Multi–Valued Optimization on Post Logic Networks

April 16, 2002

Abstract

We address the problem of developing optimization methods for multi-valued multi-level networks where the node functions are represented by Post expressions. Such networks are called Post networks. Post expressions are multi- valued expressions based on chain-based Post algebra. Another kind of expression used for general MV multi-level logic, called i-set expressions, has been studied and a complete set of multi-level optimization methods on their corresponding networks has been developed. We examined these optimization methods and successfully adapted some, including two important semi-algebraic optimization methods, to operate directly on Post networks. The difficulties encountered in adapting the remaining methods are analyzed.

1 Introduction

Multi-valued logic is a generalization of classical binary logic. In MV logic, each variable can have more than 2 values. An MV network can be seen as a network of nodes where each node in the network represents an MV-function with a single multi-valued output. This function can be represented is many different ways. There is an edge connecting node i to node j if and only if the function at j syntactically depends on the variable associated with node i. Logic synthesis on such general MV logic networks can have many applications, including logic synthesis for multi-valued hardware devices, e.g., current-mode circuits, as an initial synthesis step of a hardware description before it is encoded into binary, and as a method for optimizing software.

There are different formats for the expressions representing multi-valued output functions. Different formats can lead to significant differences in the sizes of expressions for the same logic function. One format is *i-set mode*, used in some MV synthesis methods. A complete set of operations on i-set expressions has been developed [2], including algebraic MV methods, such as *node extraction, decomposition, substitution*, the *don't care-based* optimization methods for node simplification, and operations for network restructuring, such as *collapse, merge and pair decode*. These synthesis methods are implemented in a program called MVSIS [2].

However, other formats for representing the node functions of MV networks are of interest as well as associated synthesis methods for them. Other formats may provide simpler expressions for the same logic and/or may have direct applications. Also, if it is possible to convert between two representations, then operations in one mode may lead to results that, when translated back to the other mode, would not have been produced directly.

We examime a format, based on chain-based Post algebra, leading to *Post expressions*, for general multi-valued logic. Compared to i-set expressions, Post expressions have smaller size.

We first introduce the Post format and illustrate the connection between Post expressions and i-set expressions. Then the main algorithms for semi-algebraic synthesis methods on Post expressions are given. We also investigate the other synthesis methods on Post networks and compare some results obtained by i-set and Post operations.

2 Post expressions

A multi-valued function is a discrete function whose input and output variables take two or more values. Formally, an n-variable multi-valued function $f(x_1, \ldots, x_n)$ is a mapping $f : P_1 \times P_2 \times \ldots \times P_n \to M$, with the variable x_i taking values from the sets $P_i = \{0, 1, 2, \ldots, p_i - 1\}, p_i > 1, i \in \{1, 2, \ldots, n\}$, and the function f taking its value from the set $M = \{0, 1, \ldots, m - 1\}, m > 1$.

The *i-set* format is such that the values of a multi-valued variable are unordered, i.e., we don't say one value is 'greater' than another; all values have the same priority. For the i-set mode, a multi-valued output function f is represented by a set of m multi-valued input and binary output functions $\{f^0, f^1, \ldots, f^{m-1}\}$. The binary function f^i evaluates to $m-1$ (the maximum value) iff f evaluates to i.

Chain-based Post algebra is based on a totally ordered set of elements, plus the operations maximum (MAX), minimum (MIN) and literal. When used to represent multi-valued functions, the ordered set is the union of the input domains of the variables and the output domain of the function, i.e., $M \bigcup (\bigcup_{i=1}^{n} P_i)$. *Maximum* $\mathrm{MAX}(x_i, x_j)$ and *minimum* $\mathrm{MIN}(x_i, x_j), i, j \in \{1, 2, \ldots, n\}$, are binary operations of type $P_i \times P_j \rightarrow M$. The *literal* x_i^S is a unary operation of type $P_i \rightarrow \{0, m-1\}$, defined by

$$x_i^S = \begin{cases} m-1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}$$

where $x_i \in P_i$ is a multi-valued variable and $S \subseteq P_i$ is a set.[1]

Similar to the binary logic case, we can define *cube* and *sum-of- products* in *Post mode format* based on MAX and MIN, where MIN takes the place of AND, and MAX takes the place of OR.

