Reversible Circuit Synthesis Using ACO and SA based Quine-McCluskey method

Mayukh Sarkar, Prasun Ghosal
Bengal Engineering and Science University, Shibpur
Howrah 711103, WB, INDIA
Email: mayukh.sarkar1987@gmail.com, p_ghosal@it.becs.ac.in

Saraju P. Mohanty
University of North Texas
Denton, TX 76203, USA
Email: saraju.mohanty@unt.edu

***Abstract*— With the tremendous growth in VLSI technology in recent years, the Integration density of the transistors has reached billions causing the scaling of transistors to touch the subatomic dimension in deep submicron regime where laws of classical physics can not survive. Due to inherent information loss and other factors associated with irreversible computing, reversible circuits are becoming more and more important in terms of computing for present and future days. However, due to several factors, known synthesis approaches of classical Boolean logic like Karnaugh Map and Quine-McCluskey method cannot be applied directly to synthesize a reversible logic. In this paper, we propose a stochastic procedure to synthesize a reversible circuit. This procedure is based on a modified version of classical Quine-McCluskey method and is being used under the wrapper of two intelligent stochastic search techniques, Simulated Annealing and Ant Colony Optimization. The experimental results are quite encouraging.**

I. Introduction

With progressive scaling of transistors in VLSI technology the integration density of the transistors has reached billions following Moore's Law [1] causing the scaling of transistors to touch the subatomic dimension where laws of classical physics can not survive in recent years. On the other hand, irreversible computing suffers from consistent information loss. According to Landauer's Principle [2], loss of one bit of information has an energy dissipation with the lower bound of $kTln(2)$. In next few years, this energy is going play a dominant role in the area of power aware high performance computing systems. All these effects show that, the classical computing is approaching a barrier [3], and the computation based on quantum physics is becoming very much important for future days.

Computation with no loss of information is called reversible computation, and Bennett showed that [4], zero energy dissipation is possible only with reversible computing. So, the reversible computing is going to play a dominant role in the near future. It has many applications, including quantum computation. Quantum gates are, by nature, reversible and provides a powerful motivation to study reversible computation. Other applications of reversible computing can also be found in the domain of optical computing, DNA computing, low-power CMOS design etc.

Synthesis approaches for reversible circuits are different from traditional Boolean logic in several ways [5] due to several factors such as equality of I/O numbers, acyclic nature, absence of fanout, presence of garbage output etc. So, no traditional Boolean circuit synthesis methods like Karnaugh Map, Quine-McCluskey method etc. can directly be used to synthesize a reversible circuit. Several approaches for the synthesis of a reversible circuit have been proposed, most of which are deterministic approaches, $e.g.$, Saeedi et. al. [6] proposed Moving Forward Synthesis Algorithm(MOSAIC), Gupta et. al. [7] proposed an algorithm based on Positive Polarity Reed-Muller(PPRM) expression. In this paper, the Quine-McCluskey approach has been used in a modified manner, under the wrapper of Simulated Annealing(SA) and Ant Colony Optimization(ACO).

TABLE I: *Table of a Reversible Function*

a b c	a b c
0 0 0	0 0 0
0 0 1	0 1 0
0 1 0	0 0 1
0 1 1	1 0 0
1 0 0	1 1 1
1 0 1	1 0 1
1 1 0	1 1 0
1 1 1	0 1 1

II. Background

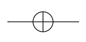
In this section, we will describe the background necessary to understand this algorithm.

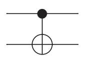
A. Reversible Functions

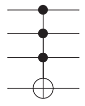
Definition 1: An n-input, n-output Boolean function f is reversible if it maps each input uniquely to each output, and vice-versa, i.e., there is an one-to-one mapping between input and output.

A reversible function can be represented in the form of a truth table or a permutation. For example, the truth table I can also be represented as the permutation $\{0, 2, 1, 4, 7, 5, 6, 3\}$. A reversible function can be realized by a reversible circuit, which in turn, is a cascade of reversible gates. If a reversible function $f : I \rightarrow O$, where I is the input set, and O is the output set, can be realized by reversible gate cascade $\{G_1, G_2, ..., G_k\}$, then the function $g : O \rightarrow I$ can be realized by same gate cascade in reversed order $\{G_k, G_{k-1}, ..., G_1\}$.

There are several universal reversible gate libraries, among which NCT (NOT, CNOT, TOFFOLI) library is used by our algorithm to synthesize a reversible specification. The figures of NOT, CNOT and n-bit Toffoli gate are shown in Figures 1a, 1b, and 1c.


