FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 23, no. 3, December 2010, 273-286

Synthesis of Reversible Circuits for Large Reversible Functions

Nouraddin Alhagi, Maher Hawash, and Marek Perkowski

Abstract: This paper presents a new algorithm MP (multiple pass) to synthesize large reversible binary circuits without ancilla bits. The well-known MMD algorithm for synthesis of reversible circuits requires to store a truth table (or a Reed-Muller - RM transform) as a 2*n* vector to represent a reversible function of n variables. This representation prohibits synthesis of large functions. However, in MP we do not store such an 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. This allows for synthesis of large scale reversible circuits (30-bits), which is not possible with any existing algorithm. In addition, our unique multi-pass approach where the circuit is synthesized with various, yet specific, minterm orders yields quasi-optimal solution. The algorithm returns a description of the quasi-optimal circuit with respect to gate count or to its "quantum cost". Although the synthesis process in MP is relatively slower, the solution is found in real-time for smaller circuits of 8 bits or less.

Keywords: Multiple pass algorithm, Reed-Muller transform, Miller, Maslov and Dueck algorithm, reversible circuits.

1 Introduction

There are currently two types of algorithms to synthesize reversible circuits: (T1) those like MMD [1–15] that start from a reversible specification, (T2) those like [16–27] that start from non-reversible specification and create ancilla bits. The second type of methods has been successful for large functions [16, 25–28] but solves basically a different problem. The MMD algorithm [14] (Miller, Maslov

Manuscript received on September 5, 2010.

The authors are with Department of Electrical and Computer Engineering Portland State University P.O.Box 751, Portland, OR 972027-0751 (e-mails: nmalhagi@yahoo.com, gmhawash@gmail.com, mperkows@cecs.pdx.edu)

and Dueck) is currently the leading reversible logic synthesizer *if no ancilla bits are used*. Mathematically, the problem is to decompose a large permutation of circuits specification to small permutations of reversible gates that are used. MMD uses the permutation vector-like reversible function specification as its input, and internal data which correspond to a truth table. This vector of minterms (binary numbers representing outputs) is explicitly used in the synthesis process. It must be therefore stored and processed in the computers memory. Since it is intrinsically bound by the natural binary order of minterms, and hence does not use search, MMD cannot be enhanced through better search algorithms or iterative/recursive routines. Since MMD processes only a single order of minterms, it is reasonably fast. In addition, MMD distinguishes itself among most other programs of this type because it achieves (theoretical) 100% convergence regardless the problem size [14]. Practically, however, it was applied to at most 12 qubit reversible functions and very few reversible functions with more than 8 variables were presented as MMD benchmarks in the literature (our non-published variant of MMD can handle 16 variables). It was found in our research, and by other researchers, that the complexity of both the synthesis process and the average circuit sizes synthesized by the original MMD software grow very quickly above 8 qubits, herein "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 [22–27, 29], 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. By convergence we mean here that the algorithm terminates with a correct result if the number of bits is not large. 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. The algorithm of Agrawal and Jha [1, 16] 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 $f = a'b'c'd'$. So this algorithm has bounds based on expressions and not numbers of inputs. Some of the algorithm variants from [1, 2, 4] 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 [18, 20, 21, 30], 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 the MMD's approach. 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 of data specifying the circuit under design. 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, 2, 4]. Other cascade types have been also proposed in the newer versions of composition-based search approaches [6, 19, 20, 30], 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 and are selected on the base of the size of the synthesized function.

2 Explanation of MMDs Main Idea

To make the paper self-contained we give a brief overview of MMD. More can be found in [3, 7–15]. 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. Every such transformation produces one reversible gate for the circuit, realized from outputs to inputs.

Example 2.1. Fig. 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. 1 lists values of the output vectors that correspond to the input vectors from the first column. For instance, 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 minterms* are minterms with matching input and output values (e.g., minterm 000 above) 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 [2, 5, 6, 20]. 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. 1), which is guaranteed by the selection of proper control bits.

