Using the Asynchronous Paradigm for Reversible Sequential Circuit Implementation

David Y. Feinstein
Innoventions, Inc.
10425 Bissonnet Street
Houston, TX, USA
david@innoventions.com

Mitchell A. Thornton*
Dept. of Computer Science and Engineering
Southern Methodist University
Dallas, TX, USA
mitch@lyle.smu.edu

Abstract

We present a brief survey of recent developments in reversible sequential circuits and quantum finite state machines based on both binary and multiple-valued solutions. We then argue the benefits of adapting the asynchronous approach for reversible sequential circuit design. We offer several new reversible implementations of key elements to be used in reversible asynchronous pipelines. We investigate the physical reversibility of the proposed design.

1. INTRODUCTION

Reversible logic design has been in the forefront of research for the past three decades due to the emergence of quantum computing and its potential of low-power design. Quantum circuits, which are inherently logically and physically reversible, promise potential speed up for some intractable problems such as prime factoring [1] and searching [2]. In some cases exponential speedup is possible depending upon the implemented computational framework. Landauer showed thermodynamically that classical irreversible logic, where information is destroyed (or re-created by fan-in), loses energy into the environment [3]. Reversible circuits, whether CMOS- or quantum-based, are likely to be the key to achieve true lossless power design.

Early theoretical work on the capabilities of quantum finite state automata investigated the unique power of such automata, which must be implemented in quantum (e.g. reversible) sequential circuits [4]. However, finite state machines necessarily employ some form of feed-back mechanism to enable the present state to be used in the computation of the next state. Such feedback mechanisms pose theoretical and implementation difficulties depending upon the underlying technology and models of computation being employed [5]. Such difficulties steered initial reversible circuit design research to deal with reversible combinatorial logic circuits, with some success [6,7,8,9]. Because many useful circuits can be realized based on the sequential circuit model, which in turn is generally based on the Turing machine computational model, there was an increasing interest in the past few years to tackle the challenge of designing reversible sequential circuits. The sequential circuit model of computation is generally implemented in a synchronous manner whereby a synchronizing clock signal is used to control the occurrence of state transitions. Chang and Cheng tried to avoid the clock altogether with asynchronous self-timed synthesis techniques that were modified to produce reversible asynchronous sequential circuits [10]. At the same time, researchers started to report formulations for simple reversible clocked latches and flip flops elements [11,12,13,14]. Many additional proposals followed with more elaborated synchronous reversible designs for shift registers [15], counters [16], and even a reversible general information processing unit that included a cache subcircuit [17].

We note that the majority of the literature in quantum logic and computing is based upon the so-called Deutsch model of computation [18]. In the Deutsch model, the quantum state vector evolves linearly in time and the need for a sequential state machine structure is non-existent. In the work described here, we investigate reversible asynchronous state machines that can be applied to other computational models such as the common Turing model. Furthermore, our conclusions are equally applicable to arbitrary implementation technologies whether they are conventional electronics or quantum in nature.

In this paper we explore the inherent advantages that the asynchronous approach has over the synchronous approach and we investigate a new asynchronous paradigm for sequential reversible logic synthesis. We demonstrate our approach by creating reversible designs for Sutherland's Micropipeline architecture [19] and the Mousetrap pipeline [20]. To the best of our knowledge, our designs include the first reversible implementations for the Muller C-element [21] and event-controlled storage elements using the Feynman and the Fredkin reversible gates [22,23]. Our approach is applicable to both binary and multiple-valued reversible circuits.

Prompted by the symmetry of the micropipeline structure, we investigate the physical reversibility of our designs, where the same circuit is used in reverse by exchanging the outputs with the inputs.

The paper is organized as follows. In Section 2 we discuss reversible logic, reversible finite state machines, and elementary reversible sequential circuits. We argue the case for using asynchronous reversible sequential circuits in Section 3. In Section 4 we review the classical

*This work was supported in part by US National Science Foundation (NSF) grant CCF-1116405.

micropipeline architecture and present our asynchronous reversible pipeline designs. We investigate physical reversibility of our design in Section 5. Conclusions and suggestions for further research appear in Section 6.