Definition 1 (Post cube) *Post cube is a product of literals or multi-valued variables of type $\alpha \cdot X_1 \cdot X_2 \cdot \ldots \cdot X_n$, where $\alpha \in M$ is a constant, product "$\cdot$" is MIN, X_i is either a multi-valued variable x_i or some literal $x_i^{S_i}$.*

Definition 2 ($>$, $=$) *Post cubes c_1, c_2 have relation $c_1 > c_2$ if the value of c_1 is greater or equal to the value of c_2 under any input vector and there exists at least one vector where the value of c_1 is greater than that of c_2. Relation $c_1 = c_2$ holds if the value of c_1 is equal to the value of c_2 under any input vector.*

Based on the definitions of $>$ and $=$, we can define $<$, $\geq$ and $\leq$.

A *sum-of-products* in Post mode is a sum (MAX) of Post cubes. Throughout the paper, we call such a SOP a *Post SOP* or simply, Post expression.[2]

Post expressions and i-set expressions are convertible, given the ordering on the value set. For a multi-valued function f which has an i-set expression $f = \{f^0, f^1, \ldots, f^{m-1}\}$, its Post expression can be $f = \sum_{i=0}^{m-1} (i \cdot g^i)$ where g^i is a binary function and satisfies $f^i \subseteq g^i \subseteq \bigcup_{j=i}^{m-1} f^j$. In other words, g^i can be obtained by simplifying f^i with $\bigcup_{j=i+1}^{m-1} f^j$ as don't cares [1].[3] Such don't cares are called *priority don't cares* and are a direct result of the ordering imposed on the output values and the use of the MAX operator. Thus the Post expression can be significantly smaller than the set of i-set expressions for the same function.[4] The i-set expression can be recovered by removing minterms in g^i which also are contained in $\sum_{j=0}^{i-1} g^j$.[5]

Consider the following multi-valued (3-valued) function:

$x_2 \backslash x_1$	0	1	2
0	0	0	1
1	2	2	2
2	0	0	1

The i-set expressions for this function is:

$$\begin{aligned} f^0 &= x_2^{0,2} \cdot x_1^{0,1} \\ f^1 &= x_2^{0,2} \cdot x_1^{2} \\ f^2 &= x_2^{1} \end{aligned}$$

The Post expression for the same function is:

$$f = 2 \cdot x_2^1 + 1 \cdot x_1^2 + 0$$

The 0 item in the Post expression is usually eliminated.

As mentioned, a set of MV optimization methods on i-set mode networks has been developed and since Post expressions and i-set expressions are convertible, we can convert Post expressions into i-set expressions, do optimizations and then convert back, thus performing optimization on Post networks. However, direct optimization methods on Post networks are desirable because in some applications Post expressions are preferred. In addition, for most MV logic functions, the Post expression is smaller than the corresponding i-set expression. Thus, generally, optimization can be sped up since the time complexity of optimization algorithms is typically proportional to the size of the input logic[6]. Also, the optimization of the Post network may result in a corresponding optimization in the i-set mode network that could not be obtained directly.

[1] The value $m-1$ is used since it is the maximum value and its use (instead of 1) is dictated because MIN is used in place of AND in the SOP Post expressions.

[2] A Post sum of products should be called a maximum of minimums or MOM.

[3] The reason is that since g^i is multiplied by i in the Post SOP, and MAX is used for the sum, then if a minterm is in both g^i and g^j where $i > j$, the appearence in g^j will have no effect on the evaluation of the Post expression for that minterm.

[4] It is important, when converting from an i-set expression to a Post expression, to find a good ordering. One heuristic is to assign the largest i-set to 0. Generally, we order the i-sets from the largest (assigned 0) to the smallest (assigned $p_i - 1$).

[5] This discussion assumes that the i-set expression is deterministic. In general, i-set expressions can represent non-deterministic relations by allowing the different i-set functions to share minterms. This sharing may be the result of don't cares, either SDC, ODC or EXDC. Post expressions can also represent non-determinism by allowing the don't care set to be used in minimizing all of the $\{g^i\}$.

[6] Our experimental results support this speed-up.

3 Semi-algebraic methods

3.1 Factoring and decomposing

Past use of Post expressions has been restricted to two-level logic [1]. Synthesis methods for multi-level logic, such as *decompose*, need to be developed on Post expressions to extend them to multi-level networks. Since factorization and decomposition are tightly related, we will first introduce an algorithm, similar to the *satisfiable matrix* algorithm for i- set mode [3], for factorization of Post expressions. Then the application of this algorithm to decomposition will be given.

