Methods of Partial Logic

Anatoly Prihozhy

*State Polytechnical Academy, 65 Block 11a, F.Skorina ave., Minsk 220027 Belarus
Fax: 375-017 2327153, e-mail: aprihozhy@bspa.unibel.by

Abstract

This paper presents novel theoretical results obtained in the field of partial logic. New operations (including a very useful minimization operation), laws, and expansions are introduced. Traditional Boolean function representation forms for the completely specified functions are generalized for the incompletely specified and partial functions.

1. Introduction

Most of the existing logic level synthesis techniques are based on the traditional two-valued logic [1]. The logic allows generation of various representations for the same logic function that can be mapped to digital circuits with different parameters. The most significant representations are as follows:

- sum of products
- product of sums
- Reed-Muller expressions
- decision diagrams.

Two key types of decision diagrams are used [1,2,4]:

- binary decision diagrams (BDDs) derived from the Shannon expansion
- functional decision diagrams (FDDs) derived from the positive and negative Davio expansions.

The ROBDDs (reduced ordered BDDs) [2,4] being a Boolean function canonical representation form, are the most popular type of decision diagrams widely used for efficient modeling, synthesis, and verification of digital circuits.

High-level synthesis systems [5-6] need efficient logic optimization techniques.

Incompletely specified functions are a very useful formalism for generating different alternatives during logic optimization process [3].

This paper presents novel theoretical results in the field of logic that uses three values: true, false, and don't care. New operations and representation forms (expressions) for the three-valued functions, laws and decomposition types in the logic are described in the paper.

2. Partial and incompletely specified variables and functions

The traditional total logic considers two values: *true (1)* and *false (0)*. The partial logic considers three values: *true (1)*, *false (0)*, and *don't care* (*dc* or –). The *don't care* value can be replaced with *true* or *false* arbitrarily. A total function $f(x_1,...,x_n)$ is a mapping $f: B^n \rightarrow B$ where $B=\{0,1\}$. A partial function $g(y_1,...,y_m)$ is a mapping $g: M^m \rightarrow M$ where $M=\{0,1,-\}$. An incompletely specified function $h(x_1,...,x_n)$ is a mapping $h: B^n \rightarrow M$. A variable which takes values from the set B will be called a total variable, and a variable which takes values from the set M will be called a partial variable.

A Value-Domain Representation (VDR) is the following encoding of a partial variable y_i with a pair $(v_i|d_i)$ of total variables:

$$y_i = \begin{cases} 0, & \text{if } v_i=0 \text{ and } d_i=1, \\ 1, & \text{if } v_i=1 \text{ and } d_i=1, \\ dc, & \text{if } v_i \in \{0,1\} \text{ and } d_i=0. \end{cases}$$

The variable v_i is called a value variable and the variable d_i is called a domain variable.

Due to VDR a partial function $z=g(y)$ of m three-valued arguments is represented by an incompletely specified function $(v|d)=g'((v_1|d_1), ...,(v_n|d_n))$ of $2m$ two-valued arguments with on-set $g^{on}=(v\&d)^{on}$, *off-set* $g^{off}=(\sim v\&d)^{on}$, and *don't-care-set* $g^{dc}=(\sim d)^{on}$.

Monadic and dyadic partial operations are transformed to operations on pairs of total functions (Table 1). The operations are used for construction of partial logic expressions and mixed total-partial logic expressions. Pairs of the expressions representing the same partial function constitute partial logic laws. The laws which transform a partial expression to a pair of total expressions, connect the partial logic to the traditional total logic. They include:

$$\sim(v_1|d_1) = (\sim v_1|d_1).$$
$$(v_1|d_1)\&(v_2|d_2) = (v_1\&v_2|d_1\&d_2+\sim v_1\&d_1+\sim v_2\&d_2),$$
$$(v_1|d_1)+(v_2|d_2) = (v_1+v_2|d_1\&d_2+v_1\&d_1+v_2\&d_2),$$
$$(v_1|d_1)\rightarrow(v_2|d_2) = (v_1\rightarrow v_2|d_1\&d_2+\sim v_1\&d_1+v_2\&d_2),$$
$$(v_1|d_1)\oplus(v_2|d_2) = (v_1\oplus v_2|d_1\&d_2).$$

The left parts of equalities contain partial operations including negation (~), conjunction (&), disjunction (+), implication (→), exclusive OR (⊕), and their right parts contain total operations with the same notations.

