**SYNTHESIS OF REVERSIBLE CIRCUITS**

**WITH NO ANCILLA BITS FOR LARGE REVERSIBLE FUNCTIONS SPECIFIED WITH BIT EQUATIONS**

Nouraddin Alhagi, Maher Hawash and Marek Perkowski

Portland State University

**Abstract.** The paper presents a new algorithm MP (multiple pass) to synthesize large reversible binary circuits without ancilla bits. The MMD algorithm requires to store a truth table (or a RM transform) as a 2n vector for a reversible function of n variables. This representation prohibits synthesis of large functions. However, in MP we do not store such exponentially growing data structure. The values of minterms are calculated in MP dynamically, one-by-one, from a set of logic equations that specify the reversible circuit to be designed allowing for synthesis of large scale reversible circuits (30-bits) which is not possible with existing algorithms. In addition, our unique multi-pass approach where the circuit is synthesized with various, yet specific, minterm orders yields optimal solution. The algorithm returns a descirption of the optimal circuit with respect to gate count or quantum cost. Althought the synthesis process is relatively slower, the solution is found in realtime for smaller circuits of 8 bits or less.

1. **Introduction**

There are currently two types of algorithms to synthesize reversible circuits: (**T1**) those like MMD [1,3,4,7,9-11,14-22] that start from a reversible specification, (**T2**) those like [6,12,13,23,24,26,28-33] that start from non-reversible specification and create ancilla bits. The second type of methods has been successful for large functions [5,6,31-33] but solves basically a different problem. The MMD algorithm [21] *(Miller, Maslov and Dueck)* is currently the leading reversible logic synthesizer if no ancilla bits are used. MMD uses the permutation vector-like reversible function specification as its data which corresponds to a truth table that is explicitly used in the synthesis process and, thus, must be stored and processed in memory. Since it is intrinsically bound by the natural binary order of mintrems, and hence does not use sarch, MMD cannot be enhanced through better search algorithms or iterative/recursive routines. Since it process a single minterm order, and, hence, MMD is reasonably fast and it distinguishes itself among other programs of this type because it achieves (theoretical) 100% convergence regardless the problem size [21]. Practically, however, it can be applied to at most 8 qubit reversible functions and very few reversible functions with more than 8 variables were presented as MMD benchmarks in the literature. It was found both in our research and by other researchers that the complexity of both the synthesis process and the average circuit sizes synthesized by MMD grow very quickly above 8 qubits, herin “large circuits”. In our research, it was difficult to evaluate the quality of our results for large circuits from reversible specifications chiefly due to the lack of a single solution for comparison. Consequently, with this paper, we set the benchmark for future research. (observe that standard non-reversible specifications are used in recent papers [27-33], and we need reversible functions such as specified by permutations). In any case, at this time MMD program is the current benchmark for the evaluation of programs for reversible circuit synthesis with no ancilla bits. A strong asset of the philosophy used in MMD, in contrast to those used in other programs is that MMD gives a warranty of convergence if the data is small enough for MMD to be able to keep them in memory. Due to the known fact that the quality of MMD may be very low for functions where the exact minimal solution is known, several research groups are constantly attempting to improve on the MMD algorithm. Agrawal and Jha’s algorithm [1,6] uses the number of terms in the Positive Polarity Reed-Muller (PPRM) expansion of synthesized functions as its cost function. As PPRM can be stored by an expression that is shorter than 2n their algorithm could in theory minimize larger functions. On the other hand this algorithm has to store many PPRM equations as it represents a tree-search algorithm. Also, non-factorized PPRMs may be in many cases of similar complexity to truth tables, for instance for function *f=a’b’c’d’.* Some of the algorithm variants from [1,3,7,9] have trouble with convergence and there is a trade-off between provable convergence and size of circuits that can be minimized. A challenge thus still exists to create an algorithm that could trade-off quality for time, but with a provable convergence for every function. In this paper we will present such an algorithm.

