techniques are not mature enough to implement complex arithmetic circuits such as the PPAs we focus on here.

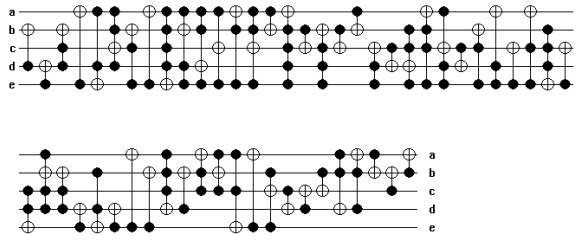


Fig. 2. The 5-variable hwb Function Implemented as a Cascade of Generalized Toffoli Gates

B. Prefix Parallel Adders

Research in binary adders focuses on the problem of fast carry generation. A naïve adder circuit implementation is the *Carry Ripple Adder* (CRA), where the carry information propagates linearly along the entire structure of full adders. As a result, while the CRA achieves the lowest cost (size) of $O(n)$, its operation is very slow, with a worst case delay of $O(n)$, where n is the number of added bits. We note that in quantum logic it is still desirable to minimize cost (the number of gates) but the underlying reason is to reduce the possibility of decoherence rather than delay minimization as is the case in classical logic circuit implementations.

The parallel prefix adder suggested by Ladner and Fischer [11] in 1980 pioneered a stream of solutions that treat the carry generation problem as a prefix computation.

Prefix computation deals with computing compositions of operators. The composition operator * must be associative, that is

$$(x * y) * z = x * (y * z). \qquad (1)$$

The prefix computation problem can be defined as follows:

Given a set S and an associative composition operator $*: S \times S \rightarrow S$ with input $x_0, x_1, \ldots, x_{n-1}$, compute $y_0, \ldots, y_{n-1} \in S$ so that $y_0 = x_0$ and $y_i = x_0 * x_1 * x_2 * \ldots * x_i$.

By the above definition, it is clear that $y_i = y_{i-1} * x_i$.

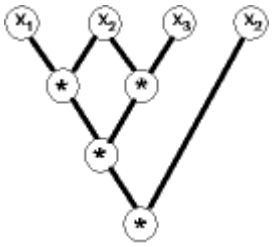
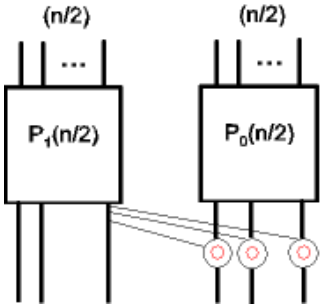


Fig. 2. Parallel prefix product with associative operator *

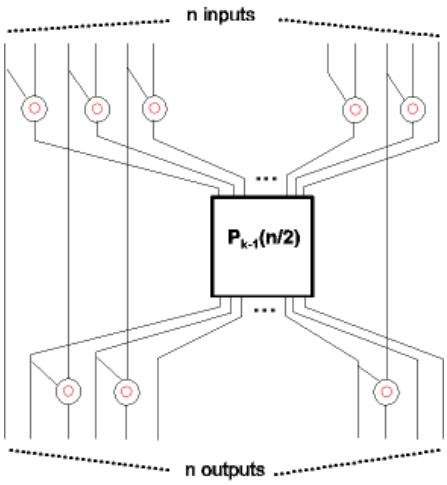
Fig. 2 demonstrates a simple product circuit that computes, in a parallel prefix fashion, the following set of values:

$$X_1 * X_2 * X_2 * X_3 * X_2$$
$$X_1 * X_2 * X_2 * X_3$$
$$X_1 * X_2$$
$$X_2 * X_3 \qquad (2)$$

Following the discussion in [11] we demonstrate how the general parallel prefix operation $P_k(n)$ is performed on a set of n-inputs at the k level. Fig. 3a shows how $P_0(n)$ is constructed recursively from $P_1(n/2)$ and $P_0(n/2)$. The left recursive construction $P_1(n/2)$ has its' subscript k equal to 1 since it is connected in the second level to $P_0(n/2)$.



