An Interior Point Approach to Boolean Vector Function Synthesis

A.P. Kamath [1] N.K. Karmarkar[2] K.G. Ramakrishnan[2] M.G.C. Resende [2]

Abstract— **The Boolean vector function synthesis problem can be stated as follows: Given a truth table with n input variables and m output variables, synthesize a Boolean vector function that describes the table. In this paper we describe a new formulation of the Boolean vector function synthesis problem as a particular type of Satisfiability Problem. The Satisfiability Problem is translated into an integer programming feasibility problem, that is solved with an interior point algorithm for integer programming. Preliminary computational results are presented.**

Introduction

The *Boolean Vector Function Synthesis Problem* has applications in logic, artificial intelligence, machine learning, and digital integrated circuit design. In this paper, we describe a Satisfiability Problem formulation of the Boolean Vector Function Synthesis Problem. This formulation can be approached with a wide range of algorithms. In this paper, preliminary computational results are presented using an interior point algorithm for integer programming to solve instances of the problem.

Consider the Boolean function $\mathcal{F} : \{0,1\}^n \rightarrow \{0,1\}$. An element of the domain of $\mathcal{F}$ is called a *minterm* of $\mathcal{F}$. The set of minterms for which $\mathcal{F}$ evaluates to 1 (0) is called the ON-set (OFF-set). An *incompletely specified* Boolean function is one for which the number of minterms in the ON- and OFF-sets is less than 2^n.

An *algebraic expression* for an incompletely specified Boolean function $\mathcal{F}$ is a Boolean expression (written with Boolean sums ($\vee$) and products ($\wedge$)) that evaluates to 1 (0) for all minterms in the ON-set (OFF-set) and evaluates to either 1 or 0 for all other minterms. An algebraic expression for $\mathcal{F}$ can always be written in sum-of-products (disjunctive normal) form. Each product term in the algebraic sum-of-products expression is called a *disjunct.* Let P_i^0 and P_i^1, respectively, denote the index set of 0 and 1 values of the i-th element of the ON-set of $\mathcal{F}$. The *canonical expansion*,

$$y = \bigvee_{i \in \text{ON-set}} \left\{ \bigwedge_{j \in P_i^1} x_j \wedge \bigwedge_{j \in P_i^0} \overline{x}_j \right\}$$

is a sum-of-products algebraic expression for $\mathcal{F}$. The major drawback of the canonical expansion is that it has |ON-set| disjuncts, each having n variables.

Given an ON-set and OFF-set of minterms, the *Boolean Function Synthesis Problem* is to find an algebraic sum-of-products expression for $\mathcal{F}$ having a specified number of disjuncts. The corresponding decision problem is NP-complete [1] [2].

The classical approach to tackle the Boolean Function Minimization Problem (where one wishes to minimize the number of disjuncts in the sum-of-products form) was developed by Quine [3] [4] and McCluskey [5]. Because exact versions of the Quine-McCLuskey method fail to handle large instances, many heuristic approaches have been developed. They include MINI [6], PRESTO [7], and ESPRESSO-MV [1]. ESPRESSO-MV is widely used in the circuit design industry.

In [8], Kamath, Karmarkar, Ramakrishnan and Resende considered an interior point mathematical programming approach to the Boolean Function Synthesis Problem. In this paper, we extend that approach to the Boolean Vector Function Synthesis Problem. The synthesis problem is formulated as a type of Satisfiability Problem that can be described as an integer programming problem, using a standard transformation. We apply the interior point algorithm for integer programming described in [9] and [10] to synthesize Boolean vector functions. In particular, we use an implementation derived from the one described in [11], that is suited for finding a satisfiable truth assignment in instances of the Satisfiability Problem. The interior point algorithm is used to attempt to find a feasible ± 1 integer solution w to the following integer program:

$$\begin{aligned} B^\top w &\leq b \\ w_j &= \pm 1, \; j = 1, \ldots, n, \end{aligned} \tag{1}$$

where $B^\top \in \Re^{m \times n}$, $b \in \Re^m$, and $w \in \Re^n$. In [11] it is shown how the Satisfiability Problem can be formulated as an integer programming problem of this form. Let $A = \left[B \vdots I \vdots - I \right]$, where I is an $n \times n$ identity matrix and let $c^\top = (b^\top, 1, \ldots, 1)$.