After many failed attempts at creating better minimizers based on other search strategies [2,13,24,26], we decided to improve MMD. The main weakness of MMD is that it is limited to functions of the size that their truth table (exponential size) can fit in memory. This limits practically MMD's approach to about 13 variables. Because of its design principle, even with big speed penalty MMD just cannot minimize larger functions. Thus an improved algorithm has to use an entirely different representation. When it was decided to use an internal representation other than a truth table or a spectrum with 2n minterms, the problem was “what is the best representation that would still guarantee convergence?” Kerntopf used a new type of decision diagrams but did not prove the convergence and, as a result, his method only worked for 3 variables. In unpublished research we used ESOPs and FPRMs rather than PPRM but we were not able to find a heuristic that would work better than the variants from [1,3,7,9]. Other cascade types have been also proposed in the newer versions of composition-based search approaches [11,23,24,2] but there were troubles with either the size of solutions or convergence. Here we present a search method that is both convergent, allows for synthesis of large functions, and produces near minimal solutions. This algorithm includes variants which are various generalizations of MMD.

**2. Explanation of the main idea of MMD**

To make the paper self-contained we give a brief overview of MMD. More in [4,14-22].

The main idea of all algorithms for reversible circuits synthesis of type T1 is to transform bit-by-bit a reversible function to its identity function.

Example 2.1. Fig. 2.1 illustrates the basic flow of MMD algorithm. The first column lists all input minterms of the function in the natural numerical order(linear): 0, 1, 2, 3, etc. The second column in Fig. 2.1 lists values of the output vectors that correspond to the input vectors from the first column. For instance, that the input minterm a’ b’ c’ = 000 is mapped to the output minterm A’ B’ C’ = 000 and input 001 is mapped to the output minterm 100. Self-mapping minterm are minterms with matching input and output values (e.g., minterm 000) The synthesis process applies successive gates to the output column (ABC), bit-by-bit, to generate the corresponding minterm of the input column (abc). Recall that Toffoli (Feynman) gates are used that are self-inverse gates (M-1 = M), so they process information the same way from inputs to outputs and from outputs to inputs. The MMD algorithm shown here is thus the “backward searching” or “output to input searching” algorithm. Since the first minterm is self-mapping, MMD skips to the second minterm applying a controlled- Feynman gate to *bit c*, shaded, conditional on *bit a* being set, underscored. After the application of each gate, the output column minterms (of intermediate functions) become more and more similar to the first column – the column of input vectors. The question is *“what does it mean to be more similar?”* It is an advantage of general search methods that various measures of complexity or coincidence or similarity have been used [9,10,11,24]. This may lead to better and faster solutions but it is hard or impossible to prove convergence. The MMD algorithm has however a very simple and working solution to this problem. It requires that intermediate columns remain exactly the same as the input column in some subset of rows from the top. The *completed rows*, start from row 0, then row 1, row 2 etc. up to the minterm under construction When some subset of rows from top are completed, they are not allowed to change (shown in shaded areas in Fig. 2.1) which is guaranteed by the selection of proper control bits. This is the main idea of MMD algorithm and actually the only algorithmic idea of this method (excluding templates). The proof that this algorithm is convergent is obvious, as every step creates one more bit in a row from top that is the same in the intermediate column as in the first column. This way, after at most n \* 2n - 1 steps (intermediate columns) the last column becomes exactly the same as the first column, and thus, the remaining function to be realized is an identity function (a better bound was also proven by Maslov but it is not relevant here). As we see, the strength of this algorithm is easy convergence, but since the complexity is exponential, MMD is limited in application to a small number of bits. So far, however, MMD continues to represent the benchmark to meet as no better algorithm had been proposed. The symbol *a 🡪 c* in the column 1 means that whenever *a = 1* in the previous column, the *bit c* is flipped from 0 to 1 and from 1 to 0. Hence, this transition from column to column executes the Feynman gate c = c ⊕ a. The reader may check that the number of completed rows is either the same or larger from column to column. In this example the upper complexity bound is *n \* 2 n – 1 which* for our 3-bit example yields *(3 \* 2 3 – 1) 23* gates. Note that our example simulation resulted in only 6 gates. Here MMD happened to work well. But there are examples [2] where the gate number is close to the upper bound although the minimal number of gates is lower.





