SYMBOLIC MINIMIZATION OF LOGIC FUNCTIONS

Giovanni De Micheli

IBM - T.J.Watson Research Center
P.O.Box 218, Yorktown Heights, NY 10598

ABSTRACT

Symbolic minimization consists of determining a two-level sum-of-products representation of a switching function in a minimal number of product-terms, independently of the Boolean encoding of all (or part of) the inputs and outputs. When the objective is to determine a binary-valued description, the minimal symbolic representation determines a set of constraints for the encoding of the input and outputs into Boolean variables. Symbolic minimization is related, but not equivalent, to **multiple-valued** logic minimization. It represents a new concept in switching theory. Formal definitions and properties of a symbolic representation of a switching function are presented for the first time. An algorithm for symbolic minimization is described. The algorithm is implemented in a computer program called ***CAPPUCCINO***, which is built on top of a modified version of the heuristic logic minimizer ***ESPRESSO-II***.

1. INTRODUCTION

In standard logic design, Boolean representations of switching functions are obtained from a functional description, such as a Hardware Description Language program, by representing each variable by Boolean symbols. The optimization of logic functions, and in particular two-level logic minimization, is performed on the Boolean representation. The result of logic optimization is heavily dependent on the representation of the variables. As an example, the complexity (in particular the minimal cardinality of a two-level implementation) of the combinational component of a finite-state machine depends on the assignment of Boolean variables to the internal states. Here, the design of switching functions is restricted to combinational two-level sum-of-products representations. Logic optimization is therefore equivalent to logic minimization, i.e. finding a representation of minimal cost, where the cost is the number of product-terms involved.

A technique to avoid the dependence on the variable representation in the optimization process consists of two steps: i) determine a minimal representation of a switching function independently on the encoding of its inputs and outputs; ii) encode the inputs and outputs so that they are compatible with the minimal representation.

This method can be applied to solve the following problems of logic design:

P1) Find an encoding of the inputs (or some inputs) of a combinational circuit that minimizes its cost.
P2) Find an encoding of the outputs (or some outputs) of a combinational circuit that minimizes its cost.
P3) Find an encoding of both the inputs and the outputs (or some inputs and some outputs) of a combinational circuit that minimizes its cost.
P4) Find an encoding of both the inputs and the outputs (or some inputs and some outputs) of a combinational circuit that minimizes its cost and such that the encoding of the inputs is the same as the encoding of the outputs (or the encoding of some inputs is the same as the encoding of some outputs.)

Finding an optimal state assignment of a sequential circuit is equivalent to problem P4, when the sequential circuit is implemented by feeding back (possibly through registers) some outputs of a combinational circuit to its inputs. Similarly, finding the encodings of the signals connecting two (or more) combinational circuits, that minimize the total cost, can be reduced to problem P4. The author presented in [DEMI84c] [DEMI85a] an approximation to the solution of the state assignment problem (P4), in which the cost was minimized with regard only to the encoding of the inputs. In particular, the technique presented in [DEMI84c][DEMI85a] solved only problem P1. Problem P2 was attacked by Nichols [NICH65], but the algorithm he presented could deal only with small-scale circuits.

The difficulty in solving problems P2-P4 is related to finding a minimal two-level representation of a switching function independently of the encoding of both inputs and outputs. **Symbolic minimization** consists of determining the minimum encoding-independent two-level sum-of-products representation of a switching function. Is is minimum in number of product-terms and independent of the encoding of all (or part of) the inputs and outputs. The minimal symbolic representation determines a set of constraints for the encoding of the inputs and outputs.

This paper describes how symbolic minimization is defined and achieved. Formal definitions of the representation of symbolic functions are reported in Section 2. An algorithm for symbolic minimization is described in Section 3. The related constrained encoding problems are sketched in Section 4. Extensive description and results are reported in [DEMI85a] [DEMI85b].