Starting with an interior point solution

$$w^0 \in \left\{ w \in \Re^n | A^\top w < c \right\},$$

a trust region method, similar to the one described in Moré and Sorensen [12], is applied to the nonconvex optimization problem

$$\text{minimize} \;\; \log(n - w^\top w) - \frac{1}{m} \sum_{k=1}^{m} \log(c_k - a_k^\top w).$$

At each iteration, a quadratic approximation of the potential function

$$\varphi(w) = \log(n - w^\top w) - \frac{1}{m} \sum_{k=1}^{m} \log(c_k - a_k^\top w)$$

is optimized over an ellipsoid inscribed in the polytope defined by

$$\left\{ w \in \Re^n | A^\top w \leq c \right\},$$

and centered at the current iterate, to produce a descent direction. The new iterate is determined by moving in that direction by a fixed step length, such that the new point is in the interior of the ellipsoid. A rounding heuristic is applied to the fractional solution and feasibility of the rounded solution is tested. A description of the rounding heuristic and initial interior point solution used in this study can be found in [11].

[1] Stanford University, Stanford CA 94305 USA
[2] AT&T Bell Laboratories, Murray Hill NJ 07974 USA

Satisfiability Problem Formulation

A formulation of the Boolean Vector Function Synthesis Problem as a Satisfiability Problem is given next. In the Boolean Vector Function Synthesis Problem, we consider an incompletely specified Boolean function $\mathcal{F} : \{0,1\}^n \rightarrow \{0,1\}^m$, and wish to find m algebraic sum-of-products expressions (one for each output variable), restricting disjuncts in each expression to come from a set of at most k distinct disjuncts, where $k > 0$ is given. The input and output vectors have components $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$, respectively. In our formulation, two types of decisions are to be made. First, what are each of the k disjuncts. Second, for each output component, which of the k disjuncts appear in the algebraic sum-of-products expression for that output.

We assume that as input we are given a partial truth table of the Boolean function $\mathcal{F}$ that we wish to synthesize. The truth table is a set of A *i/o-pairs*, where each i/o-pair is made up of an *in-tuple* and an *out-tuple*. The in-tuple is a Boolean n-vector, representing an element of the domain of $\mathcal{F}$, and the out-tuple is a Boolean m-vector, representing the corresponding output of $\mathcal{F}$ for that in-tuple. Let the a-th in-tuple be x^a and the i-th input component of the a-th in-tuple be x_i^a and let the a-th out-tuple be y^a and the q-th output component of the a-th out-tuple be y_q^a. A function is said to be incompletely specified if $A < 2^n$.

In our formulation, we use decision variables s_{ji} and s'_{ji} to determine which literal (x_i or $\overline{x}_i$, if any) is part of the j-th disjunctive term,

$$s_{ji} = \begin{cases} 0 & \text{if } x_i \text{ is in the } j\text{-th disjunctive term} \\ 1 & \text{otherwise} \end{cases}$$

$$s'_{ji} = \begin{cases} 0 & \text{if } \overline{x}_i \text{ is in the } j\text{-th disjunctive term} \\ 1 & \text{otherwise,} \end{cases}$$

and w_{jl} to determine which disjunct is part of the l-th output component y_l,

$$w_{jl} = \begin{cases} 1 & \text{if the } j\text{-th disjunct appears in output } y_l \\ 0 & \text{otherwise.} \end{cases}$$

For each i/o-pair $a = 1, \ldots, A$, input variable x_i, $i = 1, \ldots, n$, and disjunct $j = 1, \ldots, k$, define for convenience of notation,

$$\sigma_{ji}^a = \begin{cases} s'_{ji} & \text{if } x_i^a = 1 \\ s_{ji} & \text{if } x_i^a = 0. \end{cases}$$

For input x^a, the truth value of the j-th disjunct can be represented by $\wedge_{i=1}^n \sigma_{ji}^a$.

Auxiliary variable z_{jq}^a is defined to be true if the j-th disjunct is selected to be part of the q-th output and the truth value of the j-th disjunct is true. Therefore,

$$z_{jq}^a = w_{jq} \wedge (\wedge_{k=1}^n \sigma_{jk}^a).$$

Hence, if the j-th product term is part of the q-th output component, i.e. if $w_{jq} = 1$, then z_{jq}^a will take on the value of the j-th product term given the a-th in-tuple. Otherwise, if it is not part of the q-th output component, then $z_{jq}^a = 0$ and hence will not affect the q-th output. Having defined z_{jq}^a, the q-th output y_q^a is the Boolean sum of the z_{jq}^a variables, i.e.

$$y_q^a = \vee_{j=1}^k z_{jq}^a.$$

The clauses in the Satisfiability Problem formulation impose conditions which the variables must satisfy in order to determine a Boolean vector function that meets the specification of the truth table.