*Figure 2.2: The solution circuit found from MMD in Fig. 2.1 drawn and created from outputs to inputs. The arrow shows the flow of signal from inputs to outputs. This method is possible because each reversible gate used in this figure is its own self-inverse.*

3. MMDS and MMDSN Orderings

The main concept of MMD, the natural binary minterm ordering was challenged in [26] as the only 100% convergent order. It was found that MMD’s minterm ordering falls into a subset of orderings that do not exhibit certain important property that was called the “control line blocking”. This observation lead to the creation of the “MMDS ordering” [26]. To make this paper self-contained, all these ideas will be defined below but first we need to motivate the new concepts. Without any backtracking, any bi-directional search or any template matching, the MMDS ordering used exhaustively were superior for 3-bit circuits [26]. The MMDS orderings can be used with any number of inputs and have larger gains compared to MMD when the number of inputs increases. However, the number of MMDS orderings is too high to use all these orderings for synthesis. In this paper, we introduce a subset of the MMDS ordering, herein MMDSN orderings, which greatly reduces the number of terms examined while providing near minimal solution superior to MMD.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| abc | ABC | 1 | 2 | 3 | 4 | 5 | 6 |
| 000 | 000 | 000 | 000 | 000 | 000 | 000 | 000 |
| 001 | 100 | **1**01 | 00**1** | 001 | 001 | 001 | 001 |
| 010 | 101 | **1**00 | 100 | **1**10 | 0**1**0 | 010 | 010 |
| 011 | 001 | 001 | 10**1** | **1**11 | 0**1**1 | 011 | 011 |
| 100 | 110 | **111** | 01**1** | 011 | 1**1**1 | **1**01 | **1**00 |
| 101 | 010 | 010 | 010 | 010 | 1**1**0 | **1**00 | **1**01 |
| 110 | 011 | 011 | 11**1** | **1**01 | 101 | **1**11 | **1**10 |
| 111 | 111 | **1**10 | 110 | **1**00 | 100 | **1**10 | **1**11 |
|  |  | *a🡪c* | *c🡪a* | *a🡪b* | *b🡪a* | *a🡪b* | *a🡪c* |

*Figure 2.1 MMD method illustrated with truth tables of intermediate functions. Notation a 🡪 c means c = c* ⊕ *a means “ flip c if a=1”. Control lines are underlined and affected bits are shaded.*

MMD stipulates that the function is arranged in a natural binary code order by inputs assignments. Each iteration adds a gate in order to correctly transform the outputs to match the inputs without changing any of the previously *completed* (from top row) output minterms. Other innovative algorithms utilized greedy algorithm where gates are chosen to reduce the cost function from input to output. For example, Hamming Distance determines the choice of gates to transform the output function to the original function or to identity function. Such algorithms did not always converge, unlike, MMD, which, as it might give the worst solutions, it always converges. The question is, how these two main ideas of natural ordered search of MMD and greedy search can be combined to improve the quality of results and always achieve the convergence. Such combination is the goal of our research, part of which is discussed here.

The good ordering should not conflict with the main MMD’s idea [21,22] of not changing any previously set outputs. This idea is also what guarantees MMD’s convergence.

***Definition.1.***

***Control Line Blocking condition occurs when all control lines of the current minterm are a subset of the control lines of a previously completed minterm in the input order.***

When this condition occurs it makes it impossible to change any output bits during the current iteration without altering the output bits which have been previously *completed*. Occurrence of this condition hinders convergence.

***Mathematical Check =>***

***if #later = #later & #earlier***

***then there is control line blocking***

Example of control line blocking  
 101 = 101 & 111

Example of no control line blocking  
 001 = 101 & 011

Therefore, any ordering of inputs that does not lead to the occurrence of the blocking condition can be used in an improved MMD algorithm. The method to find all non-blocking permutations for any number of inputs was found in [26]. No control line blocking seems to be a very restrictive rule. For a three-input function there are initially 8! (40,320) permutations. Therefore there are the same number of various orderings. Instantly that number is reduced to 6! Since 000 must come first and 111 must come last. Using the software, 48 permutations, called MMDS orders, were found to exhibit no control line blocking for all 3\*3 reversible functions. Included in this set is the original MMD ordering.

The binary vectors of cells (minterms) of a 3 \* 3 reversible function can be represented as a well-known Hasse Diagram, where a bit-by-bit domination relation ( 1 ≥ 0, 1 ≥ 1, 0 ≥ 0) is used as an ordering relation (see Fig. 3.1a, b). While binary vectors are used in Fig. 3.1a, the Fig. 3.1b uses natural numbers being counterparts of these binary vectors. The rule says “*never to take a dominating node (number) before a dominated node”.* Thus 5 cannot be taken before 1, for instance. As we see, MMD order satisfies these rules. Another good orders are shown in Figs. 3.1c and 3.2.