2. BASIC CONCEPTS AND DEFINITIONS

Symbolic functions are switching functions whose variables can take a finite set of values. Each value is represented by a **word** (or mnemonic), i.e. by a string of symbols. A symbolic variables s has a set of admissible values S. The symbol ϕ is reserved to denote that variable s does not take any value of S. For functions of n input variables and m output variables, let S_i^I, $i = 1, 2, \dots, n$ and S_i^O, $i = 1, 2, \dots, m$ be the set of admissible values for the corresponding variables s_i^I and s_i^O. Then the domain of the symbolic function is the Cartesian product $S^I \equiv S_1^I \times S_2^I \times \cdots \times S_n^I$ and the range is the Cartesian product $S^O \equiv S_1^O \times S_2^O \times \cdots \times S_m^O$. The elements of the domain are denoted by s^I and those of the range by s^O.

A **completely specified symbolic function** of n input variables and m output variables is a function $f: S^I \rightarrow S^O$, that maps each element of the domain to an element of the range. An **incompletely specified symbolic function** is a function having the property that, for some inputs, some output variables can take any value in the corresponding range. The collection of these points of the domain is called the **don't care set** of that particular output variable.

Note that the **Boolean** or **binary-valued** functions are symbolic functions, whose variables can take the values $S = \{0, 1\}$. The domain is $\{0, 1\}^n$. The range of a completely specified function is $\{0, 1\}^m$. For incompletely specified functions, let the symbol * represent the *don't care* condition. Then the range is $\{0, 1, *\}^m$. Similarly, the variables of **multiple-valued** functions can take the values $S = \{0, 1, \dots, p-1\}$, where p is the radix of the representation [RINE77]. Algebras have been developed for both the Boolean and the multiple-valued [POST21] representations. The representation of the result of Boolean operations are based on a **linear order** of S. Let r: $S \rightarrow N$ be an enumeration consistent with the linear order, where N is the set of natural numbers. In particular:

i) Product (AND) : $s \wedge s' \equiv r^{-1} \min(r(s), r(s'))$
ii) Sum (OR) : $s \vee s' \equiv r^{-1} \max(r(s), r(s'))$
iii) Complement (NOT) : $\bar{s} \equiv r^{-1} (p - 1 - r(s))$.

Note that the order does not affect the semantics of the representation of a switching function; however the order affects heavily the size of the representation. For example, canonical representations, such as *sum-of-products* or *product-of-sums* depend on the linear order of S : in particular the minimal representations of a Boolean function and of its complement as *sum of products* have different sizes, and algorithms have been developed to exploit this fact [SASA82].

Representations of symbolic functions depend on the definitions of the operations among words. Unfortunately, no order relation is meaningful ***a priori*** among the elements of a symbolic description. For this reason, operations on symbolic representations are related to order relations among words. In the following presentation, single-output functions are considered first, i.e. $m = 1$, to simplify the notations. The extension to multiple-output functions is then shown.

Symbolic functions are represented here as *sums of products of literals*. Let $\sigma \subseteq S$ be a set of words. For any variable $s \in S$, the **symbolic literal** function is defined as follows:

$$l(s, \sigma) \equiv \begin{cases} \textbf{TRUE} \text{ if } s \in \sigma \\ \textbf{FALSE} \text{ else} \end{cases}$$

The set σ is the **pattern** of the literal.

By using a *sum of product of literals* representation, only the order in S^O affects the representation, because the literal function maps words into Boolean variables (TRUE,FALSE) independently of the order in S^I. Note that if a linear order relation is specified among the words of the range, symbolic functions representations are equivalent to multiple-valued logic functions representations [RINE77]. In particular, a multiple-valued representation can be obtained from a symbolic representation by interchanging each symbol s with $r(s)$, where $r(\bullet)$ is the appropriate enumeration.

Since an order in S^O is not necessarily given, the definitions of the representation of a symbolic function are compatible with a set, possibly empty, of **partial order relations** among the elements of the range. Let $R = \{(s, s'); s, s' \in S^O\}$ be a partial order relation on S^O. We say that s **covers** s' if either $s = s'$ or $s' = \phi$ or $(s, s') \in TR$, where TR is the transitive closure of R. The **symbolic sum** of two words s and s' is well-defined only if a covering relation exists among the elements involved. In particular:

$$s \vee s' = \begin{matrix} s & \text{if } s \text{ covers } s' \\ s' & \text{if } s' \text{ covers } s \end{matrix}$$

Else the symbolic sum is ambiguous or ill-defined.

