Technical Report: The State of Reversible Sequential Logic Synthesis

TR-CSJR2-2005
J. E. Rice
University of Lethbridge
j.rice@uleth.ca

September 27, 2005

1 Introduction

This report provides an overview of the current state of reversible sequential logic. We assume that the reader has a basic knowledge of reversible logic. An overview of necessary information is provided in [1].
Most research in reversible logic has concentrated on synthesis techniques for combinational logic. Sequential logic synthesis requires the incorporation of memory cells and feedback in a circuit, which seems to violate the definition of reversible logic. Previous researchers who have investigated this problem have approached it from two avenues; those who have proposed theoretical structures (*i.e.* gates) for building sequential building blocks such as latches and flip-flops, for example [2] and [3] and those who have proposed physical designs for memory cells (*e.g.* RAM), for example [4].
Our interest is directed mainly at logic synthesis at the gate level, and so we will concentrate on the former. This report gives an overview of the proposed structures, and suggests new work in how these might be incorporated into logic synthesis techniques aimed at producing sequential reversible logic circuits.

2 Previous Work

2.1 Toffoli

In 1980 Tommaso Toffoli wrote "Reversible Computing" [5]. In this work he characterized reversible logic in general, and included a section entitled "Reversible sequential computing". Toffoli proved that a finite automaton is reversible if its transition function is invertible. In order to realize a finite automaton with a reversible sequential circuit it suffices to build a reversible circuit realizing the transition function and use this as the combinational part of the sequential network. Toffoli also indicates that it is possible to reversibly realize any arbitrary function, thus

> ... whatever can be computed by an arbitrary finite automaton ... can also be computed by a *reversible* finite automaton...

He goes on to say that

> *Using invertible logic gates, it is ideally possible to build a sequential computer with zero internal power dissipation.*

Toffoli then discusses the implementation of Turing machines and cellular automata with relation to reversible logic. However, his work proves that it is possible to build these sequential machines using reversible logic; it does not go into detail on the gate-level structures that one might use. In addition, the concept of synthesis for (sequential) reversible logic is outside the scope of his work.

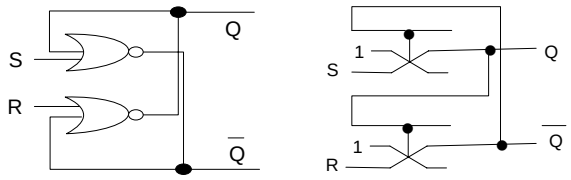
2.2 Picton's RS-Latch

Other than Toffoli's earlier discussion of sequential reversible logic, the literature holds only two papers applicable to the topic. One reason for this is that Bennett's original paper on the topic of reversible logic synthesis "Logical Reversibility of Computation" [6] states that in addition to conserving logic levels *information* should also be preserved, and so the use of feedback was prohibited. Picton [2], however, argues that the unit wire as introduced by Fredkin is nothing more than a clocked delay. Table 1 illustrates the behaviour of this reversible "gate". We see that the value of the output y at time $t+1$ equals the value of the input x at time t, and thus meets the requirements of a clocked delay. Because of this Picton argues that Bennett's proposal was thus in the form of clocked-logic and

x_t		y_{t+1}
0	$\rightarrow$	0
1		1

Table 1: Definition of a Unit Wire.

not, as implied, combinational logic. Since a clocked delay generally uses some form of feedback to create stable states this implies that the proposal also requires, rather than prohibits, feedback. Picton goes on to illustrate how the basic Fredkin gate can be used to construct an RS-latch as shown in Figure 1. The problem with this model is



(a) (b)

Figure 1: The RS-latch a) built from tradtional logic gates and b) built from two Fredkin gates as designed by Picton [2].

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) but one can not allow the arbitrary creation of data. In the illustration shown in Figure 1 there are two instances of fan-out. When fan-out occurs we are essentially creating a duplicate of the signal; however, in reversible logic this is not permitted since we cannot arbitraily generate energy from nowhere. It is not the feedback that is prohibited, but how we generate the feedback. Thus some alternative to the problem of fan-out in creating this latch is required. We address this issue in Section 3.1.