a. The First Recursive Step



b. The Recursive Step at Level k

Fig. 3. The Recursive Steps for the Parallel Prefix Computation

Fig. 3b illustrates schematically how $P_k(n)$ is created recursively at level k of a PPA. It can be seen that each additional level (increase in k) will increase the depth of the circuit. Ladner and Fischer found that $P_k(n)$ solves the general prefix problem in parallel for n inputs with

$$\text{Depth (delay)} = D(P_k(n)) \leq K + \log_2 n \qquad (3)$$

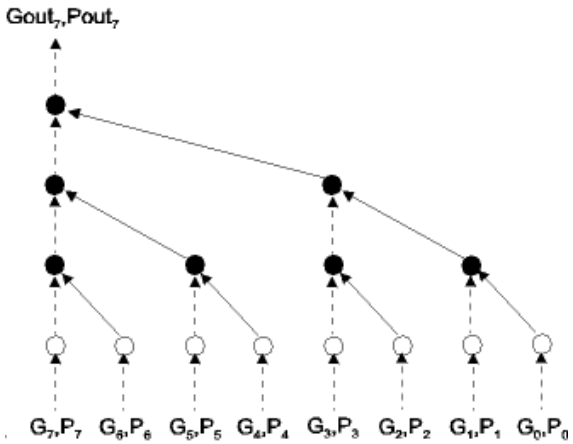
and

$$\text{Cost (size)} = C(P_k(n)) \leq 2(2+1/2^k)\, n - 4 \qquad (4)$$

for $n \geq 1$ and $0 \leq K \leq \log_2 n$. Therefore, the parallel prefix computation has a delay of $O(log_2 n)$ and size of $O(n)$. While asymptotically the $O(n)$ size is similar to that of the CRA, the resulting delay is significantly less [11].

The PPA quickly dominated adder design, with various architectures proposed by Brent and Kung [3], Kogge and Stone, Knowels [18], Han and Carlson, and more [8]. Interested readers also are referred to the detailed discussion in [5].

a. White Processor b. Black Processor



c. Carry Computation of the MSb.

Fig. 4. Brent and Kung's PPA

As we were looking for a PPA architecture that is suitable for our demonstration of a virtual simulation of reversible logic, we found much interest in the Brent and Kung adder [3]. Originally developed to achieve regular layout rules suitable for existing VLSI design methodology, this architecture provides a consistent

fanout of one on all outputs. This single fanout is crucial for reversible logic and thus seemed to be a good candidate for our study.

A simplified illustration of the Brent-Kung PPA is shown in Fig. 4, showing only the carry computation of the most significant carry bit during an 8-bit addition. The carry calculation uses two sub-functions, *g* (carry generate) and *p* (carry propagate). The architecture uses two types of nodes dubbed the white and black processors. The white processor (Fig. 4a) is merely a routing node, with *G*out=*G*in and *P*out=*P*in. The black processor performs the following computation:

$$G_{out} = G_{in} \cup (P_{in} \cap G'_{in}) \qquad (5)$$
$$P_{out} = P_{in} \cup P'_{in}.$$

The Brent-Kung PPA design incurs a cost (size) of $O(n)$ and exhibits a delay of $O(\log_2 n)$, where n is the number of bits in the addition function. The ease of implementation made it one of the first popular PPAs.

III. Our Approach

In this paper we are concerned with a virtual implementation of reversible circuits that use a direct translation of relatively complex binary functions into circuits composed of Fredkin reversible gates. We note that the Fredkin gate forms a complete set (with constants) so that any Boolean function can be realized with only Fredkin gates. Since a classical logic *n*-bit PPA requires 2*n* inputs and *n* outputs (neglecting the carry-in and the carry-out bits), a reversible counterpart will require at least 2n variables or lines. Current reversible logic synthesis based on gate cascading reported various size limits of 8-25 varaibles [15]. In fact very few benchmarks for reversible circuits with more than 25 variables are listed in the Maslov page[14]. Therefore, it is fair to assume that 32 and 16 bit adders are beyond the scope of current reversible logic synthesis based on gate cascading. Therefore, we follow a direct translation approach similar to the work in [4]. This results in an output VHDL file that is ready for *Field Programmable Gate Array* (FPGA) implementation. The emphasis is on circuit regularity as opposed to emphasis on minimization of the number of garbage outputs and ancillary inputs that is very crucial in the synthesis of reversible logic circuits. We recognized a priori that our results will include a large number of such garbage outputs and ancillary inputs, and are therefore useful only for the purpose of virtual implementation in our current investigation. Our implementation maintains direct correlation between the sub-modules of the reversible result to their standard logic counterpart.

The Altera Corporation's Quartus II software is a comprehensive suite of tools for the synthesis, simulation, and implementation of complex logic circuits. It conveniently accepts various formats of design source files, including standard VHDL and Verilog descriptions. The source files are synthesized into an RTL database, which is then optimized and minimized for target FPGA devices. A convenient feature of the Quartus II program for our research purposes is the ability to view RTL data prior to the optimization and minimization step.