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	**11**1	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
		$a \rightarrow c$	$c \rightarrow a$	$a \rightarrow b$	$b \rightarrow a$	$a \rightarrow b$	$a \rightarrow c$

Fig. 1. Figure 1. MMD method illustrated with truth tables of intermediate functions. Notation $a \rightarrow c$ means $c = c \oplus$ a means "flip *c* **if** $a = 1$". Control lines are underlined and affected bits are shaded.

This is the main idea of MMD algorithm and actually the only algorithmic idea of this method (excluding template matching which is not discussed here). 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 $n \times 2^n - 1$ 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 its 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 overall, no better algorithm had been proposed. The symbol $a \rightarrow c$ 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 Toffoli gate $c = c \oplus a$. 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 $n \times 2^n - 1$ which for our 3-bit example yields $(3 \times 2^3 - 1)$ 23 gates. Note that our example simulation resulted in only 6 gates. Here MMD happened to work well. But there are examples [30] where the gate number is close to the upper bound although the minimal number of gates is lower.

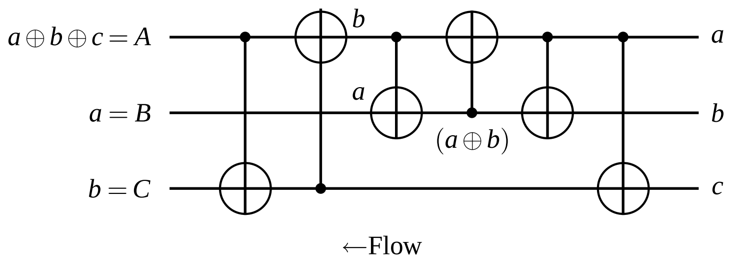

Fig. 2. The solution circuit found from MMD in Fig. 1 drawn and created from outputs to inputs. The arrow shows the flow of signal from inputs to outputs. This method is possible because each reversible gate used in this figure is its own self-inverse.

3 MMDS and MMDSN Orderings

The main concept of MMD, the natural binary minterm ordering was challenged in [21] as the only 100% convergent order. It was found that MMDs 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” [21]. 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 [21]. 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.

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 were 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 the MMD, which, as it might give the worst solutions, it is always convergent. 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 MMDs idea [14, 15] of not changing any previously set outputs. This idea is also what guarantees MMDs convergence.

A 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 $\Rightarrow$

```
    **if** #later = #later & #earlier
    **then** there is control line blocking
```

Example 1 Control line blocking exists

101 = 101 & 111

Example 1 Control line blocking does not exist

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 [21]. 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! This is because 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.

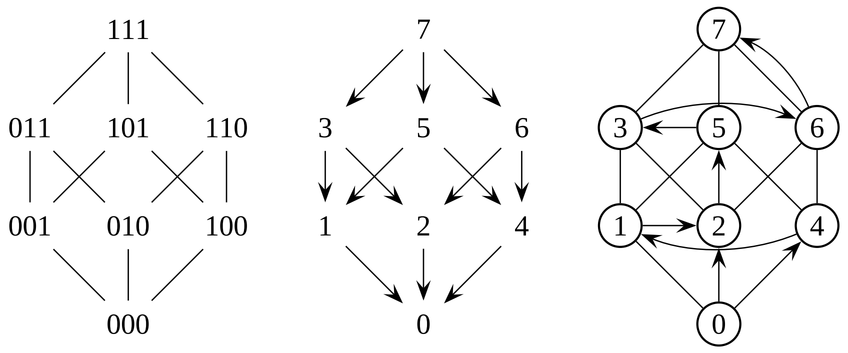

Fig. 3. New orders for MMD-like synthesis. (a) Hasse diagram with binary vectors, (b) Hasse diagram with natural numbers, (c) Ordering of nodes that violates the MMD order, illustrated on the Hasse Diagram. This is however a valid MMDS ordering.

