A New Look at Reversible Memory Elements

J. E. Rice
Dept. of Math & Computer Science
University of Lethbridge
Alberta, Canada
Email: j.rice@uleth.ca

Abstract—Although many researchers are investigating techniques to synthesize reversible combinational logic, there is little work in the area of sequential reversible logic. We present an analysis of a basic memory element, the RS-latch, and a number of possible implementations. We then go on to introduce four reversible flip-flop designs based on the reversible RS-latch implementation.

I. INTRODUCTION

In 1980 Tommaso Toffoli wrote “Reversible Computing” [1]. 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 “Using invertible logic gates, it is ideally possible to build a sequential computer with zero internal power dissipation.” Other researchers such as Frank [2] have also been interested in the idea of sequential reversible logic.

The question at this point is why reversible computing at all? A quotation from Frank [3] helps answer the question:

...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...

Despite, however, the great potential of reversible logic, and these endorsements from the leaders in the field, little to no work has been done in this area of sequential reversible logic. Many researchers are investigating combinational logic synthesis techniques but have seemed to overlook the necessity of memory elements if one is planning to implement many of our day to day circuits in reversible technologies.

This paper visits the small amount of existing work in the area, and extends this work to introduce a new reversible implementation of the RS latch. We then go on to present flip-flop implementations based on this latch. We hope that this work will begin to fill the gap in this area of research.

<table>
<thead>
<tr>
<th>gate</th>
<th>behaviour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not</td>
<td>$(x) \rightarrow (x \oplus 1)$</td>
</tr>
<tr>
<td>Feynman</td>
<td>$(y, x) \rightarrow (y, x \oplus y)$</td>
</tr>
<tr>
<td>Toffoli</td>
<td>$(z, y, z) \rightarrow (z, y, x \oplus yz)$</td>
</tr>
<tr>
<td>swap</td>
<td>$(x, y) \rightarrow (y, x)$</td>
</tr>
<tr>
<td>Fredkin</td>
<td>$(z, y, z) \rightarrow (z, x, y)$ if $z = 1$</td>
</tr>
</tbody>
</table>

TABLE II

The behaviour of a selection of more commonly used reversible logic gates.
logic sequential element in the form of a JK flip-flop, and Picton [6] suggests a reversible RS latch. He uses the basic Fredkin gate to build this latch, as shown in Figure 2. Our paper also concentrates on the use of a basic memory element such as the RS latch, as it is the traditional building block for the clocked flip-flop structures that are our goal. 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) but one can not allow the arbitrary creation of data. In the illustration shown in Figure 2 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 arbitrarily 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 further on.

More recent work in the area of sequential reversible logic has been published by Thapliyal et al. [7]. These authors also discuss the construction 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.

III. ADDRESSING THE PROBLEM OF FANOUT

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 to a relatively simple solution; instead of requiring fan-out from the Q and $\bar{Q}$ signals to both the outputs and back to the inputs, why not use the outputs of the Fredkin gate for one of these uses. A solution demonstrating this is shown in Figure 3. State tables for the two reversible latches illustrate that the behaviours are the same, as shown in Table III.

IV. CONSTRUCTING A NEW REVERSIBLE 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. For instance, the NOR gate may be used in the design of the RS-latch. The NOR gate is clearly not reversible; one problem is that there is only one output and two inputs. Table IV (A) 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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>0</th>
<th>x</th>
<th>y</th>
<th>A+B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1+0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1+0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0+1</td>
</tr>
</tbody>
</table>

TABLE IV

(A) A REVERSIBLE GATE WITH THE BEHAVIOUR OF THE NOR GATE, AND (B) WITH SLIGHT MODIFICATIONS TO MATCH THE BEHAVIOUR OF THE TOFFOLI GATE.

shown in Table IV (B) we achieve a reversible function which happens to match the behaviour of the Toffoli gate. We can thus implement the RS-latch as shown in Figure 4.

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 V. After creating the state table the next stage is to create a cascade of

1Unfortunately the final version of Thapliyal et al.’s work will not be available until after this paper is submitted.

2The second implementation was suggested by Dr. Michael Miller.
Fig. 4. Two alternate implementations for the RS latch, using Toffoli gates instead of Fredkin gates.

