GENERATION OF IDEMPOTENT MONOTONE BOOLEAN FUNCTIONS

Ilya Shmulevich
University of Nijmegen
The Netherlands
e-mail: shmulevich@nici.kun.nl

ABSTRACT

This paper focuses on the class of idempotent monotone Boolean functions. Monotone Boolean functions correspond to an important class of non-linear digital filters called Stack Filters. The idempotence property implies that applying such a filter produces a root signal in one pass. We present an algorithm for testing a given monotone Boolean function for idempotence and provide the set of idempotent monotone Boolean functions of 5 variables.

1 INTRODUCTION

Monotone Boolean functions have been extensively studied in the area of non-linear digital filtering, specifically stack and morphological filtering. In fact, any Stack Filter of window-width n is uniquely specified by a monotone Boolean function of n variables. Similarly, the Stacking Property obeyed by all stack filters is the monotonicity of these Boolean functions [1],[2]. In this paper, we focus on idempotent monotone Boolean functions and discuss a procedure for testing a given monotone Boolean function for idempotence.

1.1 Monotone Boolean Functions and the Free Distributive Lattice

Let $\tilde{\alpha} = (\alpha_1, \cdots, \alpha_n)$ and $\tilde{\beta} = (\beta_1, \cdots, \beta_n)$ be n-element binary vectors (primes). We say that $\tilde{\alpha}$ precedes $\tilde{\beta}$, denoted $\tilde{\alpha} \preceq \tilde{\beta}$, if $\alpha_i \leq \beta_i$ for $1 \leq i \leq n$. Let E^n represent the n-cube. The k^{th} level ($0 \leq k \leq n$) contains only those primes with exactly k components equal to 1. The set of all primes on the k^{th} level will be denoted by $E^{n,k}$. A Boolean function is one which maps E^n into E^1.

Definition 1 *A Boolean function $f : E^n \rightarrow E^1$ is called* **monotone** *(also called* **positive***) if for any two primes $\tilde{\alpha}$ and $\tilde{\beta}$ such that $\tilde{\alpha} \prec \tilde{\beta}$, we have $f(\tilde{\alpha}) \leq f(\tilde{\beta})$.*

The set of all monotone Boolean functions of n variables consists of all closed from below subsets of E^n, the Boolean algebra on n elements. It is also ordered by the same relation as E^n and is called the Free Distributive Lattice on n generators, denoted by $FDL(n)$.

It is well known that a monotone Boolean function is uniquely defined by its set of *minimal primes*, where a prime $\tilde{\alpha} \in E^n$ is *minimal* if $f(\tilde{\alpha}) = 1$ and $f\left(\tilde{\beta}\right) = 0$ for all $\tilde{\beta} \in E^n$ such that $\tilde{\beta} \prec \tilde{\alpha}$ [3]. Again, this is a consequence of the fact that the Boolean function is monotone.

Definition 2 *Let $\tilde{\alpha} \in E^{n,k}$ be an arbitrary prime. Then the set of all primes $\tilde{\beta} \in \bigcup_{s=k+1}^{n} E^{n,s}$ such that $\tilde{\alpha} \prec \tilde{\beta}$, will be called the* **shadow** *of $\tilde{\alpha}$, and denoted by $T(\tilde{\alpha})$.*

If $f(\tilde{\alpha}) = 1$, then we say that f **selects** $\tilde{\alpha}$ and $|f|$ is the number of such primes for some function $f \in FDL(n)$. That is, $|f| = |\{\tilde{\alpha} \in E^n : f(\tilde{\alpha}) = 1\}|$. Since E^n has exactly 2^n elements, $0 \leq |f| \leq 2^n$ and $|f|$ represents the level of $FDL(n)$ on which f is located. Clearly, $FDL(n)$ splits into $2^n + 1$ levels.

1.2 Idempotent Monotone Boolean Functions

Let us consider a special class of monotone Boolean functions. Before proceeding, however, it is necessary to state the following definition.

Definition 3 *A Boolean function $f(x_1, \cdots, x_i, \cdots, x_n)$ depends* **essentially** *on the variable x_i if there exist two primes $\tilde{\alpha}$ and $\tilde{\alpha}'$ which differ only in the i^{th} coordinate, such that $f(\tilde{\alpha}) \neq f(\tilde{\alpha}')$.*

The variable x_i will be called an **essential** variable. Let the number of essential variables for a particular function f be called the **order** of f. If the order of f is equal to n, then the function will be called a **full-order** function.