As the number of input lines increases, the number of non-blocking orderings increases exponentially. For functions with four inputs, Stedman [26] reported that 78,880 different non-blocking permutations exist. We however discovered that 1,680,382 such non-blocking permutations exist. As the amount of non-blocking orders increase so does the optimality of the MMDS orderings, and as a result, the time required to synthesize. With MMDSN order, a set of rules were created to distill the best possible control choices from the set of all possible control line choices, as follows:

1. The target bit cannot be used to control the current transformation,
2. Use minimal number controls bits necessary to flip the target bit,
3. No past outputs can be changed,
4. Process 0 🡪 1 transitions first to maximize availability of control lines, and hence, guarantee convergence.

The control possibilities are then sent to the gate choice function to produce a circuit. Currently gate choice is based on Hamming Distance but it can be any cost function [1,2,10,11,12,24,31]. Using control line blocking as the only rule, a subset of all input orders can easily be found, and it can be easily proven that all non-blocking input orders will converge for all output permutations.

*Fig. 3.1: New orders for MMD-like synthesis. (a) Hasse diagram with binary vectors, (b) Hasse diagram with natural numbers, (c) Ordering of nodes that violates the MMD order, illustrated on the Hasse Diagram. This is however a valid MMDS ordering.*



*Fig. 3.2: New ordering 02134657 for MMD-like binary synthesis, a valid MMDS order which is consistent with the Hasse diagram relations of order.*

**Theorem 1.**

All non-blocking input orders converge for all output permutations.

*Proof of Convergence:*

Convergence is guaranteed in MMD and MMDS because at any given point in the algorithms all following output bits are able to be changed without altering any previously set outputs. This is guaranteed because the input orders do not exhibit control line blocking. With MMD and MMDS’ methodical approaches, as long as all output bits can be changed without altering any previously set outputs these algorithms will converge every time.

MMDS set of orders is a superset of MMD's. Our improved algorithm uses multiple MMDS input orders that exhibit no control line blocking. Included in these orders is the MMD natural binary order. MMDS ordering algorithm performs the same bit manipulating strategy for all non-blocking input orders, and reduces the circuit more than the standard MMD algorithm. This outcome is obvious, given that MMD is a subset of MMDS, so it can perform no worse than MMD.

**Definition 2.** MMDSN order is one in which the minterm 00…0 is generated first, followed by all minterms with a single one(1) in random order, followed by all minterms with two ones (1’s) in random order, and so on, succsively incrementing the number of ones (1's) in each band until we finally reach the minterm 11…1.

Example for 3 variables: MMDSN order is for instance: 000, 100, 010, 001, 110, 101, 011, 111. This is also a MMDS order but not MMD order.

**4. MP Algorithm**

Earlier attempts at improving MMD algorithm resulted in very good/minimal solutions for some circuits or non-converging/incorrect circuits for others [2,10,11,24.26]. Thus the order of selecting outputs to be covered by gates was found experimentally to be more important than the gate heuristics to choose gates. For larger number of variables, a variant of our algorithm was created based on the following principles:

***(1)*** Rather than maintaining a set of tables mapping inputs to outputs, the algorithm creates these columns implicitely, simulating minterms one-by-one. The simulator uses the equations from the specification together with the part of the already constructed reversible circuit. To demonstrate the concept, imagine two circuits similar to Fig. 2.2 cascaded back to back and simulated from inputs at each stage of minterm transformation. The first circuit, described by equations, represents the function under synthesis, and the second circuit is the outcome of synthesis. (in reverse order of gates). When the synthesis process completes, two equivalent circuits, one mirror of the another exist, where the first circuit is specified by equations, and the second by reversible gates (in reverse order of gates). When we simulate this composed circuit, for every input minterm, the same minterm is obtained at the outputs of the concatenated circuits, and hence, the concatenated circuits together are a reversible identity. Since the circuits mirror one another, the solution is represented by the second circuit of the concatenated whole.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **# bits** | **Function** | **MMDs # of gates** | **MMDs qcost** | **MP # of Gates** | **MP Qcost** |
| **4** | **ou4\_4file** | **30** | **162** | **25** | **85** |
| **4** | **ouhwb4file** | **24** | **154** | **18** | **58** |
| **5** | **ouhwb5file** | **64** | **914** | **49** | **421** |
| **6** | **ouhwb6file** | **162** | **4036** | **139** | **2341** |
| **7** | **ouhwb7file** | **374** | **13893** | **327** | **10007** |
| **7** | **ouham7file** | **324** | **13145** | **302** | **10683** |
| **8** | **ouhwb8file** | **995** | **58605** | **960** | **45273** |
| **9** | **ouhwb9file** | **2249** | **188997** | **2140** | **144481** |
| **10** | **ouhwb10file** | **4953** | **538588** | **4836** | **463578** |
| **11** | **ouhwb11file** | **10929** | **1412439** | **10908** | **1309776** |