We now state the clauses of Type-1. Since x_i and $\overline{x}_i$ cannot simultaneously be part of the j-th disjunctive term, we have that

$$s_{ji} \vee s'_{ji}$$

must be satisfied for all $i = 1, \ldots, n$ and $j = 1, \ldots, k$.

We next state clauses of Type-2. Consider the case for which $y_q^a = 0$. Then, it must be the case that $z_{jq}^a = 0$, for all $j = 1, \ldots, k$, and consequently,

$$w_{jq} \wedge (\wedge_{k=1}^n \sigma_{jk}^a) = 0,$$

for all $j = 1, \ldots, k$. These requirements can be stated as clauses

$$\bar{w}_{jq} \vee (\vee_{k=1}^n \bar{\sigma}_{jk}^a),$$

for all $j = 1, \ldots, k$ and $\{q, a\}$ such that $y_q^a = 0$.

Now consider the other case, i.e. where $y_q^a = 1$. To satisfy $y_q^a = 1$, then at least one product term in this output must evaluate to true, i.e.

$$\vee_{j=1}^k z_{jq}^a,$$

for $\{q, a\}$ such that $y_q^a = 1$. These are the Type-3 clauses.

To complete the formulation, we must relate variables z_{jq}^a with w_{jq} and σ_{ji}^a for $\{q, a\}$ such that $y_q^a = 1$ and $j = 1, \ldots, k$. Two conditions must be satisfied.

First, if $z_{jq}^a = 1$, then $w_{jq} = 1$ and $\sigma_{ji}^a = 1$, for all $i = 1, \ldots, n$, i.e.

$$z_{jq}^a \rightarrow w_{jq} \wedge (\wedge_{i=1}^n \sigma_{ji}^a).$$

Clauses of Type-4a/b force this condition. Type-4a clauses are

$$\bar{z}_{jq}^a \vee w_{jq}.$$

Type-4b clauses are

$$\bar{z}_{jq}^a \vee \sigma_{ji}^a,$$

for all $i = 1, \ldots, n$.

Second, if $z_{jq}^a = 0$, then either $w_{jq} = 0$ or for at least one index $i = 1, \ldots, n$, $\sigma_{ji}^a = 0$, i.e.

$$\bar{z}_{jq}^a \rightarrow \bar{w}_{jq} \vee (\vee_{i=1}^n \bar{\sigma}_{ji}^a).$$

Type-5 clauses guarantee this,

$$z_{jq}^a \vee \bar{w}_{jq} \vee (\vee_{i=1}^n \bar{\sigma}_{ji}^a).$$

Note that for our objective (to synthesize a Boolean function that realizes the specification given by the truth table), only clauses of Types-1, 2, 3, 4a and 4b are needed. An assignment that satisfies these clauses produces a correct function, but does not necessarily satisfy clauses of Type-5. Type-5 clauses are only needed to produce correct values for the auxiliary variables z.

To summarize, for the Boolean Vector Function Synthesis Problem, we wish to find truth assignments for s_{ji} ($j = 1, \ldots, k$ and $i = 1, \ldots, n$), s'_{ji} ($j = 1, \ldots, k$ and $i = 1, \ldots, n$), w_{jl} ($j = 1, \ldots, k$ and $l = 1, \ldots, m$), and z_{jl}^a ($a = 1, \ldots, A$, $j = 1, \ldots, k$ and $l = 1, \ldots, m$) such that the following clauses are satisfied:

- Type-1: For $i = 1, \ldots, n$ and $j = 1, \ldots, k$,

$$s_{ji} \vee s'_{ji}.$$

Table 1: Truth table without DON'T CARE

x_1	x_2	x_3	y_1	y_2
1	0	1	1	0
0	0	1	0	0
1	0	1	0	0
1	1	1	0	1
0	1	1	1	1

Table 2: Truth table with DON'T CARE

x_1	x_2	x_3	y_1	y_2
1	0	1	1	0
DC	0	1	0	0
1	1	1	0	1
0	1	1	1	1

- Type-2: For $j = 1, \ldots, k$ and $\{q, a\}$ such that $y_q^a = 0$,

$$\bar{w}_{jq} \vee (\vee_{k=1}^{n} \bar{\sigma}_{jk}^a).$$

- Type-3: For $\{q, a\}$ such that $y_q^a = 1$,

$$\vee_{j=1}^{k} z_{jq}^a.$$

- Type-4a: For $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$,

$$\bar{z}_{jq}^a \vee w_{jq}.$$

- Type-4b: For $i = 1, \ldots, n$, $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$,

