Reversible Circuit Technology Mapping from Non-reversible Specifications

Zeljko Zilic, Katarzyna Radecka ˅ and Ali Kazamiphur˅
McGill University, Concordia University˅
zeljko.zilic@mcgill.ca

Abstract

This paper considers the synthesis of reversible circuits directly from an irreversible specification, with no need for producing a reversible embedding first. We present a feasible methodology for realizing the networks of reversible gates, in a manner that builds on the classical technology mapping. We do not restrict ourselves to the restricted notion of realizing permutation functions, and construct reversible implementations where extraneous signals are efficiently reused for overcoming the inherent fanout limitation.

1. Introduction

Reversible circuits are of fundamental interest due to their capability of restoring consumed energy [1]. They are also required in quantum computing, and are of use in cryptographic and some other classical circuits [2]. To realize a reversible implementation of an arbitrary logic function, traditionally one first has to produce a *reversible embedding* onto a suitable reversible function. Only then, the synthesis of the resulting reversible specification can take place.

In this paper, we propose a reversible circuit synthesis methodology that is undertaken without finding a reversible embedding first. We will show that it suffices to start with a classical, irreversible logic circuit that is already mapped onto a library of 2-input gates using classical methods, and then apply the reversible mapping using a library of reversible logic cells.

There are two pressing problems with any reversible synthesis that have to be addressed from the outset. First, we need to be efficient with adding extraneous I/O signals that are required for reversible operation of the circuit. Second, the synthesis must account for the inability to fan out the signal in a network, which is another fundamental limitation of the reversible logic. The technique proposed here will deal with both issues in a way that actually exploits the existence of extraneous bits to route the signals that need to be replicated in a fanout.

The paper is organized as follows. In Section 2, we provide the background on reversible computing and the reversible logic synthesis. Section 3 addresses design issues of reversible library cells. Section 4 presents the key insight to facilitating the reversible technology mapping by a library of reversible cells. Section 5 provides the results of experiments designed to demonstrate the impact of the cell library design, the technology mapping choices and the optimization steps by which double gates can be packed in a single cell.

2. Background

A circuit is reversible if it realizes a bijective mapping of inputs to outputs. Hence, the number of inputs and outputs must be the same. Further, all signals must have a fanout of 1. We say that a reversible gate is *universal* if any reversible function can be realized by a circuit built solely with such a gate. It is known that the smallest universal gates are three input, three output (3x3) gates, while 2x2 gates can implement only linear mappings [9].

Reversible synthesis of an arbitrary irreversible multi-output Boolean function f invariably produces a reversible embedding function f_r. For such an embedding, it suffices that f_r is reversible, and that its restriction to primary inputs and outputs of f is equivalent to f. Reversible embeddings can produce additional number of extraneous inputs and outputs. In some contexts, additional inputs are referred to as *ancilla bits*, while extraneous outputs are called *garbage bits*. Hence, in addition to minimizing the cost of logic gates, in reversible synthesis, the number of these extraneous pins should be kept to a minimum.

The number of required extraneous bits can be derived from the maximum repetition count, f_{same}, among output bit combinations. It is easy to prove [5] that the minimum number of added output bits G equals the ceiling of the logarithm of the identical output values, $\lceil log_2 (f_{same}) \rceil$. Further, the number of added inputs A must be such that the total number of inputs and outputs is always equal:

$$\#inputs + A = \#outputs + G \quad (1)$$

Most work on reversible synthesis is restricted to *permutation mappings* by which input bit combinations simply get permuted at the outputs. These are clearly not all reversible functions. Consider, for example, a reversible mapping from a pair of bits (a, b) to (a, $a \oplus b$). While inputs can be always restored from outputs, this mapping is not a permutation of inputs.

A formulation of the permutation mapping optimization problem was given in [2], together with an exact algorithm that is double exponential in the number of inputs. Synthesis of permutation mappings where the number of extraneous bits is kept to a minimum was considered in [5]. A heuristic is developed that changes

978-3-9810801-2-4/DATE07 © 2007 EDAA

gates until the Hamming distance to the desired function disappears. A body of work employs rewrite rule-based template synthesis of permutation mappings (see, e.g. [6]). The work in [7] presents somewhat wider scope than the permutation mapping optimization, and uses the Reed-Muller transform to algebraically express functions in a form suitable for finding common terms. While this method apparently scales better, it is still capable of synthesizing relatively small circuits that are already reversibly embedded.