*Table 2. Comparison of numbers of gates and quantum costs of MMD and MP algorithms for reversible functions with various numbers of qubits. This is “large circuits” variant with k=10. No ancilla bits.*

***(2)*** A number k of randomly selected MMDSN orders are generated representing the function under synthesis. The solution with optimal cost is selected with the possibility of backtracking if the temporary cost exceeds the minimum cost determined earlier in the process.

***(3)*** When possible, template matching method from MMD is used on the result for post-processing to further improve the quantum cost.

1. **Results of MP for Four Variables.**

For functions of four variables, new benchmarks NA1 – NA50, we created a variant of algorithm that generates all MMDS orders and finds the best order, i.e. the reversible circuit with the minimum cost. The results for functions of four variables are shown in Table 1 (we created our own benchmarks since there are very few 4-variable benchmarks in literature). This table demonstrates that algorithm MP is better than MMD in terms of total quantum costs. It is slower, though. In few cases that MMD was better than MP in Table 1 were because the MMD used template matching local optimization and MP was not using any local optimization after the search. However, for small functions we can always use the original MMD after MP algorithm to use template matching and achieve better results. Observe that MMD runs just once and has no search. Thus running MP with all or at least many orders and next using MMD-based template matching creates algorithm that is always better than MMD but running slower. It would be an easy way “to win” with MMD. However, additional advantage of MP approach is that we can have a trade-off – the longer we run the new combined algorithm the better is potentially our result. This property is missing in both MMD and Agrawal/Jha approaches.

1. **Results of the MP for more than four Variables.**

Table 2 shows the results with k=10. The user can set k to any value to get the trade-off between synthesis time and quantum cost improvement. So far we were not able to compare MP with MMD on larger functions as MMD does not accept functions of 30 variables being not able to store a vector with 230 rows. As we see in Table 2, the improvement here is best for functions with less than 7 variables, which means that k should be increased. To understand the limitation of our approach for very large functions we created 10 randomly generated reversible functions of 30 variables each [2], which were entered as equations for each bit separately (this variant of MP is not format compatible with MMD and other programs). Such large benchmarks do not exist in literature. For instance, function Chal30, shuffled 1073741824 times (230 times), number of gates generated 430296 (calculated by our program not RCV, circuit was so large that quantum cost software did not work). We run 20 orderings, it took 1 hour and 9 minutes to run. Results for other benchmarks from this set are similar. A web page with these new big benchmarks will be created.

1. **Conclusions.**

We presented a new algorithm MP to synthesize reversible circuits in the spirit of MMD. As the algorithm is a generalization of MMD, it can never create solutions worse than those by MMD. But it can create results of smaller cost and can find solutions to problems that are too large for MMD to handle. Our algorithm does not require to store the large truth table or other exponential representation as it calculates the values in the run from the logic equations. Although MP still needs an exponential number of simulations, it does not need to store exponential data. Also we use many orders of minterm creation which leads to more efficient circuits. However, we pay the price of a slower synthesis process. The results have been also extended to synthesis of incompletely specified functions and ancilla bits [2] and state machines [12,13]. As the reversible logic is still a research rather than industrial topic, speed of obtaining the solution seems to be less important than exploring larger circuits and being able to evaluate their quality. The trade-off that exists in MP between the time and cost of solution helps in this research.