$$\bar{z}_{jq}^a \vee \sigma_{ji}^a.$$

- Type-5: For $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$,

$$z_{jq}^a \vee \bar{w}_{jq} \vee (\vee_{i=1}^{n} \bar{\sigma}_{ji}^a).$$

The Satisfiability problem has $k(2n + m(1 + A))$ variables and $k(n + Y^0) + Y^1(1 + k(2 + n))$ clauses, where Y^0 is the number of pairs $\{q, a\}$ for which $y_q^a = 0$ and Y^1 is the number of pairs $\{q, a\}$ for which $y_q^a = 1$.

We next extend the Satisfiability model to allow for DON'T CARE (DC) entries in the truth table. DCs in the in-tuple can be used to make a more compact truth table representation of the Boolean function. To illustrate this, consider the example specification given in Table 1, where the second and third i/o-pairs differ exactly in one input component. Table 1 can have a more compact representation (Table 2) with the two i/o-pairs replaced by a single i/o-pair with a DC in the component under consideration. DCs in the out-tuple are used for modeling purposes, where, given some in-tuple, we "don't care" what a given component of the output will evaluate to.

Since DCs affect only some in-tuple components x_i^a of the truth table of $\mathcal{F}$, only clauses that depend on these entries are affected, i.e. clauses in which the variable σ_{ji}^a appears. Those are clauses of Types 2, 4b and 5. Let us consider each clause type individually. Before doing so, let us define the following two sets that form a partition of the index set $\{1, \ldots, n\}$:

$$\begin{aligned} I^a &= \{i \mid x_i^a = \text{DC}\}, \\ J^a &= \{i \mid x_i^a = 0 \text{ or } 1\}. \end{aligned}$$

For Type-2 clauses, the requirement $y_q^a = 0$ implies that for all $j = 1, \ldots, k$ we have $z_{jq}^a = 0$. Therefore

$$w_{jq} \wedge (\wedge_{k \in J^a} \sigma_{jk}^a) \wedge (\wedge_{k \in I^a} \sigma_{jk}^a) = 0, \tag{2}$$

for all possible values of σ_{ji}^a. Let $n^0 \in I^a$. Condition (2) can be written as

$$w_{jq} \wedge (\wedge_{k \in J^a} \sigma_{jk}^a) \wedge (\wedge_{k \in I^a \setminus n^0} \sigma_{jk}^a) \wedge \sigma_{jn^0}^a = 0. \tag{3}$$

Since $x_{n^0}^a = \text{DC}$, $\sigma_{jn^0}^a$ can be either s_{jn^0} or s'_{jn^0}, (3) can be further be expressed as

$$\begin{aligned} w_{jq} \wedge (\wedge_{k \in J^a} \sigma_{jk}^a) \quad &\wedge \quad (\wedge_{k \in I^a \setminus n^0} \sigma_{jk}^a) \wedge s_{jn^0} = 0, &(4) \\ w_{jq} \wedge (\wedge_{k \in J^a} \sigma_{jk}^a) \quad &\wedge \quad (\wedge_{k \in I^a \setminus n^0} \sigma_{jk}^a) \wedge s'_{jn^0} = 0. &(5) \end{aligned}$$

Since, by clauses of Type-1, $s_{jn^0} \vee s'_{jn^0} = 1$, for $j = 1, \ldots, k$, conditions (4–5) can be reduced to

$$w_{jq} \wedge (\wedge_{k \in J^a} \sigma_{jk}^a) \wedge (\wedge_{k \in I^a \setminus n^0} \sigma_{jk}^a) = 0.$$

Repeating this elimination process of σ_{jk}^a for all $k \in I^a$ causes clauses of Type-2 to reduce to

$$\bar{w}_{jq} \vee (\vee_{k \in J^a} \bar{\sigma}_{jk}^a),$$

for $j = 1, \ldots, k$ and $\{q, a\}$ such that $y_q^a = 0$. This reduces the number of literals per Type-2 clause by $|I^a|$.

Now consider clauses of Type-4b. Since for $i \in J^a$ the component of the in-tuple takes either value 0 or 1, the clauses corresponding to those indices remain unaltered, i.e. we have

$$\bar{z}_{jq}^a \vee \sigma_{ji}^a,$$

for $i \in J^a$, $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$. For $i \in I^a$ we must consider all possibilities and therefore expand out clauses of Type-4b to become

$$\bar{z}_{jq}^a \vee s_{ji},$$

$$\bar{z}_{jq}^a \vee s'_{ji},$$

for $i \in I^a$, $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$. This causes an increase of $kY^1 \times |I^a|$ clauses per Type-4b clause.