Recent work [11] on quantum circuit synthesis has reduced the problem to the classical multiple-valued logic synthesis and reachability analysis. In this case, the reversible gates are non-binary and are not necessarily only permutation mappings. While offering a promise of mapping the irreversible specifications to a reversible cell library, it still applies only to permutation mappings.

The goal of synthesizing reversible circuits directly from irreversible specification, similar to the classical technology mapping, has been considered since the study by Toffoli [1], but so far it has not been achieved. In classical technology mapping [3], i.e., the covering a Boolean network with library cells (usually single-output Boolean functions), one often uses the fact that the dynamic programming is known to produce optimal results for trees, or that the network-flow based algorithms can be optimal for programmable logic blocks.

2.1. Common Reversible Gates

Reversible logic synthesis has relied on gates such as Toffoli gate. Using classical logic gates, the operation of the three smallest Toffoli gates is described in Figure 1.a).

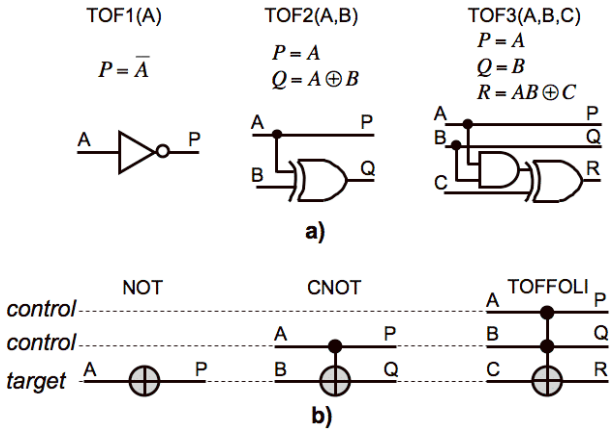

Figure 1: Family of Toffoli Gates

Toffoli gates with 1 and 2 inputs are commonly referred to as the NOT and CNOT gates. The usual symbols for Toffoli gates in Figure 1.b) indicate that the Toffoli *target* output is inverted if all the *control* bits are set to 1. Bidirectional signal flow is possible in reversible circuits; our use of Boolean gates to describe gates as in Figure 1.a) does not preclude bidirectionality.

The 3x3 Tofolli gate is *universal*, i.e., any reversible circuit can be built by employing only this gate. As evident from Figure 1, by selecting appropriate combinations of input control lines, Toffoli gate output $R(A, B, C)$, can realize AND, NOT and XOR functions:

AND: $R(A, B, 0) = AB$,

NOT: $R(A, 1, 0) = \overline{A}$,

XOR: $R(1, B, C) = B \oplus C$ or $R(A, 1, C) = A \oplus C$.

While Toffoli gate is a controlled inverter, the widely considered Fredkin gate is a controlled swap gate in which all control signals need to be 1 to swap the two target bits. Also considered is the Kerntopf gate, with three outputs, *P, Q* and *R,* defined as:

$$P(A,B,C) = 1 \oplus A \oplus B \oplus C \oplus AB,$$
$$Q(A,B,C) = 1 \oplus AB \oplus B \oplus C \oplus BC,$$
$$R(A,B,C) = 1 \oplus A \oplus B \oplus AC.$$

The simultaneous availability of many functions (there are 18 different cofactors!) at the outputs of the Kerntopf gate offers a lot of flexibility for synthesis and for implementing various adders [8]. However, there are no known inexpensive realizations [9], so it is believed to be significantly more costly than Toffoli and Fredkin gates.

2.2. Universal And/Or Reversible Gate

We refer to the gate proposed in [10], Figure 2.a), as a Universal AND/OR Reversible Gate. The gate can be realized with just two CNOT and one TOFFOLI gates, Figure 2.b). Hence, it is more area-efficient than the Kerntopf gate. Additionally, as input signals pass through less logic, the proposed gate is inherently faster than Kerntopf gate.