**References**

1. A. Agrawal, and N.K. Jha, Synthesis of reversible logic, *Proc.* *DATE,* Paris, France, Febr. 2004, pp. 1530-1591
2. N.Alhagi, A Synthesis Method to Synthesize Reversible Functions using Arbitrary Gates and Relational Specification, *Ph.D. Thesis*, PSU, 2010.
3. J. Donald and N.K. Jha, Reversible Logic Synthesis with Fredkin and Peres Gates, ACM Journal on Emerging Technologies in Computing Systems, Vol. 4, No. 1, Article 2, March 2008.
4. G. W. Dueck and D. Maslov, "Reversible Function Synthesis with Minimum Garbage Outputs," 6th International Symposium on Representations and Methodology of Future Computing Technologies (RM), Trier, Germany, March 2003, pp. 154-161.
5. D. Große, X. Chen, and R. Drechsler, Exact Toffoli network synthesis of reversible logic using boolean satisfiability, *Proc. IEEE Dallas/CAS Workshop,* pp. 51–54, 2006.
6. D. Große, R. Wille, G.W. Dueck, and R. Drechsler. Exact multiple control toffoli network synthesis with SAT techniques. *IEEE Trans. on CAD*, 28(5):703–715, 2009.
7. P. Gupta, A. Agrawal, and N. Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Trans. on CAD,* 25(11):pp. 2317–2330, 2006.
8. J. Hu G-S Ma, G. Feng, Efficient Algorithm for Positive-polarity Reed-muller Expansions of reversible circuits, *Proc. 18th Intern. Conf. on Microelectronics (ICM)* 2006, pp. 63-66.
9. J. Donald and N. K. Jha, ``Reversible logic synthesis with Fredkin and Peres gates," ACM Journal of Emerging Technologies in Computing Systems, Vol.4, issue 1, Mar. 2008.
10. P. Kerntopf, A new heuristic algorithm for reversible logic synthesis, *Proc. DAC,* pp. 834-837, June 2004.
11. A. Khlopotine, M. Perkowski, and P. Kerntopf, Reversible logic synthesis by gate composition, *Proc. IWLS* 2002, pp. 261 – 266.
12. M. Kumar, B. Year, N. Metzger, Y. Wang, and M. Perkowski, Realization of Incompletely Specified Functions in Minimized Reversible Circuits, *Proc. RM* 2007.
13. M. Kumar, Realization of Incompletely Specified Reversible Functions *MS*. *Thesis. PSU. 2008.*
14. D. Maslov and G. Dueck, Improved quantum cost for k-bit Toffoli gates, *IEE Electron. Lett*., vol. 39, no. 25, pp. 1790–1791, Dec. 2003.
15. D. Maslov and G. W. Dueck, Garbage in reversible design of multiple output functions, in *Proc. 6th Int. Symp. Representations and Methodology of Future Computing Technologies,* Mar. 2003, pp. 162–170.
16. D. Maslov, Reversible Logic Synthesis, *PhD dissertation*, 2003.
17. D. Maslov and G.W. Dueck, Reversible cascades with minimal garbage, *IEEE Transactions on CAD*, 23(11): pp. 497-1509, November 2004.
18. D. Maslov, G.W. Dueck, and D.M. Miller, Techniques for the Synthesis of Reversible Toffoli Networks, *ACM Transactions on Design Automation of Electronic Systems*, Vol. 12, No. 4, Article 42, Sept. 2007.
19. D. Maslov, G.W. Dueck, D. M. Miller, and C. Negrevergne, Quantum Circuit Simplification and Level Compaction, IEEE Trans. on CAD, Vol. 27, No. 3, March 2008, pp 436-444.