Table 1

Partial logic operations

N	Operation name	Values: 0 1 – 0 1 – 0 1 – / 0 0 0 1 1 1 – – –	Notation
1	Negation	1 0 –	$\sim(v_1\|d_1)$
2	Conjunct.	0 0 0 0 1 – 0 – –	$(v_1\|d_1)\&(v_2\|d_2)$
3	Disjunct.	0 1 – 1 1 1 – 1 –	$(v_1\|d_1)+(v_2\|d_2)$
4	Implicat.	1 0 – 1 1 1 1 – –	$(v_1\|d_1)\rightarrow(v_2\|d_2)$
5	Excl. OR	0 1 – 1 0 – – – –	$(v_1\|d_1)\oplus(v_2\|d_2)$

3. Minimization operation

In the pair $(v|d)$ function d is fixed. The function v can be replaced with another total function v_t such that

$$(v_t|d) = (v|d)$$

or

$$v_t\&d = v\&d.$$

In other words, VDRs $(v_t|d)$ and $(v|d)$ represent the same incompletely specified function. If V is the set of functions v_t then for each $v_t \in V$ the inequality

$$(v\&d)^{on} \subseteq v_t^{on} \subseteq (v+\sim d)^{on}$$

holds.

A minimization operation $min(v|d)$ is a mapping $min: F\times F \rightarrow F$ where F is the set of total functions $f: B^n \rightarrow B$ In fact, the operation selects one function from the set V. Various definitions for $min(v|d)$ are possible. They depend on which representation forms for v and d are used. In work [7] a definition of the operation on BDDs and in particular on ROBDDs is given. The operation allows decrease in the number of nodes and edges in the BDD v as shown in Fig.1.

Figure 1: Minimization operation on BDDs, an example

4. Partial logic laws

The following laws in the partial logic generalize known laws in the traditional logic:

$(v|d) = \sim\sim(v|d),$
$(v_1|d_1)\&(v_2|d_2) = (v_2|d_2)\&(v_1|d_1),$
$(v_1|d_1)\&((v_2|d_2)\&(v_3|d_3)) =$
$\quad((v_1|d_1)\&(v_2|d_2))\&(v_3|d_3),$
$(v_1|d_1)+(v_2|d_2) = (v_2|d_2)+(v_1|d_1),$
$(v_1|d_1)+((v_2|d_2)+(v_3|d_3)) =$
$\quad((v_1|d_1)+(v_2|d_2))+(v_3|d_3),$
$(v_1|d_1)\&((v_2|d_2)+(v_3|d_3)) =$
$\quad((v_1|d_1)\&(v_2|d_2))+((v_1|d_1)\&(v_3|d_3)),$
$(v_1|d_1)+((v_2|d_2)\&(v_3|d_3)) =$
$\quad((v_1|d_1)+(v_2|d_2))\&((v_1|d_1)+(v_3|d_3)),$
$\sim((v_1|d_1)\&(v_2|d_2)) = \sim(v_1|d_1)+\sim(v_2|d_2),$
$\sim((v_1|d_1)+(v_2|d_2)) = \sim(v_1|d_1)\&\sim(v_2|d_2),$
$(v_1|d_1)\&((v_1|d_1)+(v_2|d_2)) = v_1|d_1,$
$(v_1|d_1)+((v_1|d_1)\&(v_2|d_2)) = v_1|d_1,$
$(v_1|d_1)\text{->}(v_2|d_2) = \sim(v_1|d_1)+(v_2|d_2),$
$(v|d)\&(v|d) = v|d,$
$(v|d)+(v|d) = v|d,$
$(v|d)\&\sim(v|d) = 0|d,$
$(v|d)+\sim(v|d) = 1|d,$
$(v|d)\&(1|1) = v|d,$
$(v|d)+(0|1) = v|d,$
$(v|d)\&(0|1) = 0|1,$
$(v|d)+(1|1) = 1|1,$
$(0|1)\text{->}(v|d) = 0|1,$
$(v|d)\text{->}(v|d) = 1|d,$
$(v|d)\text{->}(1|1) = 1|1,$
$(v_1|d_1)\text{->}(v_2|d_2) = \sim(v_2|d_2)\text{->} \sim(v_1|d_1).$

New laws of the partial logic are as follows:

$(v_1|d)\&(v_2|d) = v_1\&v_2|d,$
$(v_1|d)+(v_2|d) = v_1+v_2|d,$
$(v|d_1)\&(v|d_2) = v|d_1\&d_2+\sim v\&(d_1+d_2),$
$(v|d_1)+(v|d_2) = v|d_1\&d_2+v\&(d_1+d_2),$
$v|v = 1|v,$