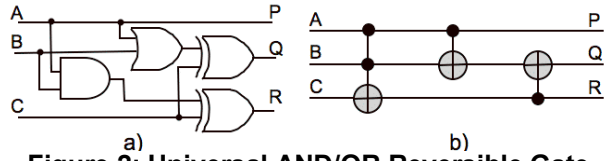

Figure 2: Universal AND/OR Reversible Gate

The gate outputs, expressed as AND, OR, NAND, NOR, etc. functions of inputs A and B, and control C, are:

$$F(A,B,C) = \{P(A,B,C), Q(A,B,C), R(A,B,C)\} =$$
$$[P(A,B,1) = A, \quad Q(A,B,1) = \overline{A+B}, \quad R(A,B,1) = \overline{AB}];$$
$$[P(A,B,0) = A, \quad Q(A,B,0) = A+B, \quad R(A,B,0) = AB].$$

Hence, this universal reversible gate provides an inexpensive block that packs together an AND and OR logic functions, whose polarity can be programmed. It reduces the number of output garbage bits by 1 since it implements two Boolean functions at its outputs.

3. Cell Library for Reversible Mapping

To design a library of cells useful for reversible logic synthesis, we rely on the well-known fact that any two-input classical gate can be embedded onto a 3x3 reversible gate. Since there are the universal reversible

gates with three inputs (and three outputs), we deduce that in order to produce a reversible cell library adequate for reversible logic synthesis it is sufficient to consider 2-input classical gates and their reversible embeddings. To realize such a minimum reversible cell library, we need to select a reversible embedding for all 2-input Boolean functions. Armed with such a library, we can then undertake the reversible technology mapping to cover the originally irreversible logic network by reversible cells.

3.1. Reversible Library Creation

By enumerating all 16 2-input Boolean functions and discarding the input permutations and output polarity (i.e., by enumerating their NP-equivalent classes [1]), one needs to realize the following:

- 2 constant functions (1 and 0),
- 2 buffer functions (BUF, INV),
- 2 XOR functions (XOR, XNOR),
- 5 AND functions (AND, NAND, one input inversion),
- 5 OR functions (OR, NOR and a single input inverse).

To have a complete reversible cell library, it suffices to realize the library of reversible embeddings for each of the above equivalence classes. For instance, for a two-input AND gate, Figure 3 shows two simple reversible embeddings. Since AND and OR functions produce identical output function values in three cases (f_{same} =3), the number of added (garbage) bits is $\lceil log_2 3 \rceil = 2$.

In the most straightforward embedding of AND, Figure 3.a), two primary inputs get repeated at the output, from which they can always be recovered. The third, ancilla input is a constant (0). A more flexible embedding is by the Toffoli gate, Figure 3.b), where the additional control input C changes the polarity of R. Here, the same reversible gate acts as both AND and NAND gate, covering both output polarities of the same function equivalence class.

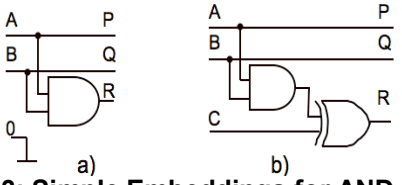

Figure 3: Simple Embeddings for AND Gate

Please note that each such embedding can naturally be done in the optimal way with respect to the total number of gate ports. For AND and OR function classes, these would be implemented by 3x3 blocks, for XOR class, 2x2 reversible gates would suffice, while for BUF/INV, the 1x1 gates would be used.

The reversible cell library can also have more powerful and flexible reversible embeddings, including cells that realize two distinct functions at their outputs. For a two-input AND gate, Figure 4 shows five single- and multi-output embeddings. Figure 4.c) contains a dual AND/OR gate, while in Figure 4.d) the polarity of one of the outputs is controlled by the input C. Finally, Figure 4.e) has both polarities controlled. Note that the cell c) is a special case of both d) and e), where C is assigned 0.

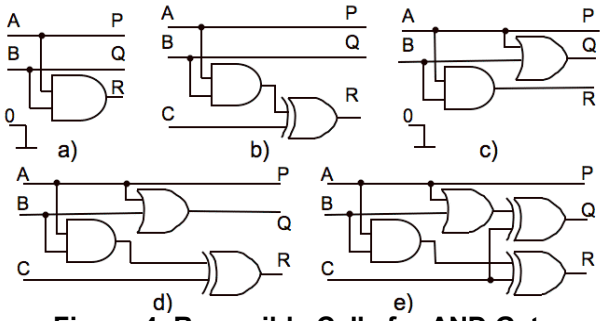

Figure 4: Reversible Cells for AND Gate

Table 1 compares the five proposed reversible embeddings of the AND gate. While cases c)-e) provide two output functions in the same cell, they can fan out only one input (pass one input to the output). Cells b), d) and e) further provide programmable output inversion.