2. PRELIMINARIES

2.1. Reversible Logic Circuits

Definition 1: A binary or multiple-valued logic (MVL) gate/circuit is *logically reversible* if it maps each input pattern to a unique output pattern, or, can be modeled as a bijective switching function. □

The bijective mapping is defined by the *transformation matrix* of the circuit. An $n \times n$ reversible circuit with n inputs and n outputs can be modeled with a $r^n \times r^n$ transformation matrix, where r represents the radix of the discrete switching function domain.

Definition 2: A circuit is *physically reversible* if the circuit can physically operate by connecting the inputs to the outputs and vice versa. □

Many different binary and MVL reversible gates have been proposed [24]. Fig. 1a shows the versatile binary Fredkin gate which passes inputs B and C to output Q and R when the control input A is at 0. B and C are swapped to R and Q respectively when A=1. This controlled swap operation is shown in the common Fredkin symbol in Fig. 1b. Fig. 1c shows the binary Feynman gate (FG) which passes A to the P output and provides the A⊕B operation at the Q output. The Feynman gate is often called the Controlled-Not gate. Since a quantum implementation of reversible logic prohibits fan-out, the FG gate is often used to create one copy of A (Fig 1d) or more (Fig. 1e). The fixed '0' inputs are called ancillary inputs and are common in reversible logic design.

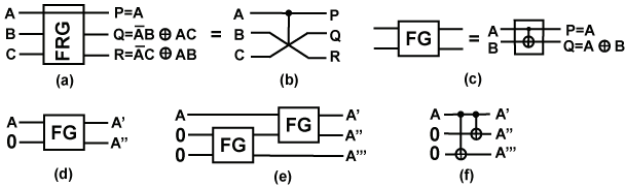

Figure 1. The Fredkin and Feynman Gates

Definition 3: An n-variable reversible gate cascade is a circuit composed of adjacent reversible gates that operate on the same n variables represented by horizontal lines across the circuit. □

Fig. 1f illustrates the cascade format for Fig. 1e. Both formats are used in the literature .Each gate in a cascade may be connected to one or more of the horizontal lines.

2.2. Reversible Finite Automata

A finite automata (FA) can be defined as a tuple A=(Q,Σ,δ,I,F) where Q represents a finite set of states of which $I \subseteq Q$ is the set of initial states and $F \subseteq Q$ is the set of final states, Σ is the input alphabet, and δ is the transition function $\delta: Q \times (\Sigma \cup \{\lambda\}) \rightarrow P(Q)$ where P(Q) is the power set of Q and λ is the empty string. Following the Turing model, the automaton A accepts an input string of length k if there exist a series of transitions δ_i, $i=\{0,1,\ldots,k\}$ beginning in an initial state and ending in a final state. The set of all the strings accepted by A comprises the language L(A).

Definition 4: The tuple $A^r=(Q,\Sigma,\delta^r,I,F)$ is the reversed automaton of A=(Q,Σ,δ,I,F) where $\delta^r(q,a) = \{p \in Q: q \in \delta(p,a)\}$. Note that by the definition of δ, $a \in \Sigma$ and $p \in Q$. □

A deterministic finite automata results when we restrict I to include only one unique state and define δ as $\delta: Q \times \Sigma \rightarrow Q$. In an early approach to determine if a quantum computer can simulate a deterministic FSM for all input strings of length k or less, the FSM is made reversible by adding an output tape where the FSM is writing symbols whenever it enters a node. Then the FSM can work in reverse by exchanging the roles of inputs and outputs [25]. Kondacs and Watrous investigated the power of 2-way quantum finite automata (2QFA) with two directional tape arranged in a circle, and they proved that such a machine surpasses classical FSM machines in view of their ability to accept the non-regular language $L = \{a^m b^m | m \geq 1\}$ with bounded error and linear time [4]. Recent work on the QFA application for sequential quantum circuits suggests the style shown in Fig. 2a [5]. The boxes marked with I perform initialization for the input, output and the state while the boxes marked with M perform measurements. |I>, |q>, |O> represent the QFSM input, state and output respectively. Note that the dotted lines indicate feedback for clarity only, as the qubits are re-used in the next computation step of the QFSM [5].