(a) NOT gate


(b) CNOT gate


(c) 4-bit Toffoli gate

B. Quine-McCluskey Method

Quine-McCluskey is an widely used approach for the synthesis of classical irreversible Boolean circuit. In this approach, first the minterm table of the required function f is created by adding those inputs, for which the value of f is 1. Details of this technique are omitted due to paucity of this space. The standard Quine-McCluskey method is used in modified manner in this algorithm. We have created the minterm tables for each output bit by checking whether the corresponding input bit has changed or not. Also, the prime implicant chart is not being used. Heuristic evaluated by using minterm tables are used by Simulated Annealing to determine the next best gate of the gate cascade.

III. Proposed Algorithm

In this algorithm, initially, each ant starts with a blank circuit, and adds a gate to the circuit based on a transition probability. After adding a gate, the ant applies the Simulated Annealing based Quine-McCluskey(SA-QM) method to find out minimum circuits possible from the resulting circuit. As soon as it finds out any suitable circuit, it imposes the constraints of length and cost of the minimum circuit returned by SA-QM to itself and the other ants coming next, and further adds more gates to check for other possible circuits.

A. Simulated Annealing Based Quine-McCluskey Method

In this algorithm, the inputs are truth table, a circuit, length and cost constraints. This method starts with a particular temperature t. It then checks each gate one by one by adding them to the circuit, and applies the circuit to the truth table. From the resulting truth table, it calculates a heuristic value using Quine-McCluskey approach, and removes the gate. Finally, the gate that maximizes the heuristic is added to the circuit with probability $e^{(h(g)-h_{max})/temperature}$, where $h(g)$ is the heuristic for the gate g, and h_{max} is the maximum heuristic obtained before checking for gate g. The gate heuristic $h(g)$ for a gate g and a truth table t_{in} is found as follows.

From the truth table t_{in}, to find the gate heuristic $h(g)$ for gate g, gate g is applied to t_{in} to get the output truth table t_{out}. The minterm table m_b for each I/O bit b is created by adding the input entries in m_b, for which the bit b gets changed in corresponding output. The minterm tables are then minimized recursively according to Quine-McCluskey approach. From the final minterm tables of all I/O bits, the heuristic is calculated as follows.

$$h(g) = 100 \times (don't\_care\_ratio + one\_bit\_ratio) - total\_length - Hamming\_distance \quad (1)$$

```
1: procedure SA-QM($TruthTable, Circuit, Length, Cost$)
2:     $temperature \leftarrow INIT\_TEMP$
3:     $i \leftarrow 1$
4:     $circuit\_list \leftarrow blank\_list$
5:     while $i \leq ITERATION$ do
6:         $temp \leftarrow temperarture$
7:         $circuit \leftarrow Circuit$
8:         $length\_loop \leftarrow 1$
9:         while $length\_loop \leq Length$ do
10:            $heuristic \leftarrow -\infty$
11:            for all possible gate $g$ do
12:                Add $g$ to the $circuit$
13:                Apply $circuit$ to the $TruthTable$
14:                if $circuit$ solves $TruthTable$ then
15:                    Add $circuit$ to the $circuit\_list$
16:                    break from inner while loop
17:                end if
18:                Determine heuristic value $h(g)$ of the resulting Truth Table
19:                if $h(g) > heuristic$ then
20:                    $best\_gate \leftarrow g$
21:                    $heuristic \leftarrow h(g)$
22:                else
23:                    Set $best\_gate \leftarrow g$ and $heuristic \leftarrow h(g)$ with probability $e^{(h(g)-heuristic)/temperature}$
24:                end if
25:                Remove $g$ from the $circuit$
26:            end for
27:            if $cost(circuit) + cost(best\_gate) > Cost$ then
28:                break                                ▷ This iteration failed
29:            end if
30:            Add $best\_gate$ to the $circuit$
31:            $temperature \leftarrow temperature - DECR_1$
32:            if $temperature \leq 0$ then
33:                $temperature \leftarrow temperature + DECR_1 +$ some Random value
34:            end if
35:            increment $length\_loop$ by 1
36:        end while
37:        $temperature \leftarrow temp - DECR_2$
38:        if $temperature \leq 0$ then
39:            $temperature \leftarrow temperature + DECR_2 +$ some Random value
40:        end if
41:        increment $i$ by 1
42:    end while
43:    return the minimum cost circuit among all $circuits$ in $circuit\_list$
44: end procedure
```