2.3 Frank's Reversible Logic Operations

According to Dr. Frank in [3],

> ... it is *not* necessary that (1) the gate hardware can perform only one operation, (2) we divide the subsystems accessed by the gate into fixed "input" and "output" signals, (3) we require the number of output signals to be equal to the number of output signals, (4) that the input signals must be consumed, (5) that the truth table of a gate with n input bits contain an output column that is a permutation of all 2^n cases.

Frank goes on to state that

- sometimes a gate can totally ignore a signal while at other times it can use a signal as both an input and an output;

- the information present on input signals need not be consumed by the operation performed;
- the numbers of signals modified does not require any particular relationship to the number of input signals, and
- that "gates" may contain internal state information.

The true requirement for a reversible logic gate is

> ... 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.

This allows definitions of reversible gates such as $rSET(x)$, or reversible SET, whose behaviour is defined as follows: given a precondition that the input signal x's value is intially 0, change it to 1. This is still reversible given that the precondition is satisfied. Similarly $rCLR(x)$, $rCOPY(x)$, and many other reversible operations based on initial conditions may be defined. Frank provides suggestions for CMOS implementations for his operations. His model also includes $crCLR(c, d)$ which is a conditional reversible CLEAR. This gate's behaviour is defined as: given a prediction that the initial state is not $c = 1, d = 0$, perform $rCLR(d)$ otherwise leave d unchanged. Similarly he defines $crSET(c, d)$: given the prediction that the initial state is not $c = d = 1$, if $c = 1$ perform $rSET(d)$ otherwise leave d unchanged. These two operations can then be composed together into networks that implement, in entirely reversible logic, any logic function utilizing sequential and/or combinational logic.

3 New Work

The design of sequential reversible logic, in my opinion, is a problem that can be approached from two directions. Regardless of the approach, the results should be the same; however, it is often useful to have two views from which to examine a problem.

The first approach to be considered in this report begins by examining existing traditional logic implementations of sequential logic elements such as latches and flip-flops. By replacing the traditional gates in these elements with reversible gates, the result will be reversible sequential logic elements. We start with examining Picton's RS-latch in Section 3.1.

The second approach examines the problem strictly from a logic synthesis point of view; given a truth table defining the desired logic we must come up with a cascade of reversible logic elements that transforms the inputs to the desired outputs. Thus, to determine a cascade for sequential logic elements, we must provide a state table defining the inputs and outputs, and a reversible logic transformation meeting these requirements. Section 3.2 examines this process.

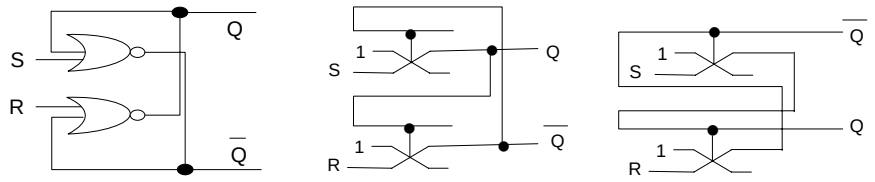
Finally, once we have a selection of reversible sequential logic elements, we can then begin incorporating them into a synthesis process with the goal of producing sequential logic circuits implemented solely with reversible logic elements. Section 3.4 begins this discussion.

3.1 An Alternative to the Use of Fan-out In Picton's Model

As pointed out in the previous section, fan-out is required for Picton's model of a RS-latch built using two Fredkin gates. An examination of the functionality of this latch leads us to a relatively simple solution; instead of requiring us to fan-out the Q and $\overline{Q}$ signals to both the outputs and back to the outputs, why not use the outputs of the Fredkin gate for one of these uses. A solution demonstrating this is shown in Figure 2. If we compile state tables for the two reversible latches we can see that the behaviours are the same, as shown in Table 2.

3.2 New Reversible Designs for the RS-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.



(a) (b) (c)

Figure 2: An RS-latch built from (a) traditional logic gates, (b) as designed by Picton, and (c) modified to avoid fanout.

