Krassimir Markov, Vitalii Velychko, Oleksy Voloshin
(editors)

Information Models of Knowledge

I T H E A®
K I E V – S O F I A
2010

Krassimir Markov, Vitalii Velychko, Oleksy Voloshin (ed.)

Information Models of Knowledge

ITHEA®

Kiev, Ukraine – Sofia, Bulgaria, 2010

ISBN 978-954-16-0048-1

First edition

Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA

ITHEA IBS ISC: 19.

This book maintains articles on actual problems of research and application of information technologies, especially the new approaches, models, algorithms and methods fot information modeling of knowledge in: Intelligence metasynthesis and knowledge processing in intelligent systems; Formalisms and methods of knowledge representation; Connectionism and neural nets; System analysis and sintesis; Modelling of the complex artificial systems; Image Processing and Computer Vision; Computer virtual reality; Virtual laboratories for computer-aided design; Decision support systems; Information models of knowledge of and for education; Open social info-educational platforms; Web-based educational information systems; Semantic Web Technologies; Mathematical foundations for information modeling of knowledge; Discrete mathematics; Mathematical methods for research of complex systems.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Ukraine

ISBN 978-954-16-0048-1

C\o Jusautor, Sofia, 2010

MULTIPLE FOLDING OF VLSI REGULAR STRUCTURE VIA BOOLEAN SATISFIABILITY

Liudmila Cheremisinova

***Abstract**: The problem under consideration is to reduce the area of the layout of regular VLSI structures by means of their folding. A novel reformulation of the folding problem as the Boolean satisfiability problem solved with any standard SAT-solver is proposed. The method is developed that allows creating a Boolean equation presenting folding problem in CNF form.*

***Keywords**: design automation, area optimization, VLSI structure folding.*

***ACM Classification Keywords**: B.6.3 [Logic Design]: Design Aids – Optimization; B.7.2 [Integrated circuits]: Design Aids –Layout.*

Introduction

When designing control logic of custom VLSI, regular structured logic is widely used. This logic refers to logic forms that exhibit a high degree of regularity in their layout and interconnections. The widespread used hardware forms for structured design of digital VLSI systems are Programmable Logic Arrays (PLAs), Gate Matrices and, regular structures on the base of MOS-transistors connected in sequence (RMOS-structures) and others [Ullman, 1984]. All these forms have a two-dimensional structure consisting of an array of rows and columns. There are transistors in intersections of some rows and columns. The other special features of the structures under consideration are specified by the technology of VLSI manufacturing. The price paid for the structural regularity is larger chip area because obtained layouts are as a rule sparse: a large percentage of the row-column intersections are not personalized. Several techniques have been proposed for reducing the area required. Two approaches are usually used to reduce the area occupied by array-based structures: logic minimization that provides logic expressions with minimal number of products (and literals) and topological minimization reclaiming unused space.

The proposed paper deals with the problem of optimizing the area of the regular structure layout of regular structure by means of its folding [Hachtel, 1982], [DeMicheli, 1983], [Lecky, 1989], [Cheremisinova, 2004]. Folding is based on merging several columns (and/or rows) of an array-based structure into a single column (row). One special case of folding is simple column (and/or row) folding that involves merging a pair of columns (and/or rows) into a single column (row). The objective of one-dimensional column (row) folding is to find a permutation of the rows (columns) such that the set of columns (rows) could be placed in the minimum number of vertical lines. A generalization of the simple folding is multiple folding [DeMicheli, 1983] when more than two columns may share the same vertical line. This availability supposes rearrangement of different pieces (initial columns) of the same vertical line and thus multiple column (row) folding aims at determining a permutation of the regular structure rows (columns) which allow to implement in each vertical line (horizontal line) of the folded structure a set of initial columns (rows).

The proposed algorithm will be formulated regarding multiple column folding but it is valid for row folding too. Folding a column supposes to split that column into several segments so that several array inputs or outputs may share the same column of the folded two-dimensional structure. A multiple column folding is implemented by reordering the regular structure rows such that all of the rows populated along one column cover physically one segment of a vertical line that does not intersect with segments for other columns of the folding set.

Recent advances in solving Boolean satisfiability problems caused a significant resurgence of the application of satisfiability solvers (SAT-solvers) in different electronic design automation domains. In the last years, great improvements were achieved in both the speed and capacity of SAT-solvers [Eén], [Mahajan, 2005], [Goldberg, 2002], which are now very fast and can handle huge problems. The new efficient SAT-solvers open new possibilities for applying this technology by translating hard design problems to equivalent SAT problems. So the existence of effective SAT-solvers makes it attractive to translate folding problem into Boolean problem solvable by SAT-solvers.

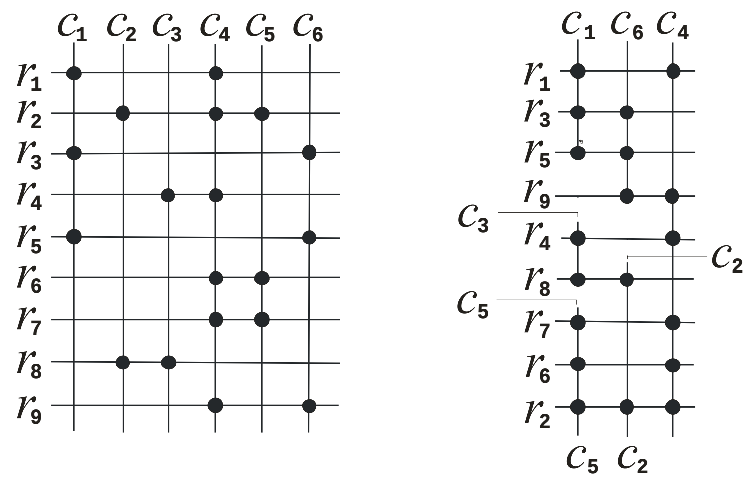
Devadas has been reformulated a routing problem as a Boolean satisfiability problem in his pioneer paper [Devadas, 1989], later the same manner the transformation of the problem of PLA column folding has been done in [Quintana, 1995]. Though the Boolean satisfiability is an NP-hard problem (as well as the folding problem [Luby, 1982]), there are successful, practical attacks on even large satisfiability problems by using Binary Decision Diagrams (BDDs) [Bryant, 1986] and heuristic search [Eén], [Mahajan, 2005], [Goldberg, 2002]. The papers [Devadas, 1989], [Quintana, 1995] attacked their problems using BDD representation of verified Boolean function whose internal structure captures the solved problem. In contrast to these approaches the method presented here turn a folding problem into an instance of a Boolean satisfiability problem solved with any standard SAT-solver. SAT-solvers normally operate on Boolean formulas in Conjunctive Normal Form (CNF), so the method is proposed that allows creating a Boolean function (that presents folding problem) in CNF form.