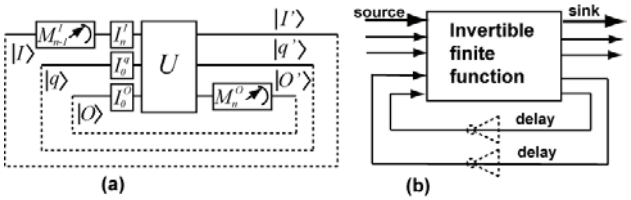

Figure 2. (a)QFSM (b) Toffoli/Muller/Huffman Model

In his seminal work on reversible computing [26], Tofolli modeled reversible finite automata as shown in Fig. 2b, where the state feedback lines impose delays on the new state. He used the term 'source' for ancillary inputs and 'sink' for what is now called garbage outputs. It is interesting to note that the Toffoli model follows Muller's asynchronous input-output style with wire delays in the feedback path (but without the source, sink, and invertible restrictions) [27]. A variant style called the

Huffman fundamental mode replaces the wire delays with the NOT gates (drawn with dashed lines in 2b). The more restrictive fundamental mode style requires the system to be in a state where all the signals (input, output and internal) are stable, and the environment must wait after it changes an input for the system to stabilize before making any other change.

2.3. Reversible Sequential Circuits

Fig. 3a illustrates a reversible level sensitive D Flip-Flop which implements the characteristics function of a latch [14]

$$Q_{out} = D_{in} * CLK + Q_{fb} * !CLK \quad (1)$$

It is easy to verify the latch behavior of this circuit by comparing to the equivalent circuit in Fig. 3b that utilizes the classical switch symbol. When the CLK control value is at 0 (shown), the circuits retains the last value of D_{in} stored in the latch when CLK=1. A reversible Master/Slave D flip flop is shown in 3c where the top left FG gate generates the reversed clock !CLK which is needed for the transition (trigger) behavior. Note that the two FRG gates provide the extra copies CLK' and !CLK' that may be fed to adjacent MS flip-flops. Other designs for sequential reversible D-latches and variety of flip-flops based on Fredkin and Toffoli gates appear in [11,28]. All the D-latch designs, including the one in Fig. 3a, follow the Muller's asynchronous input output mode of Fig. 2b.

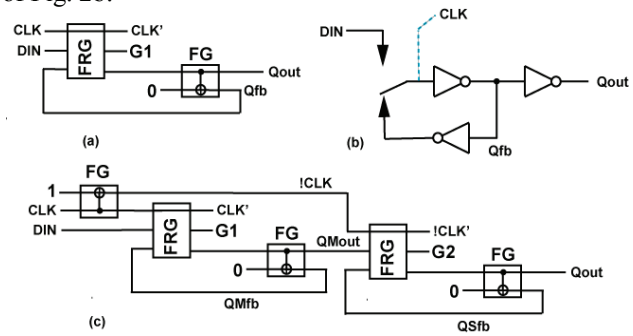

Figure 3. D Flip Flop/Latch design

Definition 5: The *quantum cost* of a circuit *C* is the number of elementary quantum operations that are used to implement the complete specification [22]. The quantum costs of the FG and FRG gates are one and five respectively [29]. □

Outputs that are not used (e.g. G1 in 3a and G1 and G2 in 3c) are called *garbage* outputs. A good reversible design attempts to minimize the number of the garbage outputs.

Multiple-valued sequential latches can be design along the same methodology shown in Fig. 3, for example, using the multiple-valued Fredkin gates to implement the NAND equivalent of the SR latch [30].

