Microprocessors and Microsystems 56 (2018) 193–203

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Enhancing logic synthesis of switching lattices by generalized Shannon decomposition methods

Anna Bernasconi[a], Valentina Ciriani*[,b], Luca Frontini[b], Valentino Liberali[c], Gabriella Trucco[b], Tiziano Villa[d]

[a] Dipartimento di Informatica, Università di Pisa, Italy
[b] Dipartimento di Informatica, Università degli Studi di Milano, Italy
[c] Dipartimento di Fisica Università degli Studi di Milano, Italy
[d] Dipartimento di Informatica, Università degli Studi di Verona, Italy

ARTICLE INFO

Keywords:
Logic synthesis for emerging technologies
Switching lattices
Generalized Shannon decomposition

ABSTRACT

In this paper we propose a novel approach to the synthesis of minimal-sized lattices, based on the decomposition of logic functions. Since the decomposition allows to obtain circuits with a smaller area, our idea is to decompose the Boolean functions according to generalizations of the classical Shannon decomposition, then generate the lattices for each component function, and finally implement the original function by a single composed lattice obtained by gluing together appropriately the lattices of the component functions. In particular we study the two decomposition schemes defining the bounded-level logic networks called P-circuits and EXOR-Projected Sums of Products (EP-SOPs). Experimental results show that about 34% of our benchmarks achieve a smaller area when implemented using the P-circuit decomposition for switching lattices, with an average gain of at least 25%, and about 27% of our benchmarks achieve a smaller area when implemented using the EP-SOP decomposition, with an average gain of at least 22%.

1. Introduction

A switching lattice is a two-dimensional lattice of four-terminal switches linked to the four neighbors of a lattice cell, so that these are either all connected, or disconnected. A Boolean function can be implemented by a lattice associating each four-terminal switch to a Boolean literal, so that if the literal takes the value 1 the corresponding switch is ON and connected to its four neighbors, otherwise it is not connected. The function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges (see Fig. 1 for an example). The synthesis problem on a lattice thus consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size.

The idea of using regular two-dimensional arrays of switches to implement Boolean functions is old and dates back to a seminal paper by Akers in 1972 [1]. Recently, with the advent of a variety of emerging nanoscale technologies based on regular arrays of switches, synthesis methods targeting lattices of multi-terminal switches have found a renewed interest [2–4]. Consider for instance a nanowire array, where each crosspoint is controlled by an input voltage. In this paper, we consider crosspoints that behave like four-terminal switches controlled by an input signal and therefore the proposed nanowire crossbar array can be modeled as a lattice of four-terminal switches. Note that, in general, crossbars can also be modeled by programmable contacts (see for instance [5]). Nanowire crossbar arrays may offer substantial advantages over conventional CMOS when used to implement programmable architectures. Conventional implementations typically employ SRAMs for programming crosspoints; other techniques have been suggested for implementing programmable crosspoints such as bistable switches that form memory cores, molecular switches and solid-electrolyte nanoswitches (see [3] for more details and bibliographic references).

In this paper we show how the cost of implementing a switching lattice could be mitigated by applying Boolean function decomposition techniques in lattice-based implementations. Decomposition of logic functions is a widely studied field in multi-level logic; here we focus on two particular decomposition methods that are based on different generalizations of the classical Shannon decomposition and that give rise to the families of bounded-level logic networks called *P-circuits*

* Corresponding author.
E-mail addresses: anna.bernasconi@unipi.it (A. Bernasconi), valentina.ciriani@unimi.it (V. Ciriani), luca.frontini@unimi.it (L. Frontini), valentino.liberali@unimi.it (V. Liberali), gabriella.trucco@unimi.it (G. Trucco), tiziano.villa@univr.it (T. Villa).

https://doi.org/10.1016/j.micpro.2017.12.003
Received 13 January 2017; Received in revised form 24 November 2017; Accepted 7 December 2017
Available online 11 December 2017
0141-9331/

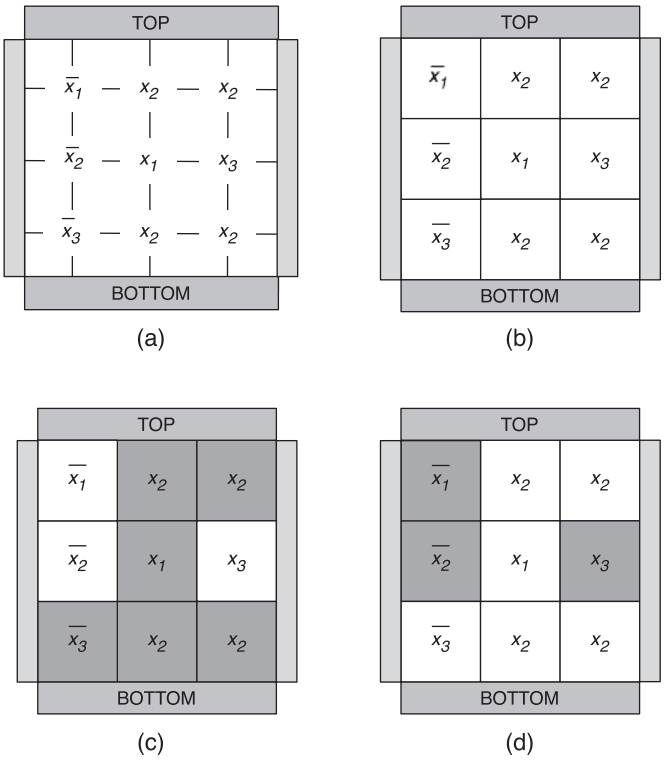

Fig. 1. A four terminal switching network implementing the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ (a); its corresponding lattice form (b); the lattice evaluated on the assignments 1,1,0 (c) and 0, 0, 1 (d), with gray and white squares representing ON and OFF switches, respectively.