We first modeled and implemented the basic Fredkin gate using CMOS transmission gates as shown in Fig. 5. VHDL implementation of the transmission gate is not a trivial matter [12]. We found that the approximation of a one-way transmission gate using the built-in Altera "TRI" primitive is useful for our application.

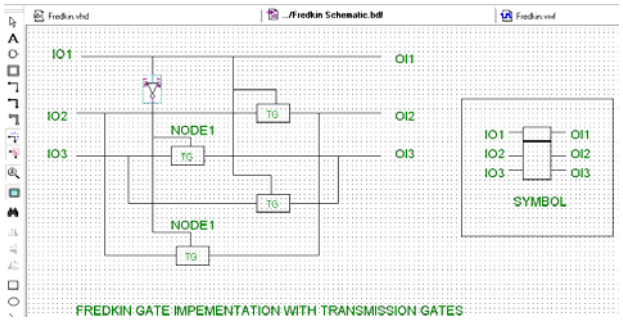


Fig. 5. VHDL Model of a Fredkin Gate

Zimmerman's PhD dissertation [19] on PPAs is an excellent research resource. He developed a comprehensive VHDL library for the implementation and investigation of various PPA architectures. The Zimmerman PPA VHDL library is parameterized to allow for the synthesis of adders with various data widths. His web site also includes a graphic interface applet that produces graphical representations of various PPA architectures.

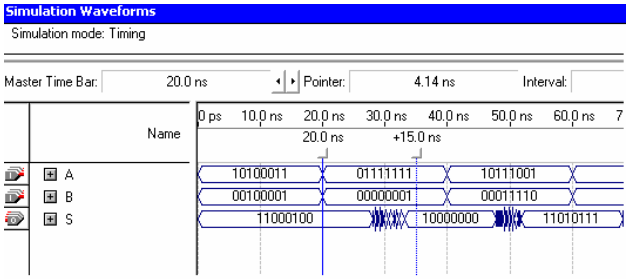


Fig. 6. Simulation of 8-bit Brent-Kung PPA Implemented with Fredkin Gates

We modified the Zimmerman VHDL files to create the Brent-Kung PPA using our Fredkin gate models. We used direct translation by replacing each original standard logic equation with a corresponding Fredkin implementation corresponding to the mappings in Fig. 1. The resulting files were then used as input to the Quartus II tool for synthesis and simulation. Fig. 6 shows the results of the timing simulation of our 8-bit

implementation of the reversible Brent-Kung PPA. This simulation tool was useful for debugging translation errors occurring during the design decomposition into Fredkin gates.