3. THE CASE FOR ASYNCHRONOUS REVRSIBLE CIRCUITS

Fig. 4 compares the classical (irreversible) synchronous (a) and asynchronous (b) approaches for complex pipelined sequential circuit design. Synchronous pipelines use a single clock for all stages, while asynchronous pipelines use handshake protocol to create a local stage control based on a request from a predecessor stage and an acknowledge from the successor stage. The relation between the request and acknowledge signals shown in Fig. 4c can follow two common protocols: the 2-phase (non-return-to-zero) and the 4-phase (return to zero) protocols. The 2-phase protocol provides two transactions, first transaction when *req* goes to '1' and the second transaction when *req* goes to '0'. In 4-phase protocol, *req* goes to '0' to indicate a return-to-zero that signals the completion of the single transaction.

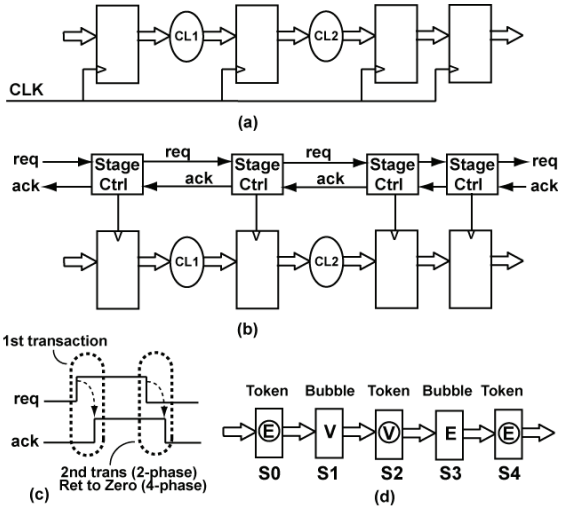

Figure 4. Synchronous vs. Asynchronous

Fig. 4d illustrates the abstract data flow in an asynchronous pipeline which is fundamentally different from the synchronous approach. During operation, Valid (V) and Empty (E) data flows between the stages under the control of the *req* and *ack* event. Fig. 4d illustrates the condition where the V and E data in stages S1 and S3 transit to stages S2 and S4, while no data (E) goes to S0. The newly transited data are called tokens, and they are marked with circles in the diagram. The old duplicates of data in S1 and S3 are marked as bubbles. In this model, a circuit must have at least one bubble to avoid a dead-lock stage [31].

The synchronous design requires the same clock (or phased locked copies) to be applied to all the stages. For classical irreversible circuits, the asynchronous paradigm has a clear advantage over the synchronous paradigm due to the elimination of the central clock and its resource costly distribution which is needed in order to minimize the clock skew. This advantage is much more significant

for reversible sequential circuits. In a reversible synchronous design, the reversible implementation of flip-flops (as shown in Fig 3) provides copies of CLK and CLK1 that can be used by adjacent flip-flops. However, this creates accumulated delays that very quickly exceed prohibitive clock skew budgets. The option of using a large number of PLL elements to overcome the skew is extremely resource and power expensive. Note that irreversible synchronous circuits require a large fan-out on the clock net, which is crucial to avoid the prohibitive delay when using a serially connected clock in a reversible design. In a stark contrast, reversible asynchronous designs use only local "clock like" control signals within each stage, as shown in Section 4.

Remembering that reversible/quantum circuits were conceived in an effort to significantly reduce power, one can further see that the elimination of a central clock in asynchronous reversible design offers significant low-power advantages, particularly for electronic implementations. Only the stages that are part of the active process are clocked locally in response to active events, while all inactive stages are dynamically quiet.

In an asynchronous design, the operation speed is determined by the local request/acknowledge latencies and is thus faster than the speed of synchronous circuits that depend on a single and global worst-case latency. Other advantages of asynchronous circuits include increased composability and modularity.

When considering reversible sequential circuits, one should note that an asynchronous design relies on latches (Fig. 2a) rather than M/S flip-flops (Fig. 2c). The quantum cost of latches is about half the cost of the M/S flip-flops. This is also true for MVL implementations.