Boolean SAT formulations are binary in essence. Introduced Boolean variables represent solution alternatives, and Boolean formulas represent constraints imposes by the solved problem. All variable assignments satisfying Boolean formulas are equivalent when solving the satisfiability problem. During the search of SAT solution there is no cost mechanisms to favor one over another. Thus we can only get the answer whether some solution of our problem exists. That is why in [Quintana, 1995] the problem of optimum PLA folding is solved regarding a priori assigned folding size (number of folding pairs or sets). The problem of PLA folding is considered and formulated as a problem of Boolean satisfiability, deriving a Boolean function such that an assignment of variables that satisfy it (if it exists) defines a fold of the given size. Thus, they are forced continuously to reformate the folding problem with decreasing values of folding sizes until a satisfiable problem formulation exists. Such a reformulation of the folding problem based on enumeration of folding size values becomes cumbersome for PLAs of great size, especially for “dense” PLAs.

Unlike that, in the present paper it is suggested to deal with the key problem of array-based structure folding – the problem of ascertaining whether the given folding set (the collections of columns) is implementable. There exist efficient methods of finding the folding sets (for example, in [Cheremisinova, 1999] this task is reduced to clique identification in graph of pairwise compatibility between columns) but the methods of examining the implementability of a collection of folding sets (they are observed in [Lecky, 1989]) are complicated and they are formulated concerning ordered folding collections only. Hence, in the paper we focus upon the task of reformulation the folding problem as SAT problem and solve it in more general statement. A collection of unordered column folding lists is dealt with, and the problem is reformulated as follows: it is necessary 1) to establish whether there exists ordering of elements within folding lists such that the collection of the obtained ordered column folding lists is implementable, and 2) to find the ordering of rows of the array-based structure induced by the collection of ordered column folding lists. The results described in the paper are generalization of ones proposed in [Cheremisinova, 2010] where the case of simple folding is considered and so the case of folding pairs is dealt with.

The problem of a regular structure folding

The combinational PLA is an example of an array-based structure. The overall combinational PLA is a standard two level NOR-NOR structure. Its vertical lines are assigned with the input variables (and their complements) and output variables. Inputs run vertically through the first part of a PLA matrix, called as the PLA AND plane. It generates signals on its rows, which are used as inputs to the second part of a PLA matrix called as the PLA OR plane. In Figure 1 an example of an array-based structure (that can be viewed as PLA AND plane) and its folded form are shown. Here, a dot means placing a transistor on crosspoint of vertical and horizontal lines.



c_1 c_2 c_3 c_4 c_5 c_6
r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9
c_1 c_6 c_4
r_1 r_3 r_5 r_9 r_4 r_8 r_7 r_6 r_2
c_3 c_5 c_2
c_5 c_2

Figure 1 An example of array-based structure

Before we can give a mathematically tractable definition of PLA folding problem and its solution let give some other definitions. Either AND plane or both AND and OR planes together can be described in symbolic form by a Boolean matrix ***B*** having sets $C(\boldsymbol{B})$ and $R(\boldsymbol{B})$ of its columns and rows. 1 in the position *i,j* of the matrix ***B*** means there is an appropriate crosspoint (transistor) between the *i*-row and the *j*-column in the matrix. Each column c_i of matrix ***B*** implies the set $R(c_i)$ of rows with a nonzero entry in this column: $r_i \in R(c_i) \leftrightarrow b^i_j = 1$. And let $C(r_i)$ be a set of columns with a nonzero entry in the *i*-th row of ***B***.

Any two columns c_i and c_j are *disjoint* if $R(c_i) \cap R(c_j) = \varnothing$. Two disjoint columns both do not have transistors on any particular row of the array-based structure. If there are no restrictions on the folding type two disjoint columns are compatible and can be folded together, so they form a column *folding pair*.

A column folding list $l^c_k = (c_{k1}, c_{k2},\dots, c_{km})$ is a set of disjoint (in pairs) columns c_{ij}, it is unordered in general case. An ordered column folding list $l^{co}_k = <c_{k1}, c_{k2},\dots, c_{km}>$ is a column folding list l^c_k whose elements c_{ki} are ordered. An ordered column folding list (OCFL) l^{co}_k of cardinality two is an ordered column folding pair. Any OCFL $l^{co}_k = <c_{k1}, c_{k2},\dots, c_{km}>$ can be actually implemented in the same vertical line of array-based structure moving c_{k1} above c_{k2}, c_{k2} above c_{k3}, an so on, $c_{k,m-1}$ above c_{km}. So OCFL l^{co}_k results to permutation on the set of rows: $R(c_{k1}) > R(c_{k2})$ – the rows of $R(c_{k1})$ are all above those of $R(c_{k2})$, $R(c_{k2}) > R(c_{k2})$ – $R(c_{k2})$ are all above those of $R(c_{k3})$, and so on, $R(c_{k,m-1}) > R(c_{k,m})$ – the rows of $R(c_{k,m-1})$ are all above those of $R(c_{km})$ inducing the relation on row set $R(\boldsymbol{B})$:

$$P^r(l^{co}_k) = \bigcup_{i,j} (R(c_{ki}) \times R(c_{kj})), \quad \text{i.e.} \quad P^r(l^{co}_k) = \{ r_p \times r_q / r_p \in R(c_{ki}), r_q \in R(c_{kj}), i < j \}.$$