A **symbolic product-term** of literals is the $n+1$-tuple $(\sigma_1, \dots, \sigma_n, \tau)$, where $\sigma_i \subseteq S_i^I$, $i = 1, 2, \dots, n$; $\tau \in S^O$. A **symbolic product** $p(s^I, \tau)$ of literals $l_i(s_i^I, \sigma_i)$, $i = 1, 2, \dots, n$ is a function:

$$p(s^I, \tau) = \begin{matrix} \tau \text{ if } l_i(s_i^I, \sigma_i) = \textbf{TRUE}, \ \forall i = 1, 2, \dots, n \\ \phi \text{ else} \end{matrix}$$

Two products p_1, p_2 intersect if $\exists s^I \in S^I$ such that $p_1(s^I, \tau_1) \neq \phi$ and $p_2(s^I, \tau_2) \neq \phi$. Two products are output-disjoint if $p_1(s^I, \tau_1)$ intersects $p_2(s^I, \tau_2)$ implies $\tau_1 = \tau_2$.

A symbolic function can be represented in a *sum of product of literals* form, if $\forall s^I \in S^I$ for which the function is specified, the operation of symbolic sum among products is well-defined. In particular, such a representation always exists in the following two cases: i) for any linear order in S^O; ii) if the representation is a sum of pair-wise output-disjoint products.

Sum of product of literals representations are conveniently represented in tabular forms, as a stack of product-terms. A **symbolic implicant** is a symbolic product $p(s^I, \tau)$ such that $\forall s^I \in S^I$ for which the symbolic function is specified, $f(s^I)$ covers $p(s^I, \tau)$. A **symbolic cover** of a symbolic function is a set of implicants whose sum is $f(s^I)$, $\forall s^I \in S^I$ for which the symbolic function is specified. A minimum cover is a cover of minimum cardinality, i.e. minimum number of product-terms.

Example: The following table is a symbolic cover of a control function with $n = 2$ inputs and $m = 1$ outputs.

INDEX	*AND OR ADD JMP*	*CNTA*
DIR	*AND OR*	*CNTB*
IND	*AND*	*CNTB*
IND	*OR JMP*	*CNTD*
DIR IND	*ADD*	*CNTC*
DIR	*JMP*	*CNTC*

Here, $S_1^I = \{DIR, IND, INDEX\}$; $S_2^I = \{AND, OR, ADD, JMP\}$; $S^O = \{CNTA, CNTB, CNTC, CNTD\}$. This representation is compatible with an empty set R of partial order relations on S^O, because it represents a sum of output-disjoint product-terms.

3. AN ALGORITHM FOR SYMBOLIC MINIMIZATION

Finding a minimum symbolic cover is a complex task and is at least as difficult as determining a minimum cover of a multiple-valued function. Both tasks involve the solution of a covering problem, which is a NP-complete problem. Therefore heuristic algorithms are used to determine a minimal (local minimum) solution. However, recent progress in heuristic logic minimization has led to techniques which very often yield minimum solutions in the binary [BRAY84b] and multiple-valued [RUDE85a] case.

We assume the reader is familiar with the operations involved in heuristic logic minimization [BRAY84b] [HONG74]. A minimal cover is obtained from an initial cover by iterative improvement. Operations like product-term merging, product-term expansion and removal of covered terms, reduction and reshaping of product-terms and detection of irredundant covering set of terms are used to determine a minimal cover. These operations can be applied to symbolic product-terms, provided that the logic sum of the terms involved, if any, is well-defined.

The idea behind symbolic minimization is to generate order relations among the elements of S^O during the minimization process. The symbolic minimizer detects partial order relations that are necessary to define sums of product-terms which would decrease the symbolic cover cardinality. As a result the order relations are determined ***a posteriori*** by the minimizer. The output of the minimizer is a minimal cover and a partial order on S^O [1].

The order relations are represented by a directed graph $G(V, A)$, where the node set V is in one-to-one correspondence with S^O. The edge set A is initialized empty and is constructed during the minimization process. An edge between two vertices defines a order relation between the corresponding elements of S^O. Therefore the sum of two distinct elements of S^O is well-defined if there is a directed path between the corresponding vertices.