4. REVERSIBLE ASYNCHRONOUS PIPELINES

4.1. Reversible Sutherland's Micropipeline

We propose the following reversible designs of classical pipelines to illustrate the use of the asynchronous paradigm in complex reversible sequential circuits. Fig. 5 shows the symmetrical design of the high performance Sutherland Micropipeline that uses 2-phase handshaking [19]. The pipeline provides forward and backward synchronizations via the *req* and *ack* control lines, respectively. Each register is an event-controlled register that performs the Pass and Capture operations, whose proposed reversible design is shown in Fig. 6. The local control of each register is achieved by the Muller C-element that performs the AND operation on *events* and its proposed reversible design is shown in Fig. 7.

The reversible event-controlled storage element of the micropipeline performs a transparent latch operation, with two control inputs (C=Capture, and P=Pass) and two control outputs (Cd=Capture done, and Pd=Pass done) that are delayed versions of the respective control inputs. For each stage in Fig. 5, the register latches are transparent when both C and P are '0'. When C becomes '1' due to a request from a predecessor stage, the latch becomes opaque. When P becomes '1' due to an acknowledge signal from the successor stage, the latch becomes transparent again. The first transaction is now completed with both C and P at '1'. When C goes to 0 at the second transaction of the two-phase protocol of Fig. 3c, the latches becomes opaque to capture the data, followed by the transition of P to 0 to pass the captured data and to complete the second transaction. The delay elements ensure that the Req signals arrive after the reversible logic delay and may be implemented by a path delay (e.g. wire or optical).

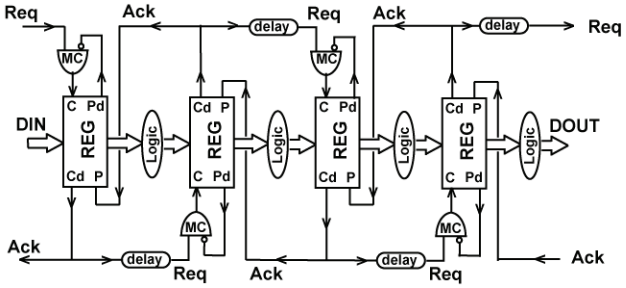

Figure 5. Sutherland's Micropipeline

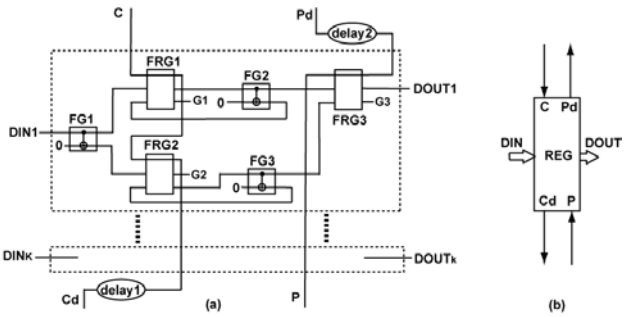

Figure 6. Reversible Event-Controlled Register

The reversible design of the micropipeline uses reversible combinatorial logic circuits between the stages that perform the required computation. The proposed reversible design of the event-controlled register is shown in Fig. 6. Each bit is controlled by two latches with opposite polarity response to the Capture control. The FRG3 gate is controlled by the Pass command to select the outputs from the proper latch (FRG1 or FRG2). The event-controlled register is extended to more bits so that the Capture and Pass controls propagate from bit to bit until reaching the delay elements placed prior to the Pd and Cd control outputs. For a quantum implementation, the quantum cost per bit C_{ecr} of the event controlled register is

$$C_{ecr} = 3*5 + 3*1 = 18 \quad (2)$$

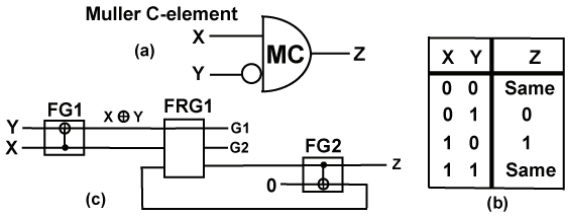

Figure 7. Reversible Muller C-element

The Capture command is controlled by the static Muller C-element whose symbol and behavior is shown in Fig. 7a and 7b, respectively. When both XY are '11' or '00', the Z output maintains its previous state. When XY changes to 01 or 10, Z captures the X value and causing event captured as discussed above. The reversible logic design of the Muller C-element is shown in Fig 7c. The quantum cost $C_{c\text{-}e}$ of the event controlled register is