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 ($1 \geq 0$, $1 \geq 1$, $0 \geq 0$) is used as an ordering relation (see Fig. 3a, b). While binary vectors are used in Fig. 3a, Fig. 3b 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. 3c and 4.

As the number of input lines increases, the number of non-blocking orderings increases exponentially. For functions with four inputs, Stedman [21] 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:

- The target bit cannot be used to control the current transformation,
- Use minimal number controls bits necessary to flip the target bit,
- No past outputs can be changed,
- Process $0 \rightarrow 1$ 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, 5, 6, 17, 20, 25, 30]. 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.

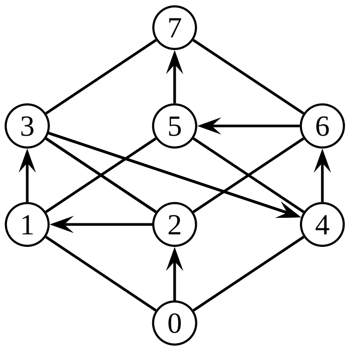

Fig. 4. New ordering 02134657 for MMD-like binary synthesis, a valid MMDS order which is consistent with the Hasse diagram relations of order.

B 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.

C Definition 2

MMDSN order is one in which the minterm $00\ldots0$ *is generated first, followed by all minterms with a single one(1) in random order, followed by all minterms with two ones (1s) in random order, and so on, successively incrementing the number of ones (1's) in each band until we finally reach the minterm* $11\ldots1$.

#	Function	MMD			MP		
bits		Gates	Q-Cost	Time(ms)	Gates	Q-Cost	Time(ms)
4	hwb4	24	120	0.577	19	91	339
5	hwb5	62	498	0.033	53	389	392
6	hwb6	164	1,800	0.075	140	1,276	613
7	hwb7	382	5,614	0.247	353	4,961	1503
8	hwb8	883	17,927	1.312	837	15,873	987
9	hwb9	2050	52,318	4.171	1993	48,817	4,170
10	urf3	3426	119,986	12.595	3334	110,910	58,306
11	urf4	10527	456,139	75.780	10336	403,184	384,589

Fig. 5. Comparison of numbers of gates and quantum costs of MMD and MP algorithms for reversible functions with various numbers of bits. This is "large circuits" variant with k=5000. No ancilla bits.

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.

Observe that MMD is not included in MMDSN, but it can be artificially added as one more 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 [5, 6, 20, 21, 30]. 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 implicitly, 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 cascaded back to back and simulated from inputs at each stage of minterm transformation (Figure 6). 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.

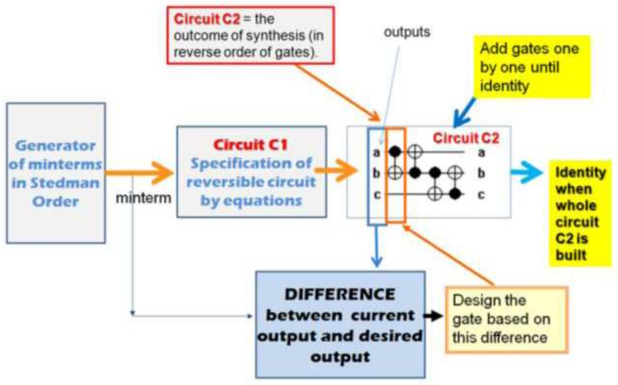
Circuit C2 = the outcome of synthesis (in reverse order of gates).
outputs
Add gates one by one until identity
Generator of minterms in Stedman Order
minterm
Circuit C1 Specification of reversible circuit by equations
Circuit C2
a
b
c
Identity when whole circuit C2 is built
DIFFERENCE between current output and desired output
Design the gate based on this difference

Fig. 6. The overall scheme of MP algorithm. The generator at the left can generate minterms in arbitrary order, MMD, MMDS, MMDSN or other.

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.