20. D. Maslov, Web Page: <http://webhome.cs.uvic.ca/~dmaslov/>
21. D. M. Miller and G.W. Dueck, Spectral Techniques for Reversible Logic Synthesis, *Proc. RM* 2003, pp. 56-62.
22. D. M. Miller, D. Maslov and G. W. Dueck, A Transformation Based Algorithm for Reversible Logic Synthesis, *Proc. DAC*, Anaheim, pp. 318–323, June 2003.
23. A. Mishchenko and M. Perkowski, Fast Heuristic Minimization of Exclusive Sum-of-Products, *Proc. Reed-Muller Workshop,* 2001, Starkville, Mississippi, pp. 242-250.
24. A. Mishchenko and M. Perkowski, “Logic Synthesis of Reversible Wave Cascades”, *Proc. IEEE/ACM IWLS*, June 4-7, 2002, pp. 197 – 202.
25. M. Mohamadi, and M. Eshghi, Heuristic methods to use don’t cares in automated design of reversible and quantum logic circuits. *Quantum Inf. Process*. J. 7(4), pp. 175–192, 2008.
26. **Ch. Stedman, B. Yen and M. Perkowski,** Synthesis of Reversible Circuits with Small Ancilla Bits for Large Irreversible Incompletely Specified Multi-Output Boolean Functions, Proc. 14th International Workshop on Post-Binary ULSI Systems, May 18, 2005, Calgary, Canada.
27. J. E. Rice K. B. Fazel, M.A. Thornton, and K. B. Kent Toffoli Gate Cascade Generation Using ESOP Minimization and QMDD-based Swapping. Proc. RM, May 23-24, 2009, pp. 63-72.
28. M. Saeedi, M. Sedighi, M. S. Zamani, A Novel Synthesis Algorithm for Reversible Circuits, Proc. ICCAD, San Jose, California, pp. 65-68. 2007.
29. M. Saeedi, M. Sedighi, M. Saheb Zamani, A New Methodology for Quantum Circuit Synthesis: CNOT-Based Circuits as an Example,” *Proc*. *IWLS*, 2007. Paradise Point Resort and Spa, San Diego, CA, USA, May 30-June 1.
30. M. Saeedi, M. S. Zamani, M. Sedighi, On the Behavior of Substitution-Based Reversible Circuit Synthesis Algorithms: Investigation and Improvement*,” Proc*. *ISVLSI*, 9-11 March 2007, pp. 428-436.
31. R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An online resource for reversible functions and reversible circuits. Proc. 38th ISMVL, pp. 220–225. Dallas, TX, USA, 2008. <http://www.revlib.org>.
32. R. Wille, R. Drechsler, BDD-based Synthesis of Reversible Logic for Large Functions. Proc. DAC, pp. 270-275. San Francisco, California
33. R. Wille, M. Saeedi, R. Drechsler, Synthesis of Reversible Functions Beyond Gate Count and Quantum Cost. IWLS 2009.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | MP | |  |  |  |  |  | MMD |  |  |  |
| Function | Gate Count | Garbage | Total Quantum | Time (Sec) |  | Function | Gate Count | Garbage | Total Quantum | Time (Sec) |  |
| NA1 | 16 | 0 | 66 | 197 |  | NA1 | 26 | 0 | 140 | 1 |  |
| NA2 | 18 | 0 | 86 | 195 |  | NA2 | 16 | 0 | 84 | 1 |  |
| NA3 | 16 | 0 | 68 | 197 |  | NA3 | 14 | 0 | 62 | 1 |  |
| NA4 | 17 | 0 | 57 | 198 |  | NA4 | 21 | 0 | 89 | 1 |  |
| NA5 | 16 | 0 | 50 | 193 |  | NA5 | 21 | 0 | 118 | 1 |  |
| NA6 | 16 | 0 | 52 | 190 |  | NA6 | 20 | 0 | 98 | 1 |  |
