Advance Access publication on January 17, 2008 doi:10.1093/comjnl/bxm116

An Introduction to Reversible Latches

J. E. RICE*

Department of Mathematics & Computer Science, University of Lethbridge, Lethbridge, AB, Canada
**Corresponding author: j.rice@uleth.ca*

Reversible logic has been suggested as one solution to the problem of power consumption in today's electronic devices. This paper addresses the issue of designing reversible latches and provides an overview and analysis of some proposed designs.

Keywords: Boolean logic circuits; reversible logic; sequential circuits

Received 11 April 2007; revised 1 October 2007

1. INTRODUCTION

Many of us have likely experienced the frustration of one's cell phone battery dying, or running low on battery power for your laptop when there is no plug-in available. Electronic devices have become portable, and are intended to be taken everywhere. The problem is that there is not always a source of electricity available, and so the demand for devices that use less power, so that batteries last longer, is quickly growing. Some researchers believe that the well-known Moore's law is at an end due to the inability for us to deal with the power-requirements of future chips [1]. For decades now the electronics industry has been able to produce devices that are smaller, faster and use less power year after year. Now, however, we may have hit a brick wall, and incremental improvements will bring only small advances.

An entirely new paradigm may provide the solution. The use of *reversible logic* in building chips may provide a solution. As stated by Frank [2],

> ... computers based mainly on reversible logic operations can reuse a fraction of the signal energy that theoretically can approach arbitrarily near to 100% as the quality of the hardware is improved...

He and many other researchers believe that we are coming close to having technologies that will support reversible computing, and that reversible devices will have far lower power requirements than do traditional devices. Additionally, a most prominent application of reversible logic lies in quantum computers, and in fact reversible logic has been called "quantum computing's practical cousin" [3]. As Thapliyal *et al.* describe, a quantum computer will be viewed as a quantum network composed of quantum logic gates [4]. Each gate will be designed to perform some elementary operation on quantum systems called qubits. Each qubit represents an elementary unit of information that can be thought of as corresponding to the traditional bit values of 0 and 1. These operations must be unitary, and any unitary operation is reversible. However, traditional designs for arithmetic operations such as addition and multiplication make heavy use of irreversible logic gates such as AND and OR. Thus, quantum arithmetic must be built from reversible logical components.

The idea of reversible computing was discussed many years ago by Landauer [5] and Bennett [6], and further refined by Toffoli [7]. Recent work has focused heavily on synthesis techniques for reversible logic. Partly because fan-outs are not permitted in reversible logic circuits, the synthesis process for reversible logic differs significantly from that of traditional irreversible logic. Research in this area is following many directions, some of which are detailed in [8–15], and implementations have been suggested by [16–19], to name a few. There is, however, little mention of sequential logic in a reversible context, and it is this area that this work addresses.

We first address the issues that some researchers raise regarding the notion of sequential logic in a reversible context. For instance, it can be argued that the very model of quantum computing is so different from that of traditional computing that the state machine model upon which sequential circuits have been based will be made obsolete. Another problem that has been raised is that in quantum computation models data are stored in qubits, for which there is no known method of implementing feedback. There are, however, other possible models that reversible logic can be based upon, although we leave that particular investigation for other work. To further refute the claim that sequential

reversible networks are not possible, we refer to Frank's 2005 paper entitled 'Approaching the Physical Limits of Computing'. In this work Frank states that

> The true requirement that must be imposed on the logical functionality of a reversible logic gate is simply this: that for each distinct operation that the gate can be directed to perform, no two initial logical states ... that can possibly arise in the normal course of the machine's operation ... can be transformed to the same final state [1].

Given this, Frank goes on to describe a number of possible gate behaviours, including reversible SET and conditional reversible SET gates. The latter of these requires knowledge of the initial condition of the state, much like a traditional latch. Thus we argue that it is not the concept of a reversible sequential network that is in question, but simply the implementation; that is, the target technology. At this juncture implementing technologies for reversible logic are still in the realm of the unknown, and so we choose not to limit ourselves to those few suggestions that have been proposed in this area.

It may indeed be the case that the future models of computing evolve to be very different from the traditional state machine-based models that we rely upon today; however, we believe that even if this is the case then during an industry switch from traditional computing models to new models a transition period will be necessary. Work such as this may fill in the gaps in our knowledge and provide a smoother transition during such a period.

The small amount of previous work in the area of synthesis for sequential reversible logic includes [4, 20–24]. Many of these papers suggest designs for reversible memory elements, and in some cases the design investigated is that of a basic latch, as in this paper. In this work, we review and analyse the simplest of the proposed designs, the SR latch.

2. BACKGROUND

2.1. Reversible logic

We first present some basic concepts underlying the idea of reversible logic. According to Shende *et al.*, the following definition holds.

DEFINITION 2.1. *A gate is reversible if the (Boolean) function it computes is bijective* [13].

In other words, a function is reversible if there is a one-to-one and on-to mapping from the inputs to the outputs (and vice versa) of the function. At the very least, a reversible function must have the same number of inputs as it does outputs. For instance, the traditional NOT gate is reversible, but the traditional AND gate is not, as shown in Table 1. We can see how the NOT gate is bijective; for each value of x there is a unique value for $\bar{x}$ and every value of $\bar{x}$ results from applying the function to x. However, the AND gate is not bijective as there are two values for yz that result in the same output, and so it is not injective and cannot be bijective.