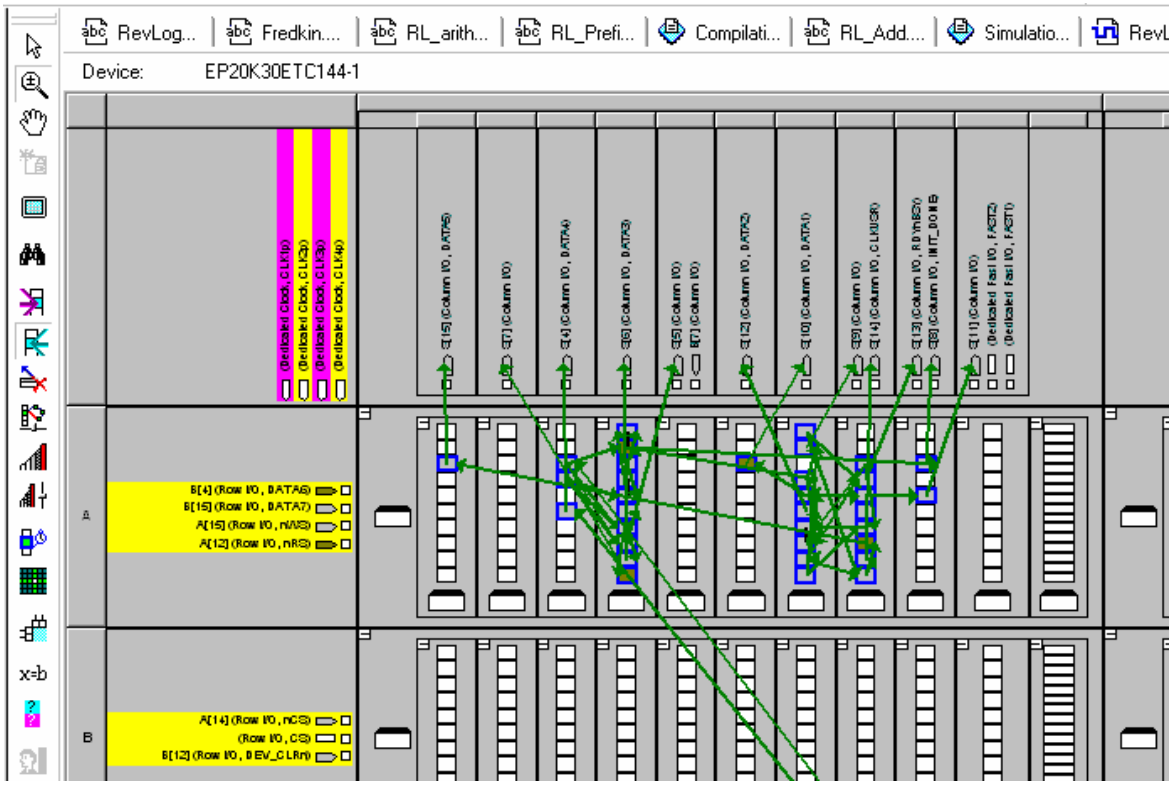


Fig. 7. Place and Route Map of a 16-bit Brent-Kung PPA

Our research methodology was to compare the size and the delay simulation results between reversible and standard logic versions of the Brent-Kung PPA of various sizes. An interesting pitfall of our approach quickly appeared when we tried to place and route the resulting synthesis of both reversible and classic logic versions into Altera's FPGA chips. Our motivation was to use the number of *Logic Element* (LE) units reported at the end of device compilation as a measure of size [1]. Fig. 7 shows the Place and Route map result when a reversible 16-bit Brent-Kung PPA is implemented in an Altera EP20K30E FPGA device. Each small square within the chip map represents one LE unit.

When we placed and routed comparable reversible and standard logic versions of same operand size PPAs, the compilation tool reported a similar number of LEs (see Table 1). This clearly indicates an information loss problem since the extra logic required by the reversible circuits must result with more size. It seems that the problem lies in the optimization and minimization phase of the synthesis operation that the design undergoes within the Quartus II tool prior to the Place & Route phase. It turns out that the optimization phase simply converts the reversible logic files back to a similar set of optimized equations that were obtained for the standard logic implementation. While this in fact validates our translation process, it is of course undesirable for our size comparison purposes. This problem was avoided with the RTL viewer which allows us to view the non-optimized VHDL files and to manually count the number of Fredkin gates required in order to compare sizes.

IV. Experimental Results

We obtained our initial results shown in Table 1 using the LE count provided by the Quartus II compilation reports for operand sizes of 4, 8, 16, 32, and 64. As discussed in the previous section, there is a loss of information due to the optimization and minimization of both the reversible and the standard logic source files. This results in very similar LE counts for each operand size comparison.

TABLE 1
Information Loss Due to Place and Route Optimization of the RTL Data

Operand size N	I/O pins	Reversible logic in LE units	Standard logic in LE units
4	12	16	18
8	24	45	44
16	48	71	72
32	96	103	99
64	192	222	212

Fig. 8 demonstrates the RTL viewer display for the 8-bit reversible Brent-Kung PPA. Each of the small square elements represents a Fredkin gate. The large square is a higher hierarchy sub-circuit which is comprised of additional Fredkin gates. During the tedious manual counting process, we must analyze all such sub-circuits in order to accurately count all the Fredkin gates of the benchmark. Similar efforts are needed in counting the classical gates in the RTL views of the standard logic implementations. It should be noted that other commercial modules for the Quartus II, like the Mentor Graphics Leonardo Spectrum tool, allow users to treat a module as a black box. Such a tool can be used to eliminate the tedious manual counting process. However, this tool was not available to us during this research.

Table 2 summarizes our results for operand sizes of 4, 8, 16, and 32. Here we see a consistent size cost increase when comparing the reversible and standard logic implementations. The I/O count column relates only to the lines connected to the FPGA pads for time delay simulation. Currently our interface to the Altera tool does not provide an automatic means to count the internally unused inputs and outputs required by the Fredkin gates. Since our simplified translation synthesis technique does not minimize these extra inputs and outputs, a rough empirical estimate for the total I/O is approximately 2.5 times the number of the Fredkin gates in addition to the actual pad I/Os listed in the second column of Table 1.