5 Results of MP for Four Variables

For functions of four variables, we created a set of randomly generated four-bit reversible functions, AHP1-AHP50, and synthesized them using the original MMD, MMDS and our MMDSN orders. For MMDS and MMDSN, we tested the AHP functions against all possible permutations and calculated the minimum possible gate count as shown in Table 1. It is evident that our selective order consistently produce superior results compared to the single MMD order for a negligible time penalty. Notice, however, that although the MMDSN order did not generate the optimal gate count generated by MMDS, the time advantage of MP is huge at 4 bits, and would be astronomical at greater number of bits. Even at higher number of bits, MMDSN order consistently produces better results than MMD within tolerable time. For example, at 11 bits, MP accrued a savings of 191 gates taking about six minutes to synthesize 5000 MMDSN sequences selected at random,

Table 1. Comparison of MMD, MMDS and MMDSN orders on 50 random functions of 4 variables.

Function	MMDSN			MMD			MMDS		
	#Gates	Q-Cost	Time(ms)	#Gates	Q-Cost	Time(ms)	#Gates	Q-Cost	Time(ms)
AHP- 0	18	102	8.393	20	144	1.074	15	55	178,097
AHP- 10	16	68	6.991	29	209	0.022	14	42	182,428
AHP- 100	22	150	8.040	25	149	0.018	18	98	205,910
AHP- 102	21	109	7.653	28	192	0.019	19	103	362,359
AHP- 104	19	99	7.408	28	192	0.020	17	73	392,670
AHP- 106	21	129	7.567	24	116	0.016	17	77	438,121
AHP- 108	20	108	8.078	21	129	0.015	17	77	464,066
AHP- 1000	16	80	7.497	19	111	0.014	14	54	468,883
AHP- 1002	21	113	7.513	31	223	0.014	18	78	526,966
AHP- 1004	20	136	7.056	23	167	0.029	15	79	539,691
AHP- 1006	17	109	7.495	24	172	0.030	17	93	575,764
AHP- 1008	19	95	6.682	31	215	0.024	18	90	593,118
AHP- 1010	18	85	6.953	30	230	0.028	17	74	621,180
AHP- 1012	23	131	7.146	28	168	0.031	18	70	626,634
AHP- 1014	23	139	8.069	27	179	0.031	19	75	639,966
AHP- 1016	18	126	6.748	23	167	0.030	15	79	646,605
AHP- 1018	17	105	6.939	25	197	0.030	15	63	408,780
AHP- 1020	18	106	7.317	25	193	1.803	16	96	284,467
AHP- 1022	19	111	7.697	24	156	0.153	14	54	268,481
AHP- 1024	22	138	6.622	30	218	0.148	16	76	253,849
AHP- 1026	14	66	7.252	17	113	0.154	14	66	229,625
AHP- 1028	14	86	7.343	20	148	0.157	13	81	222,084
AHP- 1030	21	137	7.776	27	167	0.124	16	80	211,866
AHP- 1032	20	108	6.726	27	187	0.106	17	93	214,853
AHP- 1034	19	123	7.132	22	138	0.102	15	71	220,812
AHP- 1036	19	107	7.257	26	186	0.093	17	81	206,786
AHP- 1038	18	106	7.927	18	106	0.083	13	65	210,267
AHP- 1040	16	96	6.478	22	174	0.078	11	39	217,464
AHP- 1042	22	146	7.263	25	173	0.080	19	99	204,661
AHP- 1044	19	107	7.325	23	159	0.096	16	92	196,889
AHP- 1046	19	107	7.739	23	147	0.092	15	71	210,829
AHP- 1048	18	94	6.484	20	120	0.096	17	89	201,351
AHP- 1050	23	123	7.325	34	230	0.083	19	83	219,222
AHP- 1052	18	110	7.557	26	166	0.080	16	84	241,366
AHP- 1054	17	81	7.226	24	164	0.047	16	76	215,861
AHP- 1056	17	93	7.757	28	196	0.813	15	67	228,621
AHP- 1058	18	118	6.991	23	155	0.015	15	55	200,601
AHP- 1060	19	151	8.110	21	161	0.019	15	83	252,009
AHP- 1062	19	107	7.268	31	247	0.020	16	76	236,668
AHP- 1064	23	131	7.357	29	189	0.017	20	84	237,049
AHP- 1066	18	122	7.055	31	235	0.017	15	75	240,952
AHP- 1068	22	134	8.606	21	97	0.017	19	99	272,891
AHP- 1070	18	106	7.707	22	158	0.018	16	80	386,639
AHP- 1072	20	112	7.611	23	159	0.019	16	72	313,911
AHP- 1074	22	126	8.236	26	194	0.017	18	106	263,204
AHP- 1076	21	121	8.644	28	184	0.020	18	74	264,143
AHP- 1078	21	105	7.690	30	222	0.021	18	78	277,411
AHP- 1080	21	145	7.879	21	109	0.016	17	93	289,429
AHP- 1082	21	133	8.109	29	233	0.016	16	92	230,252
AHP- 1084	23	119	8.797	31	187	0.014	20	104	263,490
AHP- 1086	20	116	7.367	27	195	0.014	17	85	232,918

