Global Lower Bounds for the VLSI Macrocell Floorplanning Problem using Semidefinite Optimization *

P. L. Takouda, M. F. Anjos
Department of Management Sciences
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
ptakouda@uwaterloo.ca
anjos@stanfordalumni.org

A. Vannelli
Department of Electrical and Computer Engineering, University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
vannelli@cheetah.vlsi.uwaterloo.ca

Abstract

We investigate the application of Semidefinite Programming (SDP) techniques to the VLSI macrocell floorplanning problem. We propose a new mixed-integer SDP formulation of the problem which leads to new SDP relaxations. This approach has been implemented and we report global lower bounds for some MCNC benchmark macrocell problems.

1 Introduction

The VLSI macrocell floorplanning problem consists of partitioning a given rectangular chip into N modules with fixed areas so as to minimize the total cost associated with interactions between these modules and with N_p fixed I/O pads located all around the chip. It is one of the most challenging problems in VLSI CAD. In particular, we are interested in the *fixed-outline* floorplanning. This problem, where the chip's dimensions are fixed, is receiving growing attention, since it addresses some of difficulties with classical floorplanning in the context of ASIC and is adapted to hierarchical design in ASIC and SoC (see [1], [6]). Even though our primary interest is in fixed-outline floorplanning, our formulation can be extended to the classical outline-free case.

In this paper, we present a new formulation for the problem using mixed-integer semidefinite programming motivated by the tremendous progress in the area of semidefinite programming (SDP) and the recent work [2] on single-row layout. More precisely, we derive a model where rectilinear distances are used and the design of the layout of the modules on the chip is formulated using integer and semidefinite constraints.

The paper is organised as follows. In the next section, we introduce semidefinite optimization problems and semidefinite relaxations of a binary optimization problem. Section 3 introduces the new mixed integer and semidefinite optimization model for the problem. In Section 4, an SDP relaxation is derived, and in Section 5, we present computational results for three MCNC macrocell floorplanning benchmark problems.

Remark 1.1 *The chip is a $w_F \times h_F$ rectangle with area a_F. We set the origin of our system of coordinates at the center of the chip. A module i is a $w_i \times h_i$ rectangle with area a_i and centroid (x_i, y_i). $w_i^{\max}$ (and respectively $w_i^{\min}$, $h_i^{\max}$, $h_i^{\min}$) stands for the maximum length (resp. minimum length, maximum width, minimum width) of each module. The pads are determined by their coordinates, and are denoted using the letters p, q, r while the modules are denoted by i, j, k. We finally assume that the cost of the connection per unit of distance between modules i, j is denoted by c_{ij} and the cost per unit of distance between a module i and a pad p is γ_{ip}.*

2 Semidefinite Relaxations

In the space $\mathcal{S}_n$ of symmetric matrices with real entries, one can define a scalar product as $\langle A, B\rangle = \text{trace}(A \cdot B)$, $A, B \in \mathcal{S}_n$, usually called Frobenius scalar product. In $\mathcal{S}_n$, a symmetric matrix A is said to be *positive semidefinite* and denoted by $A \succeq 0$ if all its eigenvalues are nonnegative.

*This research was supported by grant NAL/00636/G from the Nuffield Foundation (UK), grant A2002/3 from the University of Southampton, NSERC of Canada Operating Grant 15296, and a Bell University Laboratories Research Grant at the University of Waterloo.

Definition 2.1 *In $\mathcal{S}_n$, a semidefinite programming problem (SDP) is the following optimization problem:*

$$\begin{array}{lll} \min & \langle C, X\rangle & C \in \mathcal{S}_n \text{ (linear objective)}, \\ & \langle A_i, X\rangle = b_i, & i = 1, \ldots, m; A_i \in \mathcal{S}_n, b_i \in \mathbb{R} \\ & X \succeq 0 & \end{array}$$

SDP is a linear optimization problem in $\mathcal{S}_n$, and can be viewed as the generalization of linear programming (LP). In particular, interior-point algorithms for LP can be extended *mutatis mutandis* to efficiently solve SDP problems. As a result, SDP has recently received much attention from the optimization community.The recent handbook of SDP [14] gives an extensive list of applications together with a detailed treatment of the theory and algorithms.

One major area of application of SDP has been the derivation of improved relaxations for binary optimization problems. A *binary program* is an optimization problem where some of variables are restricted to take one of two possible values (usually $0, 1$ or $-1, 1$). It is common for such problems to derive relaxations and solve these relaxations to obtain global bounds on the optimal value of the problem. These relaxations are then used with rounding schemes or within a branch & bound framework to obtain global optimal solutions.