Finally, consider clauses of Type-5. Using an argument similar to the one used for clauses of Type-2, clauses of Type-5 become

$$z_{jq}^a \vee \bar{w}_{jq} \vee (\vee_{i \in J^a} \bar{\sigma}_{ji}^a),$$

for $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$. This reduces the number of literals per Type-5 clause by $|I^a|$.

DCs in the output are handled automatically, since the clauses make no reference to out-tuple components for which $y_q^a = \text{DC}$, resulting in fewer number of clauses.

To summarize, in the synthesis problem where we allow DON'T CARE entries in the truth table, we wish to find truth assignments for s_{ji} ($j = 1, \ldots, k$ and $i = 1, \ldots, n$), s'_{ji} ($j = 1, \ldots, k$ and $i = 1, \ldots, n$), w_{jl} ($j = 1, \ldots, k$ and $l = 1, \ldots, m$), and z_{jl}^a ($a = 1, \ldots, A$, $j = 1, \ldots, k$ and $l = 1, \ldots, m$) such that clauses of Types-1, 3 and 4a given for the formulation without DCs are satisfied, together with the following clauses:

- Type-2: For $j = 1, \ldots, k$ and $\{q, a\}$ such that $y_q^a = 0$,

$$\bar{w}_{jq} \vee (\vee_{k \in J^a} \bar{\sigma}_{jk}^a).$$

- Type-4b[1]: For $i \in J^a$, $j = 1, \ldots, k$, and $\{q, a\}$ such that $y_q^a = 1$,

$$\bar{z}_{jq}^a \vee \sigma_{ji}^a.$$

Table 3: A small example

x_1	x_2	x_3	y_1	y_2
1	0	1	1	0
0	0	1	0	0
1	0	0	1	1
1	1	1	0	0
0	1	1	1	1

Table 4: σ values for small example

a	i	σ^a_{ji}	i	σ^a_{ji}	i	σ^a_{ji}
1	1	s'_{j1}	2	s_{j2}	3	s'_{j3}
2	1	s_{j1}	2	s_{j2}	3	s'_{j3}
3	1	s'_{j1}	2	s_{j2}	3	s_{j3}
4	1	s'_{j1}	2	s'_{j2}	3	s'_{j3}
5	1	s_{j1}	2	s'_{j2}	3	s'_{j3}

- Type-4b[2]: For $i \in I^a$, $j = 1, \ldots, k$, and $\{q, a\}$ such that $y^a_q = 1$,

$$\bar{z}^a_{jq} \vee s_{ji},$$
$$\bar{z}^a_{jq} \vee s'_{ji}.$$

- Type-5: For $j = 1, \ldots, k$, and $\{q, a\}$ such that $y^a_q = 1$,

$$z^a_{jq} \vee \bar{w}_{jq} \vee (\vee_{i \in J^a} \bar{\sigma}^a_{ji}).$$

Again, as before, Type-5 clauses can be ignored in practice.

A Small Example

Consider the logic specification given in Table 3. We seek a 3 product term expression for this Boolean vector function. This problem has the following dimensions: $n = 3$, $m = 2$, $k = 3$, $A = 5$ and the σ^a_{ji} values given in Table 4.

The Satisfiability formulation for this problem is given next.

- There are 9 Type-1 clauses:

$$\begin{array}{lll} s_{11} \vee s'_{11} & s_{21} \vee s'_{21} & s_{31} \vee s'_{31} \\ s_{12} \vee s'_{12} & s_{22} \vee s'_{22} & s_{32} \vee s'_{32} \\ s_{13} \vee s'_{13} & s_{23} \vee s'_{23} & s_{33} \vee s'_{33} \end{array}$$

