Optimal Design by Resolving Boolean Equations

Bernd Steinbach, Christian Posthoff

Abstract - The optimal design of a non-deterministic finite state machine by solving and resolving Boolean equations is shown, taking advantage of the relationships between single Boolean functions, sets of Boolean functions, Boolean equations, and their solutions. The Boolean Differential Calculus is used to point out the key ideas of the applied resolvability theory.

Keywords - optimal design, resolvability, function sets

I. INTRODUCTION

The coding of the states of a finite state machine (FSM) by Boolean variables is necessary in order to implement this machine by a digital circuit. If the number of states is not a power of 2, not all of the code words are needed for coding the essential states of the FSM. In this case there are some additional states in the real system, which can be used for the optimization of the circuit. If the state variables are not reachable from the additional inessential states, the circuit does not work well, if unfortunately the starting state is one of the inessential states.

A very easy but not optimal assumption to solve this problem is the demand that each inessential state switches in each case to a predefined essential state. More freedom for optimization and correct functionality is given if transitions from all inessential states to all essential states are allowed.

The main task of the logic design is the transformation of a given behavioral description into structural information. An easy method to describe the required behavior can be based on a simple table, called list of phases (LOP). The columns of this table are labeled by the logic input variables $x_i$ and the logic output variables $y_i$. The rows include the values 0 and 1, and such a row vector is called phase.

A combinational circuit realizes on the output $y_i$ the Boolean function $f(x)$. A Boolean function maps all patterns of the $n$-dimensional Boolean space $B^n$ into $B = \{0, 1\}$. A Boolean equation equates two Boolean functions. In the Boolean equation (1) the function on the left-hand side is the output variable $y_i$ and the function on the right-hand side describes the behavior for this output.

$$y_i = f(x)$$ (1)

The solution of a Boolean equation is a set of Boolean vectors having the property that their substitution into (1) results in $1 = 1$ or $0 = 0$. Note, first, the solution of a Boolean equation can be represented as LOP, and second, the output function is explicitly described by (1). We have a complete structural description of the circuit, if such an explicit equation (1) is known for each output of all gates.

If both sides of a Boolean equation are connected with the same Boolean function by a linear operation like XAND, its solution set does not change. Thus, using the transformation (2), the equations (1) and (3) have the same solution set, but the function for the output $y_i$ is implicitly described by (3).

$$y_i \odot f(x) = f(x) \odot f(x)$$ (2)

$$F(x, y_i) = 1$$ (3)

It was shown by (2) that the equation (3) can describe the Boolean function $f(x)$ for the output $y_i$, but the function $F(x, y_i)$ must fulfill a certain condition. Vice versa, the more general equation (3) can implicitly describe a set of Boolean functions $f(x)$ that give us on one hand the freedom of optimization, but on the other hand, the problem of resolving of (3) with respect to $y_i$ arises.

Having more than one output in the combinatorial circuit, the equation (3) can be generalized to the system equation (4).

$$F(x, y) = 1$$ (4)

The modeling method mentioned above can be extended from combinational to sequential circuits. In addition to the output functions a sequential circuit needs memory functions. The memory functions $g(x, s)$ specify the state $s'$ on the next time step depending on the input pattern $x$ and the recent state $s$. In the LOP columns for the independent variables $x$ and the function values $s'$ must be added. The associated system equation (5) includes two sets of variables, the independent variables $(x, s)$ and the dependent variables $(s', y)$.

$$F(x, s, s', y) = 1$$ (5)

A system equation as (3), (4) or (5) describes the allowed behavior. We assume that there are two systems having the system functions $F_1(x, s, s', y)$ and $F_2(x, s, s', y)$, respectively. The global behavior $F(x, s, s', y)$ of both subsystems can be calculated by (6), because the allowed behavior of the whole system covers the behavior of the first and the second subsystem. Connected wires of both subsystems must be labeled by the same variable.

$$F(x, s, s', y) = F_1(x, s, s', y) \land F_2(x, s, s', y)$$ (6)

The system functions $F_1(x, s, s', y)$ may describe a certain behavior of one part of the future circuit or the real behavior of an existing hardware block such as a gate or a flip-flop.

An important advantage of the system function is that transformation in both directions, from the inputs to the outputs and vice versa, is possible. Mostly the transformation from the inputs to the outputs is used. In this case the system equation must be resolved with respect to the output variables. Designing a sequential network based on given flip-flops, the system function of the flip-flop is needed to transform the memory function into the function controlling the flip-flop.
and the global system equation must be resolved with respect to
the input variables of the flip-flop.