This relation is partial because it is irreflexive, asymmetric and transitive by its definition. Any unordered column folding list (CFL) $l^c_k = (c_{k1}, c_{k2},\dots, c_{km})$ induces up to $m!$ possible OCFLs $l^{co}_{kl} = <c_{k,l1}, c_{k,l2},\dots, c_{k,lm}>$ begot by different permutations of columns in l^c_k. Thus CFP l^c_k induces one of $m!$ possible partial relations on the row set $R(\boldsymbol{B})$. For example, the mentioned array-based structure (Figure 1) allows the CFL $l^c = (c_1, c_3, c_5)$ that induces six OCFLs:

$$l^{co}_1 = <c_1, c_3, c_5>;\ l^{co}_2 = <c_1, c_5, c_3>;\ l^{co}_3 = <c_3, c_1, c_5>;\ l^{co}_4 = <c_3, c_5, c_1>;\ l^{oc}_5 = <c_5, c_1, c_3>;\ l^{co}_6 = <c_5, c_3, c_1>. \quad (1)$$

and correspondingly six partial row orderings, the first of them is as follows:

$$\{r_1, r_3, r_5\} > \{r_4, r_8\};\ \{r_4, r_8\} > \{r_2, r_6, r_7\}.$$

A column folding set (CFS) $L^c_k = \{l^c_{k1}, l^c_{k2},\dots, l^c_{kn}\}$ is a set of disjoint column folding lists (ordered L^{co}_k or unordered L^c_k). The number of columns entering into all CFLs $l^c_{ki} \in L^c_k$ is called as the size of CFS L^c_k. An ordered column folding set (OCFS) $L^{co}_k = \{l^{co}_{k1}, l^{co}_{k2},\dots, l^{co}_{kn}\}$ induces a set of ordering relations among the rows that is the union of ordering relations induced by OCFLs l^{co}_k belonging to the OCFS $P^r(l^{co}_k)$:

$$P^r(L^{co}_k) = \bigcup_{i=1}^{n} (P^r(l^{co}_{ki})).$$

This relation $P^r(L^{co}_k)$ is irreflexive, asymmetric but not transitive in general case. The transitive closure $R^t(P^r)$ of $P^r(L^{co}_k)$ is irreflexive, transitive but can be not asymmetric.

It is proven [DeMicheli, 1983] that an OCFS L^{co}_k is *implementable* topologically (by a folded array-based structure) if the transitive closure $R^t(P^r)$ of the relation $P^r(L^{co}_k)$ is a partial ordering on $R(\boldsymbol{B})$, that is $R^t(P^r)$ is asymmetric. In other words, an OCFS L^{co}_k is implementable topologically if there exist linear order of the rows $R(\boldsymbol{B})$ extending the row ordering $P^r(L^{co}_k)$. For example, for the considered array-based structure (Figure 1) there are four CFLs:

$$l^c_1 = (c_1, c_3, c_5);\ l^c_2 = (c_1, c_2);\ l^c_3 = (c_2, c_6);\ l^c_4 = (c_3, c_6)$$

two of them are disjoint (l^c_1 and l^c_3) and may constitute CFS of maximal size: $L^c = \{(c_1, c_3, c_5), (c_2, c_6)\}$, it induces 12 OCFS L^{co}_k (l^c_1 allows 6 different orderings (1) and l^c_2 allows 2 different orderings: $<c_2, c_6>$ and $<c_6, c_2>$). One of the induced OCFS $L^{co}_1 = \{<c_1, c_3, c_5>, <c_2, c_6>\}$ generates the relation

$$P_1^r(L^{co}_1) = R(c_1) \times R(c_3) \cup R(c_3) \times R(c_5) \cup R(c_2) \times R(c_6) =$$

$$= \{r_1, r_3, r_5\} \times \{r_4, r_8\} \cup \{r_4, r_8\} \times \{r_2, r_6, r_7\} \cup \{r_2, r_8\} \times \{r_3, r_5, r_9\}.$$

Its transitive closure $R^t(P_1^r)$ contains conflict pairs (r_3, r_2) and (r_2, r_3); so $R^t(P_1^r)$ is not a partial ordering on $R(\boldsymbol{B})$ (because it is not asymmetric) and OCFS L^{co}_1 is not implementable. But the other induced OCFS $L^{co}_2 = \{<c_1, c_3, c_5>, <c_6, c_2>\}$ generates the relation

$$P_2^r(L^{co}_1) = R(c_1) \times R(c_3) \cup R(c_3) \times R(c_5) \cup R(c_6) \times R(c_2) =$$

$$= \{r_1, r_3, r_5\} \times \{r_4, r_8\} \cup \{r_4, r_8\} \times \{r_2, r_6, r_7\} \cup \{r_3, r_5, r_9\} \times \{r_2, r_8\},$$

whose transitive closure $R^t(P_2^r)$ is asymmetric, so it is partial relation and OCFS L^{co}_2 is implementable.

An implementable OCFS L^{co}_k specifies the structure of the folded array, and its size is referred to as the size of the folding: the number of OCFLs in L^{co}_k corresponds to the number of columns that will replace $\sum_{i=1}^{n} |l^{co}_{ki}|$ columns of the initial array-based structure. For example, the size of implementable OCFS $L^{co}_2 = \{<c_1, c_3, c_5>, <c_6, c_2>\}$ is equal five and two columns of the folded regular structure replace five columns of the initial structure.

So the formal statement of optimal folding problem is as follows: given a Boolean matrix representing array-based structure, find an implementable ordered folding set of maximum size.

A column folding set (consisting of unordered folding lists) $L^c_k = \{l^c_{k1}, l^c_{k2},\ldots, l^c_{kn}\}$ is *implementable* if there exists an implementable OCFS L^{co}_k which OCFLs l^{co}_{ki} are got by ordering CFLs $l^c_{ki} \in L^c_k$.