TABLE 1. The truth tables for the traditional NOT and AND gates.

x	$\bar{x}$	y	z	$y \cdot z$
0	1	0	0	0
1	0	0	1	0
		1	0	0
		1	1	1

TABLE 2. The behaviour of a selection of more commonly used reversible logic gates.

Gate	Behaviour
Not	$(x) \rightarrow (x \oplus 1)$
Feynman	$(x, y) \rightarrow (x, x \oplus y)$
Toffoli	$(x, y, z) \rightarrow (x, y, xy \oplus z)$
swap	$(x, y) \rightarrow (y, x)$
Fredkin	$(x, y, z) \rightarrow (x, z, y)$ *iff* $x = 1$

There are, however, a variety of possibilities for Boolean reversible functions, even limiting ourselves to a few inputs (e.g. three). Table 2 lists the behaviour of each of the most commonly used reversible gates. The behaviour describes how each input becomes transformed to produce the outputs when the gate is applied. The swap and Not gates are the very simplest, with the Feynman, Toffoli and Fredkin in essence extending these basic gates to incorporate additional inputs. This work concentrates primarily on latches designed from Fredkin and Toffoli gates, and the symbols used for these gates are shown in Fig. 1. The inputs and outputs in Fig. 1 are labelled in such a way to provide a reminder of the behaviour of the gates. Thus in Fig. 1(A), we see that the topmost signal passes through the gate unchanged, as indicated by the fact that it is labelled as x on both sides of the gate. The same holds true for the next signal, y, while the third signal is affected by the gate as described by the equation given on the right side of the gate. For instance, if the input values for x, y and z are 110 then the output values will be x, y, $xy \oplus z$ or 111 after the Toffoli gate is applied.

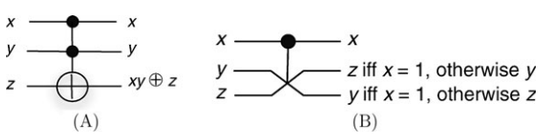

FIGURE 1. (A) The symbol for the Toffoli gate. (B) The symbol for the Fredkin gate.

2.2. Latches

The current model of computing is based on the concept of a state machine. Such a machine exhibits different behaviours depending on the value of its current state. Thus, some structure is required to keep track of this state. The basic building block used in today's computer is the flip-flop, which themselves are built out of latches. There are a variety of types of latches, but the basic type upon which other designs are generally modified is the Set-Reset, or SR latch. A SR latch allows the outputs, labelled Q^+ and $\overline{Q^+}$, to either be 'set', in which case the next state values are $Q^+ = 1$ and $\overline{Q^+} = 0$ or 'reset', in which case $Q^+ = 0$ and $\overline{Q^+} = 1$. The primary inputs of such a structure are S (set) and R (reset). A diagram illustrating a NOR-gate implementation of a SR latch is given in Fig. 2(A).

A variety of reversible latches have been introduced in previous work. This paper examines the behaviour of the latches presented in [22]. These reversible latches have been designed to emulate a traditional SR latch. The behaviour of the SR latch is characterized by the truth table given in Table 3.

2.3. Previous work

A number of researchers including Toffoli [7] and Frank [1] discuss the potential for sequential reversible logic, but do not present any structures for its realization. Fredkin and Toffoli [25] appear to be the first to suggest a conservative logic sequential element in the form of a J$\bar{K}$ flip-flop, and Picton [23] suggests a reversible SR latch. He uses the basic Fredkin gate to build this latch, as shown in Fig. 2(B). This work also concentrates on the use of a basic memory element such as the SR latch, as it is the traditional building block for the clocked flip-flop structures that are our eventual goal.

3. REVERSIBLE LATCHES

In this work we will introduce, characterize and analyse the behaviour of two reversible latch designs: the Toffoli-based latch shown in Fig. 3(A) and the Fredkin-based latch shown in Fig. 3(B). As noted in the section above, Picton was the first to introduce a reversible latch based on Fredkin gates. However, if we examine the illustration of his latch in Fig. 2 we see that, like the traditional latch, the design requires fan-out from the signals for Q^+ and $\overline{Q^+}$. The problem with Picton's model is that the concept of reversible logic is predicated on the fact that not only can one not allow the destruction of data (e.g. a signal value), as in the situation with the traditional AND gate, but one can not allow the arbitrary creation of data. This means that fan-out is not permitted. However, the problem can be corrected by making use of one of the garbage outputs, and in fact the design in Fig. 3 (B) does exactly this.

TABLE 3. The next state values for the SR latch.

Inputs $S\ R$	Next state $Q^+\ \overline{Q^+}$
0 0	$Q\ \bar{Q}$
0 1	0 1
1 0	1 0
1 1	not permitted

Thapliyal, Srinivas and Zwolinski [4] have also published work in the area of sequential reversible logic, and have introduced their own constructions of basic memory elements such as flip-flops. However, their designs are ultimately different from those presented in this work, although possibly useful for comparison purposes.

3.1. The Fredkin-based SR latch

The Fredkin-based SR latch was developed by using Picton's original latch as a reference, and addressing the problem of fan-out in his design. As indicated above, the solution suggested here is to make use of one of the unused outputs of the Fredkin gate for one of these purposes. The modified design is as shown in Fig. 3(A). Table 4 shows that this design does, in fact, satisfy the requirements for the behaviour of a SR latch. There are 16 possible starting values of the

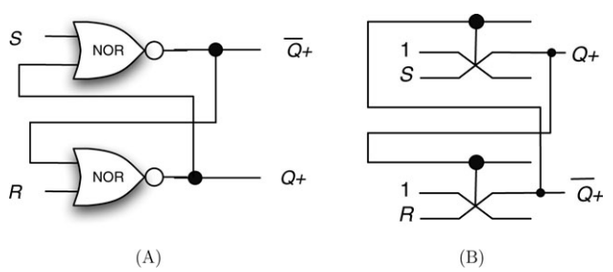

FIGURE 2. The SR latch (A) built from traditional logic gates and (B) built from two Fredkin gates as designed by Picton [23].