Table 1: Comparing Embeddings of AND Gate

Cell in Figure 4	a)	b)	c)	d)	e)
Output functions	1	1	2	2	2
Output polarities	1	2	1	2	2
Input passed to out	2	2	1	1	1

4. Reversible Technology Mapping

We now present a method of building the reversible implementations of circuits in a manner equivalent to the classical technology mapping, where a Boolean network is to be covered by a network of gates in a manner that the overall network is reversible. The key benefit of this approach is that there is no need for finding a reversible embedding prior to the synthesis, and for undertaking costly permutation mapping optimizations [5], [7]. In consequence, much larger circuits (in terms of input count) can be created.

In reversible technology mapping considered here, the goal is to map a non-reversible circuit into a network of reversible cells, such that the overall network is reversible. This goal requires the following:

1. Network mapping using the classical minimization methods and classical, irreversible cell library l,
2. Realizing reversible embedding library l_r of l, (the library can be reused for mappings of other circuits).
3. Covering (reversible mapping of) the irreversible network from Step 1 by a reversible cell network l_r.

During the actual reversible mapping, there is a choice in selecting the cells, as any reversible cell realizing the given function from l can be applied. For example, any of the 5 cells in Figure 4 can be used for mapping an AND function. The OR function can be implemented in a reversible form by cells in Figure 4.c-e), XOR function can be achieved by cells in Figure 4.b and -Figure 4, d-e). A better XOR embedding, however, is a 2x2 Toffoli gate, Figure 1, performing mapping $(a,b) \rightarrow (a, a \oplus b)$.

In Step 3 above (covering process), the first concern is to ensure that a resulting network is a reversible function itself. Towards this goal, we prove the following property of the networks composed of reversible gates.

Lemma 1: *An acyclic network N of reversible cells is itself a reversible logic function if:*

*-**(Fanout-free)** Fanout of all nodes is kept at 1.*

*-**(Connectedness)** There is a path between all network signals to both primary inputs and outputs of N.*

Proof: Consider the primary outputs of a network *N*. Since all gates of *N* are reversible, the inputs to those gates can be recovered fully from the outputs. The process starts with the primary outputs of *N*, and continues through the backward traversal, using the fact that each internal signal line (including extraneous bits added in reversible synthesis) is on a path from the outputs. Further, as there is no fanout to eventually require conflicting values, and no cycles in *N*, this means that one can always traverse the whole network *N* from any location toward the primary inputs, and, in consequence, recover the values of the primary inputs. Hence, the network is reversible. ■

Lemma 1 can be directly applied to realize a reversible technology mapping from an irreversible network that is mapped onto an acyclic network of gates from the cell library *l*. All that is required is to replace in topological order, the library cells by their reversible version from l_r. Each replacement must naturally maintain the connectedness and fanout conditions from Lemma 1.

The extraneous bits can actually be used to provide the fanout, as required by the original irreversible network. For example of AND function, the garbage outputs (signals *P* and *Q*, Figure 4.a-b), can be used to fan out the inputs which are unaltered inputs to the gate, or even some logic function of inputs to the gate (signal *Q* in Figure 4.c-e)). If these bits, however, cannot be reused elsewhere, then they automatically create extra garbage outputs of the overall circuit.

With extraneous input bits, the situation is somewhat different. While there still need to be a dedicated path from primary inputs to each such extraneous bit, in this case, Lemma 1 simply implies that no such input is left detached from the augmented set of primary inputs.