OCFS implementability checking via Boolean satisfiability formulation

Further the problem is stated as follows: given a column folding set (ordered) $L^{co} = \{l^{co}_1, l^{co}_2,\ldots, l^{co}_n\}$, it is necessary to verify whether it is implementable. If it is, the corresponding partial order relation on the regular structure rows will be found. OCFS L^{co} can be obtained by one of the known methods (for example, by the method from [DeMicheli, 1983] or from [Cheremisinova, 1999]) where these task is reduced to maximum cliques identification in the graph of compatibility relation between array columns).

As it follows from above discussion an OCFL $l^{co}_k = <c_{k1}, c_{k2},\ldots, c_{km}>$ induce the following partial ordering:

$$(R(c_{k1}) > R(c_{k2})) \cap (R(c_{k2}) > R(c_{k3})) \cap \ldots \cap (R(c_{k,m-1}) > R(c_{km})) \qquad (2)$$

The task of checking whether OCFS L^{co} is implementable is reduced as shown below to checking whether there exists such an ordering of rows that provides fulfillment of the condition (2) for all OCFL $l^{co}_k \in L^{co}$. Let us show how to reduce the task to solving logic equation.

Let form a set R_P of rows that enter into at least one of the sets $R(c_{ij})$ ($c_{ij} \in l^{co}_i$, $l^{co}_i \in L^{co}$), $i \in \{1,2,\ldots,n\}$, $j \in \{1,2,\ldots,i_m\}$. Positions of rows of the set $R(\boldsymbol{B}) \setminus R_P$ in the folded array-based structure have no influence on the possibility of the structure folding regarding column folding lists from L^{co}. Therefore we may consider further only rows of R_P. Moreover, we do not need to consider the rows each of which is a member of the only $R(c_{kj})$ ($c_{kj} \in l^{co}_k$, $l^{co}_k \in L^{co}$), they may be excluded from R_P because their placing is fixed only relative to the rows of a single set $R(c_{ki})$ and has no conflicts regarding placing rows from other sets $R(c_{sq})$ ($s \neq k$). In this way we can reduce the number of variables and terms in a formed equation.

To determine ordering of $|R_P|$ rows we encode each of the rows of the set R_P in $q = \lceil \log_2 |R_P| \rceil$ Boolean variables $x_1, x_2, \ldots, x_q$ (where $\lceil t \rceil$ is the least integer that is not less than the value of t). The value of the code $x^i_1 x^i_2 \ldots x^i_q$ for the row r_i specifies its serial number that shows in which physical horizontal line the row r_i will be in the folded array-based structure. Thus the conditions implied by orderings $r_i > r_j$ or $r_i < r_j$ can be represented by the following functions:

$$f_{ij}^{>} = (x^i_q x^i_{q-1} \ldots x^i_1) > (x^j_q x^j_{q-1} \ldots x^j_1) \quad \text{or} \quad f_{ij}^{<} = (x^i_q x^i_{q-1} \ldots x^i_1) < (x^j_q x^j_{q-1} \ldots x^j_1). \tag{3}$$

So the restriction (2), induced by an OCFL $l^{co}_k = <c_{k1}, c_{k2}, \ldots, c_{km}>$ and imposed on the order of array rows, takes the following form:

$$\bigcap_{l=1}^{m-1} \Big(\bigcap_{p=l+1}^{m} \Big(\bigcap_{r_i \in R(c_{kl})} \Big(\bigcap_{r_j \in R(c_{kp})} f_{ij}^{>} \Big)\Big)\Big). \tag{4}$$

An ordered folding set $L^{co} = \{l^{co}_1, l^{co}_2, \ldots, l^{co}_n\}$ is implementable if the following equation is satisfiable:

$$\bigcap_{k=1}^{n} \Big(\bigcap_{l=1}^{m-1} \Big(\bigcap_{p=l+1}^{m} \Big(\bigcap_{r_i \in R(c_{kl})} \Big(\bigcap_{r_j \in R(c_{kp})} f_{ij}^{>} \Big)\Big)\Big)\Big) = 1 \tag{5}$$

If the equation (5) has an assignment to the variables $x^i_l \rightarrow \{0,1\}$ that makes it to equal 1 than the considered OCFS L^{co} is implementable. And the satisfying assignment specifies the serial numbers of the rows of R_P. Otherwise, when no satisfying assignment for the equation (5) exists, the considered OCFS L^{co} is not implementable.

For illustration the technique consider OCFS $L^{co}_2 = \{<c_1, c_3, c_5>, <c_6, c_2>\}$ from the example above. As $R(c_1) = \{r_1, r_3, r_5\}$; $R(c_3) = \{r_4, r_8\}$; $R(c_5) = \{r_2, r_6, r_7\}$; $R(c_2) = \{r_2, r_8\}$; $R(c_6) = \{r_3, r_5, r_9\}$ we have $R_P = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$ that can be reduced to $R_P = \{r_2, r_3, r_5, r_8\}$ because some rows enter into the only set $R(c_i)$. Thus we have $R'(c_1) = \{r_3, r_5\}$; $R'(c_3) = \{r_8\}$; $R'(c_5) = \{r_2\}$; $R'(c_2) = \{r_2, r_8\}$; $R'(c_6) = \{r_3, r_5\}$. Four rows from R_P are encoded using two Boolean variables x_2 and x_1, their values for a row r_i are x^i_2 and x^i_1. After encoding equation (5) takes the form:

$$\begin{aligned}(f_{3,8}^{>} \wedge f_{5,8}^{>} \wedge f_{8,2}^{>}) \wedge (f_{3,2}^{>} \wedge f_{3,8}^{>} \wedge f_{5,2}^{>} \wedge f_{5,8}^{>}) = f_{3,8}^{>} \wedge f_{5,8}^{>} \wedge f_{8,2}^{>} \wedge f_{3,2}^{>} \wedge f_{5,2}^{>} = \\ = (x^3_2 x^3_1 > x^8_2 x^8_1) \wedge (x^5_2 x^5_1 > x^8_2 x^8_1) \wedge (x^8_2 x^8_1 > x^2_2 x^2_1) \wedge (x^3_2 x^3_1 > x^2_2 x^2_1) \wedge (x^5_2 x^5_1 > x^2_2 x^2_1)\end{aligned} \tag{6}$$