	$\overline{Q}(t)$	1	S	$Q(t)$	1	R	$\overline{Q}(t+1)$	$Q(t+1)$
initial state:	0	1	0	0	1	0	0	0
next state:	1	1	0	1	1	0	1	1
next state:	0	1	0	0	1	0	0	0 (unstable)
initial state: (SET)	0	1	1	0	1	0	0	0
next state:	1	1	1	1	1	0	1	1
next state:	0	1	1	1	1	0	0	1 (stable)
initial state: (RESET)	0	1	0	0	1	1	0	0
next state:	1	1	0	1	1	1	1	1
next state:	1	1	0	0	1	1	1	0 (stable)

Table 2: The state table for the Fredkin gate-based RS latch, both the Picton version and the modified version.

For instance, the NOR gate may be used in the design of the RS-latch. The truth table of this gate is shown in Table 3. The NOR gate is clearly not reversible; one problem is that there is only one output and two inputs.

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Table 3: The behaviour of the NOR gate as used in the traditional logic implementation of a RS-latch.

Table 4 shows the additions required to create a reversible equivalent of the NOR gate's behaviour. Three outputs are required, and so an additional input must be added. Note that we set this input to 0 and so this is technically only half of the function's truth table; however, we are only interested in the behaviour defined for these inputs. If we rearrange the table as shown in Table 5 which happens to match the behaviour of the Toffoli gate. The behaviour of the Toffolli gate is defined as $(z, y, x) \rightarrow (z, y, x \oplus yz)$. We can thus implement the RS-latch as shown in Figure 3[1].

Another approach to determining reversible memory elements is to define the desired element's state table, manipulate as needed in order to get a reversible function, and then perform reversible logic synthesis techniques. For instance, such a state table might begin as shown in Table 6. After creating the state table the next stage is to create a cascade of reversible gates that effects the required transitions. However, in the case above the cascade quickly becomes quite involved, and so the designs from Figures 3 and 2 are a more efficient choice.

[1]The second implementation was designed by Dr. Michael Miller.

A	B	0	x	y	$\overline{A+B}$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	0	1	1	0

Table 4: A reversible gate with the behaviour of the NOR gate.

$\overline{A}$	B	0	x	y	$\overline{A+B}$
1	1	0	0	0	1
1	0	0	0	1	0
0	1	0	1	0	0
0	0	0	1	1	0

Table 5: A slightly modified version of the reversible NOR gate which matches the definition of the Toffoli gate.

3.3 Reversible flip-flop designs based on the RS-latch

According to Sasao [8] there are four standard flip-flop designs. Although one can construct more complex latches with characteristics to those of the flip-flops, we argue that the usefulness of a non-clocked memory element is limited. Additionally, we investigate only edge-triggered flip-flops, since clocked latches may be difficult to use due to the restrictions inherent in the period and and width of the clock pulse.

3.3.1 Master-Slave flip-flops

Figure 4 illustrates the standard construction, using traditional logic, of a master-slave flip-flop. The behaviour of this flip-flop is defined as given in Table 7. Since previous sections have determined structures for reversible RS-latches, if we can also determine reversible structures with behaviours equivalent to the other elements in the master-slave flip-flop then these can be combined to create a reversible implementation. A Toffoli gate can be used for the AND structures, leaving only the problem of fan-out. In Figure 5 the fan-out from the clock is generated through the use of a Fredkin gate, which gives two identical outputs and one output that is the negation of the other two. This is useful since we also need a negated clock output, which is then fanned-out to the two inputs to the second RS-latch. Another issue lies in the question of how to generate the clock signal for a reversible circuit; however this is outside the scope of this work.

3.3.2 D flip-flop

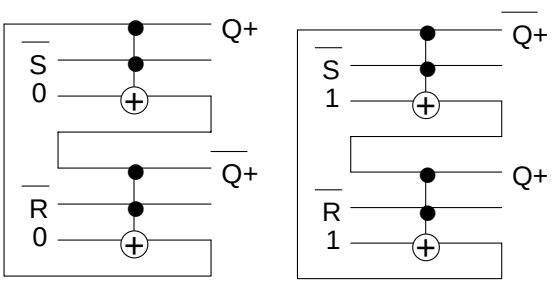
An extension of the Master-slave flip-flop is to disallow $S = R$. One way to do this is to use a single input D which is connected directly to the S input, and then $\overline{D}$ is connected to the R input. The behaviour is then as shown in Table **??**. Figure 6 shows both the traditional logic implementation of the D flip-flop and an equivalent reversible implementation.