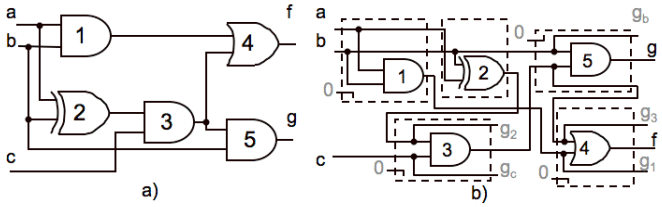

Figure 5: Reversible Mapping Example

***Example* 1**: *Consider an irreversible circuit in Figure 5.a). In the construction of a reversible mapping, Figure 5.b), each original gate i is replaced with a reversible library cell (dashed box around the corresponding gate i). The cells are interconnected such that the fanout is kept at 1. For three higher-fanout nodes, the input a and the output of node 3 is replicated through a reversible implementation of the AND gate 1, while input b with fanout 3 needs to be replicated twice through a cascade of two reversible gates – AND gate 1 and XOR gate 3. The extraneous inputs are also added to each AND and OR gate. Usually these would be the control inputs, however, when they are assigned a fixed value, it is denoted in Figure 5.b) by a grounded "0". The garbage outputs g_j (in grey color) are all reversible gate outputs that do not get used in the network. By counting the number of input, output and extraneous signals upon reversible technology mapping, we note that Eq. 1 holds, with A=4, G=5.*

4.1. Assessing Number of Extraneous Bits

With the proposed reversible technology mapping, we can provably know the number of extraneous bits generated. First, there is a relation regarding the number of extraneous bits and the type of the cells employed in the technology mapping.

Lemma 2: *The total number of extraneous input bits A added by the reversible technology mapping for 2-input irreversible library l is equal to the number of reversible cells for AND or OR functions:*

$$A = \#(AND/OR)$$

Proof: Among all irreversible 2-variable function NP classes, only the AND/OR function class gates require 1 extraneous input bit during a reversible embedding. ■

We can bring more relations between the extraneous bits and the irrreversible gate count in the original network. Applying previous lemma to identity from Eq. 1 proves the following relation.

Lemma 3: *The total number of output garbage bits G is*

$$G = (\#\,inputs) + (\#[AND/OR]) - (\#\,outputs).$$

From this, we conclude that the added cost of both ancilla input bits and garbage output bits is due to the use of reversible library cells that represent AND or OR gates (including their inversions and input permutations). The number of extraneous signals can be reduced if the classical irreversible technology mapping can avoid AND and OR gates, and instead use XOR gates. We will further show in Section 4.3 that the choice of reversible cells can also reduce the extraneous signal count through packing of multiple AND and OR functions.

4.2. Solution to the Fanout Problem

The most obvious obstacle in designing any reversible circuit lies in the inability to fan out (i.e., distribute) the node outputs in the network as traditionally understood, Figure 6.a). The solution to the fanout problem should be the least costly in terms of additional gates and ancilla/garbage bits. With our reversible cells approach, the inputs that are replicated at the cell outputs as garbage

bits are a vehicle to provide an increased signal fanout. When such signals are fed to a reversible cell downstream, then, up to some limitations discussed shortly, we obtain their copy free of charge for further use in the network, Figure 6.b). To distribute (fan out) signal *x,* we then construct a cascade of reversible gates. Logic function of each gate in the cascade is arbitrary; however, each gate must be capable of distributing further *x*.

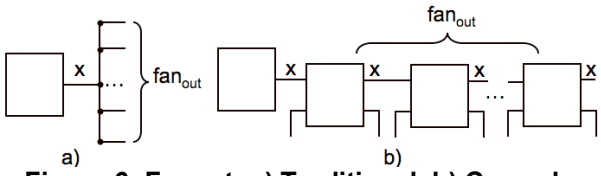

Figure 6: Fanout: a) Traditional, b) Cascade

For all single-function AND/OR cells in Figure 4 (including Toffoli gate), this approach can fan out both signal inputs to the next cell, while for multi-function, as well as XOR/XNOR gates, only one input can be distributed in this way. When both primary inputs need to be replicated, it is critical to devise a mechanism to drive both such inputs in the fanout cascades. There are two cases to consider in enlarging the number of inputs to be replicated.

Case 1): A packed, two-function, AND/OR cell is replaced by a cascade of two simpler cells, one for AND and one OR, with the primary inputs to the second cell in the cascade being provided by outputs of the first one, which can now fan out both inputs. In this case, the optimization of packing two functions, AND and OR, in a single cell cannot be applied.

Case 2) XOR cell provides only one garbage bit to be used as a fanout of one of XOR gate inputs. To allow both inputs to be fan out, we construct a XOR "fanout gadget", Figure 7, where the additional reversible XOR gate is inserted to recover and fan out the second output from the XOR gate. This is the only case in which an extra gate is added to provide the fanout.