This paper will show how the resolvability of the system
equation can be checked and how the explicit description of
the circuit functions can be calculated. The application of the
presented theory is illustrated using the optimal design of a
non-deterministic finite state machine.

II. RESOLVING A LOGIC EQUATION USING THE
BOOLEAN DIFFERENTIAL CALCULUS

The Boolean Differential Calculus (BDC) was introduced
in [1] and is used in this paper to describe the resolution of
logic equations. Using the cofactors \( f(x_i=0) \) and \( f(x_i=1) \) of a
Boolean function \( f(x_1, \ldots, x_n) \), the formula (7) defines the
simple maximum operation.

\[
\max_{x_i} f(x_1, \ldots, x_i, \ldots, x_n) = \begin{cases} 
  f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) & \text{if } f(x_1, \ldots, x_i = 0, \ldots, x_n) \\
  f(x_1, \ldots, x_i = 1, \ldots, x_n) & \text{if } f(x_1, \ldots, x_i = 0, \ldots, x_n) \text{ is undefined}
\end{cases}
\]

(7)
The simple maximum does not depend on the variable \( x_i \)
and is equal to 1 for such patterns \((x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)\)
where the two values of the variable \( x_i \) lead at least once to the
value 1 of the given function. The maximum operation can be
executed iteratively for several variables. In this way the \( m \)-th
maximum is defined as can be seen in (8).

\[
\begin{align*}
\max_{x_{i_1}, \ldots, x_{i_m}} f(x_{01}, \ldots, x_{i_1}, \ldots, x_{i_m}) &= \max_{x_{i_1}, \ldots, x_{i_m}} \left( \max_{x_{i_1}, \ldots, x_{i_m}} f(x_{01}, \ldots, x_{i_1}, \ldots, x_{i_m}) \right) \quad (8)
\end{align*}
\]

The conditions for which a function value 1 in an \( m \)-th
maximum operation occurs is that at least one 1 exists in the
subspace \( \mathbb{Z}_0 = \text{constant} \). The result of the \( m \)-th maximum
operations depends only on the variable set \( \mathbb{Z}_0 \).

We consider the homogenous characteristic logic equation

\[
f(x_1, \ldots, x_n, y_1, \ldots, y_m) = 1.
\]

(9)

For any problem-dependent reason, the variable \( x_1, \ldots, x_n \)
are considered to be independent and the variables \( y_1, \ldots, y_m \)
are expected to depend on \( x_1, \ldots, x_n \). The transformation of (9)
into the set of functions (10) is called resolvizing a logic equation.

\[
y_1 = \phi_1(x_1, \ldots, x_n)
\]

\[
\vdots
\]

\[
y_m = \phi_m(x_1, \ldots, x_n)
\]

(10)
For each vector \((x_1, \ldots, x_n)\), it has to be checked whether this
vector can be found in the solution set. In principle, there are
three possibilities.

a) If a given vector \((x_1, \ldots, x_n)\) cannot be found, there are no
values for \( y_1, \ldots, y_m \) together with this vector \((x_1, \ldots, x_n)\) in
the solution set; hence, the functions (10) do not exist.
b) For each vector \((x_1, \ldots, x_n)\), exactly one vector \((x, y)\) can
be found in the solution set. In this case, all the functions (10)
are defined by the equation (9) in a unique way.
c) For each vector \((x)\), there is at least one vector \((x, y)\), in the
solution set, however, for some vectors \((x)\) there are two or
more vectors in the solution set. In this case all functions
(10) exist, and for at least one of the functions (10) there
are several possibilities.

Theorem 1. The equation (9) can be solved with regard to
\( y_1, \ldots, y_m \) if and only if

\[
\max_{y_1, \ldots, y_m} f(x_1, \ldots, x_n, y_1, \ldots, y_m) = 1.
\]

(11)

This theorem ensures that all vectors \((x_1, \ldots, x_n)\) appear at
least once in the solution set. In this case the solution
functions can be characterized by the following theorem.