To this end, let $f : E^n \rightarrow E^1$ be a monotone Boolean function. Let us define n functions $f_1, \cdots, f_n : E^{2n-1} \rightarrow E^1$ as follows:

$$f_k(x_1, \cdots, x_{2n-1}) = f(x_k, \cdots, x_{k+n-1}) \, \forall \, x_k, \cdots, x_{k+n-1} \tag{1}$$

The above functions will be called the **expanded** functions, since their domains have been expanded to E^{2n-1}.

Please note that fictitious variables have been purposefully introduced into these definitions so that all $f_i,\ i = 1, \cdots, n$ can be elements of the same set, namely, $FDL(2n-1)$ even though the order of each f_i is not more than n. Also note that there was no assumption being made regarding the order of f itself, which may very well be less than n. For the ensuing discussion, let us restrict our attention to full-order functions. Finally, let us assume that from now on n is an odd number, since the definition of idempotence becomes ambiguous for even n. Now, we are ready to give the definition of an idempotent monotone Boolean function.

Definition 4 *If*

$$f\ (f_1(x_1, \ldots, x_{2n-1}),\ f_2(x_1, \ldots, x_{2n-1}), \cdots \\ \cdots, f_n(x_1, \ldots, x_{2n-1})) = \\ f\left(x_{\frac{n+1}{2}}, \ldots, x_{\frac{3n-1}{2}}\right)\ \textit{for all}\ x_{\frac{n+1}{2}}, \ldots, x_{\frac{3n-1}{2}}$$

then $f \in FDL(n)$ *is an* **idempotent monotone Boolean function**.

As can be seen from the above definition, we are considering a composition of functions. Since functions f, $f_1, f_2, \cdots, f_n$ are all monotone functions, it can easily be shown that the composition of these functions as shown in Definition 4 is also a monotone function.

2 REPRESENTATION OF MONOTONE BOOLEAN FUNCTIONS

Before we can check a given monotone Boolean function for idempotence, we must have an efficient method for storing such a function in a computer. The most economical and computationally convenient way to represent a function f is to simply store the vector $\tilde{\tau}_f$ as a binary string of length 2^n, where n is the number of variables. So, for example the function $f(x_1, x_2, x_3) = x_2 + x_1x_3$ would be represented by

$$\tilde{\tau}_f = 11\underline{1}01\underline{1}00 \tag{2}$$

The bits are arranged in the usual binary coded decimal (BCD) representation starting from right to left. The bits which are underlined are the minimal primes of the function. The rest of the 1^s represent the shadow. It should be noted that to perform a conjunction or disjunction of two functions f and g, we simply perform the logical bitwise AND or OR operation respectively on $\tilde{\tau}_f$ and $\tilde{\tau}_g$. This is one reason why the above representation is computationally convenient.

3 TESTING FOR IDEMPOTENCE

In order for us to test a given function f for idempotence, we must first form the n expanded functions $f_1, \cdots, f_n$ defined in (1). Since $f_1, \cdots, f_n$ are all functions of $2n-1$ variables, we will need binary vectors of lengths 2^{2n-1} to represent them. Furthermore, since each f_i is defined in terms of f, we can construct the vectors $\tilde{\tau}_{f_i},\ i = 1, \cdots, n$ in terms of $\tilde{\tau}_f$. To see how to do this, we simply need to consider the truth table that defines each f_i. Each f_i has only at most n essential variables and thus has $n-1$ fictitious variables. So, if $x_k, \cdots, x_{k+n-1}$ are the essential variables (for function f_k), then for any particular values of $x_k, \cdots, x_{k+n-1}$, the value of f_k will be repeated 2^{n-1} times in the vector $\tilde{\tau}_{f_k}$. This is due to the fact that for each $x_k, \cdots, x_{k+n-1}$, the $n-1$ fictitious variables combine to make 2^{n-1} possible values.

To construct $\tilde{\tau}_{f_1}$, each bit of $\tilde{\tau}_f$ will be repeated 2^{n-1} times in a row. This is because for f_1, the essential variables are the leftmost ones (most significant bits) and the fictitious ones are the rightmost ones (least significant bits). For $\tilde{\tau}_{f_2}$, each bit of $\tilde{\tau}_f$ will be repeated 2^{n-2} times in a row. However, since the truth table is basically split up into *two* parts: one when the fictitious variable $x_1 = 0$ and one when $x_1 = 1$, $\tilde{\tau}_{f_2}$ has two identical parts. In general, to construct $\tilde{\tau}_{f_k}$, each bit of $\tilde{\tau}_f$ repeats 2^{n-k} times and when we run out of bits in $\tilde{\tau}_f$, we start again, repeating this process 2^{k-1} times. As one can see, $2^{n-k} \cdot 2^{k-1} = 2^{n-1}$, which is the total number of times each bit repeats. The following example illustrates this construction for the vector defined in (2).