$$C_{c\text{-}e} = 1*5 + 2*1 = 7 \quad (3)$$

4.2. Reversible Mousetrap Pipelines

The Mousetrap pipeline was recently designed by Singh and Nowick in order to reduce the complexity of the capture-pass protocol and to allow the use of simple latches for the event-controlled registers [19]. The Mousetrap pipeline architecture is reproduced in Fig. 8 indicating the replacement of the C-element by the NXOR gates and passing the request signals via the same latches of the event-controlled storage registers. Each stage must capture its current data before its predecessor is allowed to pass new data. This is accomplished by specifying a locally determined delay to the backward synchronization carried by the acknowledge chain.

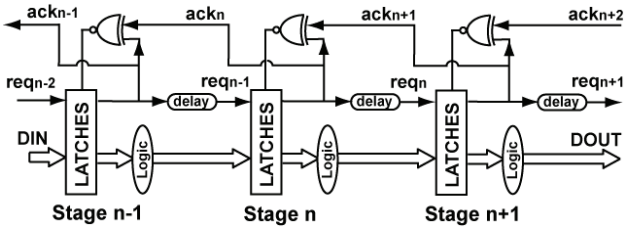

Figure 8. The Mousetrap pipeline architecture

The forward synchronization follows the request chain. Initially, all req_i and ack_i are at '0', so that the NXOR outputs are '1', keeping all latches transparent. As req_{n-2} and data propagate from left to right, the latched transition of req_{n-2} to '1' is fed (prior the delay that time the req_{n-1}) to the lower input of the XNOR gate of stage n-1. The XNOR output goes to '0' to capture the data in stage n-1. The delayed transition of the ack_n signal from the successor stage n toggles the XNOR of stage n-1back to '1', thus making the stage transparent again.

The reversible design of the Mousetrap uses reversible combinatorial logic circuits between the stages. The proposed reversible implementation of each Mousetrap stage is shown in Fig. 9. The first latch uses the FRG1 gate and a 3-copy cascade to perform the request chain forward synchronization with the NXOR gate. The quantum cost of each stage of k bits $C_{s(k)}$ is

$$C_{s(k)} = (5+2+3) + k * (5+1) \quad (4)$$

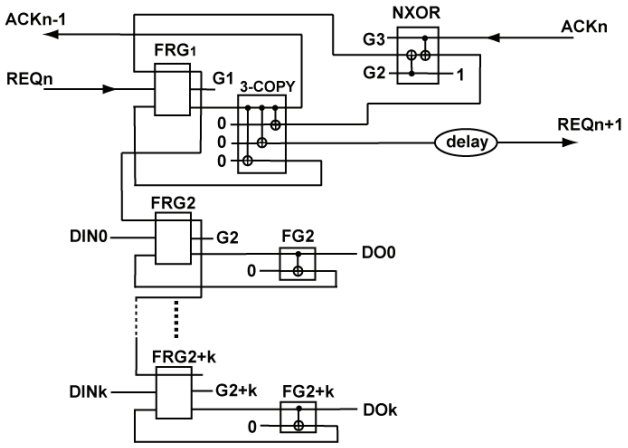

Figure 9. Reversible implementation of the Mousetrap stage

5. PHYSICAL REVERSIBILITY

In examining the symmetrical structure of the reversible pipelines in Section 4, one may consider whether the design is physically reversible (Definition 2). Rice performed a detailed investigation of the physical reversibility of her implementation of reversible D-latches, finding that (a) the reversed circuit does not perform the same functionality; and (b) that the ancillary inputs may not become garbage outputs when the circuit is operated in inverse mode (i.e. 'backwards') [28].