- There are 15 Type-2 clauses:

$$\begin{array}{ll} \bar{w}_{12} \vee \bar{s}'_{11} \vee \bar{s}_{12} \vee \bar{s}'_{13} & \bar{w}_{11} \vee \bar{s}_{11} \vee \bar{s}_{12} \vee \bar{s}'_{13} \\ \bar{w}_{12} \vee \bar{s}_{11} \vee \bar{s}_{12} \vee \bar{s}'_{13} & \bar{w}_{11} \vee \bar{s}'_{11} \vee \bar{s}'_{12} \vee \bar{s}'_{13} \\ \bar{w}_{12} \vee \bar{s}'_{11} \vee \bar{s}'_{12} \vee \bar{s}'_{13} & \bar{w}_{22} \vee \bar{s}'_{21} \vee \bar{s}_{22} \vee \bar{s}'_{23} \\ \bar{w}_{21} \vee \bar{s}_{21} \vee \bar{s}_{22} \vee \bar{s}'_{23} & \bar{w}_{22} \vee \bar{s}_{21} \vee \bar{s}_{22} \vee \bar{s}'_{23} \\ \bar{w}_{21} \vee \bar{s}'_{21} \vee \bar{s}'_{22} \vee \bar{s}'_{23} & \bar{w}_{22} \vee \bar{s}'_{21} \vee \bar{s}'_{22} \vee \bar{s}'_{23} \\ \bar{w}_{32} \vee \bar{s}'_{31} \vee \bar{s}_{32} \vee \bar{s}'_{33} & \bar{w}_{31} \vee \bar{s}_{31} \vee \bar{s}_{32} \vee \bar{s}'_{33} \\ \bar{w}_{32} \vee \bar{s}_{31} \vee \bar{s}_{32} \vee \bar{s}'_{33} & \bar{w}_{31} \vee \bar{s}'_{31} \vee \bar{s}'_{32} \vee \bar{s}'_{33} \\ \bar{w}_{32} \vee \bar{s}'_{31} \vee \bar{s}'_{32} \vee \bar{s}'_{33} & \end{array}$$

- There are 5 Type-3 clauses:

$$\begin{array}{lll} z^1_{11} \vee z^1_{21} \vee z^1_{31} & z^1_{13} \vee z^1_{23} \vee z^1_{33} & z^2_{13} \vee z^2_{23} \vee z^2_{33} \\ z^1_{15} \vee z^1_{25} \vee z^1_{35} & z^2_{15} \vee z^2_{25} \vee z^2_{35} & \end{array}$$

- There are 15 Type-4a clauses:

$$\begin{array}{lll} \bar{z}^1_{11} \vee w_{11} & \bar{z}^3_{11} \vee w_{11} & \bar{z}^3_{12} \vee w_{12} \\ \bar{z}^5_{11} \vee w_{11} & \bar{z}^5_{12} \vee w_{12} & \bar{z}^1_{21} \vee w_{21} \\ \bar{z}^3_{21} \vee w_{21} & \bar{z}^3_{22} \vee w_{22} & \bar{z}^5_{21} \vee w_{21} \\ \bar{z}^5_{22} \vee w_{22} & \bar{z}^1_{31} \vee w_{31} & \bar{z}^3_{31} \vee w_{31} \\ \bar{z}^3_{32} \vee w_{32} & \bar{z}^5_{31} \vee w_{31} & \bar{z}^5_{32} \vee w_{32} \end{array}$$

- There are 45 Type-4b clauses:

$$\begin{array}{lll} \bar{z}^1_{11} \vee s'_{11} & \bar{z}^1_{21} \vee s'_{21} & \bar{z}^1_{31} \vee s'_{31} \\ \bar{z}^3_{11} \vee s'_{11} & \bar{z}^3_{21} \vee s'_{21} & \bar{z}^3_{31} \vee s'_{31} \\ \bar{z}^3_{12} \vee s'_{11} & \bar{z}^3_{22} \vee s'_{21} & \bar{z}^3_{32} \vee s'_{31} \\ \bar{z}^5_{11} \vee s'_{11} & \bar{z}^5_{21} \vee s'_{21} & \bar{z}^5_{31} \vee s'_{31} \\ \bar{z}^5_{12} \vee s'_{11} & \bar{z}^5_{22} \vee s'_{21} & \bar{z}^5_{32} \vee s'_{31} \\ \bar{z}^1_{11} \vee s'_{12} & \bar{z}^1_{21} \vee s'_{22} & \bar{z}^1_{31} \vee s'_{32} \\ \bar{z}^3_{11} \vee s'_{12} & \bar{z}^3_{21} \vee s'_{22} & \bar{z}^3_{31} \vee s'_{32} \\ \bar{z}^3_{12} \vee s'_{12} & \bar{z}^3_{22} \vee s'_{22} & \bar{z}^3_{32} \vee s'_{32} \\ \bar{z}^5_{11} \vee s'_{12} & \bar{z}^5_{21} \vee s'_{22} & \bar{z}^5_{31} \vee s'_{32} \\ \bar{z}^5_{12} \vee s'_{11} & \bar{z}^5_{22} \vee s'_{22} & \bar{z}^5_{32} \vee s'_{31} \\ \bar{z}^1_{11} \vee s'_{13} & \bar{z}^1_{21} \vee s'_{23} & \bar{z}^1_{31} \vee s'_{33} \\ \bar{z}^3_{11} \vee s'_{13} & \bar{z}^3_{21} \vee s'_{23} & \bar{z}^3_{31} \vee s'_{33} \\ \bar{z}^3_{12} \vee s'_{13} & \bar{z}^3_{22} \vee s'_{23} & \bar{z}^3_{32} \vee s'_{33} \\ \bar{z}^5_{11} \vee s'_{13} & \bar{z}^5_{21} \vee s'_{23} & \bar{z}^5_{31} \vee s'_{33} \\ \bar{z}^5_{12} \vee s'_{13} & \bar{z}^5_{22} \vee s'_{23} & \bar{z}^5_{32} \vee s'_{33} \end{array}$$