Some definitions are necessary before introducing this algorithm.

Definition 3 (supercube) *The supercube of a set of Post cubes $C = \{c_1, c_2, \ldots, c_n\}$, denoted $\sigma(C)$ is the cube that satisfies:*

1. *$\sigma(C) \geq c_i$ for any $c_i \in C$;*
2. *$\sigma \geq \sigma(C)$ for any Post cube σ satisfying condition 1.*

Definition 4 (Post satisfiable matrix) *A matrix A_{mn} is satisfiable if any entry $c_{i,j}$ of A is a Post cube and $c_{i,j} = \sigma(C_{rin}) \cdot \sigma(C_{cjm})$ where C_{rin} represents the set of cubes $\{c_{i1}, \ldots, c_{in}\}$ and C_{cjm} represents the set of cubes $\{c_{1j}, \ldots, c_{mj}\}$.*

Clearly, if a set of cubes can be arranged into a Post satisfiable matrix, they have a factored form. We give an algorithm to build up such matrices. The most important part of this algorithm is the method for bounding in a branch and bound technique. The following theorem provides a bounding method.

Theorem 1 *A matrix A_{mn} is satisfiable if and only if each entry $c_{i,j}$ satisties the following two conditions:*

1. *$c_{i,j} = \sigma(C_{rij}) \cdot \sigma(C_{cji})$;*
2. *$c_{i,k} = \sigma(C_{rij}) \cdot \sigma(C_{cki}), c_{l,j} = \sigma(C_{rlj}) \cdot \sigma(C_{cji})$ for any $k < j, l < i$.*

Proof: By induction on the dimensions of A.

1. $m = 1, n = 1$. Condition 1 holds since $\sigma(C_{rmm}) = \sigma(C_{cnn}) = c_{m,n}$. No entry is specified by Condition 2.
2. Assume that matrices $A_{i,j-1}$ and $A_{i-1,j}$ are satisfiable. Adding entry $c_{i,j}$ only affect the supercubes of row i and column j. So we still have $c_{l,k} = \sigma(C_{rlj}) \cdot \sigma(C_{cki})$, for any $k < j, l < i$. Condition 1 and 2 claim that this is also true for cubes on row i and column j. So matrix $A_{i,j}$ is satisfiable. **QED**

For a Post SOP form $f = c_1 + c_2 + \ldots + c_n$, the Post satisfiable matrix is built up column by column. At each position $A(i,j)$, only cubes that satisfy the two conditions specified by Theorem 1 are considered. Such checking has a linear time complexity to the size of Post satisfiable matrix.

We have shown that the Post satisfiable matrix method can be used on Post expressions for factorization. Once we find a factored form for function $f = \sum_{i=1}^{m}(d_i) \cdot \sum_{j=1}^{n}(q_j) + r$, f can be decomposed as $f = \sum_{j=1}^{n}(q_j \cdot g) + r, g = \sum_{i=1}^{m}(d_i)$. The resulting f and g are both Post expressions. So the Post matrix method can be applied iteratively for decomposition.

Example 3.1 Consider the Post expression:

$$\begin{aligned} f &= 3 \cdot x \cdot u^1 + 2 \cdot x^{\{0,3\}} \cdot w^1 + 2 \cdot x \cdot u^1 \cdot y^1 \\ &\quad +2 \cdot x \cdot y^1 \cdot w^1 + 1 \cdot x^0 \cdot v^1 + 1 \cdot x^{\{1,2\}} \cdot v^1 \cdot y^1 \end{aligned}$$

One possible Post satisfiable matrix can be:

	$3 \cdot x \cdot u^1$	$1 \cdot x^{\{0,1,2\}} \cdot v^1$	$2 \cdot w^1$
$3 \cdot x^{\{0,3\}}$	$3 \cdot x^3 \cdot u^1$	$1 \cdot x^0 \cdot v^1$	$2 \cdot x^{\{0,3\}} \cdot w^1$
$2 \cdot x \cdot y^1$	$2 \cdot x \cdot u^1 \cdot y^1$	$1 \cdot x^{\{1,2\}} \cdot v^1 \cdot y^1$	$2 \cdot x \cdot y^1 \cdot w^1$

f can be decomposed as:

$$\begin{aligned} f &= 3 \cdot x \cdot u^1 \cdot g + 2 \cdot w^1 \cdot g + 1 \cdot x^{\{0,1,2\}} \cdot v^1 \cdot g \\ g &= 3 \cdot x^{\{0,3\}} + 2 \cdot x \cdot y^1 \end{aligned}$$

Note that g is a multi-valued output function. In contrast, decomposition on i-set expressions yields only binary output functions. Thus decomposition on Post expressions can provide results that cannot be produced by direct application of decomposition of i-set expressions.