Investigating the implementations in Fig. 6,7 and 9 we also find similar observations as Rice. It is quite clear that none of the intended functionality is retained when the circuits are run backwards, and clearly some ancillary inputs need to be "disconnected" from the '0' state. Of course, complex circuits are never run in reverse. However, when we try to change the design so it does achieve some "useful" function when run in reverse, we encounter a major challenge unique to implementation of reversible electronic circuits – the implementation of bidirectional I/O. The challenge arises from the impossibility to implement tri-state logic (e.g. a transmission gate) in reversible logic.

We demonstrate this challenge in the *hypothetical* design of the physically reversible (bi-directional) D-latch of Fig. 10. When DIR='0', the latch works from left to right (D/Q is D, Q/D is Q). When we set the DIR' to '1' in order to run the circuit backward, the D/Q line becomes Q, and Q/D becomes D. The gates FRG1 and FRG4 are marked with a thick rectangle to indicate that they must be able to run backwards. FRG2 selects the D

input for the latch (FRG3 and the two attached Feynman gates). To achieve bi-directional operation, the latch Q must be sent to FRG4 or to FRG1, depending on the DIR control. In this hypothetical design, we need two transmission gates T1 and T2, which are marked with '??' to indicate that they violate the physical reversibility constraint of the circuit. This is a major obstacle for electronic reversible circuit design that must implement I/O operation. Another challenge is the dotted line from FRG4 to FRG2 which, depending on the technology (CMOS or quantum), may violate the electrical design rules during DIR='0', when the "floating" input of FRG2 is connected to a "floating" input of FRG4.

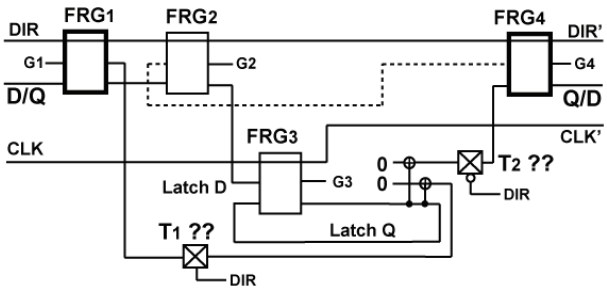

Figure 10. The physically reversible D-latch challenge

6. CONCLUSION AND FUTURE WORK

This paper examined recent developments in quantum finite states and reversible sequential circuits. We explored the benefits of the asynchronous design paradigm for reversible sequential design. We proposed new reversible implementation of key asynchronous circuits and demonstrated their use to construct relatively complex reversible asynchronous pipeline structures.

In future research we plan to adapt classical asynchronous design tools for reversible sequential circuits design. In particular, we will examine augmenting the UNCLE synthesis tool [32] for NCL logic [33] to support the ideas presented here. This work will require the study of using and implementing threshold logic gates with inherent hysteresis as opposed to the use of Muller C-elements as an atomic synchronizing element.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete log and factoring", In *Proc. of the 35th Annual Symposium on Foundations of Computer Science*, pp. 124-134, 1994.

[2] L. Grover, "A Fast Quantum Mechanical Algorithm for Database Search", In *Proc. ACM Symp. on Theory of Computing*, pp. 212-219, 1996.

[3] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," In *IBM J. Res. and Dev.*, vol. 3, pp. 183-191, 1961.

[4] A. Kondacs and J. Watrous, "On the Power of Quantum Finite State Automata", In *IEEE Symposium on Foundations of Computer Science*, pp. 66-75, 1997.

[5] M. Lukac and M. Perkowski, "Quantum Finite State Machines as Sequential Quantum Circuits", In *ISMVL'09*, pp. 92-97, 2009.

[6] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits", In *ICCAD'02*, pp. 353-360.

[7] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis", In *DAC'03*, pp. 318-323.

[8] P. Kerntopf, "A new heuristic algorithm for reversible logic synthesis", In *DAC'04*, pp. 835-837, 2004.