- There are 15 Type-5 clauses:

$$\begin{array}{lcl} z^1_{11} \vee \bar{w}_{11} & \vee & \bar{s}'_{11} \vee \bar{s}_{12} \vee \bar{s}'_{13} \\ z^3_{11} \vee \bar{w}_{11} & \vee & \bar{s}'_{11} \vee \bar{s}_{12} \vee \bar{s}_{13} \\ z^3_{12} \vee \bar{w}_{12} & \vee & \bar{s}'_{11} \vee \bar{s}_{12} \vee \bar{s}_{13} \\ z^5_{11} \vee \bar{w}_{11} & \vee & \bar{s}_{11} \vee \bar{s}'_{12} \vee \bar{s}'_{13} \\ z^5_{12} \vee \bar{w}_{12} & \vee & \bar{s}_{11} \vee \bar{s}'_{12} \vee \bar{s}'_{13} \\ z^1_{21} \vee \bar{w}_{21} & \vee & \bar{s}'_{21} \vee \bar{s}_{22} \vee \bar{s}'_{23} \\ z^3_{21} \vee \bar{w}_{21} & \vee & \bar{s}'_{21} \vee \bar{s}_{22} \vee \bar{s}_{23} \\ z^3_{22} \vee \bar{w}_{22} & \vee & \bar{s}'_{21} \vee \bar{s}_{22} \vee \bar{s}_{23} \\ z^5_{21} \vee \bar{w}_{21} & \vee & \bar{s}_{21} \vee \bar{s}'_{22} \vee \bar{s}'_{23} \\ z^5_{22} \vee \bar{w}_{22} & \vee & \bar{s}_{21} \vee \bar{s}'_{22} \vee \bar{s}'_{23} \\ z^1_{31} \vee \bar{w}_{31} & \vee & \bar{s}'_{31} \vee \bar{s}_{32} \vee \bar{s}'_{33} \\ z^3_{31} \vee \bar{w}_{31} & \vee & \bar{s}'_{31} \vee \bar{s}_{32} \vee \bar{s}_{33} \\ z^3_{32} \vee \bar{w}_{32} & \vee & \bar{s}'_{31} \vee \bar{s}_{32} \vee \bar{s}_{33} \\ z^5_{31} \vee \bar{w}_{31} & \vee & \bar{s}_{31} \vee \bar{s}'_{32} \vee \bar{s}'_{33} \\ z^5_{32} \vee \bar{w}_{32} & \vee & \bar{s}_{31} \vee \bar{s}'_{32} \vee \bar{s}'_{33} \end{array}$$

The Satisfiability problem for the small example has 66 variables and 104 clauses. A satisfiable truth assignment was found in .18s on a 33MHz Silicon Graphics Indigo workstation, producing the Boolean vector function:

$$\begin{array}{rcl} y_1 & = & \bar{x}_3 + \bar{x}_1 x_2 + x_1 \bar{x}_2 \\ y_2 & = & \bar{x}_3 + \bar{x}_1 x_2 \end{array}$$

Table 5: Truth table of a 2-bit adder

x_1	x_2	z_1	z_2	y_1	y_2	y_3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Table 6: 8 input Boolean functions

			SAT			
m	k	A	vars	clauses	itr	time
2	2	300	608	6078	1	8.0s
2	3	300	486	5424	27	123.3s
2	4	300	824	8636	45	221.0s
2	5	300	1455	14233	202	766.1s
2	5	300	2605	23663	1	47.0s
4	3	350	1116	13024	5	74.0s
8	3	500	3168	37824	1	94.2s

Computational Results

We conclude this paper by reporting on preliminary computational testing of the integer programming algorithm described in [10] and [11] on instances of the Boolean Vector Function Synthesis Problem. We use the standard integer programming formulation for the Satisfiability problem [8]. The runs were carried out on a 33 MHz MIPS3000 Silicon Graphics Indigo workstation. The integer programming code was compiled with Fortran compiler `f77` and the C language compiler `cc` using compiler flags `-O2 -Olimit 800`. CPU times were measured with the system call `times()`.