| NA7 | 17 | 0 | 69 | 201 |  | NA7 | 18 | 0 | 92 | 1 |  |
| NA8 | 18 | 0 | 60 | 190 |  | NA8 | 18 | 0 | 80 | 1 |  |
| NA9 | 15 | 0 | 79 | 181 |  | NA9 | 18 | 0 | 94 | 1 |  |
| NA10 | 18 | 0 | 60 | 192 |  | NA10 | 22 | 0 | 92 | 1 |  |
| NA11 | 18 | 0 | 66 | 199 |  | NA11 | 27 | 0 | 143 | 1 |  |
| NA12 | 16 | 0 | 68 | 197 |  | NA12 | 25 | 0 | 113 | 1 |  |
| NA13 | 16 | 0 | 60 | 195 |  | NA13 | 19 | 0 | 99 | 1 |  |
| NA14 | 15 | 0 | 61 | 193 |  | NA14 | 18 | 0 | 70 | 1 |  |
| NA15 | 19 | 0 | 69 | 199 |  | NA15 | 27 | 0 | 139 | 1 |  |
| NA16 | 17 | 0 | 67 | 191 |  | NA16 | 20 | 0 | 92 | 1 |  |
| NA17 | 19 | 0 | 89 | 193 |  | NA17 | 22 | 0 | 122 | 1 |  |
| NA18 | 15 | 0 | 51 | 188 |  | NA18 | 24 | 0 | 106 | 1 |  |
| NA19 | 16 | 0 | 58 | 199 |  | NA19 | 23 | 0 | 119 | 1 |  |
| NA20 | 16 | 0 | 60 | 200 |  | NA20 | 23 | 0 | 115 | 1 |  |
| NA21 | 16 | 0 | 74 | 189 |  | NA21 | 20 | 0 | 108 | 1 |  |
| NA22 | 15 | 0 | 57 | 196 |  | NA22 | 20 | 0 | 108 | 1 |  |
| NA23 | 11 | 0 | 51 | 182 |  | NA23 | 18 | 0 | 92 | 1 |  |
| NA24 | 15 | 0 | 55 | 194 |  | NA24 | 23 | 0 | 99 | 1 |  |
| NA25 | 18 | 0 | 56 | 198 |  | NA25 | 21 | 0 | 95 | 1 |  |
| NA26 | 14 | 0 | 62 | 188 |  | NA26 | 21 | 0 | 103 | 1 |  |
| NA27 | 13 | 0 | 51 | 190 |  | NA27 | 20 | 0 | 106 | 1 |  |
| NA28 | 17 | 0 | 77 | 199 |  | NA28 | 25 | 0 | 121 | 1 |  |
| NA29 | 19 | 0 | 55 | 200 |  | NA29 | 26 | 0 | 130 | 1 |  |
| NA30 | 15 | 0 | 75 | 194 |  | NA30 | 19 | 0 | 105 | 1 |  |
| NA31 | 18 | 0 | 92 | 198 |  | NA31 | 21 | 0 | 99 | 1 |  |
| NA32 | 16 | 0 | 70 | 189 |  | NA32 | 21 | 0 | 117 | 1 |  |
| NA33 | 17 | 0 | 63 | 199 |  | NA33 | 19 | 0 | 89 | 1 |  |
| NA34 | 16 | 0 | 70 | 189 |  | NA34 | 27 | 0 | 147 | 1 |  |
| NA35 | 16 | 0 | 64 | 192 |  | NA35 | 17 | 0 | 87 | 1 |  |
| NA36 | 17 | 0 | 61 | 199 |  | NA36 | 23 | 0 | 93 | 1 |  |
| NA37 | 17 | 0 | 59 | 199 |  | NA37 | 22 | 0 | 116 | 1 |  |
| NA38 | 16 | 0 | 58 | 189 |  | NA38 | 22 | 0 | 108 | 1 |  |
| NA39 | 15 | 0 | 71 | 194 |  | NA39 | 20 | 0 | 64 | 1 |  |
| NA40 | 14 | 0 | 46 | 192 |  | NA40 | 21 | 0 | 85 | 1 |  |
| NA41 | 16 | 0 | 54 | 189 |  | NA41 | 23 | 0 | 115 | 1 |  |
| NA42 | 15 | 0 | 53 | 194 |  | NA42 | 19 | 0 | 59 | 1 |  |
| NA43 | 15 | 0 | 71 | 194 |  | NA43 | 25 | 0 | 117 | 1 |  |
| NA44 | 18 | 0 | 74 | 198 |  | NA44 | 23 | 0 | 83 | 1 |  |
| NA45 | 17 | 0 | 67 | 199 |  | NA45 | 21 | 0 | 109 | 1 |  |
| NA46 | 19 | 0 | 77 | 201 |  | NA46 | 27 | 0 | 143 | 1 |  |
| NA47 | 17 | 0 | 65 | 199 |  | NA47 | 21 | 0 | 95 | 1 |  |
| NA48 | 15 | 0 | 59 | 194 |  | NA48 | 23 | 0 | 93 | 1 |  |
| NA49 | 16 | 0 | 66 | 189 |  | NA49 | 24 | 0 | 124 | 1 |  |
| NA50 | 16 | 0 | 62 | 189 |  | NA50 | 21 | 0 | 103 | 1 |  |

*Table 1. Comparison of MMD and MP results on 50 randomly generated functions of 4 variables*

*[[1]](#endnote-2)*

1. Need to check what's possible in realtime???? - 1 min or less, 5 minutes or less??? What's realtime. [↑](#endnote-ref-2)