3.3.3 JK flip-flop

In many cases it is desirable to be able to retain the value of Q in the memory element, or even to negate it. The JK flip-flop allows this by making use of additional feedback. The behaviour is then as defined in Table 9. Figure 7 shows a traditional implementation of this flip-flop and an equivalent reversible implementation.

3.3.4 T flip-flop

Again, a modification to the inputs of the JK flip-flop is possible in order to build a slightly different flip-flop. If we let $T = J = K$ then the flip-flop behaviour becomes as given in Table 10. Figure 8 shows a traditional implementation of this flip-flop and an equivalent reversible implementation.



(i) (ii)

Figure 3: Two alternate implementations for the RS latch, using Toffoli gates instead of Fredkin gates.

$Q\overline{Q}SR$	$Q^+\overline{Q}^+XY$
0 0 - -	disallowed
0 1 0 0	0 1 0 0
0 1 0 1	0 1 0 0
0 1 1 0	1 0 0 0
0 1 1 1	1 1 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 1 0 1
1 0 1 0	1 0 1 0
1 0 1 1	1 1 0 1
1 1 0 0	1 1 1 0
1 1 0 1	0 1 1 1
1 1 1 0	1 0 1 1
1 1 1 1	1 1 1 1

Table 6: A state table for the RS-latch.

3.4 Incorporating Sequential Structures into Reversible Logic Synthesis Techniques

Although Thapliyal *et. al.* will be presenting a paper at MAPLD2005 [7] entitled "A Beginning in the Reversible Logic Synthesis of Sequential Circuits" which details the design of various reversible sequential logic elements, it appears that no researchers have yet investigated the problem of incorporating sequential structures into reversible logic synthesis techniques. All of the logic synthesis techniques are restricted thus far to combinational logic. Both Picton's model and Frank's model of reversible sequential gates seem viable; thus the next step is to incorporate these gates into known techniques for reversible logic synthesis. However, it first seems reasonable to review techniques used in traditional logic synthesis for the incorporation of sequential logic.
Sasao [8] gives the following procedure for the realization of sequential networks. Given a state table describing the sequential network to be implemented one must

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	forbidden

Table 7: The behaviour of the master slave flip-flop.

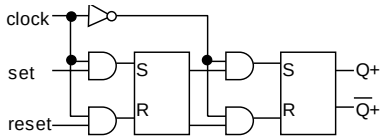


Figure 4: A master-slave flip-flop designed using RS-latches.

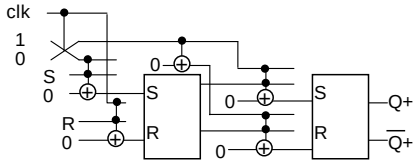


Figure 5: A reversible implementation of a master-slave flip-flop.

1. design the state table from the word description,
2. simplify the state table,
3. assign codes to the state and derive the transition table; also, derive the output table showing the output functions,
4. select the flip-flops to be used,
5. derive the excitation functions for the selected flip-flops, and finally
6. realize the logic networks for the excitation functions and the output functions.

Based on Picton's and Frank's work it seems that any state table can be implemented using either our modified version of Picton's RS latch or using the $crSET$ and $crCLR$ as described by Frank. Thus many of the steps above should remain the same for reversible sequential logic. The areas where work remains are in the selection of flip-flops to be used, derivation of their excitation functions, and the realization of the logic networks. Realization of the logic networks for excitation ond output functions must, if the network is to be implemented using reversible logic, be performed using reversible logic synthesis techniques. This will be a topic for future research.

4 Conclusion

This report gives an overview of the state of current research into sequential logic synthesis for reversible logic circuits. We have discussed the work of Picton and Frank, and suggested a modification to the latch model presented by Picton. We also briefly review the design process for traditional sequential logic, and indicate areas where work for reversible sequential logic must be done. We provide examples of reversible logic flip-flops and derive excitation equations from them.