Fig. 1: Simulated Annealing Based Quine-McCluskey

where,

$don't\_care\_ratio = \frac{total\ number\ of\ don't\ care\ terms}{total\ length\ of\ all\ tables}$

$one\_bit\_ratio = \frac{total\ number\ of\ one\ bit}{total\ length\ of\ all\ tables}$

$total\_length = total\ length\ of\ all\ tables$

$Hamming\_distance = Hamming\ distance\ of\ t_{out}$

B. Ant Colony Optimization

In this approach, the synthesis problem is considered as exploring a DFS tree, in which every node at any particular level corresponds to a particular gate. A *pheromone* and *heuristic* is associated with each node. The root of the tree is not associated with any gate, having 0 *pheromone* and *heuristic*. Whenever the children of any node is explored for the first time, the *pheromone*s are initialized by a fixed value $INITIAL\_PHEROMONE$ and *heuristic*s are calculated by a procedure discussed in next subsection.

The algorithm starts with the input truth table and a blank circuit. We will assume, we have n number of processors available. Every time n number of ants are sent with an initial length and cost constraints. After exploring each node of the DFS tree, each ant invokes simulated annealing based Quine-McCluskey method, updates the $pheromone$ of the node individually, and continues with the next level, until length constraint is reached. If SA-QM returns a circuit, it updates it's own, as well as global length and cost constraints for next n ants. It continues until all ants finish their journey.

Among all $b2^{b-1}$ children of a particular node, the ant reaches the t^{th} child node g_t with probability,

$$p(g_t) = \frac{pheromone_t^{\alpha} \times heuristic_t^{\beta}}{\sum_{i=1}^{b2^{b-1}} pheromone_i^{\alpha} \times heuristic_i^{\beta}} \quad (2)$$

where $pheromone_i$ represents the pheromone level, and $heuristic_i$ represents the heuristic probability of the i^{th} gate node. After reaching the node, and adding the corresponding gate, it invokes SA-QM, and updates the pheromone of g_t based on the result of SA-QM, as follows,

$$\begin{aligned} pheromone_t = {} & EVAPORATION \times pheromone_t \\ & + new\_pheromone \end{aligned} \quad (3)$$

where,

$$new\_pheromone = \begin{cases} \dfrac{1}{min\_cost}, & \text{if SA-QM succeeds} \\ 0, & \text{if SA-QM fails} \end{cases} \quad (4)$$

where in the first case, invoked SA-QM returns with a circuit of cost min_cost.

C. Heuristic Probability Determination of DFS Tree Node

The path from root to any $node$ of the DFS tree represents an unique circuit, *e.g.*, the root represents blank circuit, the node with gate g at level 1 represents a circuit with gate g only. The heuristic probability used by ants while exploring the children of some $node$ is represented by the probability of adding the next gate in the circuit after $node$.

Assume the result of applying the circuit c, on the truth table t_{in} be the truth table t_{out}. We will first find out the probability for each bit to be changed first in t_{out}.

Assume that the input permutation of t_{out} be $(i_1, i_2, i_3, ..., i_{2^n})$ and the output permutation be $(z_1, z_2, z_3, ..., z_{2^n})$, where t_{out} is $n \times n$ truth table. Assume for some $k, 1 \leq k \leq 2^n$, where $i_k \neq z_k$, i_k and z_k differs in b_k bit positions. So, i_k can be changed to z_k in $b_k!$ ways, assuming no bit is changed more than once, as i_k can be changed to z_k by changing the b_k differing bits in any order.

Among these $b_k!$ ways, the number of ways in which any particular bit, say t^{th} bit is changed first, is,

$$w(t) = \begin{cases} (b_k - 1)!, & \text{if } t^{th} \text{ bit is changed from } i_k \text{ to } z_k \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

From the above discussion, we get the following immediate lemma.