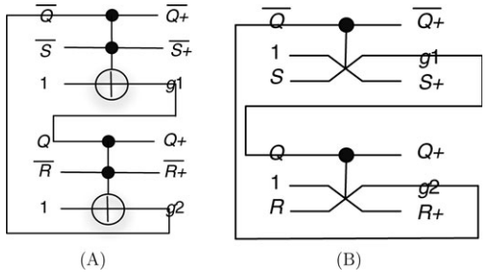

FIGURE 3. (A) SR latch based on Toffoli gates. (B) SR latach based on Fredkin gates.

TABLE 4. Behaviour of the Toffoli-based SR latch.

	Initial values $\overline{S}$R 11 $Q\overline{Q}$	After gate processing S^+R^+ g_1 g_2 $Q^+\overline{Q}^+$	New input values $\overline{S}$R 11 $Q\overline{Q}$
0	00 11 00	00 11 00	00 11 11
1	00 11 01	00 01 01	00 11 01
2	00 11 10	01 10 10	00 11 10
3	00 11 11	11 00 11	00 11 00
4	01 11 00	01 11 00	01 11 11
5	01 11 01	11 01 01	01 11 01
6	01 11 10	01 11 10	01 11 11
7	01 11 11	11 01 11	01 11 01
8	10 11 00	10 11 00	10 11 11
9	10 11 01	10 11 01	10 11 11
10	10 11 10	11 10 10	10 11 10
11	10 11 11	11 10 11	10 11 10
12	11 11 00	11 11 00	11 11 11
13	11 11 01	11 11 01	11 11 11
14	11 11 10	11 11 10	11 11 11
15	11 11 11	11 11 11	11 11 11

inputs to the Fredkin latch, as shown in Table 4. For some input combinations there may be a delay while the gates process the initial values, then the new values are propagated back to the inputs and another processing step must take place. Additionally, some input combinations will cause the latch to oscillate between two non-meaningful states. We are most interested in ascertaining that the latch carries out the behaviour detailed in the characteristic table of the SR latch (Table 3).

Let us first examine what occurs when S and R are both set to 0. What should happen is that the current state of the latch (that is, the current values for Q and $\bar{Q}$) is reflected in the next state (that is, the values for Q^+ and $\overline{Q^+}$). It appears that this does occur for all possible initial values of Q and $\bar{Q}$ if we simply look at the values after gate processing for Q^+ and $\overline{Q^+}$. Actually, due to the nature of the Fredkin gate, Q^+ will always reflect the unchanged value of Q, and similarly for $\bar{Q}$ and $\overline{Q^+}$. It is more important to look at the new input values, as this tells us whether the latch will be stable at its current values, or if the values will change after the next gate processing step. We see in lines 1 and 2 of Table 4 that the new input values for Q and $\bar{Q}$ are the same as the original values, and this tells us two things: that the latch will be stable, or remain unchanged after the next processing step, and that the behaviour is correct in that the current values of Q and $\bar{Q}$ are stored unchanged in the latch. However, in lines 0 and 3 we see that the new input values for Q and $\bar{Q}$ are not the same as the original values. In fact, if the latch somehow gets into a non-meaningful state where $Q = \bar{Q}$ and we set S = R = 0 the latch will oscillate between $Q\bar{Q} = 00$ and $Q\bar{Q} = 11$. This does not strictly match the description in the characteristic table, but we assume that this will not matter since the latch should never be in this type of non-meaningful state.

Lines 4–7 of the table allow us to verify the correct 'reset' behaviour for the latch; that is, when $SR = 01$ the latch should have values of $Q^+\overline{Q^+} = 01$, and Table 4 shows that this does happen, eventually, for all possible initial values of Q and $\bar{Q}$. Similarly lines 8–11 verify the 'set' behaviour of the latch. Lines 12–15 are not particularly useful in that they show what will occur when S and R are both set to 1, which is in fact, a combination that should not be permitted.

In general, however, Table 4 shows that for all 'permitted' combinations for S and R and all meaningful initial values of Q and $\bar{Q}$ the latch will eventually reach the desired stable state.

3.2. The Toffoli-based SR latch

One approach to determining reversible memory elements is to take existing memory elements, built from traditional logic, and replace the traditional components with reversible components. For instance, the NOR gate may be used in the design of the SR latch, as is shown in Fig. 2(A). The NOR gate is clearly not reversible; one problem is that there is only one output and two inputs. Table 5(A) shows one possible way to create a reversible equivalent of the NOR gate's behaviour. Note that for our use of this gate we will always set the z input to 0, and so we have shown the four inputs with $z = 0$ at the top of the truth table. Since we now have a reversible NOR gate we can create a reversible latch out of it, as shown in Fig. 4(A). However we have, again run into the problem of fan-out, and so Fig. 4(B) shows a reversible latch based on the reversible NOR gate without fan-out. The reversible latch shown in Fig. 4 is perfectly acceptable, however, it turns out that a small modification to the reversible NOR gate—inverting the x and y inputs—will turn it into a Toffoli gate, and so the Toffoli-based latch shown in Fig. 3(A) is, in fact, the equivalent of this 'revnor'-based latch.