$\sim v|v = 0|v,$
$v\&d|d = v|d,$
$v+d|d = 1|d,$
$v\&\sim d|d = 0|d,$
$v+\sim d|d = v|d,$
$v|v\&d = 1|v\&d,$
$v|\sim v\&d = 0|\sim v\&d,$
$v|v\&d = (v|d)+(v|0),$
$v|\sim v\&d = (v|d)\&(v|0),$
$v|v+d = (v|1)+(0|d),$
$v|\sim v+d = (v|1)\&(1|d),$
$\sim v|v+d = (\sim v|1)+(1|d),$
$\sim v|\sim v+d = (\sim v|1)\&(0|d),$
$f|x_i = f(x_i=1)|x_i,$
$f|\sim x_i = f(x_i=0)|x_i,$
$v|v\oplus d = \sim d|v\oplus d,$
$v|v\equiv d = d|v\equiv d,$
$\sim v|v\equiv d = \sim d|v\equiv d,$
$f(v)|v\oplus d = f(\sim d)|v\oplus d,$
$f(v)|v\equiv d = f(d)|v\equiv d,$
$min(\sim v|d) = \sim min(v|d),$
$v\&d \le min(v|d) \le v+\sim d.$

5. Partial logic expansions

The work [1] proofs that only the Shannon and Davio expansions are useful for representation and manipulation of functions in the total logic. The Shannon expansion of function $f(x)$ in the logic is as follows:

$$f(x) = x_i \& f_{xi=1} + \sim x_i \& f_{xi=0}$$

where x_i is a total variable, $f_{xi=1}$ is a cofactor of function $f(x)$ on $x_i=1$ and $f_{xi=0}$ is a cofactor of the function on $x_i=0$.

The following generalization for the Shannon expansion holds in the partial logic:

$$f(x) = \alpha(x) \,\&\, min(f(x)|\alpha(x)) + \sim \alpha(x) \,\&\, min(f(x)|\sim\alpha(x))$$

In the expansion variable x_i is replaced with an arbitrary total function $\alpha(x)$ and the cofactors are replaced with the operations minimizing $f(x)$ on $\alpha(x)$ and $\sim\alpha(x)$ respectively. When $\alpha(x)=1$ then

$$1 \,\&\, min(f(x)|\,1) + 0 \,\&\, min(f(x)|\,0) = f(x).$$

When $\alpha(x)=0$ we obtain the same result.

The partial logic expansion

$$f(x) = min(f(x)|\sim\alpha(x)) \oplus \alpha(x) \,\&\, (min((x)f|\alpha(x)) \oplus min(f(x)|\sim\alpha(x)))$$

is a generalization for the positive Davio expansion. A generalization for the negative Davio expansion is derived from the positive one by means of replacement of the function $\alpha(x)$ with its negation $\sim\alpha(x)$.

6. Function representation forms

The table form is the basic one for the representation of the partial and incompletely specified Boolean functions.

The partial Sum of Products is represented by the following expression

$$f = \Big(\sum_{f(a)=1}^{m} \&_{j=1}^{a_j} (y_j\,|1)\Big) + \Big(\sum_{f(a)=0}^{m} \&_{j=1}^{a_j} (0|y_j)\Big) =$$

$$\Big(\sum_{f(a)=1}^{m} \&_{j=1}^{a_i} y_j\,|1\Big) + \Big(0\,|\sum_{f(a)=0}^{m} \&_{j=1}^{a_i} y_j\Big),$$

where $a=(a_1,\ldots,a_m)$ and

$$y_j = \begin{cases} \sim y_j, & \text{if } a_j=0, \\ y_j, & \text{if } a_j=1. \end{cases}$$

if $a_j \in B$, and

$$y_j = \begin{cases} \sim vj \,\&\, dj, & \text{if } a_j=0, \\ y_j \,\&\, dj, & \text{if } a_j=1, \\ \sim d_j, & \text{if } a_j=\text{'-'}. \end{cases}$$

if $a_j \in M$, where v_j, d_j are two-valued variables encoding the three valued variable y_j.

The partial Product of Sums is represented by the following expression

$$f = \Big(\prod_{f(a)=0}^{m} \&_{j=1}^{\sim a_j} (y_j\,|1)\Big) \,\&\, \Big(\prod_{f(a)=-}^{m} \&_{j=1}^{\sim a_j} (1\,|y_j)\Big) =$$

$$\Big(\prod_{f(a)=0}^{m} \&_{j=1}^{\sim a_i} y_i\,|1\Big) \,\&\, \Big(1\,|\prod_{f(a)=-}^{m} \&_{j=1}^{\sim a_i} y_j\Big).$$

7. If-decision diagrams

The if-decision diagrams (IFDs) and functional if-decision diagrams (FIFDs) [7] are constructed though using the generalized Shannon and Davio expansions. Basic fragments of IFDs and FIFDs are shown in Fig.2.