Table V

<table>
<thead>
<tr>
<th>QSR</th>
<th>Q+Y</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>disallowed</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

A state table for the RS-latch.

Fig. 5. The (A) behaviour, (B) a traditional logic implementation, and (C) a reversible equivalent for a master-slave flip-flop.

TABLE V

<table>
<thead>
<tr>
<th>D</th>
<th>Q+</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 6. The (A) behaviour, (B) traditional logic implementation, and (C) equivalent reversible implementatin for the D flip-flop.

V. REVERSIBLE CLOCKED FLIP-FLOPS

According to Sasao [8] there are four standard flip-flop designs. Although one can construct more complex latches with characteristics similar 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 width of the clock pulse.

1) Master-Slave flip-flops: Figure 5 illustrates the standard construction, using traditional logic, of a master-slave flip-flop. 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 (C) 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.

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 Figure 6 (A). Figure 6 also shows both the traditional logic implementation of the D flip-flop and an equivalent reversible implementation.

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 Figure 7 (A). Figure 7 also shows a traditional implementation of this flip-flop and an equivalent reversible implementation.

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 Figure 8 (A). Figure 8 also shows a
feedback is the final structure truly reversible?

elements; however, by introducing not one but two stages of
a reversible equivalent for the T flip-flop.

Fig. 8. The (A) behaviour, (B) a traditional logic implementation, and (C) a
reversible implementation.

traditional implementation of this flip-flop and an equivalent
reversible implementation.

Fig. 7. The (A) behaviour, (B) a traditional logic implementation and (C) a
reversible equivalent for the JK flip-flop.

VI. DISCUSSION

There is some argument as to whether a flip-flop such as
the JK flip-flop shown in Figure 7 (C) can be considered
reversible. Certainly it is constructed entirely of reversible
elements; however, by introducing not one but two stages of
feedback is the final structure truly reversible?

VII. CONCLUSION AND FUTURE WORK

This paper presents a discussion of reversible sequential
logic. Previous work in the area is examined, with some
suggestions as to modifications being presented. A new re-
versible implementation for a reversible RS latch is introduced,
and four standard types of edge-triggered flip-flops are built
exclusively from reversible logic gates.

One particular area of future work, now that reversible
memory elements have been presented, is to incorporate these
into a logic synthesis process. Existing work on combinational
logic synthesis has been presented by researchers such as
Dueck et. al. [9], Perkowski et. al. [10], Kerntopf [11], and
Shende et. al. [4]. It is a logical next step to make use of the
most suitable of these in a sequential logic synthesis technique,
and future work will address this area.

In addition, the process followed in this paper is fairly
pedantic; we simply used known traditional logic implementa-
tions for sequential elements and replaced each of the internal
elements with reversible equivalents. As Fredkin et. al. indi-
cate in [5], this may not result in the most efficient reversible
(or conservative) implementation. Thus further investigation
into determining optimal implementations for edge-triggered
clocked flip-flops is continuing.

ACKNOWLEDGMENT

The author would like to thank Drs. Claudio Moraga and
Michael Miller for enlightening and thought-provoking discus-
sions and emails on this matter.

REFERENCES

[1] T. Toffoli, Automata, Languages and Programming. Springer Verlag,
Proceedings of the International Symposium on Multiple-Valued Logic
[3] ———, “Introduction to reversible computing: motivation, progress, and
challenges,” in Proceedings of the 2nd Conference on Computing
logic circuit synthesis,” in IEEE/ACM International Conference on
of Theoretical Physics, pp. 219–253, 1982.
Logic Synthesis of Sequential Circuits,” in Proceedings of Military
and Aerospace Programmable Logic Devices (MAPLD) International
Conference, 2005.
synthesis of networks of toffoli/fredkin gates,” in Proceedings of the
IEEE Canadian Conference on Electrical and Computer Engineering,
2003.
[10] M. Perkowski, L. Joziwak, A. Mikhchenko, A. Al-rabadi, A. Coppola,
A. Buller, X. Song, M. Khan, S. Yanushkevich, V. P. Shmerko,
and M. Chrzanowska-Jeske, “A General Decomposition for Reversible
Logic,” in Proceedings of the International Workshop on Methods and
sis,” in Proceedings of the Design Automation Conference (DAC), 2004,
pp. 834–837.