Having thus shown the development of the Toffoli-based SR latch we will now turn to verifying its behaviour. As for

TABLE 5. Truth tables for (A) the traditional, irreversible NOR gate, and (B) a reversible equivalent.

xy	$\overline{x+y}$	xyz	xy	$(\overline{x+y})\oplus z$
00	1	000	00	1
01	0	010	01	0
10	0	100	10	0
11	0	110	11	0
		001	00	0
		011	01	1
		101	10	1
		111	11	1

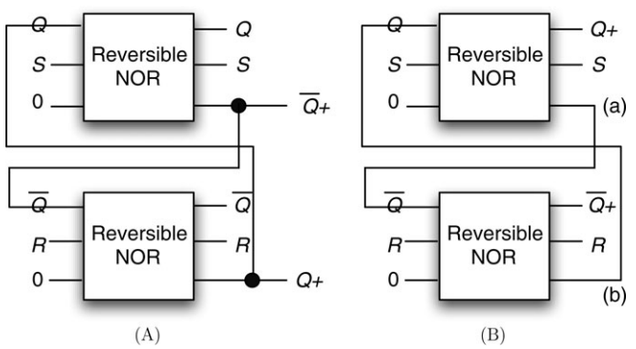

FIGURE 4. (A) A reversible latch based on the reversible NOR gate, and (B) the same latch with fan-out removed.

the Fredkin-based latch, there are 16 possible combinations. These are shown in Table 6. The reader will note that the ordering in which the inputs are given does not match that of Table 4, and that is because negated input values for S and R are required. The ordering shown for each of Tables 4 and 6 gives results for keeping the latch unchanged in lines 0–3, the reset operation in lines 4–7, and the reset operation in lines 8–11. We can see that the behaviour for the Toffoli latch does match that of the Fredkin latch as well as meeting the requirements for a SR latch. As we saw for the Fredkin latch, some input combinations result in the desired behaviour immediately after the gates process the inputs, while others require an additional gate-processing step before the desired behaviour is seen. Again, some input combinations cause the latch to oscillate between two non-meaningful states as we see in lines 0 and 3.

TABLE 6. Behaviour of the Toffoli-based SR latch.

	Initial values $\overline{SR}$ 11 $Q\overline{Q}$	After gate processing $\bar{S}^+\bar{R}^+$ g_1 g_2 $Q^+\overline{Q}^+$	New input values $\overline{SR}$ 11 $Q\bar{Q}$
0	11 11 00	11 11 00	00 11 11
1	11 11 01	11 01 01	00 11 01
2	11 11 10	11 10 10	00 11 10
3	11 11 11	11 00 11	00 11 00
4	10 11 00	10 11 00	10 11 11
5	10 11 01	10 01 01	10 11 01
6	10 11 10	10 11 10	10 11 11
7	10 11 11	10 01 11	10 11 01
8	01 11 00	01 11 00	01 11 11
9	01 11 01	01 11 01	01 11 11
10	01 11 10	01 10 10	01 11 10
11	01 11 11	01 10 11	01 11 10
12	00 11 00	00 11 00	00 11 11
13	00 11 01	00 11 01	00 11 11
14	00 11 10	00 11 10	00 11 11
15	00 11 11	00 11 11	00 11 11

3.3. A short comparison of the Fredkin and Toffoli designs

Given that we have three possible reversible designs for a SR latch, how might one choose between them? Two criteria come to mind in making a selection: (1) behaviour and (2) speed.

3.3.1. Behaviour comparison

All three of the suggested latches behave in similar ways:

- the latches store values of $Q\bar{Q} = 01$ and $Q\bar{Q} = 10$, but become unstable if required to store non-meaningful values such as $Q\bar{Q} = 00$ or $Q\bar{Q} = 11$,
- the latches will all reset or set the latch appropriately for any given starting values for Q and $\bar{Q}$, and
- the latches will all result in some stable but non-meaningful state when S = R = 1.

The only difference is that when S = R = 1 the resulting values for Q^+ and $\overline{Q^+}$ are 11 for the Toffoli latch and 00 for both the revnor-based latch and the Fredkin latch.

3.3.2. Timing comparison

As indicated above, there is in some cases a delay before the desired results become evident. For instance, for the input combination $SRQ\bar{Q} = 0100$, the immediate result for the Fredkin latch is that $Q^+\overline{Q^+}$ is set to 00. However, after the new output values are propagated back to the inputs and an additional gate processing step takes place, then the desired values of $Q^+\overline{Q^+} = 01$ are seen.

Interestingly enough, it turns out that both the Toffoli and the Fredkin-based latches behave in very similar ways; that is, an extra step of processing is required for the same starting values for S, R, Q and $\bar{Q}$. The revnor-based latch also requires additional processing steps, but for differing starting values. However, the number of combinations requiring additional processing steps is the same.

In general, there are two combinations when setting the latch which will require additional delay, and two combinations when resetting the latch. This would have to be factored in to any clocked design, such as a flip-flop, that was to be built from these latches.

3.3.3. Overall comparison

The behaviour comparison suggests very little to choose from amongst the three latches.

In terms of design, factors which impact area would include the number of gates required and the number of garbage lines. The revnor-based latch in Fig. 4(B) requires two gates and two garbage lines, as does the Toffoli-based design in Fig. 3(A) and the Fredkin-based design in Fig. 3(B).