Theorem 2. If the equation (9) can be solved with regard to
\( y_1, \ldots, y_m \) then the elements of the solution set for one function
\( y_i(x_1, \ldots, x_n) \), \( i = 1, \ldots, m \) are elements of a lattice. The lower
and upper bound of these lattices are given as follows:

\[
q(x) = \inf_{y_i} f(x_1, \ldots, x_n, y_1, \ldots, y_m) = 1 \quad (12)
\]

\[
r(x) = \sup_{y_i} f(x_1, \ldots, x_n, y_1, \ldots, y_m) = 1 \quad (13)
\]
The vector \( y_0 \) contains the variables \( y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_m \).
The lower bound \( \inf_{y_i} V_i \) describes the ON-set \( q(x_1, \ldots, x_n) \) of
the lattice and the upper bound \( \sup_{y_i} V_i \) the complement of the
OFF-set, labeled by \( r(x_1, \ldots, x_n) \). The function \( y_i \) is unique if
and only if the lower bound and the upper bound are equal to
each other. Hence, there is a clear understanding of how to
check the resolvability and how to define solution functions.
A more detailed presentation can be found, e.g. in [1] and [3].
Note, there may be dependencies between the solution lattices
of selected functions \( y(x_1) \) and \( y(x_2) \), but each function of each
lattice is allowed in some combinations.

III. APPLICATIONS

The expression of a logic function in a disjunctive form
may be represented by a list of ternary vectors (TVL) [2]. The
columns of this list are labeled by the variables of the
expression and each vector in this list defines a conjunction.
The element ‘1’ expresses a non-negated variable and the
element ‘0’ expresses a negated variable, respectively. A dash
‘–’ means that the associated variable does not occur in the
conjunction. One benefit of such a TVL is that this data
structure may also describe the set of all solutions of a logic
equation in a very compact way. A complete set of more than
hundred algorithms for Boolean operations and operations of
the BDC was introduced in [2].

In the following a mod-3-counter will be designed using
DV-flip-flops and taking advantage from the non-deterministic
description of the behavior. Based on assumptions from the
introduction, transitions from inessential states to all essential
states are allowed. Figure 6.1 shows the graph of the non-
deterministic description of the mod-3-counter. Two Boolean
variables are necessary to code the essential states 00, 10 and
01.

\[
\begin{array}{c}
01 \\
11
\end{array}
\]

\[
\begin{array}{c}
00 \\
10
\end{array}
\]

Fig. 6.1. Non deterministic FSM of the mod-3-counter
The remaining code 11 adds one additional inessential state to the circuit to be realized. The counter runs without any control by an input. If the circuit realizes the transition $11 \rightarrow 11$, the counter cannot work, if it starts in the state 11. Each other transition $11 \rightarrow 00, 11 \rightarrow 10$ or $11 \rightarrow 01$ guarantees that the counting cycle will be reached. Which one of these three transitions should be used to find the simplest circuit structure? The following calculation shows, how this question can be answered by resolving the system equation of a non-deterministic FSM. The non-deterministic list of phases of the mod-3-counter LOP_M3C is shown in the following list of ternary vectors, were the dash ‘-’ represents both the value ‘0’ and ‘1’, respectively.

\[
\begin{array}{cccc}
s_1 & s_2 & s'_1 & s'_2 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & - & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_M3C &=& 1 & - & 0 & 1 \\
&= & 0 & 1 & 0 & 0 & 1 & 0 \\
&= & 0 & 0 & 1 & 0 \\
\end{array}
\]

- Design of the input functions of the first DV-flip-flop

The first DV-flip-flop is defined by the LOP FF_DV1.

\[
\begin{array}{cccc}
s_1 & s'_1 & d & v \\
0 & 0 & 0 & - \\
- & 0 & 0 & 1 \\
- & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
FF_DV1 &=& 1 & 1 & - & 0 \\
&= & 0 & 0 & - & 0 \\
&= & 1 & 1 & - & 0 \\
\end{array}
\]

Based on (6), the common behavior LOP_FF1 of the mod-3-counter and the first DV-flip-flop is calculated by the intersection of LOP_M3C and FF_DV1.

\[
\begin{array}{cccc}
s_1 & s_2 & s'_1 & s'_2 & D & v \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & - & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & - \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_FF1 &=& 0 & 1 & 0 & 0 & - \\
&= & 1 & 1 & 1 & 0 & 0 \\
&= & 1 & 1 & 0 & - & 0 \\
&= & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