We call relaxation of a binary optimization problem the optimization problem obtained by replacing the constraints enforcing some variables to be binary by "simpler" but weaker constraints. The most common relaxation is the well-known LP relaxation.

Let the binary variables be denoted by $x_i \in \{-1, 1\}, i = 1, \ldots, n$. For an integer $k = 1, \ldots, n$, define a k^{th} *order lifting or k-lifting* to be a formulation of the problem involving products $x_{i_1} x_{i_2} \ldots x_{i_k}$ of k given binary variables. Such k-liftings are behind some of the most powerful relaxation procedures that have been used in recent years. Recently, a new relaxation procedure using k-liftings and SDP was proposed and shown to be stronger than the previous procedures (see [8], [9]). The idea is to derive relaxations where for $k \geq 1$, the binary variables are replaced by the following constraints:

$$\begin{array}{c} y = (1, x_1, \ldots, x_n, x_i x_j, \ldots, x_{i_1} x_{i_2} \cdots x_{i_k})^T, \\ Y \succeq 0, \;\; Y = y \cdot y^T, \end{array} \tag{1}$$

where the entries of y are all the products of m variables for $0 \leq m \leq k$. In other words, the relaxation is an SDP problem, whose matrix variable Y must satisfy the constraints $\text{diag}(Y) = (1, \ldots, 1)^T$, $\text{rank}(Y) = 1$, $Y \succeq 0$ where $\text{diag}(Y)$ is the vector of the diagonal entries of Y. This procedure has been used successfully to obtain strong relaxations for some challenging combinatorial problems (see [13], [10], [7]).

3 A mixed integer SDP formulation for floorplanning

We now derive a new formulation of the macrocell floorplanning problem using semidefinite optimization.

3.1 Area constraints

We relax the area constraint $w_i h_i = a_i$ for each module as

$$h_i w_i \geq a_i. \tag{2}$$

This is a standard relaxation in optimization, and (2) can then easily be expressed as a semidefinite constraint:

$$\begin{bmatrix} h_i & \sqrt{a_i} \\ \sqrt{a_i} & w_i \end{bmatrix} \succeq 0, \tag{3}$$

which is well defined since $a_i > 0$. It is clear that (3) $\Rightarrow$ (2). In fact, (3) $\Leftrightarrow$ (2) since $w_i, h_i \geq 0$. Using (3), there is **no** error due to linearization. Furthermore, (3) simultaneously requires that $h_i, w_i \geq 0$.

Theorem 3.1 *Let a_F be the area of the chip, and $a_i, i = 1, \ldots, N$ be the areas of the modules. If* (3) *holds and $a_F = \sum_i a_i$, then the exact area constraints are satisfied.*

Proof: From (3) and the fact that all the modules lie in the chip, we have:

$$0 \leq \sum \underbrace{w_i h_i - a_i}_{\geq 0} = \sum w_i h_i - \sum a_i \leq \underbrace{a_F - \sum a_i}_{=0}.$$

This obviously implies $w_i h_i = a_i$. ∎

Requiring $\sum_i a_i = a_F$ is not restrictive, because most benchmark problem satisfy it. Moreover, this requirement is directly in the spirit of fixed-outline floorplanning where, in addition to minimizing the connectivity costs, one also wants to minimize dead-space in the layout.

3.2 Fit-in-the-chip constraints

For module i to lie inside the chip, the following inequalities have to be satisfied:

$$-\tfrac{1}{2}(w_F - w_i) \leq x_i \leq \tfrac{1}{2}(w_F - w_i), \tag{4}$$

$$-\tfrac{1}{2}(h_F - h_i) \leq y_i \leq \tfrac{1}{2}(h_F - h_i). \tag{5}$$

It is straightforward that (4) is equivalent to:

$$x_i^2 \leq \left[\tfrac{1}{2}(w_F - w_i)\right]^2, \tag{6}$$

$$\Leftrightarrow \begin{pmatrix} \frac{1}{2}(w_F - w_i) & x_i \\ x_i & \frac{1}{2}(w_F - w_i) \end{pmatrix} \succeq 0. \tag{7}$$

And similarly we have for (5):

$$\begin{pmatrix} \frac{1}{2}(h_F - h_i) & y_i \\ y_i & \frac{1}{2}(h_F - h_i) \end{pmatrix} \succeq 0. \tag{8}$$

This shows that the fit-in-the chip constraints can also be exactly written as semidefinite constraints.