Symbolic minimization is achieved by an iterative improvement of the initial cover. Let $S^O = \{s_i^O; i = 1, 2, \dots, q\}$ (the labelling is arbitrary). Let ON_i, $i = 1, 2, \dots, q$ be the subset of the initial symbolic cover consisting of the product-terms with $\tau = s_i^O$. The initial cover is a set of output-disjoint product-terms, because no order in S^O is specified before the minimization. Therefore, product-terms in ON_i do not intersect product-terms in ON_j, if $i \neq j$. Each set ON_i is a symbolic single-output single-valued function. A minimal representation of ON_i, denoted by M_i, can be obtained by using a multiple-valued-input, binary-valued output minimization technique.

The leading idea behind symbolic minimization is that the product-terms in the minimized covers M_i, $i = 1, 2, \dots, q$ are not constrained to be output-disjoint, by introducing appropriate order relations among the elements of S^O. For example, suppose that $(s_j^O, s_i^O) \in R$. Then, while minimizing ON_i to M_i, ON_j is considered as a subset of the don't care set DC_i, because the sum of the product-terms of ON_i and ON_j cannot yield s_i^O.

To minimize ON_i, an explicit representation of the offset (or the don't care set) is needed. The offset corresponding to ON_i, denoted by OFF_i, is the subset of the domain which must have empty intersection with M_i. In particular, it is the subset of S^I that is mapped by the function f to a value different than s_i^O and covered by s_i^O. If $G(V, A)$ represents the partial order, then the off-sets can be defined as: $OFF_i = \cup_J ON_j$; $J = \{ j \text{ s.t.} \exists \text{ a path from } v_i \text{ to } v_j \text{ in } G(V, A)\}$. The complement of $ON_i \cup OFF_i$ is the don't care-set DC_i of the function, which is used in the minimization process.

The symbolic minimization algorithm has a main iteration loop. At iteration i of the loop, M_i is is obtained by minimizing (ON_i, OFF_i), using a routine that performs multiple-valued-input binary-valued output minimization. The corresponding don't care-set DC_i includes by construction all the sets ON_j for which no path exists in $G(V,A)$ from v_i to v_j. As a result, minimization may be very efficient in reducing the cardinality of M_i because of the particularly advantageous don't care set. If, during the minimization, M_i intersects ON_j, the relation (s_j^O, s_i^O) is recorded, by adding (v_j, v_i) to the edge set of the graph. This has two consequences. First the relation (s_j^O, s_i^O) implies that s_j^O covers s_i^O. Second it prevents that, at a further iteration of the algorithm, a minimized cover M_j intersects ON_i, which would induce a contrasting relation (s_i^O, s_j^O). At step j of the loop, $M_j \cap ON_i = \phi$, because $ON_i \subseteq OFF_j$. Therefore $G(V,A)$ is acyclic by construction. The structure of the algorithm is as follows:

```
SYMBOLIC MINIMIZATION LOOP
Data $ON_i$, $i = 1, 2, \dots, q$
Data $G(V, A)$
$A = \phi$
while ( $q$ or fewer iterations have been done ) {

    $i$ = select__function
    $OFF_i = \cup_J \overline{ON_j}$;     $J = \{j \mid \exists$ a path from $v_i$ to $v_j$ in $G(V, A)\}$
    $M_i$ = minimize ($ON_i$, $OFF_i$)
    $A = A \cup \{(v_j, v_i)$ s.t. $M_i \cap ON_j \neq \phi\}$
}
```

Procedure **select__function** sorts the sets ON_i according to a heuristic criterion. The minimization of multiple-valued-input binary-valued-output functions is done using a modified version of program ***ESPRESSO-II*** [BRAY84b]. The don't-care sets DC_i are computed by the minimizer, by complementing $ON_i \cup OFF_i$. The overall minimized cover is represented by the concatenation of $M_i, i = 1, 2, \dots, q$. The final graph $G(V, A)$ represents the partial order on S^O. The sets ON_i, $i = 1, 2, \dots, q$ are obtained by partitioning the initial cover. As a pre-processing step before the symbolic minimization loop, the sets ON_i, $i = 1, 2, \dots, q$ are minimized with $OFF_i = \cup_J ON_j$; $J = \{j \neq i\}$. This helps in reducing the number of product-terms, without introducing any order relation.