and maintaining the solution with the best gate count. Although the current implementation of the MP algorithm does not utilize parallel processing, the algorithm is very suitable for parallelization through threading, multiprocessing or within a cloud infrastructure. Such capability would allow for synthesizing a selecting a larger iteration variant (k) and thus produce even larger circuits. The reader should note that in this study, neither MMD or MP used local optimization techniques, e.g. template matching, which would ideally reduce the number of gates even further. Although MP would run even slower with template matching, its inclination to parallelization would easily minimize such an impact. An 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 MMD, Agrawal/Jhas algorithm, and all other published approaches.

6 Results of the MP For More Than Four Variables

Figure 5 shows the results with $k = 5000$ produced with a single threaded application on a Windows 7 operating system running on a Intel® Core™2 Duo 2.93 GHz processor. The application allows the user to k to any value to get the trade-off between synthesis time and quantum cost improvement. As of this writing, we were not able to compare MP with original MMD on larger functions since MMD does not accept functions of 30 variables as it is not able to store a vector with 2^{30} rows in memory. As we see in Figure 5, 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 a sample reversible function, AHP30_1, of 30 variables [30], which was input as separate equation for each bit (this variant of MP is not format compatible with MMD and other programs). The synthesis generated a quantum array of 4496 gates and took 2 hours and 45 minutes to complete. The function was a simple cascade of Toffoli gates where each variable controls its immediate successor. Our choice of a simplistic function, at this time, sets for us a foundation for future research as we plan on extending our method to other functions of 30 variables or more. Results sited in this paper are currently available on `http://www.quantumlib.org:21012`.

7 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 on the fly 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 [30] and state machines [17, 18]. 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. Currently, MP is the reversible logic synthesizer that can synthesize the largest functions and that allows to obtain the solutions with the smallest quantum cost of all existing logic minimizers for this task.

References

[1] A. Agrawal and N. K. Jha, “Synthesis of reversible logic,” in *Proc. DATE*, Paris, France, Feb. 2004, pp. 1530–1591.

[2] 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, Mar. 2008.

[3] G. W. Dueck and D. Maslov, “Reversible function synthesis with minimum garbage outputs,” in *6th International Symposium on Representations and Methodology of Future Computing Technologies (RM)*, Trier, Germany, Mar. 2003, pp. 154–161.

[4] P. Gupta, A. Agrawal, and N. Jha, “An algorithm for synthesis of reversible logic circuits,” *IEEE Trans. on CAD*, vol. 25, no. 11, pp. 2317–2330, 2006.

[5] P. Kerntopf, “A new heuristic algorithm for reversible logic synthesis,” *Proc. DAC*, pp. 834–837, June 2004.

[6] A. Khlopotine, M. Perkowski, and P. Kerntopf, “Reversible logic synthesis by gate composition,” *Proc. IWLS*, pp. 261–266, 2002.