As discussed above, each design has appropriate behaviour, the only difference being the state achieved when the values for S and R are both set to 1. This information may be of use to a designer who has need of this type of behaviour

but, in general, does not give us any concrete choice between the latch designs.

The timing comparisons suggest that there may be reasons to choose one latch over another. Each latch requires an additional step processing certain inputs, but the inputs requiring extra processing time differ amongst the latches. Again, in general usage where all possible input combinations are equally likely, this is not particularly useful, but for specific designs a designer should, and could, with this information, choose the latch that will require the least number of processing steps for the input combinations most likely to be used.

In general, we cannot state that one of our designs is always a better choice than the others, but we can provide information that for particular applications may suggest the choice of one design over the other two.

3.4. Less successful latch designs

It has been suggested that to avoid fan-out, instead of modifying the Picton latch in Fig. 2 as we have done in this work, use of additional gates to provide the fan-out could be used. Fig. 5 shows this design. This is certainly an option, however, this would introduce an additional set of gates and thus additional processing delay.

4. OTHER CONSIDERATIONS WITH REVERSIBLE LATCHES

The concept of reversible logic implies, in its very name, that the function described by the logic gates can be applied in reverse. This is clearly true for the NOR gate and the other reversible gates described in Table 2; whether the signals on the left side *or* the right side of the gates are considered 'inputs', the functionality of the gate is well defined and can be operated or 'driven' in either direction. However, in traditional logic this is not the case; gates and circuits operate in one direction, with possible feedback in the case of sequential logic. Thus there are two questions to consider with the reversible latch designs presented in this work: are they truly reversible, and if so is it possible to characterize their behaviour when driven in reverse?

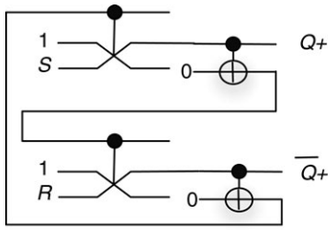

FIGURE 5. An alternate solution to the fan-out problem in Picton's latch.

To answer the first question we refer to Toffoli's report [7], in which he states

> By definition, a finite automaton is reversible if its transition function is invertible. Thus in order to realize a finite automaton by means of a reversible sequential network, it will be sufficient to take its transition function, construct a reversible realization of it, and use this as the combinational part of the desired sequential network.

In examining the combinational part of our sequential design, we should point out that the most commonly used reversible gates exhibit the following characteristic: if X is a particular combination of input values, and f is the function carried out by the gate, then

$$f(f(I)) = I.$$

However, this characteristic is *not* a requirement for a reversible gate, and indeed our SR latch designs do not have this behaviour. However, for all input values on either the left- or right-hand of the latch designs (i.e. being driven forwards or in reverse) both designs show the characteristics required of reversible gates: they are surjective (onto) and injective (1 to 1).

So, since both of our designs behave as SR latches when driven from left to right, and both satisfy certain characteristics that make the structures reversible, can we then use either latch as a (SR) latch when driving them from right to left (in reverse)? If this is the case then this suggests an opportunity that no other work has identified: the opportunity to design circuits whose behaviour will be known whether driven from left to right or right to left. In effect, we may be able to design one circuit that can carry out two actions: one when driven in one direction and one when driven in reverse.

4.1. Fredkin latch

Since we have two different proposals for latch structures we will examine each design separately. In the Fredkin latch, when operating the latch in reverse, or 'driving' it from right to left we can control inputs $\overline{Q^+}$, S^+, Q^+ and R^+ while g_1 and g_2 are connected to Q and $\bar{Q}$. Thus the goal is to have some way to set, reset and hold the latch as described in Table 3 using only these inputs to control the latch.

There are a number of possibilities to consider, but using the forward behaviour of the latch as a model it seems reasonable to consider holding two of the inputs at fixed values and operate with the other two controlling the latch. For instance, we could hold Q^+ and $\overline{Q^+}$ at fixed values and operate with S^+ and R^+ controlling the latch, or hold S^+ and R^+ at fixed values and use Q^+ and $\overline{Q^+}$ to control the latch. If we fix $Q^+\overline{Q^+}$ each at 1 then the resulting values are as shown in Table 7.

TABLE 7. An example of driving the Fredkin-based latch backwards with Q^+ and $\overline{Q^+}$ both fixed at 1.

	Initial values S^+R^+ $Q^+\overline{Q^+}$ g_1 g_2	After gate processing 1_a1_bSR	New input values S^+R^+ $Q^+\overline{Q^+}$ g_1 g_2
0	00 11 00	0000	00 11 11
1	00 11 01	0001	00 11 11
2	00 11 10	0010	00 11 11
3	00 11 11	0011	00 11 11
4	01 11 00	0100	01 11 11
5	01 11 01	0101	01 11 11
6	01 11 10	0110	01 11 11
7	01 11 11	0111	01 11 11
8	10 11 00	1000	10 11 11
9	10 11 01	1001	10 11 11
10	10 11 10	1001	10 11 11
11	10 11 11	1011	10 11 11
12	11 11 00	1100	11 11 11
13	11 11 01	1101	11 11 11
14	11 11 10	1110	11 11 11
15	11 11 11	1111	11 11 11

To distinguish between the two left-hand inputs labelled with inputs values of 1 in Fig. 3 we refer to the top input as 1_a and the bottom input as 1_b. All other inputs are referred to by the labels used in Fig. 3.