[1] In previous approaches to solve problem P4 [DEMI84c] [DEMI85a], exploiting the order in S^O was not considered and symbolic minimization was approximated by considering only sums of output-disjoint products. Sub-optimal results were obtained due to this restriction.

Example: The following is a minimal symbolic cover.

INDEX	*AND OR ADD JMP*	*CNTA*
DIR IND	*AND OR*	*CNTB*
DIR IND	*ADD JMP*	*CNTC*
IND	*JMP OR*	*CNTD*

Here, $R = \{(CNTD, CNTB);(CNTD, CNTC)\}$. Note that the fourth product-term has an intersection with the second and third one. The partial order relation allows to reduce the cover cardinality by two.

Symbolic minimization can be extended to multiple-output functions ($m > 1$) by considering covers with multiple-output product-terms and extending the definitions and operations appropriately. In particular, m partial orders on the sets $S_i^O, i = 1, 2, \dots, m$ are recorded in corresponding graphs $G_i(V_i, A_i)$, $i = 1, 2, \dots, m$. This technique can be applied to mixed symbolic-Boolean representations. In this case, some of the sets S_i^I, $i = 1, 2, \dots, n$, S_i^O $i = 1, 2, \dots, m$ are the ordered set $\{0, 1\}$. For finite-state machine minimization (problem P4), it is relevant to minimize multiple-output functions having only one symbolic output (the state information), all the others being binary. In this case, the symbolic minimization algorithm described above with $m = 1$ can be used with a slight modification: the offset of the binary-valued-output functions are appended to OFF_i at each iteration i of the loop.

The symbolic minimization algorithm has been implemented in an APL computer program, called ***CAPPUCCINO*** because it is built on top of minimizer ***ESPRESSO-II***.

4. ENCODING PROBLEMS

Symbolic minimization is used as a first step in solving problems P1-P4. If the objective is a multiple-valued circuit implementation, then the (minimal) symbolic representation can be mapped into a multiple-valued representation with the same cardinality by interchanging the symbols s with $r(s)$, where $r(\bullet)$ is the appropriate enumeration. If $s \in S^O$, then $r(\bullet)$ is an enumeration consistent with R.

However, usually, the objective of problems P1-P4 is to determine a binary-valued circuit implementation. The minimal symbolic representation defines the constraints of two encoding problems, whose solutions are binary-valued encodings that allow to implement the switching function with as many binary-valued product-terms as the minimal symbolic cover.

Encoding problem E1. Given the set of literal patterns corresponding to each input variable in the (minimal) symbolic representation, find an encoding of the words which that variable can take, such that each literal pattern can be identified by a Boolean subspace containing all and only the encoding of the words in the pattern [DEMI85a].

Encoding problem E2. Given the partial order on the set of words that each output variable can take, find an encoding of the words, such that $\forall s, s'; (s, s') \in R$ the encoding of s covers bit-wise the encoding of s'.

The solution of problem P1 requires the solution of encoding problem E1 after symbolic minimization. In this case only the inputs of the function f are represented by symbols, while the outputs are represented by Boolean variables in the original representation. Since the order in S^O is given, symbolic minimization is equivalent to multiple-valued-input binary-valued-output logic minimization [RUDE85a]. The author proved in [DEMI85a] that problem E1 admits always a solution; an interesting sub-problem is to find minimal-length encodings. A heuristic algorithm was presented in [DEMI84c] and [DEMI85a].

Problem P2 is the complementary problem. The solution of problem P2 requires the solution of encoding problem E2 after symbolic minimization. In this case only the outputs of the function f are represented by symbols, while the inputs are represented by Boolean variables in the original representation. Problem E2 admits always a solution; also in this case an interesting sub-problem is to find minimal-length encodings. A heuristic algorithm has been developed to determine minimal-length solutions [DEMI85b].

Problem P3 consists of solving both the encoding problems E1 and E2 independently, after symbolic minimization.