[7] 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.

[8] 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.

[9] D. Maslov, “Reversible logic synthesis,” Ph.D. dissertation, 2003.

[10] D. Maslov and G. Dueck, “Reversible cascades with minimal garbage,” *IEEE Transactions on CAD*, vol. 23, no. 11, pp. 497–1509, Nov. 2004.

[11] D. Maslov, G. Dueck, and D. 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.

[12] D. Maslov, G. Dueck, D. M. Miller, and C. Negrevergne, “Quantum circuit simplification and level compaction,” *IEEE Trans. on CAD*, vol. 27, no. 3, pp. 436–444, Mar. 2008.

[13] D. Maslov. [Online]. Available: http://webhome.cs.uvic.ca/ dmaslov/

[14] D. M. Miller and G. Dueck, "Spectral techniques for reversible logic synthesis," *Proc. RM*, pp. 56–62, 2003.

[15] 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.

[16] D. Große, R. Wille, G. Dueck, and R. Drechsler, "Exact multiple control toffoli network synthesis with sat techniques," *IEEE Trans. on CAD*, vol. 28, no. 5, pp. 703–715, 2009.

[17] M. Kumar, B. Year, N. Metzger, Y. Wang, and M. Perkowski, "Realization of incompletely specified functions in minimized reversible circuits," *Proc. RM*, 2007.

[18] M. Kumar, "Realization of incompletely specified reversible functions," Master's thesis, PSU, 2008.

[19] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive sum-of-products," in *Proc. Reed-Muller Workshop*, Starkville, Mississippi, 2001, pp. 242–250.

[20] ——, "Logic synthesis of reversible wave cascades," *Proc. IEEE/ACM IWLS*, pp. 197 – 202, June 4-7, 2002.

[21] C. Stedman, B. Yen, and M. Perkowski, "Synthesis of reversible circuits with small ancilla bits for large irreversible incompletely specified multi-output boolean functions," in *Proc. 14th International Workshop on Post-Binary ULSI Systems*, Calgary, Canada, May 18, 2005.

[22] M. Saeedi, M. Sedighi, and M. S. Zamani, "A novel synthesis algorithm for reversible circuits," San Jose, California, 2007, pp. 65–68.

[23] ——, "A new methodology for quantum circuit synthesis: Cnot-based circuits as an example," in *Proc. IWLS*, Paradise Point Resort and Spa, San Diego, CA, USA, May 30-June 1, 2007.

[24] M. Saeedi, M. S. Zamani, and M. Sedighi, "On the behavior of substitution-based reversible circuit synthesis algorithms: Investigation and improvement," in *Proc. IEEE Computer Society Annual Symposium on VLSI, ISVLSI, 2007*, Porto Alegre, Brazil, Mar. 9-11, 2007, pp. 428–436.

[25] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "Revlib: An online resource for reversible functions and reversible circuits," in *Proc. 38th ISMVL*, Dallas, TX, USA, 2008, pp. 220–225. [Online]. Available: http://www.revlib.org

[26] R. Wille and R. Drechsler, "Bdd-based synthesis of reversible logic for large functions," in *Proc. DAC*, San Francisco, California, pp. 270–275.

[27] R. Wille, M. Saeedi, and R. Drechsler, "Synthesis of reversible functions beyond gate count and quantum cost," in *IWLS 2009*.

[28] D. Große, X. Chen, and R. Drechsler, "Exact toffoli network synthesis of reversible logic using boolean satisfiability," in *Proc. IEEE CAS Workshop*, Dallas, USA, 2006, pp. 51–54.

[29] J. E. Rice, K. B. Fazel, M. Thornton, and K. B. K. Toffoli, "Gate cascade generation using esop minimization and qmdd-based swapping," *Proc. RM*, pp. 63–72, May 23-24, 2009.

[30] N. Alhagi, "A synthesis method to synthesize reversible circuits," Ph.D. dissertation.