The behavior information of the state function $s'_1$ is transformed in LOP_FF1 to the values of the input functions $d$ and $v$ of the first DV-flip-flop. These functions depend on the state variables $s_1$ and $s_2$. In case of $(s_1, s_2) = (01)$ both functions $d$ and $v$ can choose the values 0 or 1, but the combination $d=1$ and $v=1$ is not allowed in this case. Without any restrictions it is allowed to resolve the equation $F_{FF1}(s, s', d, v) = 1$ only for one of the input functions $d$ or $v$. We begin with the input function $d$. Carrying out the $m$-th maximum with respect to the variables $s'_1$, $s'_2$ and $v$ reduces LOP_FF1 to LOP_D1, which describes the dependency of $d$ from $s_1$, $s_2$. The equation $F_{D1}(s, v) = 1$, associated to LOP_D1, is resolvable with respect to $d$.

\[
\begin{array}{cccc}
s_1 & s_2 & d \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 1 & - \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_D1 &=& 1 & 0 & 0 \\
&= & 0 & 1 & 0 \\
&= & 1 & 1 & 1 \\
\end{array}
\]

All solution functions $d_1$ are visualized in the following Karnaugh – map, and a simple solution function is selected.

\[
\begin{array}{c}
s_2 \\
\hline
1 & 0 & 0 \\
1 & 1 & 1 \\
0 & 1 & s_1 \\
\end{array}
\]

\[
\begin{array}{cccc}
d_1 = s_1 \\
\end{array}
\]

The list of phases of the selected function LOP_D1S is

\[
\begin{array}{cccc}
s_1 & s_2 & d \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 1 & - \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_D1S &=& 0 & - & 1 \\
&= & 1 & - & 0 \\
\end{array}
\]

and the intersection of the previous LOP_FF1 and LOP_D1S creates the restricted remaining behavior of the mod-3-counter. Note, the function value of $v$ in the case of $(s_1, s_2) = (01)$ is now fixed to 0, but the choice of the function $d_1$ does not restrict the allowed behavior in the inessential state 11.

\[
\begin{array}{cccc}
s_1 & s_2 & s'_1 & s'_2 & d & v \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & - & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_FF1 &=& 0 & 1 & 0 & 0 & 1 \\
&= & 1 & 1 & 1 & 0 & 0 \\
\end{array}
\]

Next, the input function $v_1$ is calculated. Carrying out the $m$-th maximum with respect to the variables $s'_1$, $s'_2$ and $d$ reduces LOP_FF1 to LOP_V1, which describes the dependency of $v$ from $s_1$, $s_2$. The equation $F_{V1}(s, v) = 1$, associated to LOP_V1, is resolvable with respect to $v$.

\[
\begin{array}{cccc}
s_1 & s_2 & v \\
0 & 0 & - \\
- & 1 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_V1 &=& - & 1 & 0 \\
&= & 1 & 1 & 1 \\
\end{array}
\]

All solution functions $v_1$ are visualized in the following Karnaugh – map, and a simple solution function is selected, taking the values 1 only.

\[
\begin{array}{cccc}
s_2 & v_1 \\
0 & \bar{1} \\
1 & 0 \bar{1} \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 1 & s_1 \\
\end{array}
\]

The list of phases of the selected function LOP_V1S is

\[
\begin{array}{cccc}
s_1 & s_2 & v \\
0 & 0 & - \\
- & 1 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_V1S &=& - & 1 & 0 \\
&= & 1 & 1 & 1 \\
\end{array}
\]

and the intersection of previous LOP_FF1 and LOP_V1S creates the remaining behavior of the mod-3-counter.

\[
\begin{array}{cccc}
s_1 & s_2 & s'_1 & s'_2 & d & v \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

Note, the choice of the input function $v_1$ restricted the non-deterministic finite state machine of the mod-3-counter to a deterministic one. After removing $d$ and $v$ by the $m$-th maximum we got LOP_M3C again, and these columns can be reused to design the input functions of the 2nd DV-flip-flop.

\[
\begin{array}{cccc}
s_1 & s_2 & s'_1 & s'_2 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
LOP_M3C &=& 1 & 0 & 0 & 1 \\
&= & 0 & 1 & 0 & 0 \\
&= & 1 & 1 & 1 & 0 \\
\end{array}
\]

- Design of the input functions of the second DV-flip-flop

The second DV-flip-flop is defined by the LOP FF_DV2. The design steps are the same as for the first DV-flip-flop.

\[
\begin{array}{cccc}
s_2 & s'_2 & d \\
0 & 0 & - & 0 \\
- & 0 & 0 & 1 \\
- & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
FF_DV2 &=& 1 & 1 & - & 0 \\
&= & 0 & 0 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
\end{array}
\]