Lemma 1: For any initial permutation $(i_1, i_2, i_3, ..., i_{2^n})$ *and final permutation* $(z_1, z_2, z_3, ..., z_{2^n})$, *if* $j_1, j_2, ..., j_m$ *be*

```
procedure ACO($Truth\_Table, Initial\_Cost,$
        $Initial\_Length$)
    $ant \leftarrow 0$
    $cost \leftarrow Initial\_Cost$
    $length \leftarrow Initial\_Length$
    $circuit \leftarrow blank\_circuit$
    while $ant \leq NUMBER\_OF\_ANTS$ do
        Send $NO\_OF\_PROCESSORS$ of $ants$
         in parallel.
        $node \leftarrow ROOT\_OF\_DFS$
        $length\_loop \leftarrow 1$
        while $length\_loop \leq length$ do
            if children of $node$ are uninitialized then
                initialize children of $node$.
            end if
            Select next $node$ $g_t$ with
            probability $p(g_t)$
            Add $node.gate$ to $circuit$
            if $circuit$ solves $Truth\_Table$ then
                $cost \leftarrow cost(circuit)$
                $length \leftarrow length(circuit)$
                update $pheromone_t$ with
                $new\_pheromone \leftarrow \frac{1}{cost}$
                break
            end if
            $min\_circuit \leftarrow SA - QM$
            $(Truth\_Table, circuit, cost, length)$
            if $min\_circuit$ exists and
            $cost(min\_circuit) < cost$ then
                $cost \leftarrow cost(min\_circuit)$
                $length \leftarrow length(min\_circuit)$
                update $pheromone_t$ with
                $new\_pheromone \leftarrow \frac{1}{cost}$
            else
                update $pheromone_t$ with
                $new\_pheromone \leftarrow 0$
            end if
            $length \leftarrow length - 1$
            $length\_loop \leftarrow length\_loop + 1$
        end while
        $ant \leftarrow ant + NO\_OF\_PROCESSORS$
    end while
    return $min\_circuit$
end procedure
```

Fig. 2: Ant Colony Optimization

the indices, for which $i_{j_k} \neq z_{j_k}$, *and at these indices the permutations differ in* $b_1, b_2, ..., b_m$ *bit positions, then for any bit t, the probability that the immediate next gate in the circuit controls the* t^{th} *bit is.*

$$p(t) = \frac{\alpha_1(b_1 - 1)! + \alpha_2(b_2 - 1)! + ... + \alpha_m(b_m - 1)!}{b_1! + b_2! + ... + b_m!} \quad (6)$$

where,

$$\alpha_k = \begin{cases} 1, & \textit{if } t^{th} \textit{ bit is changed from } i_{j_k} \textit{ to } z_{j_k} \\ 0, & \textit{otherwise} \end{cases} \quad (7)$$

For an $n \times n$ truth table, the number of CNT gates having control at any particular bit t is 2^{n-1}. So, the probability of next immediate gate g is $\frac{p(t)}{2^{n-1}}$, where the gate g controls the bit t.

IV. Experimental Result

We evaluated the proposed algorithm on several reversible circuits by implementing the algorithm using Java on a 6-core $i7$ ($NO\_OF\_PROCESSORS = 6$) machine with 8GB memory. For 3×3 circuits we used 1000 ants ($NO\_OF\_ANTS = 1000$) and for 4×4 circuits we used

4000 ants ($NO\_OF\_ANTS = 4000$) for the ACO part. For a $n \times n$ circuit, the initial length($Initial\_Length$) is taken as $n2^{n-1}$, as observing a large amount of circuits, the following hypothesis can be developed.

Hypothesis 1: For any $n \times n$ truth table, there exists at least one circuit with maximum of $n2^{n-1}$ gates, that can synthesize the truth table.

In ACO, the initial cost($Initial\_Cost$) is set as positive infinity. While exploring any new node, the initial pheromone ($INITIAL\_PHEROMONE$) is being taken as 0.8. The probability parameters are taken as $\alpha = 0.5$ and $\beta = 0.5$. The pheromone evaporation rate($EVAPORATION$) is being taken as 0.7. In the SA based QM method, we set initial temperature($INITIAL\_TEMPERATURE$) as 20.0, number of iterations ($ITERATION$) as 10, local temperature decrease rate ($DECR_1$) as 0.4, and global temperature decrease rate ($DECR_2$) as 0.5. A proper tuning of these parameters can further improve the result.