Example 1 *Let* $\tilde{\tau}_f = 11\underline{1}01\underline{1}00$. *Then,*

$$\begin{array}{lcl} \tilde{\tau}_{f_1} & = & 1111111111\underline{1}0000111111\underline{1}00000000 \\ \tilde{\tau}_{f_2} & = & 1111110011110000111111\underline{0}0111\underline{1}0000 \\ \tilde{\tau}_{f_3} & = & 1110110011101100111011001\underline{1}01\underline{1}00 \end{array}$$

Notice that in $\tilde{\tau}_{f_1}$ *every bit of* $\tilde{\tau}_f$ *is repeated four times in a row. In* $\tilde{\tau}_{f_2}$ *every bit is repeated two times in a row, but this is done twice. In* $\tilde{\tau}_{f_3}$ *every bit is repeated one time, but this is done four times altogether. The minimal primes for each* f_i *are also underlined.*

Now, we are ready to check whether or not Definition 4 holds. To do this, we simply form $\tilde{\tau}_{\tilde{f}}$, where $\tilde{f}$ is defined as $\tilde{f} = f(f_1, \cdots, f_n)$. Since $\tilde{f}$ is a function of $2n-1$ variables, the length of $\tilde{\tau}_{\tilde{f}}$ will be equal to that of the vectors $\tilde{\tau}_{f_i}$. Thus, for every $\tilde{\alpha} \in E^{2n-1}$, we simply evaluate f_1 through f_n at $\tilde{\alpha}$ and then evaluate $\tilde{f}$, treating the values of $f_1, \cdots, f_n$ (for that particular $\tilde{\alpha}$) as a prime in E^n. Let $b(\tilde{\alpha})$ be the integer represented by $\tilde{\alpha}$. Recall that the bits of $\tilde{\tau}_{f_i}$ were arranged in BCD representation and so to evaluate $\tilde{f}$ at some particular $\tilde{\alpha}$, we simply look at the components of $\tilde{\tau}_{f_1}, \cdots, \tilde{\tau}_{f_n}$ at position $b(\tilde{\alpha})$ (from right to left). These components form some prime $\tilde{\beta} \in E^n$ and the value of $\tilde{f}$ at $\tilde{\alpha}$ is simply the component of $\tilde{\tau}_f$ in position $b\left(\tilde{\beta}\right)$. The following outlines this algorithm:

form vectors $\tilde{\tau}_{f_1}, \cdots, \tilde{\tau}_{f_n}$
for $i = 0$ to $2^{2n-1} - 1$

form the vector $\tilde{\beta} = (\tilde{\tau}_{f_1}(i), \cdots, \tilde{\tau}_{f_n}(i))$
$\tilde{\tau}_{\tilde{f}}(i) = \tilde{\tau}_f\left(b\left(\tilde{\beta}\right)\right)$
next i

Let us consider the example above. We would like to find out the value of $\tilde{f}$ at the prime $\tilde{\alpha} = (00101)$. Since $b(\tilde{\alpha}) = 5$, we look at the 5^{th} position of the three vectors $\tilde{\tau}_{f_1}, \tilde{\tau}_{f_2}, \tilde{\tau}_{f_3}$. They have values $0, 1, 1$ at that particular $\tilde{\alpha}$. So, let $\tilde{\beta} = (011)$. To find the value of $\tilde{f}$, we look at position $b\left(\tilde{\beta}\right) = 3$ of $\tilde{\tau}_f$. The value is equal to 1. Of course, using the above procedure for every $\tilde{\alpha} \in E^{2n-1}$, we can generate the whole vector $\tilde{\tau}_{\tilde{f}}$. For this example, the vector $\tilde{\tau}_{\tilde{f}}$ is identical to $\tilde{\tau}_{f_2}$ (easily checked by hand). This implies that the equation in Definition 4 is satisfied and that the function f is idempotent. In general, after forming the vectors $\tilde{\tau}_{f_1}, \cdots, \tilde{\tau}_{f_n}$, checking a function for idempotence simply consists of referencing the original vector $\tilde{\tau}_f$, $2n-1$ times, and comparing $\tilde{\tau}_{\tilde{f}}$ to $\tilde{\tau}_{f_{\frac{n+1}{2}}}$.