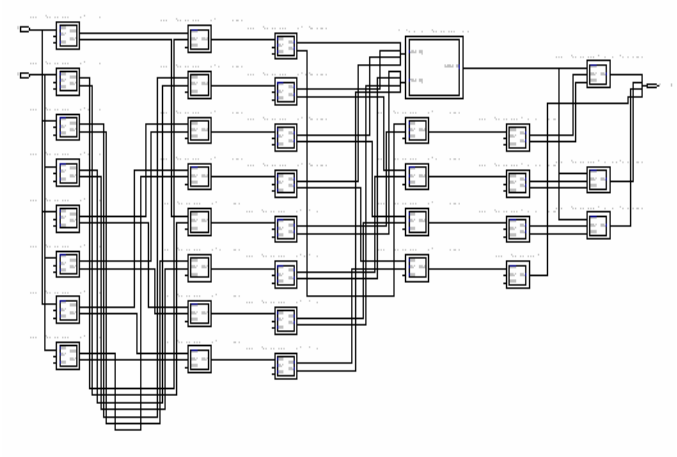


Fig. 8. The RTL Viewer Output for the 8-bit Brent-Kung PPA of Fig. 6

TABLE 2
REVERSIBLE AND STANDARD LOGIC COMPARISON BY THE RTL VIEWER

Operand Size N	I/O pins	Reversible Logic in Fredkin Gates	Standard Logic gates
4	12	53	26
8	24	120	62
16	48	259	135
32	96	547	312

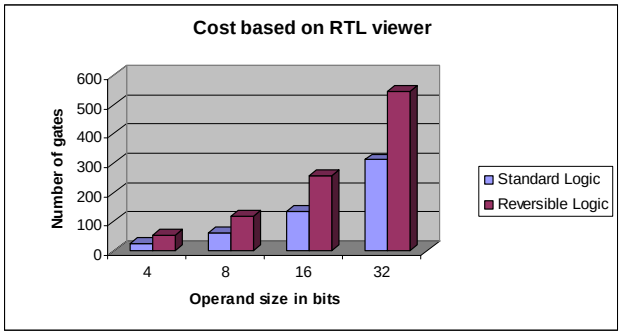


Fig. 9. Comparison of Number of Fredkin and Classical Logic Gate Requirements for each Operand Wordsize

Fig. 9 illustrates the size comparison. As predicted by equation (4), the size grows linearly with respect to the operand size increase for both implementations. Of course, additional results with larger circuits will be needed to confirm this trend beyond our current range of 64 bits. The number of Fredkin gates in the reversible implementations seems to be about two times that of the implementation using classical logic gates. Approximating the silicon size of a Fredkin gate to be about twice that of a standard logic gate, we conclude that our reversible virtual implementations require about four times the cost of the standard logic implementations.

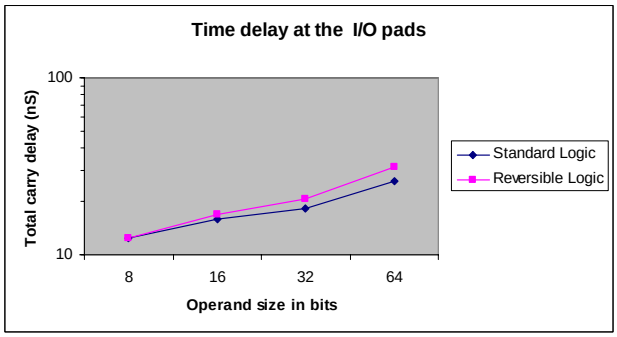


Fig. 10. Comparison of Time Delay Results for Reversible and Classical Logic Implementations

Using the Quartus II simulation tool, we have compared the time delay for the reversible and standard logic implementation for different operand sizes. The results appear in Fig. 10 and they generally follow the expected $O(\log_2 n)$ time for the Brent-Kung PPA in (3). There is a slight delay overhead for our reversible virtual implementations. While these results are important to validate our model, the actual delay with future quantum circuits are likely to exhibit different variances.

V. CONCLUSIONS

In view of the small size limitation of current gate cascade synthesis tools for reversible logic, we demonstrate a simplified synthesis technique based on

direct logic block translation into reversible gates. While our approach clearly cannot achieve the efficient synthesis of the gate cascade synthesis, it allows us to peek into the behavior of moderately complex reversible arithmetic circuits such as the Brent-Kung PPA. Comparison of our virtual reversible logic implementation of the Brent and Kung PPA with the standard logic implementation shows that it still follows the $O(\log_2 n)$ delay and $O(n)$ cost of the standard logic implementation. Due to the complexity of the Fredkin gate as compared to a standard logic gate, the cost of our reversible logic implementation is roughly four times that of the standard logic implementation.