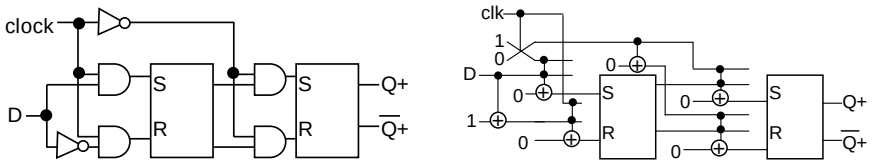
References

[1] J. E. Rice. Projects in Reversible Logic. Technical Report TR-CSJR1-2005, Dept. of Math & Computer Science, University of Lethbridge, July 2005.

[2] P. Picton. Multi-Valued Sequential Logic Design using Fredkin Gates. *Multiple-Valued Logic*, pages 241–251, 1996.

D	Q^+
0	0
1	1

Table 8: The behaviour of the D flip-flop.



(i) (ii)

Figure 6: (i) A traditional logic implementation for the D flip-flop, and (ii) an equivalent reversible implementation.

[3] M. P. Frank. Approaching the Physical Limits of Computing. In *Proceedings of the International Symposium on Multiple-Valued Logic (ISMVL)*, pages 168–187, 2005.

[4] J.-H. Kwon, J. Lim, and S.-I. Chae. A three-port nRERL register file for ultra-low-energy applications. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design (ISLPED '00)*, pages 161–166, 2000.

[5] T. Toffoli. *Automata, Languages and Programming*, chapter Reversible Computing, pages 632–644. Springer Verlag, 1980.

[6] C. H. Bennett. Logical Reversibility of Computation. *IBM Journal of Research and Development*, 6:525–532, 1973.

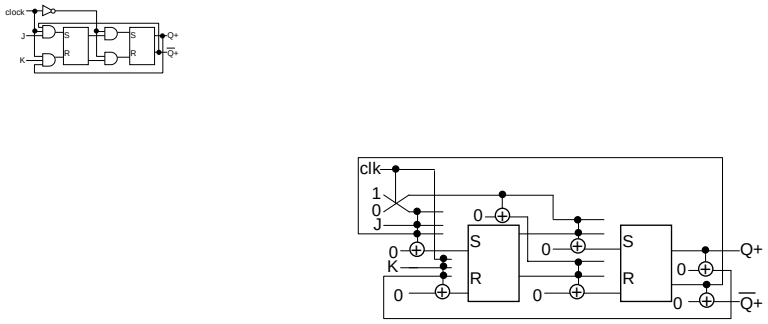
[7] H. Thapliyal and M. B. Srinivas. A Beginning in the Reversible Logic Synthesis of Sequential Circuits. In *Proceedings of Military and Aerospace Programmable Logic Devices (MAPLD) International Conference*, 2005.

[8] T. Sasao. *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, 1999.

[9] J. E. Rice. The State of Reversible Sequential Logic Synthesis. Technical Report TR-CSJR2-2005, Dept. of Math & Computer Science, University of Lethbridge, July 2005.

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Table 9: The behaviour of the JK flip-flop.

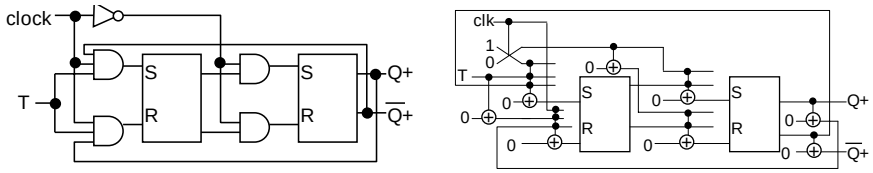


(i) (ii)

Figure 7: (i) A traditional logic implementation of the JK flip-flop, and (ii) a reversible equivalent.

T	Q^+
0	Q
1	$\overline{Q}$

Table 10: The behaviour of the T flip-flop.



(i) (ii)

Figure 8: (i) A traditional logic implementation of the T flip-flop, and (ii) a reversible equivalent.