3.2 Exact division

Besides factorization and decomposition, the Post satisfiable matrix method can also be used for inexact and exact division [3]. Even though exact division has more constraints, the Post satisfiable matrix method is slower than on factorization using inexact division. It is necessary to develop faster methods for exact division because it is used very frequently in fast extraction and resubstitution. An algorithm similar to *maximum graph-matching method* used on i-set multi- valued expressions is introduced next for exact division in Post mode. Two lemmas guarantee the correctness of this algorithm.

Lemma 1 *If $d_1 \cdot q = c_1$ and $d_2 \cdot q = c_2$, where d_1, d_2, q, c_1, c_2 are Post cubes, then $q \geq \sigma(c_1, c_2)$.*

Proof: If there exists some q that satisfies $d_1 \cdot q = c_1$, then $d_1 \geq c_1$ and $q \geq c_1$. By the same reason, we have $q \geq c_2$. This fact with the definition of supercube leads to $q \geq \sigma(c_1, c_2)$. **QED**

Lemma 2 *If $d_1 \cdot q = c_1$ and $d_2 \cdot q = c_2$ can be satisfied, then they must be able to be satisfied by $q = \sigma(c_1, c_2)$.*

Proof: From Lemma 1, $q \geq \sigma(c_1, c_2)$. So $d_i \cdot q \geq d_i \cdot \sigma(c_1, c_2)$, $i = 1, 2$. Assume that $\sigma(c_1, c_2)$ cannot satisfy $d_i \cdot q = c_i$, then $d_i \cdot \sigma(c_1, c_2) > c_i$. So $d_i \cdot q > c_i$ and it contradicts that q satisfies $d_i \cdot q = c_i$,$i = 1, 2$. **QED**

The exact division problem can be specified as: given a two-cube divisor $d = d_1 + d_2$ and a multi-valued function $f = c_1 + c_2 + \ldots + c_n$, find a quotient q with the maximum number of cubes such that $f = d \cdot q + r$.

We illustrate the algorithm with an example where f is the function specified in Example 3.1 and $d = 3 \cdot x^{\{0,3\}} + 2 \cdot x \cdot y^1$. This algorithm includes 3 main steps:

1. Find all candidate cubes for d_1 and d_2. Any cube c in f is a candidate cube for d_i if $c \leq d_i$.

   In the example, the candidate cubes for $d_1 = 3 \cdot x^{\{0,3\}}$ are $\{3 \cdot x^3 \cdot u^1,\ 2 \cdot x^{\{0,3\}} \cdot w^1,\ 1 \cdot x^0 \cdot v^1\}$, and the candidate cubes for $d_2 = 2 \cdot x \cdot y^1$ are $\{2 \cdot x \cdot u^1 \cdot y^1,\ 1 \cdot x^{\{1,2\}} \cdot v^1 \cdot y^1,\ 2 \cdot x \cdot y^1 \cdot w^1\}$.

2. Find all compatible pairs among the candidate cubes. Two cubes c_{1j} and c_{2k} are said to be compatible if there exists a Post cube q, such that $c_{1j} = d_1 \cdot q$ and $c_{2k} = d_2 \cdot q$. Compute $\sigma(c_{1j}, c_{2k})$, denoted q', and check if $c_{1j} = d_1 \cdot q'$ and $c_{2k} = d_2 \cdot q'$ hold. If so, c_{1j} and c_{2k} are compatible.

   In the example, we have 3 compatible pairs: $(3 \cdot x^3 \cdot u^1,\ 2 \cdot x \cdot u^1 \cdot y^1)$, $(2 \cdot x^{\{0,3\}} \cdot w^1,\ 2 \cdot x \cdot y^1 \cdot w^1)$, $(1 \cdot x^0 \cdot v^1,\ 1 \cdot x^{\{1,2\}} \cdot v^1 \cdot y^1)$.