CFS implementability checking via Boolean satisfiability formulation

Further we consider that unordered column folding set $L^c = \{l^c_1, l^c_2, \ldots, l^c_n\}$ is given where $l^c_k = (c_{k1}, c_{k2}, \ldots, c_{km})$. The problem is stated as follows: verify whether CFS L^c is implementable and, if it is, the appropriate implementable OCFS should be found and the corresponding partial order relation on the array rows. This case differs from that considered above only by what to check CFS implementability we should look over different orders of columns within all CFLs $l^c_i \in L^c$, i.e. we should generate all permutations on a set of columns for each folding list $l^c_i \in L^c$.

That follows from the fact that a CFL $l^c_k = (c_{k1}, c_{k2},\ldots, c_{km})$ induces the set $P(k_m)$ of size $k_m!$ of possible OCFLs $l^{co}_{ki} = (c_{k,i1}, c_{k,i2},\ldots, c_{k,im})$ ($m = 1, 2,\ldots, k_m!$) corresponding to different permutations of columns in l^c_k. Thus CFL l^c_k induces the following partial relations on the row set $R(\boldsymbol{B})$ that is the extension of (2):

$$\bigcup_{l^{\text{co}}_{\text{k,s}} \in P(k_m)} ((R(c_{k,s1}) > R(c_{k,s2})) \cap (R(c_{k,s2}) > R(c_{k,s3})) \cap \ \ldots \cap (R(c_{k,sm-1}) > R(c_{k,sm}))). \tag{7}$$

After encoding rows of the set R_P formed the same manner as in previous section the condition for CFL l^c_k takes the following form that is more complex than that of (4) for OCFL l^{co}_k:

$$\bigcup_{l^{\text{co}}_{\text{k,s}} \in P(k_m)} (\bigcap_{l=1}^{m-1} (\bigcap_{p=l+1}^{m} (\bigcap_{r_i \in R(c_{k,sl})} (\bigcap_{r_j \in R(c_{k,sp})} f_{ij}^{>})))). \tag{8}$$

With (8) in mind, by analogy with (5), an unordered folding set $L^c = \{l^c_1, l^c_2,\ldots, l^c_n\}$ is implementable if the following equation is satisfiable:

$$\bigcap_{k=1}^{n} (\bigcup_{l^{\text{co}}_{\text{k,s}} \in P(k_m)} (\bigcap_{l=1}^{m-1} (\bigcap_{p=l+1}^{m} (\bigcap_{r_i \in R(c_{k,sl})} (\bigcap_{r_j \in R(c_{k,sp})} f_{ij}^{>}))))) = 1 \tag{9}$$

If the equation (9) has an assignment to the variables $x^i_l \rightarrow \{0,1\}$ that makes it to equal 1 than the considered CFS L^c is implementable. And the satisfying assignment specifies the serial numbers of the rows of R_P, which show, in its turn, how the CFLs $l^c_k = (c_{k1}, c_{k2},\ldots, c_{km})$ should be ordered. Otherwise, when no satisfying assignment for the equation (9) exists, the considered CFS L^c is not implementable.

To illustrate the technique let consider CFS $L^c = \{(c_1, c_3, c_5), (c_2,c_6)\}$ from the above example. Thus we have $R_P = \{r_2, r_3, r_5, r_8\}$, $R'(c_1) = \{r_3, r_5\}$; $R'(c_3) = \{r_8\}$; $R'(c_5) = \{r_2\}$; $R'(c_2) = \{r_2, r_8\}$; $R'(c_6) = \{r_3, r_5\}$, CFS L^c induces 12 OCFS L^{co}_k. After encoding four rows from R_P the equation (9) takes the form:

$$\begin{gathered}((f_{3,8}^{>} \wedge f_{5,8}^{>} \wedge f_{8,2}^{>}) \vee (f_{3,2}^{>} \wedge f_{5,2}^{>} \wedge f_{2,8}^{>}) \vee (f_{8,3}^{>} \wedge f_{8,5}^{>} \wedge f_{3,2}^{>} \wedge f_{5,2}^{>}) \vee (f_{8,2}^{>} \wedge f_{2,3}^{>} \wedge f_{2,5}^{>}) \vee \\ \vee (f_{2,3}^{>} \wedge f_{2,5}^{>} \wedge f_{3,8}^{>} \wedge f_{5,8}^{>}) \vee (f_{2,8}^{>} \wedge f_{8,3}^{>} \wedge f_{8,5}^{>})) \wedge ((f_{2,3}^{>} \wedge f_{2,5}^{>} \wedge f_{8,3}^{>} \wedge f_{8,5}^{>}) \vee (f_{3,2}^{>} \wedge f_{3,8}^{>} \wedge f_{5,2}^{>} \wedge f_{5,8}^{>}))\end{gathered} \tag{10}$$

Transforming the equation for OCFS implementability checking into CNF form

When testing CFS whether it is implementable we need to ascertain SAT and to find a solution (if it exists), that gives OCFS and array row ordering. Effective SAT-solvers have been developed to perform the search for a satisfying solution (or prove that it does not exist). The best of SAT-solvers that exist today ([Eén], [Mahajan, 2005], [Goldberg, 2002] and others) are capable of handling millions of logical clauses and thousands of variables. SAT-solvers usually require their input to be in CNF form, thus it is necessary to convert the left part of the equations (5) and (9) into CNF form. CNF formula is expressed as the product of clauses, where each clause is the sum of literals. A literal is either variable or its negation.