As in previous tables, Table 7 gives the initial input values and the values that will result immediately after gate processing, as well as the new input values that will result after Q and $\bar{Q}$ have had a chance to propagate back to the inputs. Unfortunately our intermediate output values are not all that useful, as we can see that whatever initial values g_1 and g_2 may have had will always be overwritten by the values we set on Q^+ and $\overline{Q^+}$. Thus lines 3, 7, 11 and 15 show the behaviours that will actually take place when the values stabilize. What we see is that fixing Q^+ and $\overline{Q}^+$ at 11 forces a swap of g_1 with S^+ and of g_2 with R^+, resulting in 1_a reflecting the value(s) put on S^+ and 1_b reflecting the value(s) put on R^+. This behaviour does not match that of a SR latch; however, there are other latches to consider. The D latch, for instance, is characterized by the behaviour shown in Table 8. The behaviour shown above is very similar to that of the D latch, with the notable difference being that we control our latch with two inputs while the traditional D latch only has one. Traditionally, however, the D latch is constructed from a SR latch for which there is only one input; S is connected to the unmodified input signal while R is connected to that same signal inverted as shown in Fig. 6. Thus the Fredkin-based SR latch could in fact be used as a D latch when functioning in reverse. In fact, there are other configurations in which the proposed SR latch structure could be useful when used in reverse. For instance, if we set $Q^+\overline{Q^+} = 00$ then we get $\mathrm{S} = S^+$ and $\mathrm{R} = R^+$, or if we set $S^+ = R^+ = 11$ then we get behaviour as shown in Table 9. Obviously Table 9 does not show all possible starting combinations for Q^+, $\overline{Q^+}$, g_1 and g_2, but since the values for Q and $\bar{Q}$ will always get propagated around to g_1 and g_2 respectively, the initial values for g_1 and g_2 get overwritten and thus do not affect the final values.

TABLE 8. The characteristic table for a D latch.

D	$Q^+\overline{Q^+}$
0	01
1	10

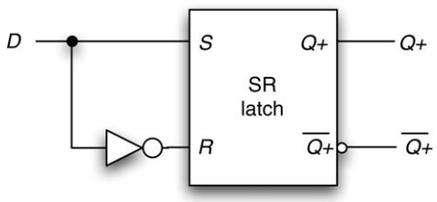

FIGURE 6. Constructing a D latch from a SR latch.

We can take this analysis in another direction and instead look at the equations for the outputs of each of the gates. If the inputs for a Fredkin gate are labelled x, y and z, and the outputs are labelled x', y' and z', then the outputs can be characterized (using traditional, irreversible AND and OR gates) as follows:

$$x' = x$$
$$y' = \bar{x}y + xz$$
$$z' = xy + \bar{x}z$$

Thus in our Fredkin latch we have

$$\bar{Q} = \overline{Q^+}$$
$$1_a = \overline{(\overline{Q^+})}g_1 + \overline{Q^+}S^+$$
$$\mathrm{S} = \overline{Q^+}g_1 + \overline{(\overline{Q^+})}S^+$$

TABLE 9. The reverse behaviour of the Fredkin SR latch when S^+ and R^+ are fixed at 1.

$Q^+\overline{Q^+}$	g_1 g_2	1_a1_bSR
00	−−	0011
01	−−	1101
10	−−	1110
11	−−	1111

TABLE 10. A complete characterization of the reverse behaviour of the Fredkin-based SR latch.

$Q^+\overline{Q^+}$	1_a	1_b	S	R
00	0	0	S^+	R^+
01	S^+	1	0	0
10	1	R^+	S^+	R^+
11	S^+	R^+	1	1

and

$$Q = Q^+$$
$$1_b = \overline{(Q^+)}g_2 + Q^+R^+$$
$$R = Q^+g_2 + \overline{(Q^+)}R^+.$$

Since we know that $g_1 = Q^+$ and $g_2 = \overline{Q^+}$ we can use this to derive Table 10, which completely characterizes the behaviour of the Fredkin-based SR latch when operated in reverse.

4.2. Toffoli latch

We can carry out a similar investigation with the Toffoli latch. Table 11 shows the behaviour that occurs if we set both Q^+ and $\overline{Q}^+$ to 1 and drive the latch in reverse. As shown in Table 11 one option for the reverse behaviour of the Toffoli latch, again we have the situation where Q^+ $\overline{Q^+}$ overwrite any initial values on g_1 and g_2, and so lines 3, 7, 11 and 15 show the behaviour when the latch stabilizes to its final values.

TABLE 11. An example of driving the Toffoli-based latch backwards with Q^+ and $\overline{Q^+}$ both fixed at 1.