[6–9] and *EXOR-Projected Sums of Products* (EP-SOPs) [10–13]. These two methods have been selected, among other known decomposition techniques, mainly because Shannon-based decomposition methods allow to keep the number of logic levels bounded, in addition to the fact that they have been already exploited with good results in CMOS technology.

In the framework of switching lattices synthesis, where the available minimization tools are not yet as developed and mature as those available for CMOS technology, reducing the synthesis of a target Boolean function to the synthesis of smaller functions could represent a very beneficial approach. This expectation has been confirmed by our experimental results, which demonstrate that in about 35% of the analyzed cases the synthesis of switching lattices based on a decomposition of the logic function into smaller sub-functions allows to obtain a smaller area in the final resulting lattice.

This paper is an extended version of the conference paper in [14], where only the decomposition with P-circuits was described, and is organized as follows. Preliminaries on switching lattices are reviewed in Section 2, while P-circuits and EP-SOP forms are described in Section 3. Section 4 shows how the proposed decomposition schemes can be exploited for the synthesis of switching lattices. Section 5 provides the experimental results and Section 6 concludes the paper.

2. Switching lattices

In this section we briefly review some basic notions and results on switching lattices [1,3,4].

A switching lattice is a two-dimensional array of four-terminal switches. The four terminals of the switch link to the four neightbours of a lattice cell, so that these are either all connected (when the switch is ON), or disconnected (when the switch is OFF).

A Boolean function can be implemented by a lattice in terms of connectivity across it:

- Each four-terminal switch is controlled by a Boolean literal;
- each switch may be also labelled with the constant 0, or 1;
- if the literal takes the value 1, the corresponding switch is connected to its four neighbors, else it is not connected;
- the function evaluates to 1 if and only if there exists a connected path between two opposing edges of the lattice, e.g., the top and the bottom edges;
- input assignments that leave the edges unconnected correspond to output 0.

For instance, the 3 × 3 network of switches in Fig. 1 (a) corresponds to the lattice form depicted in Fig. 1 (b), which implements the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$. If we assign the values 1, 1, 0 to the variables x_1, x_2, x_3, respectively, we obtain paths of gray square connecting the top and the bottom edges of the lattices (Fig. 1 (c)), indeed on this assignment f evaluates to 1. On the contrary, the assignment $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, on which f evaluates to 0, does not define any path from the top to the bottom edge (Fig. 1 (d)).

The synthesis problem on a lattice consists in finding an assignment of literals to switches in order to implement a given target function with a lattice of minimal size. The size is measured in terms of the number of switches in the lattice.

A switching lattice can similarly be equipped with left edge to right edge connectivity, so that a single lattice can implement two different functions. This fact is exploited in [2,3] where the authors propose a synthesis method for switching lattices simultaneously implementing a function f according to the connectivity between the top and the bottom plates, and its dual function f^D according to the connectivity between the left and the right plates. Recall that the dual of a Boolean function f depending on n binary variables is the function f^D such that $f(x_1, x_2, ..., x_n) = \overline{f^D}(\overline{x}_1, \overline{x}_2, ..., \overline{x}_n)$. This method produces lattices with a size that grows linearly with the number of products in an irredundant sum of product (SOP) representation of f, and consists of the following steps:

1. Find an irredundant, or a minimal, SOP representation for f and f^D: $SOP(f) = p_1 + p_2 + ... + p_s$ and $SOP(f^D) = q_1 + q_2 + ... + q_r$;
2. form a $r \times s$ switching lattice and assign each product p_j $(1 \le j \le s)$ of $SOP(f)$ to a column and each product q_i $(1 \le i \le r)$ of $SOP(f^D)$ to a row;
3. for all $1 \le i \le r$ and all $1 \le j \le s$, assign to the switch on the lattice site (i, j) one literal which is shared by q_i and p_j (the fact that f and f^D are duals guarantees that such a shared literal exists for all i and j).

This synthesis algorithm thus produces a lattice for f whose size depends on the number of products in the irredundant SOP representations of f and f^D, and it comes with the dual function implemented for free. For instance, the lattice depicted in Fig. 1 has been built according to this algorithm, and it implements both the function $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$ and its dual $f^D = x_1\overline{x}_2x_3 + \overline{x}_1x_2 + x_2\overline{x}_3$.

The time complexity of the algorithm is polynomial in the number of products. However, the method does not always build lattices of minimal size for every target function, since it ties the dimensions of the lattices to the number of products in the SOP forms. In particular this method is not effective for Boolean functions whose duals have a very large number of products. Another reason that could explain the non-minimality of the lattices produced in this way is that the algorithm does not use Boolean constants as input, i.e., each switch in the lattice is always controlled by a Boolean literal.

In [4], the authors proposed a different approach to the synthesis of minimal-sized lattices, which is formulated as a satisfiability problem in quantified Boolean logic and solved by quantified Boolean formula solvers. This method uses the previous algorithm to find an upper bound on the dimensions of the lattice. It then searches for successively better implementations until either an optimal solution is found, or else a preset time limit has been exceeded. Experimental results show how