To convert (5) into CNF form it is enough to get CNF form for the function $f_{ij}^{>} = (x^i_q x^i_{q\text{-}1} \ldots x^i_1) > (x^j_q x^j_{q\text{-}1} \ldots x^j_1)$ (3) only. For example for $q = 1$ (only one encoding variable) we have two-argument function $f_{ij}^{>} = (x^i_1) > (x^j_1)$, its minimum CNF formula is written over two variables and has two clauses:

$$C(f_{ij}^{>}) = x^i_1 \ \bar{x}^j_1.$$

For $q = 2$ we have four-argument function $f_{ij}^{>} = (x^i_2 x^i_1) > (x^j_2 x^j_1)$, its minimum CNF formula is written over four variables and has five clauses:

$$C(f_{ij}^{>}) = (x^i_2 \vee x^i_1) \wedge (\ \bar{x}^j_2 \vee x^i_1) \wedge (x^i_2 \vee \ \bar{x}^j_1) \wedge (\ \bar{x}^j_2 \vee \ \bar{x}^j_1) \wedge (x^i_2 \vee \ \bar{x}^j_2).$$

In general case CNF formula for $q = t$ depends on $2t$ arguments. It is not difficult to show that the CNF formula for $q = t + 1$ can be recursively received from the CNF formula for $q = t$ by transforming each r-th clause d^t_r of the first CNF into two clauses $(x^i_{t+1} \vee d^t_r)$, $(\ \bar{x}^j_{t+1} \vee d^t_r)$ and adding into the resulting CNF one more additional clause

($x^i_{t+1} \vee \bar{x}^i_{t+1}$). So the minimum CNF for $q = t + 1$ will consist of $n_{t+1} = 2n_t + 1$ clauses, where n_t is the number of clauses in CNF for $q = t$.

After substituting CNFs for the function $f_{ij}^{>}$ we transform (5) into the following Boolean equation whose left part is CNF:

$$\bigcap_{k=1}^{n} \Big(\bigcap_{l=1}^{m-1} \Big(\bigcap_{p=l+1}^{m} \Big(\bigcap_{r_i \in R(c_{kl})} \Big(\bigcap_{r_j \in R(c_{kp})} C(f_{ij}^{>})\Big)\Big)\Big)\Big) = 1 \qquad (11)$$

Thus the task of checking whether an ordered column folding set is implementable is reduced to the task of checking whether CNF formula (11) is satisfiable.

To make clear the above transformations, it is useful to proceed with transformations of the formula (6) for OCFS $L^{co}_2 = \{<c_1, c_3, c_5>,<c_6,c_2>\}$ implementability checking into the following equation in CNF form:

$$\begin{aligned} f_{3,8}^{>} \wedge f_{5,8}^{>} \wedge f_{8,2}^{>} \wedge f_{3,2}^{>} \wedge f_{5,2}^{>} = {} & ((x^3_2 \vee x^3_1) \wedge (\bar{x}^8_2 \vee x^3_1) \wedge (x^3_2 \vee \bar{x}^8_1) \wedge (\bar{x}^8_2 \vee \bar{x}^8_1) \wedge (x^3_2 \vee \bar{x}^8_2)) \wedge \\ & \wedge ((x^5_2 \vee x^5_1) \wedge (\bar{x}^8_2 \vee x^5_1) \wedge (x^5_2 \vee \bar{x}^8_1) \wedge (\bar{x}^8_2 \vee \bar{x}^8_1) \wedge (x^5_2 \vee \bar{x}^8_2)) \wedge \\ & \wedge ((x^8_2 \vee x^8_1) \wedge (\bar{x}^2_2 \vee x^8_1) \wedge (x^8_2 \vee \bar{x}^2_1) \wedge (\bar{x}^2_2 \vee \bar{x}^2_1) \wedge (x^8_2 \vee \bar{x}^2_2)) \wedge \\ & \wedge ((x^3_2 \vee x^3_1) \wedge (\bar{x}^2_2 \vee x^3_1) \wedge (x^3_2 \vee \bar{x}^2_1) \wedge (\bar{x}^2_2 \vee \bar{x}^2_1) \wedge (x^3_2 \vee \bar{x}^2_2)) \wedge \\ & \wedge ((x^5_2 \vee x^5_1) \wedge (\bar{x}^2_2 \vee x^5_1) \wedge (x^5_2 \vee \bar{x}^2_1) \wedge (\bar{x}^2_2 \vee \bar{x}^2_1) \wedge (x^5_2 \vee \bar{x}^2_2)) = 1. \end{aligned} \qquad (12)$$

Variable assignment satisfying the CNF is $x^3_2 = x^3_1 = 1$; $x^5_2 = 1$; $x^5_1 = 0$; $x^8_2 = 0$; $x^8_1 = 0$; $x^2_2 = x^2_1 = 0$. So the rows of $R_P = \{r_3, r_5, r_8, r_2\}$ are encoded as r_3 – 11, r_5 – 10, r_8 – 01, r_2 – 00. The solution specifies partial row ordering r_3, r_5, r_8, r_2 and linear ordering that is, for example, r_1, r_3, r_5, r_9 r_4, r_8, r_7, r_6, r_2, which leads to the folded array-based structure shown in Figure 1.