[9] R. Wille and D. Groβe, "Fast exact Toffoli network synthesis of reversible logic", *ICCAD'07*, pp. 60-63, 2007.

[10] L.K. Chang and F. –C. Cheng, "Automatic Synthesis of Composable Sequential Quantum Boolean Circuits", In *ICCD'05*.

[11] J. E. Rice, "The State of Reversible Sequential Logic Synthesis", Technical Report TR-CSJR2-2005, University of Lethbridge, 2005.

[12] H. Thapliyal et al., "A Beginning in the Reversible Logic Synthesis of Sequential Circuits," In *MAPLD'05*.

[13] J. E. Rice, "A New Look at Reversible Memory Elements," In *ISCAS'06*.

[14] H. Thapliyal and M. Zwolinski, "Reversible Logic to Cryptographic Hardware: A New Paradigm," In *MWSCAS'06,* pp. 342-346. 2006.

[15] N.M. Nayeem et al., "Efficient Design of Shift Registers Using Reversible Logic",In *ICSPS'09*, pp 474-477, 2009.

[16] M.H.A. Khan and M. Perkowski, "Synthesis of Reversible Synchronous Counters", In *ISMVL 2011*, pp 242-247, 2011.

[17] M. Lukac, Ben Shuai, M. Kameyama, and D.M. Miller, "Information-Preserving Logic Based on Logical Reversibility to Reduce the Memory Data Transfer Bottleneck and Heat Dissipation", In *ISMVL'11*, pp. 131-138, 2011.

[18] D. Deutsch, "Quantum Theory, the Church-Turing principle and the universal quantum computer," In *Proc. Royal Society London Series A*, 400, 1985, pp. 96-117.

[19] I.E. Sutherland, "Micropipelines", In *Comm. ACM*, vol. 32, no.6, pp. 720-738, 1989.

[20] M. Singh and S.M. Nowick, "Mousetrap: High-Speed Transition-Signaling Asynchronous Pipelines", In *IEEE Trans. VLSI,* vol. 15, no.6, pp. 684-698, 2007.

[21] D.E. Muller and W.S. Bartky, "A Theory of Asynchronous Circuits," In *Proc. Int. Symp. on Theory of Switching*, vol. 29, pp. 204-243, 1959.

[22] A.Barenco et al., "Elementary gates for quantum computations", In *Phys. Rev. A*., Vol. 52, no. 5, pp. 3457-3467, 1995.

[23] E. Fredkin and T. Toffoli, "Conservative Logic," In *Int. J. Theoretical Physics*, vol. 21, nos. 3/4, pp. 219-253, 1982.

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

[25] M.R. Dunlavey, "Simulation of finite state machines in a quantum computer",1998, arXiv:quant-ph/9807026v1.

[26] T. Toffoloi, "Reversible Computing," In *Automata, Languages, and Programming*, Springer Verlag, pp. 632-644, 1980.

[27] D.E. Muller, "Asynchronous logics and application to information processing", In *Proc Symp on Application of Switching Theory in Space Technology*, pp 289-297, 1963.

[28] J.E. Rice, "Introduction to Reversible Latches", *The Computer Journal,* Vol. 51 No. 6, 2008 pp. 700-709.

[29] J. Smolin and D.P. DiVincenzo, "Five Two-Qubit Gates are Sufficient to Implement the Quantum Fredkin Gate", In *Physical Review A,* vol 53, no. 4, pp 2855-2856, 1996.

[30] P. Picton, "Multi-Valued Sequential Logic Design using Fredkin Gates," Multiple-Valued Logic Journal vol. 1, pp. 241–251, 1996.

[31] J. Sparso, *Asynchronous Circuit Design: A Tutorial*, 2006, Available online.

[32] R.B. Reese, UNCLE (Unified NCL Environment), Technical Report MSU-ECE-10-001 November 2010, available online: http://www.ece.msstate.edu/~reese/uncle/UNCLE.pdf,

[33] S. Smith and J. Di, *Designing Asynchronous Circuits Using the NULL Convention Logic (NCL)*, Morgan & Claypool Publishers, 2009.