3. Make a bipartite graph: each candidate cube is a vertex; there is an edge from c_{1j} and c_{2k} if and only if they are compatible. Solve the maximum graph-matching problem.

   The bipartite graph in the example contains 3 edges and all of them are in the solution.

This problem has a complexity of $O(n^3)$ where n is the number of vertices in the graph. q can be constructed from the solution of the graph-matching problem. Suppose the edge between vertices c_{1i} and c_{2i} is in the solution, then $q_i = \sigma(c_{1i}, c_{2i})$ is one cube in q. For our example, $q = 3 \cdot x \cdot u^1 + 2 \cdot w^1 + 1 \cdot x^{\{0,1,2\}} \cdot v^1$.

One criticism about this method is that q_i may have other forms, and the one we get tends to have as few values for each literal as possible. It is unclear which form would work best.

In summary, we developed algorithms for factorization, decomposition and division on Post expressions. These are key algorithms for algebraic optimization methods, such as factor, decompose, node extract and resubstitute. These algebraic methods can be used to derive an appropriate structure for a multi-valued multi-level network. We have extended Post expression optimization from 2-level logic to multi-level logic.

4 Other Minimization methods

There are other optimization methods besides algebraic methods that we also want to extend to Post expressions, such as *collapse, merge, eliminate* and the *don't care-based* optimization methods. Some of these extensions are shown to be successful while some are not. Next we give a brief explanation.

One disadvantage of Post expressions is that the function of a literal is not explicitly available. For example, from the Post expression,

$$f = 2 \cdot x_2^1 + 1 \cdot x_1^2$$

we cannot get the function for f^1 directly. When such a function is needed, we need to convert the Post expression to the i-set expressions. This disadvantage is the main reason for the unsuccessful extensions.

The operations *collapse or eliminate* are used to modify the structure of a MV multi-level network. They can be used to reduce the depth of a network by collapsing some node into its fanouts. Unfortunately, in this type of operation, the functions of literals are usually needed. In order to collapse node f into its fanouts, it seems necessary to replace any literal f^i in its fanouts by its corresponding function. As an example, suppose one fanout of $f = 2 \cdot x_2^1 + 1 \cdot x_1^2$ is $g = 2 \cdot f^1 + 1 \cdot f \cdot y^1$. To directly replace f^1 with x_1^2 leads to the wrong result where input vector $(x_1, x_2, y) = (2, 1, 1)$ produces $g = 2$ while the correct value of g is 1. So there seems no easy way to extend this operation to Post expressions. If we desire to optimize a Post network, the only alternative we have seems to be to convert to i-set mode, collapse, and then convert back to Post mode.

Besides algebraic optimizations, important optimization methods include the don't care-based optimizations methods [4]. Once the structure of the whole logic has been decided, the multi-valued function at each node can be optimized using some flexibility allowed for this node. This flexibility is given by satisfiability don't cares (SDC), observability don't cares (ODC) and observability partial cares (OPC). SDC expresses variable combinations that can't happen. For a MV function node y in i-set mode $(f^0, f^1, \ldots, f^n)$, the SDC for y can be expressed as:

$$SDC_y^i = \sum_{l=0}^{n} y^{U-\{l\}} f^l$$

where U represents the universal set $\{0, 1, \ldots, n\}$. Directly from the Post expression $y = n \cdot g^n + \ldots + 1 \cdot g^1$ where $f^l \subseteq g^l \subseteq \bigcup_{j=l}^{n} f^j, l = 0, 1, \ldots, n$, we can get

$$SDC_y^P = \sum_{l=0}^{n} y^{\{0,\ldots,l-1\}} g^l$$

Note that because of the use of priority don't cares, we lose a part of SDC to keep the calculation correct, i.e., $SDC_y^P \subset SDC_y^i$. The computation of ODC is much more complex and it needs collapsing. Thus up to now no efficient way has been found to extend ODC-based optimization methods onto Post expressions.

Thus it seems that the only feasible approach for assembling a complete set of operations that work directly on Post networks is to surround some of the operations with an automatic conversion to and from i-set expressions applying the operation to the resulting i-set expressions.