This result demonstrates how a state-of-the-art commercial platform like the Quartus II can be adapted to investigate futuristic technology like reversible logic. In particular, we show how to overcome the loss of information due to optimization by using the RTL viewer. More importantly, the problems we encountered in using this tool for synthesizing and modeling reversible logic circuits underscores the need for the development of new synthesis and simulation tools specifically for quantum device based circuits. Since classical logic synthesis tools exploit characteristics such as reconvergent fanout (which is not permissible in quantum circuits) and the optimization techniques automatically remap low-level structures into those inappropriate for quantum logic, altogether new approaches for quantum logic CAD tools are required.

Future work includes the investigation of the use of Toffoli gates instead of (or together with) Fredkin gates and will involve arithmetic architectures other than PPAs.

REFERENCES

[1] Altera Corporation, *Quartus II Software*. Available online: http://www.altera.com/products/software/products/quartus2/qts-index.html.

[2] C.H. Bennett, "Logical reversibility of computation,"*IBM, J. Res. Dev.*, Vol. 17, No. 6, pp. 525-532, 1973.

[3] R. P. Brent and H. T. Kung, "Regular layout for parallel adders", *IEEE Trans. Comp.*, Vol. C 31, no. 3, pp. 260-264, 1982.

[4] J.W. Bruce, M.A. Thornton, L. Shivakumaraiah, P.S. Kokate, and X. Li, "Efficient adder circuits based on a conservative reversible logic gate", In Proceedings of the *IEEE Computer Society Annual Symposium on VLSI*, pp. 83-88,April 2002

[5] M. D. Ercegovac and T. Lang, **Digital Arithmetic**, Morgan Kaufmann, 2004.

[6] R. Feynman, "Quantum mechanical computers," *Optics News*, vol. 11, pp. 11–20, 1985.

[7] E. Fredkin and T. Toffoli, "Conservative Logic," *Int, J. Theoretical Physics*, Vol. 21, Nos. 3/4, pp. 219-253, 1982.

[8] Z. Huang and M. D. Ercegovac "Effect of wire delay on the design of prefix adders in deep-submicron technology", *Signals, Systems and Computers*, 2000. Volume 2, pp. 1713 - 1717.

[9] K. Iwama, Y. Kambayashi and S. Yamashita, "Transformation rules for designing CNOT-based quantum circuits", In Proceedings of the *Design Automation Conference, 2002*, June 2002, pp. 419-425.

[10] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis", In Proceedings of the *41st Annual Conference on Design Automation*, pp. 835-837, June 2004.

[11] R. Ladner and M. Fischer, "Parallel prefix computation", *J. Assoc. Comp. Mach.*, Vol 27, pp. 831-838, 1980.

[12] S. S. Leung, "Behavioral modeling of transmission gates in VHDL", In Proceedings of the *26th ACM/IEEE Conference on Design Automation*, June 1989, pp 746-749.

[13] J. Lim, D.-G. Kim, and S.-I. Chae, "A 16-bit Carry-Lookahead Adder using Reversible Energy Recovery Logic for ultra-low-energy systems", *IEEE Journal of Solid-State Circuits*, Vol. 34, No. 6, pp. 898-903, June 1999.

[14] D. Maslov. Reversible logic synthesis benchmarks page. Online: http://www.cs.uvic.ca/~dmaslov/, Nov. 15, 2005.

[15] D. M. Miller, D. Maslov, and G. W. Dueck. "A transformation based algorithm for reversible logic synthesis". In Proceedings of the *Design Automation Conference*, 318-323 June 2003.

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

[17] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 22, No. 6, pp. 710-722, June 2003.

[18] M. M. Ziegler and M. R. Stan, "A unified design space for regular Parallel Prefix Adders", In Proceedings of the *Conference on Design, Automation and Test in Europe* - Volume 2, Feb 2004, IEEE Computer Society.

[19] R. Zimmermann, **Binary adder architectures for cell-based VLSI and their synthesis**, *PhD thesis*, Swiss Federal Institute of Technology (ETH) Zurich, Hartung-Gorre Verlag, 1998. Online: http://www.iis.ee.ethz.ch/~zimmi/.

[20] R. Landauer, "Irreversibility and Heat Generation in the Computing Process", *IBM J. Research and Development*, vol. 3, pp. 183-191, July 1961.