	Initial values $\overline{S^+R^+}$ $Q^+\overline{Q^+}$ g_1 g_2	After gate processing $1_a1_b\overline{\mathrm{SR}}$	New input values $\overline{S^+R^+}$ $Q^+\overline{Q^+}$ g_1 g_2
0	00 11 00	0000	00 11 11
1	00 11 01	0100	00 11 11
2	00 11 10	1000	00 11 11
3	00 11 11	1100	00 11 11
4	01 11 00	0101	01 11 11
5	01 11 01	0001	01 11 11
6	01 11 10	1101	01 11 11
7	01 11 11	1001	01 11 11
8	10 11 00	1010	10 11 11
9	10 11 01	1110	10 11 11
10	10 11 10	0010	10 11 11
11	10 11 11	0110	10 11 11
12	11 11 00	1111	11 11 11
13	11 11 01	1011	11 11 11
14	11 11 10	0111	11 11 11
15	11 11 11	0011	11 11 11

As for the Fredkin latch, we do not have the exact behaviour of a SR latch. The choice of fixing Q^+ and $\overline{Q^+}$ to 1 in the Fredkin latch was made in order to force a swap to occur; however, the Toffoli latch has a very different behaviour, and so a more general characterization is probably more useful. Assuming that the inputs of the Toffoli gate are labelled x, y and z and the outputs are labelled x', y' and z' then we have

$$x' = x$$
$$y' = y$$
$$z' = xy \oplus z.$$

For the Toffoli-based latch, then we have

$$\overline{Q} = \overline{Q^+}$$
$$\overline{S} = \overline{S^+}$$
$$1_a = \overline{Q^+S^+} \oplus g_1$$

and

$$Q = Q^+$$
$$\overline{R} = \overline{R^+}$$
$$1_b = Q^+\overline{R^+} \oplus g_2.$$

Given that we know $g_1 = Q^+$ and $g_2 = \overline{Q^+}$ we can thus derive the characterization given in Table 12.

5. COMPARISON TO OTHER WORK

We briefly comment on how our designs compare to those presented in the other literature on this topic.

As mentioned in Section 3, Picton suggested the use of two Fredkin gates to build a SR latch, and in fact, we suggest a design in Fig. 5 that introduces two extra gates to remove the fan-out problem from Picton's latch. Many comparisons are often based on numbers of gates and/or on garbage lines

TABLE 12. The complete characterization of the reverse behaviour of the Toffoli-based SR latch.

$\overline{Q^+}Q^+$	1_a	1_b
00	0	0
01	1	$\overline{R^+}$
10	$\overline{S^+}$	1
11	$\overline{S^+}$	$\overline{R^+}$

required. Clearly in this case the design in Fig. 5 compares poorly to Picton's latch, requiring an extra two gates and an extra two garbage lines. Our other designs, in Figs. 3 and 4(B), however, require the same number of gates and the same number of lines as do Picton's latch, and additionally do not have the fan-out problem inherent to Picton's design.

Chuang and Wang also suggest latch designs in [20]: a D latch, JK latch and a T latch. They go on to then use these designs in the development of more sophisticated sequential building blocks. The pros and cons of these various types of latches can be considered separately from the question of reversibility; however, in general, we note that Chuang and Wang's designs require 4 gates for the more complex latch (JK latch) and 2 gates for the simpler latches (T and D latches). Their designs in general need only one garbage line for all latches, but do not incorporate both Q and $\bar{Q}$ as do our designs (which may be considered a pro or a con, depending on the needs of the designer). Given that both T and D latches generally have only a single controlling input, these can be easily simulated with our SR latch designs. We have not yet investigated designs for JK latches, and so cannot compare our work to this design.

The other work which suggests designs for sequential building blocks is [4]. These authors present designs for a variety of flip-flops, but do not address the simpler latch structure, and thus comparisons do not give useful information.

As with many areas of research it is difficult to state that any one design is the best for all possible purposes. Rather in this work we have attempted to choose a simple, if not the simplest, design for a sequential building block and analyse its behaviour in various reversible implementations. This has given us two-fold results; confidence in the designs, as well as further possibilities for uses of these designs that were previously unconsidered, as described in Section 4.

6. CONCLUSION AND FUTURE WORK

This paper introduces the details behind two proposed reversible SR latch designs: one design based on the Fredkin gate and one design based on the Toffoli gate. Both of these designs are verified to behave correctly as compared to the traditional, irreversible SR latch and a short comparison is made. We find that the designs are very similar in behaviour, and little can be found to choose between them.

A major part of this paper introduces a novel idea; that of designing circuits that can be operated or driven, both forwards and in reverse, AND that can compute different functions in each of these directions. In relation to this idea the reverse behaviour of the Toffoli and Fredkin SR latch designs are characterized. We find that one of our structures has similar behaviour to that of a D latch when driven in reverse. This opens up the possibility of designing a circuit that can carry out two separate actions: we can design for one action to be performed when the circuit is driven forward, and a different action to be performed when the circuit is driven in reverse.

This suggests the possibility of an entirely new direction for this work: how can we design reversible gates that can leverage this possibility for dual functionality, and then how do we go about designing circuits that have the desired functionality using the same structures? One possibility is to design building blocks with known forwards and reverse behaviours, and to build reversible circuits entirely from such building blocks. However, this is an area that will require a great deal more research. Future work will consider these ideas as well as continuing to characterize the behaviour of other proposed reversible latches.

ACKNOWLEDGEMENT

The author would like to acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). She would also like to thank Dr. D. M. Miller for his discussions on this topic.

REFERENCES

[1] Frank, M.P. (2005) Approaching the Physical Limits of Computing. *Proc. Int. Symp. Multiple-Valued Logic (ISMVL)*, Calgary, Alberta, May 18–21, pp. 168–187. IEEE Computer Society, Los Alamitos, CA.

[2] Frank, M.P. (2005) Introduction to Reversible Computing: Motivation, Progress, and Challenges. *Proc. 2nd Conf. Computing Frontiers*, Ischia, Italy, May 4–6, pp. 385–390. ACM Inc., New York, NY.

[3] Frank, M.P. Reversible Computing: Quantum Computing's Practical Cousin. Lecture Notes and Abstract for Talk given at Simons Conference Lecture, Stony Brook, NY, May 2003. www.cise.ufl.edu/research/revcomp/Simons.