Transforming the equation for CFS implementability checking into CNF form

For more general case of unordered column folding set, after substituting CNFs for the functions $f_{ij}^{>}$ into (9) we construct following Boolean equation whose left part is not CNF generally speaking:

$$\bigcap_{k=1}^{n} \Big(\bigcup_{l^{co}_{k,s} \in P(k_m)} C^{>}_{k,s}\Big) = 1 \qquad (13)$$

where $C^{>}_{k,s}$ is the following CNF formula:

$$C^{>}_{k,s} = \bigcap_{l=1}^{m-1} \Big(\bigcap_{p=l+1}^{m} \Big(\bigcap_{r_i \in R(c_{k,sl})} \Big(\bigcap_{r_j \in R(c_{k,sp})} C(f_{ij}^{>})\Big)\Big)\Big). \qquad (14)$$

Boolean function on the left part of the equation (13) is almost CNF but not CNF in general case. To solve difficulty arising in transformation of Boolean formula $\Big(\bigcup_{l^{co}_{k,s} \in P(k_m)} C^{>}_{k,s}\Big)$ to a CNF form we suggest to convert it by encoding $k_m!$ CNFs $C^{>}_{k,s}$ using unary codes, for example. $k_m!$ additional coding Boolean variables $z_{k,s}$ are introduced for encoding $k_m!$ CNFs $C^{>}_{k,s}$. To choose between $k_m!$ CNFs $C^{>}_{k,s}$ we can replace $\Big(\bigcup_{l^{co}_{k,s} \in P(k_m)} C^{>}_{k,s}\Big)$ (from (13) with:

$$\Big(\bigcup_{l^{co}_{k,s} \in P(k_m)} (C^{>}_{k,s} \vee z_{k,s})\Big) \wedge \Big(\bigcup_{s=1}^{k_m!} \bar{z}_{k,s}\Big). \qquad (15)$$

Here $(C^{>}_{k,s} \vee z_{k,s})$ can be considered as CNF that is simply deduced from $C^{>}_{k,s}$ with the help of the distributive law (of Boolean algebra) of the conjunction concerning the disjunction: encoding variable $z_{k,s}$ is added into each clause $d_m \in C^{>}_{k,s}$, that is $d_m \vee z_{k,s}{}^{>}$. To code all the CFLs $l^c_k \in L^c$ we should introduce $\sum_{k=1}^{n} (k_m!)$ encoding variables $z_{k,s}$.

Thus, from the formulas (11), (13) – (15) obtained previously we construct CNF formula solving the key problem of array-based structure folding:

$$\bigcap_{k=1}^{n} \Big(\Big(\bigcup_{l^{\text{co}}_{\text{k,s}} \in P(k_m)} (C^{>}_{k,s} \vee z_{k,s})\Big) \wedge \Big(\bigcup_{s=1}^{k_m!} \bar{z}_{k,s}\Big)\Big) = 1.$$

To make clear the above transformations, it is useful to proceed with transformations of the formula (10) for CFS $L^c = \{(c_1, c_3, c_5),(c_2,c_6)\}$ implementability checking into the following equation in CNF form:

$$((f_{3,8}{}^{>} \wedge f_{5,8}{}^{>} \wedge f_{8,2}{}^{>}) \vee \ldots \vee (f_{2,8}{}^{>} \wedge f_{8,3}{}^{>} \wedge f_{8,5}{}^{>})) \wedge ((f_{2,3}{}^{>} \wedge f_{2,5}{}^{>} \wedge f_{8,3}{}^{>} \wedge f_{8,5}{}^{>}) \vee \ldots) =$$
$$= ((x^3{}_2 \vee x^3{}_1 \vee z^1{}_1) \wedge (\bar{x}^8{}_2 \vee x^3{}_1 \vee z^1{}_1) \wedge (x^3{}_2 \vee \bar{x}^8{}_1 \vee z^1{}_1) \wedge (\bar{x}^8{}_2 \vee \bar{x}^8{}_1 \vee z^1{}_1) \wedge (x^3{}_2 \vee \bar{x}^8{}_2 \vee z^1{}_1)) \wedge$$
$$\wedge ((x^5{}_2 \vee x^5{}_1 \vee z^1{}_1) \wedge (\bar{x}^8{}_2 \vee x^5{}_1 \vee z^1{}_1) \wedge (x^5{}_2 \vee \bar{x}^8{}_1 \vee z^1{}_1) \wedge (\bar{x}^8{}_2 \vee \bar{x}^8{}_1 \vee z^1{}_1) \wedge (x^5{}_2 \vee \bar{x}^8{}_2 \vee z^1{}_1)) \wedge$$
$$\wedge ((x^8{}_2 \vee x^8{}_1 \vee z^1{}_1) \wedge (\bar{x}^2{}_2 \vee x^8{}_1 \vee z^1{}_1) \wedge (x^8{}_2 \vee \bar{x}^2{}_1 \vee z^1{}_1) \wedge (\bar{x}^2{}_2 \vee \bar{x}^2{}_1 \vee z^1{}_1) \wedge (x^8{}_2 \vee \bar{x}^2{}_2 \vee z^1{}_1)) \wedge \ldots. \wedge$$
$$\wedge ((x^2{}_2 \vee x^2{}_1 \vee z^6{}_1) \wedge (\bar{x}^8{}_2 \vee x^2{}_1 \vee z^6{}_1) \wedge (x^2{}_2 \vee \bar{x}^8{}_1 \vee z^6{}_1) \wedge (\bar{x}^8{}_2 \vee \bar{x}^8{}_1 \vee z^6{}_1) \wedge (x^2{}_2 \vee \bar{x}^8{}_2 \vee z^6{}_1)) \wedge$$
$$\wedge ((x^8{}_2 \vee x^8{}_1 \vee z^6{}_1) \wedge (\bar{x}^3{}_2 \vee x^8{}_1 \vee z^6{}_1) \wedge (x^8{}_2 \vee \bar{x}^3{}_1 \vee z^6{}_1) \wedge (\bar{x}^3{}_2 \vee \bar{x}^3{}_1 \vee z^6{}_1) \wedge (x^8{}_2 \vee \bar{x}^3{}_2 \vee z^6{}_1)) \wedge$$
$$\wedge ((x^8{}_2 \vee x^8{}_1 \vee z^6{}_1) \wedge (\bar{x}^5{}_2 \vee x^8{}_1 \vee z^6{}_1) \wedge (x^8{}_2 \vee \bar{x}^5{}_1 \vee z^6{}_1) \wedge (\bar{x}^5{}_2 \vee \bar{x}^5{}_1 \vee z^6{}_1) \wedge (x^8{}_2 \vee \bar{x}^5{}_2 \vee z^6{}_1)) \wedge$$
$$\wedge (\bar{z}^1{}_1 \vee \bar{z}^2{}_1 \vee \bar{z}^3{}_1 \vee \bar{z}^4{}_1 \vee \bar{z}^5{}_1 \vee \bar{z}^6{}_1) \wedge$$
$$\wedge ((x^2{}_2 \vee x^2{}_1 \vee z^1{}_2) \wedge (\bar{x}^3{}_2 \vee x^2{}_1 \vee z^1{}_2) \wedge (x^2{}_2 \vee \bar{x}^3{}_1 \vee z^1{}_2) \wedge (\bar{x}^3{}_2 \vee \bar{x}^3{}_1 \vee z^1{}_2) \wedge (x^2{}_2 \vee \bar{x}^3{}_2 \vee z^1{}_2)) \wedge \ldots \wedge$$
$$\wedge (\bar{z}^1{}_2 \vee \bar{z}^2{}_2) = 1.$$