Example: Consider the minimal symbolic cover of the previous example. The following encodings satisfy both encoding problems:

$INDEX = 00$	$AND = 00$	$CNTA = 00$
$DIR = 01$	$OR = 01$	$CNTB = 01$
$IND = 11$	$ADD = 10$	$CNTC = 10$
	$JMP = 11$	$CNTD = 11$

The corresponding Boolean cover is:

```
00  **  00
*1  0*  01
*1  1*  10
11  *1  11
```

where a don't care in a binary input variable is represented by *. Note that the first implicant can be deleted, because 00 is the default output.

Problem P4 is similar to problem P3 but has the additional equality constraint of some encodings. Unfortunately, an encoding that satisfies simultaneously the constraints set by problems E1 and E2 may not exist. Therefore it is important to set the conditions of compatibility of the constraints related to problems E1 and E2. Let S be a set of words. Let P be the set of patterns defining problem E1. Let R be the partial order on S representing problem E2.

Theorem: A necessary and sufficient condition for the existence of a solution to both problems E1 and E2 is that $\forall s, s', s'' \in S$ s.t. $\{(s, s'),(s', s'')\} \subseteq R$, there exists no pattern $\sigma \in P$ s.t. $s, s'' \in \sigma$ and $s' \notin \sigma$.■

The proof is reported in [DEMI85b]. A solution to problem P4 requires then the detection of the mutually incompatible constraints. An encoding algorithm for solving problems E1 and E2 by a minimal length encoding is reported in [DEMI85b].

5. CONCLUDING REMARKS

Symbolic minimization and constrained encoding allow to achieve optimal logic design of combinational circuits and sequential circuits. Optimality is measured in terms of the number of product-terms of two-level *sum of product of literals* representations. This technique allows to compute minimal representation first, and then to determine a compatible encoding of the input and output variables.

An approximation to symbolic minimization was used in the past to determine optimal state assignments for finite-state machines [DEMI84c]. Though good results were achieved, it was not possible to relate the optimality of the state assignment to the encoding of the output variables of the combinational component of the finite state machine. Symbolic minimization bridges this gap and the results obtained by ***CAPPUCCINO*** have shown that minimal symbolic cover cardinalities are smaller by about 25% compared to those obtained by the previous approach.

Symbolic minimization is related to two encoding problems. Algorithms for obtaining compatible solutions have been developed and will be disclosed in the near future.

6. ACKNOWLEDGMENTS

The author wish to thank Dr. Robert Brayton for many helpful discussions. This work has been sponsored by NSF under contract No. ECS-8121446.

7. REFERENCES

[BRAY84b] R.Brayton, G.D.Hachtel, C.McMullen and A.L.Sangiovanni- Vincentelli, "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.

[DEMI84c] G.De Micheli, R.Brayton and A.L.Sangiovanni Vincentelli, "KISS: a Program for Optimal State Assignment of Finite State Machines", Proc. ICCAD Santa Clara, nov 1984.

[DEMI85a] G.De Micheli, R.Brayton and A.L.Sangiovanni Vincentelli, "Optimal State Assignment for Finite State Machines", IEEE Transactions on CAD/ICAS, Vol CAD-4, No. 3, pp.269-284, July 1985.

[DEMI85b] G.De Micheli, "Optimal Synthesis of Combinational and Sequential Circuits using Two-level Logic", (in preparation)

[HONG74] S.J.Hong,R.G.Cain and D.L.Ostapko, "MINI: a Heuristic Approach for Logic Minimization", IBM Jour. of Res. and Dev., vol. 18, pp. 443-458, sep. 1974.

[NICH65] A.Nichols and A.Bernstein "State Assignemnt in Combinational Networks" IEEE Trans on Elect. Comp., vol. EC-14, pp. 343-349, Jun. 1965.

[POST21] E.L.Post, "Introduction to a General Theory of Elementary Propositions" Amer. J. Math., vol. 43, pp. 163-185, 1921.

[RINE77] D.Rine "Computer Science and Multiple-Valued Logic", North Holland, 1977.

[RUDE85a] R. Rudell and A. Sangiovanni-Vincentelli, "ESPRESSO-MV: Algorithms for Multivlued Logic Minimization" Proc. Custom Int. Circ. Conf., Portland, Oregon, May 85

[SASA83] T.Sasao "Input Variable Assignment and Output Phase Assignment of PLAs", IBM Research Report, No. 1003, June 1983.