3.3 Aspect ratio constraints

Given $\beta_i \geq 1$, the aspect ratio requirement for module i is: $\frac{w_i}{h_i} \leq \beta_i$ and $\frac{h_i}{w_i} \leq \beta_i$.Using $w_i > 0, h_i > 0, a_i > 0, \quad w_i h_i = a_i$, it is equivalent to: $h_i^2 \leq \beta_i a_i$ and $w_i^2 \leq \beta_i a_i$, and therefore to:

$$\begin{pmatrix} \beta_i & w_i \\ w_i & a_i \end{pmatrix} \succeq 0, \begin{pmatrix} \beta_i & h_i \\ h_i & a_i \end{pmatrix} \succeq 0. \tag{9}$$

3.4 Non-overlap constraints

Finally, we consider the non-overlap constraints. These are disjunctive constraints, since they require the modules to be separated in either the x direction or the y direction. They can be expressed as follows: $\forall i, j, i < j, \underbrace{d_{ij}^x}_{|x_i - x_j|} \geq \frac{1}{2}(w_i + w_j)$ or $\underbrace{d_{ij}^y}_{|y_i - y_j|} \geq \frac{1}{2}(h_i + h_j)$. It is important to notice that only one of the two inequalities has to be satisfied for each pair of modules. These constraints are commonly modeled using binary variables [12] or complementary constraints [3]. We derive here a new formulation which, like in [5], uses only **two** binary variables per pair of modules. The first variable σ_{ij} is used to decide the direction in which the non-overlap is enforced:

$$\sigma_{ij} = \begin{cases} +1 & \text{if } i \text{ and } j \text{ are separated along } x, \\ -1 & \text{if } i \text{ and } j \text{ are separated along } y, \end{cases} \tag{10}$$

Once this direction is selected, we determine the relative position of modules i and j in that direction using a second binary variable:

$$\alpha_{ij} = \begin{cases} +1 & \text{if } i \text{ precedes } j \text{ in the selected direction,} \\ -1 & \text{if } j \text{ precedes } i \text{ in the selected direction.} \end{cases} \tag{11}$$

Define the following quantities: $Q_{ij}^x = \min\{w_F, w_i^{\max} + w_j^{\max}\}$, $Q_{ij}^y = \min\{h_F, w_i^{\max} + w_j^{\max}\}$, $Q_{ij}^y = \min\{h_F, w_i^{\max} + w_j^{\max}\}$, $U_{ij}^x = w_F - \frac{1}{2}w_i^{\min} - \frac{1}{2}w_j^{\min}$, $U_{ij}^y = h_F - \frac{1}{2}h_i^{\min} - \frac{1}{2}h_j^{\min}$, $U_{ij}^y = h_F - \frac{1}{2}h_i^{\min} - \frac{1}{2}h_j^{\min}$ and consider the following inequalities:

$$d_{ij}^x \geq \tfrac{1}{2}(w_i + w_j) - \tfrac{1}{2}(1 - \sigma_{ij})Q_{ij}^x \tag{12}$$
$$d_{ij}^x - 2S_{ij}^x = x_j - x_i \tag{13}$$
$$0 \leq S_{ij}^x \tag{14}$$
$$S_{ij}^x \leq \tfrac{1}{2}\left[(1 - \sigma_{ij}) + \tfrac{1}{2}(1 + \sigma_{ij})(1 - \alpha_{ij})\right] U_{ij}^x \tag{15}$$
$$0 \leq S_{ij}^x + (x_j - x_i) \tag{16}$$
$$S_{ij}^x + (x_j - x_i) \leq \tfrac{1}{2}\left[(1 - \sigma_{ij}) + \tfrac{1}{2}(1 + \sigma_{ij})(1 + \alpha_{ij})\right] U_{ij}^x \tag{17}$$

We claim that: *When the separation between modules i and j is in direction x, the constraints* (12)-(17) *enforce non-overlap and compute exactly the x component d_{ij}^x of the distance d_{ij}.* Indeed, when $\sigma_{ij} = 1$,

- (12) $\Rightarrow d_{ij}^x \geq \frac{1}{2}(w_i + w_j)$ which separates i and j in the x-direction;
- either $\alpha_{ij} = 1$ ($x_j \geq x_i$), and from (15), (13),(17), $S_{ij}^x = 0,\ d_{ij}^x = x_j - x_i \geq 0;\ 0 \leq x_j - x_i \leq U_{ij}^x$;
- or $\alpha_{ij} = -1$ ($x_j \leq x_i$), and $S_{ij}^x = -(x_j - x_i),\ d_{ij}^x = x_i - x_j \geq 0;\ 0 \leq x_i - x_j \leq U_{ij}^x$.