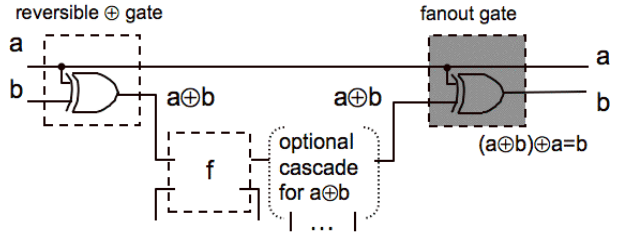

Figure 7: XOR Fanout Gadget

Lemma 4: *The length of the cascade $l_{cascade}$ required for providing fanout for a single-function node i, is lower-bounded by its fanout, and upper bounded by the fanout plus the number of XOR gates in its fanout.*

$$fanout_i \le l_{cascade} \le fanout_i + \#(XOR)_{fanout_i}$$

and the total length of all fanout cascades is bounded by

$$fanout_i \le Total.l_{cascade} \le fanout_i + \#(XOR)$$

Proof: Among reversible library cell elements, when AND/OR gates pass both signal inputs to the outputs, the passed signals can be used again through a fanout cascade. If, however, the library cell passes only one input to the output, a longer cascade might be needed. Only in the case of a XOR cell, a new fanout gate (XOR gadget) might be added, hence the upper bounds. ■

The upper bound on the length of the cascades is reached only when we need to fan out both inputs to all XOR gates. The synthesis can exploit this fact, to place the XOR nodes at the end of the fanout cascades.

One special case to consider is when an inverter INV is used in one of the fanout branches. Since INV is reversible by itself, it has only one output, the fanin signal cannot be distributed beyond reaching an inverter. The inverter is then either used at the very end of the fanout cascade, or it gets subsumed in the logic downstream.

4.3. Gate Packing Optimization

With the availability of the cells that realize multiple functions, such as AND and OR, the reversible technology mapping can perform another optimization, i.e., packing two functions into a single reversible library cell. This operation reduces the total number of gates, as well as and the number of garbage bits. One downside is that if more than one input to the gate needs to be fan out, packing will disable this possibility, and should not be applied in that case, as explained in previous section.

We express the conditions for gate packing in Table 2. The two logic functions f_1 and f_2 are described with their ON-set cubes where *a*, *b* is either 0 or 1, while "-" stands for "don't care". The case in the first row can be packed in the Universal AND/OR Reversible Gate, while the second case can be fit in the same cell plus an inverter, or, even better, into a cell in Figure 4.d). If the reversible library contains an AND/OR cell in which only one input to an AND gate is inverted, then the last two cases fit into one such cell.

Table 2: Packing Two Functions - Cube Rules

	f_1	f_2
Direct packing (cell in in Figure 4.e)	*a* - - *b*	*ab*
Invert out (De Morgan rule) (cell in Figure 4.d)	*a* – - *b*	*a*'- - *b*'
Invert input (cell with inverter before AND gate)	*a* – - *b*	*a'b*
Invert input/output (cell with two inverters)	*a* – - *b*	*a*- - *b*'

5. Experimental Results

The methods described in this paper have been implemented as a software add-on to the technology

mapper *map* of the Berkeley SIS program. We used MCNC benchmarks, for which the synthesis has completed in seconds for all but two multi-level benchmarks on a 450MHz Sun Ultra 80 workstation.

Since the algorithm operates on irreversible networks and apparently can process larger networks than other reversible synthesis algorithms, a direct comparison is not possible. The purpose of the experiments was to assess the significance of the reversible cell library selection, the impact of the initial irreversible technology mapping and that of packing an AND and OR gate (and its single-input inverted and permuted versions) into a single library cell.

The technology mapping results in Table 3 are reported in terms of the number of reversible library gates, including inverters. The reversible circuit cost of the proposed library cells has not been assumed, as this is technology dependent and the object of further studies [12]. We also performed the classical technology mapping aware of the fanout limitations in reversible circuits.

The first column (label *s*) reports the total cell usage using a small cell library, consisting of NOT, AND, OR and XOR gates with non-inverted inputs. In this case, as the inversions of individual inputs are not provided, inverters need to be inserted by technology mapping. The added use of universal AND/OR gate, by which the And and OR gates can be packed is reported in the third column ($\mathbf{s_{pack}}$). The next two columns report the resource usage in the case of fanout restriction (SIS technology mapping with a *–f* 0 option), considered with and without the gate packing. The penultimate column reports the use of a larger (*L*) cell library that includes the gates for all individual input inversions. Again, the packing impact is reported in L_{pack}. In this case, the input inversions create additional instances where packing can happen.