4 ENHANCEMENTS TO THE IDEMPOTENCE CHECKING ALGORITHM

There are several ways to increase the efficiency of the algorithm for checking a monotone Boolean function for idempotence. One improvement can be made by considering the functions defined in (1). These functions are essentially dependent on no more than n variables. Therefore, we can immediately conclude that for a function to be idempotent, $\tilde{f}$ must also be essentially dependent on the same variables as $f_{\frac{n+1}{2}}$. In the previous section, we pointed out that the fictitious variables lead to repetitions of the bits in the vectors $\tilde{\tau}_{f_i}$. Moreover, the vector $\tilde{\tau}_{f_k}$ contains 2^{k-1} identical parts. Thus, the vector $\tilde{\tau}_{\tilde{f}}$ must contain $2^{\frac{n+1}{2}-1}$ identical parts, since it must equal $\tilde{\tau}_{f_{\frac{n+1}{2}}}$. This fact allows us to immediately discard functions upon discovering that $\tilde{\tau}_{\tilde{f}}$ does not contain $2^{\frac{n+1}{2}-1}$ identical parts. For example, in the $n = 3$ case, $\tilde{\tau}_{\tilde{f}}$ must have 2 identical parts in order to be idempotent. So, we simply compare the 0^{th} bit to the 16^{th}, then the 1^{st} bit to the 17^{th} and so on.

Another useful improvement is obtained by using duality as well as reverse permutation. Once finding an idempotent monotone Boolean function, we get two other ones "for free," since we know that the dual and the reverse permuted functions must be idempotent as well [4]. Table 1 contains all idempotent monotone Boolean functions of order 5, generated by the above algorithm.

In conclusion, without any knowledge of the structure of idempotent functions in general, the only method for generating them is by first making the entire free distributive lattice available (FDL). Because of the extreme growth rate of $FDL(n)$ as a function of n, it is only possible to generate it (not enumerate) for $n \leq 7$ [5]. Therefore, further research should be conducted with the aim of discovering general structural properties of idempotent monotone Boolean functions.

$x_3x_4x_5 + x_1x_2x_3 + x_2x_3x_4$
$x_3x_5 + x_1x_3 + x_2x_3x_4$
$x_3x_4 + x_1x_2x_4x_5 + x_2x_3$
$x_3x_5 + x_1x_2x_4x_5 + x_1x_3 + x_2x_3x_4$
$x_3x_5 + x_1x_3 + x_2x_3 + x_3x_4$
$x_3x_4 + x_1x_2x_5 + x_1x_4x_5 + x_2x_3$
$x_3x_5 + x_1x_2x_4x_5 + x_1x_3 + x_2x_3 + x_3x_4$
$x_3 + x_1x_2x_4x_5$
$x_3x_5 + x_1x_3 + x_2x_4$
$x_3x_5 + x_1x_2x_5 + x_1x_3 + x_1x_4x_5 + x_2x_3 + x_3x_4$
$x_3 + x_1x_2x_5 + x_1x_4x_5$
$x_3x_5 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4$
$x_3 + x_1x_4 + x_2x_4 + x_2x_5$

Table 1: Idempotent Monotone Boolean Functions of Order 5

References

[1] P.D. Wendt, E.J. Coyle, N.C. Gallagher Jr., "Stack Filters," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 4, pp. 898-911, Aug. 1986

[2] E.J. Coyle, J.H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-36, no. 8, pp.1244-1254, Aug. 1988

[3] A.D. Korshunov, "On The Number of Monotone Boolean Functions," *Problemy Kibernetiki*, 38, pp. 5-108, 1981 (Russian)

[4] I. Shmulevich, E.J. Coyle, "On the Structure of Idempotent Monotone Boolean Functions," *Proceedings of Noblesse Workshop on Non-Linear Model Based Image Analysis*, July 1998, Glasgow, Scotland.

[5] I. Shmulevich, T.M. Sellke, M. Gabbouj, E.J. Coyle, "Stack Filters and Free Distributive Lattices," *Proceedings of 1995 IEEE Workshop on Nonlinear Signal Processing*, Halkidiki, Greece, 1995.