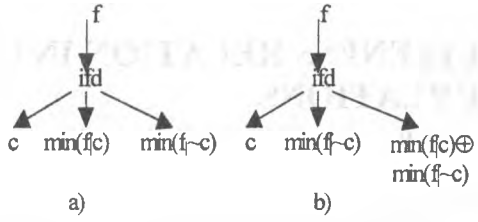
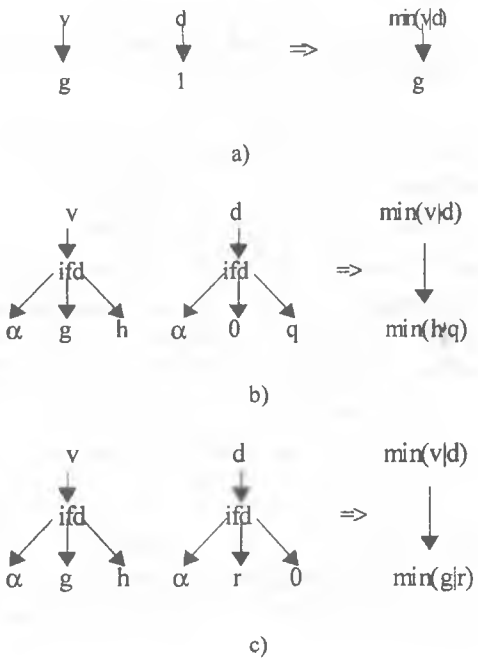


Figure 2: Construction of a) IFD and b) FIFD

The IFD is a generalization for the BDD and the FIFD is a generalization for the FDD. The IFD is represented by a rooted directed noncyclic graph the terminal nodes of which are labeled *0*, *1*, x_i, and $\neg x_i$ and the nonterminal nodes are not labeled and have exactly three successors. The diagram graph is reduced if it does not include identical subgraphs. The IFDs and FIFDs extend the set of representation alternatives and allow a compressed representation and efficient manipulation of Boolean functions.

In order to manipulate the IFDs, we define the minimization operation on this type of decision diagrams. Four cases in Fig.3 define the minimization result for different source IFDs *v* and *d*. As the figure shows, the operation may remove nodes and edges from the diagram *v* in cases b) and c).



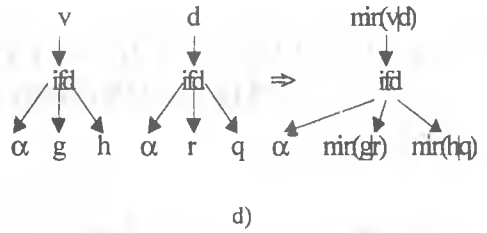


Figure 3: Minimization operation on IFDs

References

[1] B. Becker and R.Drechsler, "How Many Decomposition Types Do We Need?", presented at the ED&TC European Conference, Paris, France, 1995.

[2] R.E.Bryant, "Graph-based algorithms for Boolean function manipulation", IEEE Transactions on Computers, Vol C-35, pp. 677-691, August 1986.

[3] M. Damiani, G. DeMicheli, "Don't Care Set Specification in Combinational and Synchronous Logic Circuits", IEEE Trans. On CAD, vol.12, 1993, pp.365-388.

[4] S. Malik, A. Wang, R. Brayton, A. Sangiovanni-Vincentelli, "Logic Verification Using Binary Decision Diagrams in a Logic Synthesis Environment", in Proceedings of the Int. Conference on CAD, 1988, pp. 6-9.

[5] J.P. Mermet ed., Fundamentals and Standards in Hardware Description Languages. Boston: Kluwer Academic Publishers, 1993.

[6] A. Mozipo, D. Massicotte, P. Quinton, T. Risset, "Automatic Synthesis of a Parallel Architecture for Kalman Filtering using MMAlpha", in Proceedings of the Int. Conference on Parallel Computing in Electrical Engineering, Tech. University of Bialystok, Poland, 1998, pp.201-296.

[7] A. Prihozhy, "If-Diagrams: Theory and Application", in Proceedings of the Conference PATMOS'97, UCL, Belgium, 1997, pp.369-378.