Based on (6), the common behavior LOP_FF2 of the mod-3-counter and the second DV-flip-flop is calculated by the intersection of LOP_M3C and FF_DV2.
Note, the FSM of the mod-3-counter is now deterministically specified, but in case of \((s_1, s_2) = (00)\) both functions \(d\) and \(v\) can choose the values 0 or 1. In order to design the input function \(d\) we carry out the \(m\)-th maximum with respect to the variables \(s'1, s'2\) and \(v\), which reduces LOP_FF2 to LOP_D2. The equation \(F_D2(s, d) = 1\), associated to LOP_D2, is resolvable with respect to \(d\).

\[
\begin{array}{c|c|c|c|c|c|c}
 s1 & s2 & s'1 & s'2 & d & v \\
 0 & 0 & 1 & 0 & 1 & 0 \\
 0 & 0 & 1 & 0 & 0 & - \\
 LOP_FF2 &=& 1 & 0 & 0 & 1 & 1 \\
 0 & 1 & 0 & 0 & 0 & 1 \\
 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

All solution functions \(d_2\) are visualized in the following Karnaugh-map and a simple solution function is selected.

\[
\begin{array}{c|c|c}
 s2 & d_2 \\
 0 & 0 \\
 1 & 0 \\
\end{array}
\]

The list of phases of the selected function LOP_D2S is

\[
\begin{array}{c|c|c|c|c|c|c}
 s1 & s2 & d & v \\
 0 & 0 & - & 1 \\
 - & 1 & 0 & 0 \\
 LOP_D2S &=& - & 0 & - & 1 \\
 0 & 0 & - & - \\
\end{array}
\]

and the intersection of previous LOP_FF2 and LOP_D2S creates the remaining behavior of the mod-3-counter. Note, the function value of \(v\) for \((s_1, s_2) = (00)\) is now fixed to 0.

Next, the input function \(v_2\) is calculated. LOP_FF2 is reduced to LOP_V2 by the \(m\)-th maximum with respect to the variables \(s'1, s'2\) and \(d\). The equation \(F_V2(s, d) = 1\), associated to LOP_V2, is uniquely resolvable with respect to \(v\).

\[
\begin{array}{c|c|c|c|c|c|c}
 s1 & s2 & s'1 & s'2 & d & v \\
 0 & 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 0 & 0 & 0 & 1 \\
 LOP_FF2 &=& 1 & 0 & 0 & 1 & 1 \\
 0 & 1 & 0 & 0 & 0 & 1 \\
 1 & 1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

The solution function \(v_2\) is visualized in the following Karnaugh-map, and a simple expression is shown.

\[
\begin{array}{c|c|c|c|c|c|c}
 s2 & v_2 \\
 0 & 1 \\
 1 & 1 \\
\end{array}
\]

The list of phases of the selected function LOP_V1S is equal to LOP_V2. The intersection of the previous LOP_FF2 and LOP_V2S returns the already known deterministic behavior of the mod-3-counter.

\[
\begin{array}{c|c|c|c|c|c|c}
 s1 & s2 & s'1 & s'2 & d & v \\
 0 & 0 & 1 & 0 & 1 & 1 \\
 0 & 1 & 0 & 0 & 0 & 1 \\
 1 & 1 & 1 & 0 & 0 & 0 \\
 LOP_FF2 &=& 1 & 0 & 0 & 1 & 1 \\
 0 & 1 & 0 & 0 & 1 & 1 \\
 1 & 1 & 1 & 0 & 0 & 0 \\
\end{array}
\]

After removing \(d\) and \(v\) by the \(m\)-th maximum we get LOP_M3C which specifies the behavior shown in figure 6.2 of the designed circuit structure, shown in figure 6.3.

---

**IV. CONCLUSIONS**

The implicit representation of Boolean function sets extends the possible approaches in circuit design significantly. As shown in this paper, a system equation allows the specification of the deterministic or non-deterministic behavior of combinatorial or sequential Boolean systems. Based on the implicit representation of sets of Boolean functions it is easily possible to deal with output functions depending on the inputs and the inverse input functions depending on the outputs. The composition of subsystems to a more complex system can be done by a simple conjunction of their system functions.

In order to use such benefits in circuit design, it is necessary to know how a system equation can be resolved. Based on the Boolean Differential Calculus, this paper introduces the main ideas of the theory of resolving Boolean equations. It was shown in which case a system equation is resolvable and how the set of all possible solutions can be calculated. The paper shows by the example of a mod-3-counter how an optimal circuit structure can be found using all the freedom of this non-deterministic specification.

**REFERENCES**