Table 3: Mapping of MCNC Benchmarks

Circuit	*s*	s_{pack}	*s*0	$s0_{pack}$	*L*	L_{pack}
apex6	717	715	741	738	591	587
C1355	326	271	450	384	246	212
C2670	617	582	742	687	547	514
C3540	1037	1007	1051	1014	928	895
C5315	1478	1401	1601	1482	1306	1221
C6288	2555	2216	2795	2359	2138	1899
C7552	1848	1830	1979	1752	1630	1470
i10	2175	2100	2243	2144	1851	1778
pair	1447	1389	1483	1399	1260	1213
x3	732	730	755	749	603	601

6. Conclusions and Future Work

The work presented in this paper provides techniques for synthesizing reversibly the circuits of sizeable complexity by mapping the original irreversible circuit specification into a library of reversible cells. Up to our best knowledge, this is the first technique that allows the reversible synthesis of circuits such as multi-level MCNC benchmarks in compute time that is comparable to that of the classical technology mapping.

To achieve this goal, we have identified the rules for creating reversible library and the technology mapping producing a reversible implementation of any irreversible netlist. Created reversible cell libraries include universal cells capable of packing two irreversible gates into a single reversible library cell. We have proven the conditions for mapping the irreversible netlist into a network of reversible gates, such that the result is a reversible netlist that overcomes the inherent fanout limitation by using efficiently the extraneous pins.

It is further shown how the irreversible technology mapping using the classical cell library can impact the result. To reduce the number of extraneous pins, the technology mapping should prefer XOR gates. On the other hand, to reduce the length of the fan out, the mapping should favor AND/OR gates, or XOR gates by which only one input needs to be further distributed.

In future, scalable and robust technology mapping techniques could address closer the interaction between gate packing and fanout cascade use, as well as the extraneous bits. Other circuit features, such as the speed and reversible gate cost should be considered as well.

7. References

[1] T. Toffoli, "*Reversible Computing*", Technical Memo, MIT/LCS/TM-151, Boston, 1980.

[2] V. Shende, A. Prasad, I. Markov and J. Hayes, "Synthesis of Reversible Logic Circuits", *IEEE Trans. CAD of Integrated Circuits and Systems*, Vol. 22, No. 6, Jun. 2003, pp. 710-722.

[3] G. De Micheli. *Synthesis and Optimization of Digital Circuits*, Mc-Graw Hill, 1994.

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

[5] D. Maslov and G. Dueck, "Reversible Cascades with Minimal Garbage", *IEEE Trans. CAD of Integrated Circuits and Systems*, Vol. 23, No. 11, Nov. 2004, pp. 1497-1509.

[6] D. Maslov, G. Dueck and D. M. Miller, "Toffoli Network Synthesis with Templates", *IEEE Trans. CAD of Integrated Circuits and Systems*, Vol. 24, No. 6, Jun. 2005, pp. 807-817.

[7] P. Gupta, A. Agrawal and N. Jha, "An Algorithm for Synthesis of Reversible Logic Circuits", *IEEE Transactions on CAD of Integrated Circuits and Systems*, to appear, 2006

[8] P. Kerntopf, "Synthesis of Multipurpose Reversible Logic Gates", *Proc. Euromicro Symposium*, Sep. 2002, pp. 259-266.

[9] M. Perkowski, P. Kerntopf, "Regular Reversible Lates", *Proc Euromicro Symposium*, Sep. 2001 pp. 245 – 252.

[10] A. Khazamipur and K. Radecka, "Adiabatic Implementation of Reversible Logic", *Proc. Midwest Intl. Symposium on Circuits and Systems*, pp. 291-294, ,2005.

[11] W. Hung, X. Song, G. Yang, J. Yang and M. Perkowski, "Optimal Synthesis of Multiple Output Boolean Functions using a Set of Quantum Gates by Symbolic Reachability", *IEEE Trans. CAD*, to appear.

[12] D. Maslov and M. D. Miller, "Comparison of the Cost Metrics for Reversible and Quantum Logic Synthesis, Quant-Physics Preprint, http://arxiv.org/pdf/quant-ph/0511008.