One of the variable assignments satisfying the CNF is the same as for CNF (12). It allows to encode rows of $R_P = \{r_3, r_5, r_8, r_2\}$ as r_3 – 11, r_5 – 10, r_8 – 01, r_2 – 00 and to fold array-based structure as shown in Figure 1.

Conclusion

In this paper a novel reformulation of the multiple folding problem as the Boolean satisfiability problem solved with any SAT-solver was developed. To be exact the task of checking whether the given set of folding sets (or pairs in the case of simple folding) is implementable is considered. The distinctive features of the proposed method are as follows:

1) it can deal with a set of unordered column folding lists:

2) it orders column folding lists such a manner the resulting set of ordered folding lists to be implementable (if it is possible);

3) it specifies the ordering of rows of the regular structure induced by the found set of ordered column folding lists (if it exists).

Acknowledgement

The research was supported by the Fond of Fundamental Researches of Belarus (Project Ф09К–025).

Bibliography

[Ullman, 1984] Jeffrey D. Ullman Computational aspects of VLSI. Rockville, Md.: Computer Science Press, 1984, 495 p.

[Hachtel, 1982] G.D. Hachtel, A.R. Newton and A.L. Sangiovanni-Vincentelli. An Algorithm for optimal PLA Folding. In: IEEE Trans. Computer-Aided Design of Integrated Circuit Syst., 1982, vol. CAD–1, No 2, pp. 63–77.

[DeMicheli, 1983] G. DeMicheli and A. Sangiovanni-Vincentelli. A. Multiple Constrained Folding of Programmable Logic Arrays: Theory and Applications. In: IEEE Trans. Computer-Aided Design, 1983, vol. CAD-2, No 3, pp. 151–167.

[Lecky, 1989] O.I. Lecky, O.I. Murphy, R.G. Absher. Graph theoretic flgorithms for the PLA folding problem. In: IEEE Trans. Computer-Aided Design, 1989, vol. 8, No 9, pp. 1014–1021.

[Cheremisinova, 2004] L. Cheremisinova. Simple folding of array-based VLSI structures. In: Proc. of 6-th Intern. Workchop on Boolean problems, Freiberg (Sachsen), Sept. 19–20, 2004, pp. 245–250.

[Eén] N. Eén, and N. Sörensson. MiniSat. http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat.

[Mahajan, 2005] Y. Mahajan, Z. Fu, and S. Malik. Zchaff2004: An Efficient SAT Solver. In: Theory and Applications of Satisfiability Testing (2004 SAT Solver Competition and QBF Solver Evaluation (Invited Papers)), Springer Berlin / Heidelberg, 2005, pp. 360–375.

[Goldberg, 2002] E. Goldberg, and Y. Novikov. BerkMin: A Fast and Robust SAT-Solver. In: Design, Automation, and Test in Europe, March 2002, pp. 142–149.

[Devadas, 1989] S. Devadas. Optimal Layout via Boolean Satisfiability. In Proc of International Conference on Computer-Aided Design (ICCAD '89), 1989, November, pp. 294–297.

[Quintana, 1995] J.M. Quintana, M.J. Avedillo, M.P. Parra, and J.L. Huertas. Optimum PLA Folding through Boolean Satisfiability. In: Asian South Pacific Design Automation Conference (ASP_DAC'95), 1995, pp. 289–293.

[Luby, 1982] M. Luby, U. Vazirani, V. Vazirani, A. Sangiovanni-Vincentelli. Some theoretical results on the optimal PLA folding problem. In: Proc. of IEEE Conf on Circuits and Computers, 1982, pp. 165–170.

[Bryant, 1986] R.E. Bryant. Graph-based algorithms for Boolean function manipulation. In: IEEE Trans. Computers, 1986, vol. C-35, No 8, pp. 677–691.

[Cheremisinova, 1999] L.D. Cheremisinova. Area minimization of Weinberger matrix. In: Proc. of the Third Intern. Conf. on Computer-Aided Design of Discrete Devices, CAD DD'99, Minsk: Rep.of Belarus, Nov.10–12, 1999, vol. 1, pp. 80–87.

[Cheremisinova, 1999] L.D. Cheremisinova. Some results in optimal PLA folding. In: Proc. of the Third Intern. Conf. on Computer-Aided Design of Discrete Devices, CAD DD'99, Minsk: Rep.of Belarus, Nov.10–12, 1999, vol. 1, pp. 59–64.

[Cheremisinova, 2010] L.D. Cheremisinova. Regular structure folding based on solving logic equations. In: Proc. of 4th intern. Conf. of Tanaev's readings, March 28–29, 2010, Minsk: UIIP NAS of Belarus (in Russian: Л.Д. Черемисинова. Свертка регулярных структур на основе решения логических уравнений. В: Танаевские чтения. Доклады 4-й Междун.научной конф. (28–29 марта 2010 г., Мн.: ОИПИ НАН Беларуси).

Authors' Information

Liudmila Cheremisinova – *Principal Researcher, The United Institute of Informatics Problems of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, Belarus, e-mail: cld@newman.bas-net.by*

Major Fields of Scientific Research: Logic Design, CAD systems, optimization