We first consider the 2-Bit Adder ($y = x + z$) with the 4-input, 3-output specification given in Table 5. The Satisfiability problem formulation for this adder, using at most $k = 12$ distinct product terms has 408 variables and 1751 clauses. The interior point code took 66 iterations, producing the 11 product term vector function

$$\begin{aligned} y_1 &= x_1x_2z_2 + x_2z_1z_2 + x_1z_1 \\ y_2 &= \bar{x}_1x_2\bar{z}_1z_2 + \bar{x}_1\bar{x}_2z_1 + x_1x_2z_1z_2 + \\ &\quad x_1\bar{x}_2\bar{z}_1 + \bar{x}_1x_2z_1\bar{z}_2 + x_1x_2\bar{z}_1\bar{z}_2 \\ y_3 &= x_2\bar{z}_2 + \bar{x}_2z_2 \end{aligned}$$

in 34.3 CPU seconds.

Now consider 8-input Boolean vector functions. Table 6 summarizes the functions, Satisfiability problems and the experimental results. For each instance, the table shows the number of output variables (m), the number of product terms (k), the number of i/o-pairs in the truth table (A), the number of variables (vars) and clauses of the Satisfiability problem and the number of iterations and CPU time for the interior point algorithm to produce the Boolean function.

Table 7: Larger Boolean functions

				SAT			
n	m	k	A	vars	clauses	itr	time
10	4	5	400	2825	35641	353	8546.6s
16	2	5	300	1110	18308	30	153.0s
32	2	5	200	1245	31623	100	602.5s

Finally, in Table 7, we summarize results for three larger Boolean functions. In addition to the parameters shown in Table 6, this table gives the number of input variables (n) of the instance.

This preliminary computational experiment was limited to a narrow set of test problems. Nevertheless, it indicates that the interior point approach can synthesize complex Boolean vector functions in reasonable time. We believe this approach holds much promise for solving real-world instances. In a forthcoming paper, we extend the computational study to a wider set of problems, including some real-world circuit design problems.

References

[1] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI minimization.* Kluwer Academic, 1985.

[2] J. Gimpel, "A method of producing a boolean function having an arbitrarily prescribed prime implicant table," *IEEE Trans. Computers*, vol. 14, pp. 485–488, 1965.

[3] W. Quine, "The problem of simplifying truth functions," *Am. Math. Monthly*, vol. 59, 1952.

[4] W. Quine, "A way to simplify truth functions," *Am. Math. Monthly*, vol. 62, 1955.

[5] E. McCluskey, "Minimization of Boolean functions," *Bell Syst. Tech. J.*, vol. 35, pp. 1417–1444, 1956.

[6] S. Hong, R. Cain, and D. Ostapko, "MINI: A heuristic approach for logic minimization," *IBM J. Res. Develop.*, pp. 443–458, Sept. 1974.

[7] D. Brown, "A state-machine synthesizer-SMS," in *Proceedings of the 18th Design Automation Conference*, pp. 301–304, June 1981.

[8] A. Kamath, N. Karmarkar, K. Ramakrishnan, and M. Resende, "A continuous approach to inductive inference," *Mathematical Programming*, vol. 57, pp. 215–238, 1992.

[9] N. Karmarkar, "An interior point approach to NP complete problems," *Contemporary Mathematics*, vol. 114, pp. 297–308, 1990.

[10] N. Karmarkar, M. Resende, and K. Ramakrishan, "An interior point algorithm to solve computationally difficult set covering problems," *Mathematical Programming*, vol. 52, pp. 597–618, 1991.

[11] A. Kamath, N. Karmarkar, K. Ramakrishan, and M. Resende, "Computational experience with an interior point algorithm on the Satisfiability problem," *Annals of O.R.*, vol. 25, pp. 43–58, 1990.

[12] J. Moré and D. Sorensen, "Computing a trust region step," *SIAM J. Sci. Stat. Comput.*, vol. 4, pp. 553–572, 1983.