5 Experiments

We presented algorithms for factorization, decomposition and division on Post networks in Section 3 but implementations for only a subset have been completed. However, some operations on i-set networks can be applied directly to Post networks, such as *decomp* and *fx* [2], since these operations actually work on multi-valued-input and binary-output functions. We can use them on $g^n, g^{n-1}, \ldots, g^1$ if function f is in Post mode $f = n \cdot g^n + (n-1) \cdot g^{n-1} + \ldots + 1 \cdot g^1$.

We mentioned that Post mode operations might have several positive effects: one is that Post mode logic can be optimized directly; another is it might speed up some optimization operations on i-set networks since Post expressions are usually smaller than those for i-set mode; a third is to obtain results that would not be obtained if only one mode is used. We did some experiments using *decomp, fx* and *eliminate* in MVSIS [2] to illustrate the effectiveness of using the Post mode in improving time and size performance.

The MV examples used in the experiments are shown in Table 1. We compare two approaches, both with respect to size and total run time. Each approach is applied to the same set of i-set mode networks.

- Approach 1: *decomp, fx, eliminate*
- Approach 2: *compress, decomp, fx, uncompress, eliminate*

Compress is the operation to convert an i-set network to a Post network and *uncompress* is the operation to convert back. The total run time includes the run time for all operations in each approach. Thus Approach 2 has the additional overhead of converting to and from Post networks. Experimental results are shown in Table 2.

	#cubes	#literals	nodes
adder_mod4	24	48	2
balance	337	1348	1
iris	100	400	1
mm3	111	555	1
mm4	598	2990	1
mm5	112	506	1
ele-ctr-det	1633	3542	1446
coh-dir	996	1917	653
bakery-proc	813	1788	258
matmul	128	480	4

Table 1: Initial characteristics of the examples.

From Table 2, we see that for most examples, the times are comparable or better with Approach 2; for some examples, better results are obtained, mainly because of more extensive use of the complements of divisors. The operation *uncompress* involves the intersection of a function of lower priority with the complements of those with higher priorities. So if a divisor has been extracted from a function with higher priority, its literals are used in that function, possibly causing the complements of those literals to be introduced into the functions with lower priority. This can result in large savings since in i-set mode, we usually do not use the complement of a divisor if it contains more than 2 cubes because division by it would be very expensive.

In some examples, Approach 2 gets worse results (last two in Table 2). The reason is that good divisors in Post mode may not be as good in i-set mode. However, the speed improvement and possible savings obtained using Post mode is often desirable; it leads to the idea of using Post mode operations first if the input network is very large and then using i-set mode operations.

	Approach 1		
	#cubes	#literals	time(s)
adder_mod4	24	60	≤ 0.1
balance	-	-	-
iris	57	187	2.1
mm3	26	64	1.2
mm4	140	393	65.8
mm5	126	338	7.7
ele-ctr-det	193	439	3.0
coh-dir	254	566	4.4
bakery-proc	147	338	1.4
matmul	112	248	1.2

	Approach 2		
	#cubes	#literals	time(s)
adder_mod4	26	56	≤ 0.1
balance	230	1100	3.8
iris	31	114	0.3
mm3	16	37	≤ 0.1
mm4	41	112	0.2
mm5	78	242	1.0
ele-ctr-det	180	407	2.5
coh-dir	240	546	4.4
bakery-proc	183	650	1.4
matmul	112	304	0.6

Table 2: Experimental results[7] by Approachs 1 and 2.

6 Conclusion

We introduced Post expressions and Post networks, which are multi-valued multi-level logic networks based on chain-based Post algebra. Key algorithms for algebraic optimization were developed on Post expressions. We also analyzed if other multi-valued optimization methods can be extended to Post expressions. Difficulties encountered in attempting such extensions were analyzed.

Reference

[1] Elena Dubrova, Yunjian Jiang and Robert Brayton, Minimization of Multiple- valued Functions in Post Algebra, *International Workshop on Logic Synthesis, Tahoe City*, June 2001.

[2] M.Gao, Y.Li, J-H. Jiang, Y.Jiang, A.Mishchenko, S.Sinha, T.Villa, and R.Brayton, The Optimization of Multi-valued Multi-level Networks, *IEEE International Symposium on Multi-valued Logic*, 2002.

[3] Minxi Gao. Multi-valued Multi-level Logic Synthesis. Master thesis, 2000.

[4] Yunjian Jiang. Multi-valued Logic Network Minimization and its Application. Master thesis, 2000.

[7]'-' means that no result was obtained in reasonable time.