[4] Thapliyal, H., Srinivas, M.B. and Zwolinski, M. (2005) A Beginning in the Reversible Logic Synthesis of Sequential Circuits. *Proc. Military and Aerospace Programmable Logic Devices (MAPLD) Int. Conf.*, Washington, DC, September 7–9, submission no. 1012 (online proceedings). NASA Office of Logic Design, Washington DC.

[5] Landauer, R. (1961) Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, **5**, 183–191.

[6] Bennett, C.H. (1973) Logical reversibility of computation. *IBM J. Res. Dev.*, **6**, 525–532.

[7] Toffoli, T. (1980) Reversible Computing. Technical Report MIT/LCS/TM No. 151, MIT.

[8] Perkowski, M. *et al*. (2001) A General Decomposition for Reversible Logic. *Proc. Int. Workshop on Methods and Representations (RM)*, Starkville, Mississippi, August 10–11, pp. 119–138. RM Workshop, Starkville, MS.

[9] Agrawal, A. and Jha, N.K. (2004) Synthesis of Reversible Logic. *Proc. Design, Automation and Test in Europe Conf. and Exhibition (DATE)*, Paris, France, February 16–20, pp. 1384–1385. IEEE Computer Society, Los Alamitos, CA.

[10] Miller, D.M., Dueck, G.W. and Maslov, D. (2004) A Synthesis Method for MVL Reversible Logic. *Proc. 34th Int. Sympos. Multiple-Valued Logic (ISMVL)*, Toronto, Canada, May 19–22, pp. 74–80. IEEE Computer Society, Los Alamitos, CA.

[11] Miller, D.M., Maslov, D. and Dueck, G.W. (2003) A Transformation Based Algorithm for Reversible Logic Synthesis. *Proc. 40th Design Automation Conf. (DAC)*, Anaheim, CA, June 2–6, pp. 318–323. ACM Inc., New York, NY.

[12] Maslov, D. and Dueck, G.W. (2004) Reversible cascades with minimal garbage. *IEEE Trans. Comput. Aided Des.*, **23**, 1497–1509.

[13] Shende, V.V., Prasad, A.K., Markov, I.L. and Hayes, J.P. (2002) Reversible Logic Circuit Synthesis. *IEEE/ACM Int. Conf. Computer Aided Design* (*ICCAD*), San Jose, CA, November 10–14, pp. 353–360. IEEE, Piscataway, NJ.

[14] Kerntopf, P. (2004) A New Heuristic Algorithm for Reversible Logic Synthesis. *Proc. Design Automation Conf.* (*DAC*), San Diego, CA, June 7–11, pp. 834–837. ACM Inc., New York, NY.

[15] Mishchenko, A. and Perkowski, M. (2002) Logic Synthesis of Reversible Wave Cascades. *Proc. Int. Workshop on Logic Synthesis*, New Orleans, USA, June 4–7, pp. 197–202. IWLS Workshop, New Orleans, LA.

[16] Ye, Y. and Roy, K. (1996) Energy recovery circuits using reversible and partially reversible logic. *IEEE Trans. Circuits Syst.*, **43**, 769–778.

[17] Forsberg, E. (2005) The Electron Waveguide Y-Branch Switch: a Review and Arguments for its Use as a Base for Reversible Logic. *Proc. 2nd Conf. Computing Frontiers*, Ischia, Italy, May 4–6, pp. 404–406. ACM Inc., New York, NY.

[18] Henzler, S., Nirschl, T., Eireiner, M., Amirante, E. and Schmitt-Landsiedel, D. (2005) Making Adiabatic Circuits Attractive for Todays VLSI Industry by Multimode Operation-adiabatic mode circuits. *Proc. 2nd Conf. Computing Frontiers*, Ischia, Italy, May 4–6, pp. 414–420. ACM Inc., New York, NY.

[19] Kim, S. and Chae, S.-I. (2005) Implementation of a Simple 8-bit Microprocessor with Reversible Energy Recovery Logic. *Proc. 2nd Conf. Computing Frontiers*, Ischia, Italy, May 4–6, pp. 421– 426. ACM Inc., New York, NY.

[20] Chuang, M.-L. and Wang, C.-Y. (2006) Reversible Logic Designs for Sequential Elements. *Proc. 13th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI)*, Nagoya, Japan, April 3–4, pp. 127–133. SASIMI Workshop, Nagoya, Japan.

[21] Rice, J.E. (2006) An Analysis of Several Proposals for Reversible Latches. *Proc. 2nd Int. Joint Conf. Computer, Information, and Systems Sciences and Engineering (CISSE), e-conference*, December 4–14, CDROM 1 paper no. 548. Springer, Heidelberg, Germany.

[22] Rice, J.E. (2006) A New Look at Reversible Memory Elements. *Proc. Int. Symp. Circuits and Systems (ISCAS)*, Kos, Greece, May 21–24, pp. 1243–1256. IEEE, Piscataway, NJ.

[23] Picton, P. (1996) Multi-valued sequential logic design using Fredkin gates. *Multiple-Valued Logic*, **1**, 241–251.

[24] Patra, P. (1995) Asymptotically Zero Power in Reversible Sequential Machines. Technical Report CS-TR-94-14, Department of Computing Sciences, University of Texas at Austin.

[25] Fredkin, E. and Toffoli, T. (1982) Conservative logic. *Int. J. Theor. Phys.*, **21**, 219–253.