Table II shows the comparison with MOSAIC [6] and PPRM [7] methods for several circuits. Though our algorithm

TABLE II: *Gate Count comparison with MOSAIC and PPRM methods for Benchmark Circuits*

Function Name	Functions	Gate Count		
		MOSAIC	PPRM	SA-QM and ACO
rand_3_1	[7,0,1,2,3,4,5,6]	3	3	3
rand_3_2	[0,1,2,3,4,6,5,7]	3	3	5
rand_3_3	[0,1,2,4,3,5,6,7]	7	5	6
rand_3_4	[1,2,3,4,5,6,7,0]	3	3	3
rand_3_5	[3,6,2,5,7,1,0,4]	8	7	8
rand_3_6	[1,2,7,5,6,3,0,4]	8	6	7
rand_3_7	[4,3,0,2,7,5,6,1]	6	7	7
rand_3_8	[7,5,2,4,6,1,0,3]	6	7	7
rand_3_9	[1,0,3,2,5,7,4,6]	4	4	5
rand_4_1	[13,1,14,0,9,2,15,6,12,8,11,3,4,5,7,10]	29	16	14
rand_4_2	[0,1,2,3,4,5,6,8,7,9,10,11,12,13,14,15]	9	7	**10**
rand_4_3	[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0]	4	4	4
rand_4_4	[0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5]	4	4	4
rand_4_5	[6,2,14,13,3,11,10,7,0,5,8,1,15,12,4,9]	19	15	14

gives higher gate count for the function $rand\_4\_2$, but it gives lower circuit cost of 38 over PPRM, which gives the circuit of cost 51. The two figures 3 and 4 shows the comparison bar graphs with MOSAIC and PPRM for 3×3 and 4×4 circuits respectively. As can be seen from the bar graphs,

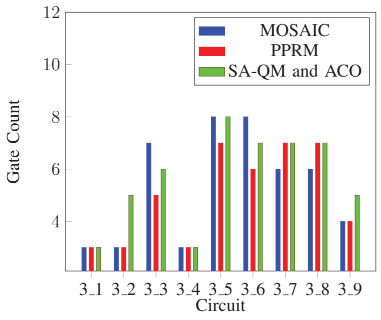


Fig. 3: Comparison Graph with MOSAIC and PPRM for 3×3 circuits

for most of the 3×3 circuits, our algorithm has generated circuit with average gate count with MOSAIC and PPRM, and for 4×4 circuits, our algorithm has generated circuits with lower or equal gate count, except one case $rand\_4\_2$, in

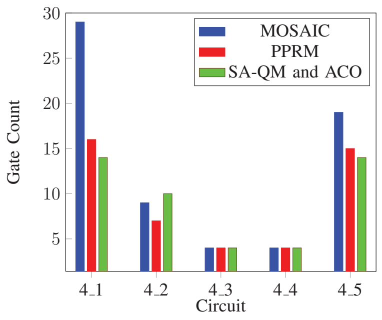


Fig. 4: Comparison Graph with MOSAIC and PPRM for 4×4 circuits

TABLE III: *Cost Comparison with Revlib Benchmark Circuits*

Functions	Cost		Cost Increased
	RevLib Minimum Cost	SA-QM and ACO	
ham_3_28	9	9	0%
3_17_6	14	14	0%
4_49_7	32	36	12.5%
hwb4_12	23	26	13%

which our cost is better, as discussed before. Table III shows the cost comparison with 2 3×3 circuits ham_3_28 and 3_17_6, and 2 4×4 circuits 4_49_7 and $hwb4\_12$, with minimum cost provided on the $revlib$ website [8] of the corresponding circuits.

V. Conclusion

Synthesis of reversible circuits has drawn the attention of many researchers in recent years due to its promising aspects in tomorrow's lossless computing. In this work, an algorithm has been proposed using Ant Colony Optimization and Simulated Annealing based Quine-McCluskey approach to synthesize a reversible circuit. This approach is producing optimal or near optimal circuits for most of the cases. Major advantage of this algorithm is that the constant factors of the algorithm can be modified or tuned by the user to get better result.

References

[1] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, Apr. 1965.
[2] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Develop.*, vol. 5, no. 3, pp. 183–191, July 1961.
[3] The International Technology Roadmap for Semiconductors. [Online]. Available: http://www.itrs.net/
[4] C. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.*, vol. 17, no. 6, pp. 525–532, Nov. 1973.
[5] A. Mishchenko and M. Perkowski, "Logic synthesis of reversible wave cascades." Proc. Int. Workshop Logic Synthesis, June 2002, pp. 197–202.
[6] M. S. Z. M. Saeedi and M. Sedighi, "Moving forward: A non-search based synthesis method toward efficient cnot-based quantum circuit synthesis algorithms." ASPDAC, Jan. 2008, pp. 83–88.
[7] N. J. P. Gupta, A. Agrawal, "An algorithm for synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2317–2330, Nov. 2006.
[8] An online resource for reversible benchmarks. [Online]. Available: http://www.revlib.org/