Theorem 3.2 *Suppose σ_{ij} and α_{ij} defined as in* (10), (11)*, then the non-overlap constraint is equivalent to*

$$(12) - (17) \tag{18}$$
$$d_{ij}^y \geq \tfrac{1}{2}(h_i + h_j) - \tfrac{1}{2}(1 + \sigma_{ij})Q_{ij}^y. \tag{19}$$
$$d_{ij}^y - 2S_{ij}^y = y_j - y_i \tag{20}$$
$$0 \leq S_{ij}^y \tag{21}$$
$$S_{ij}^y \leq \tfrac{1}{2}\left[(1 + \sigma_{ij}) + \tfrac{1}{2}(1 - \sigma_{ij})(1 - \alpha_{ij})\right] U_{ij}^y \tag{22}$$
$$0 \leq S_{ij}^y + (y_j - y_i) \tag{23}$$
$$S_{ij}^y + (y_j - y_i) \leq \tfrac{1}{2}\left[(1 + \sigma_{ij}) + \tfrac{1}{2}(1 - \sigma_{ij})(1 + \alpha_{ij})\right] U_{ij}^y \tag{24}$$

Proof: From the paragraph preceding the theorem, if $\sigma_{ij} = 1$, the equations (12), (15), (17) ensure non-overlap between modules i and j in the x direction **and** compute exactly d_{ij}^x. At the same time, (19) is equivalent to $d_{ij}^y \geq 0 \geq \frac{1}{2}(h_i + h_j) - Q_{ij}^y$ which is obvious. By (22),(24): $0 \leq S_{ij}^y,\ 0 \leq S_{ij}^y + y_j - y_i$ and then using (20), we obtain: $d_{ij}^y \geq y_i - y_j,\ d_{ij}^y \geq y_j - y_i$, which is the linear relaxation of $d_{ij}^y = |y_j - y_i|$. This proves that when $\sigma_{ij} = 1$, the constraints (12)-(24) enforce separation in the x direction and compute **exactly** the distance d_{ij}^x while performing a linear relaxation of d_{ij}^y. The case $\sigma_{ij} = -1$ is proved similarly. ∎

Note that (15),(17),(22),(24) can be expanded as:

$$S_{ij}^x \leq \tfrac{1}{4}\left(3 - \sigma_{ij} - \alpha_{ij} - \sigma_{ij}\alpha_{ij}\right) U_{ij}^x, \tag{25}$$
$$S_{ij}^x + (x_j - x_i) \leq \tfrac{1}{4}\left(3 - \sigma_{ij} + \alpha_{ij} + \sigma_{ij}\alpha_{ij}\right) U_{ij}^x \tag{26}$$
$$S_{ij}^y \leq \tfrac{1}{4}\left(3 + \sigma_{ij} - \alpha_{ij} + \sigma_{ij}\alpha_{ij}\right) U_{ij}^y, \tag{27}$$
$$S_{ij}^y + (y_j - y_i) \leq \tfrac{1}{4}\left(3 + \sigma_{ij} + \alpha_{ij} - \sigma_{ij}\alpha_{ij}\right) U_{ij}^y. \tag{28}$$

Remark that products of sigmas by alphas arise in these expressions. The strength of an SDP relaxation depends on how well it approximates those products of binary variables.

3.5 Complete formulation

To complete our formulation, we need to compute the rectilinear distances $d^r_{ip} = \underbrace{|x_i - x_p|}_{d^x_{ip}} + \underbrace{|y_i - y_p|}_{d^y_{ip}}$ between a module i and an I/O pad p. Since the x_p's and y_p's are given data and either $|x_p| \geq w_F/2$ or $|y_p| \geq h_F/2$, we have two cases. If $|x_p| \geq w_F/2$:

$$d^x_{ip} = x_p - x_i \text{ or } d^x_{ip} = x_i - x_p (\text{whichever is positive}) \quad (29)$$

$$d^y_{ip} \geq y_p - y_i \quad (30)$$

$$d^y_{ip} \geq y_i - y_p \quad (31)$$

The case $|y_p| \geq h_F/2$ is handled similarly.

Remark 3.3 *Our formulation can be modified in a straightforward way if one prefers to measure the distances using the Euclidean distance. The following constraints can be used:*

$$d \geq \sqrt{(d^x)^2 + (d^y)^2}. \quad (32)$$

These are second-order cone constraints, are a special case of semidefinite constraints, and can also be handled using interior-point algorithms (see [14]).

We now put together all the above results to obtain a mixed-integer semidefinite programming (MISDP) formulation for the VLSI macrocell floorplanning problem:

$$\begin{array}{ll} \min & \sum_{ij=1}^{N} c_{ij}(d^x_{ij} + d^y_{ij}) + \sum_{i=1}^{N}\sum_{p=1}^{N_p} \gamma_{ip}(d^x_{ip} + d^y_{ip}) \\ \text{s.t.} & \\ & \text{Area: (3)} \\ & \text{Aspect: (9)} \\ & \text{Fit-in-the-chip: (7), (8)} \\ & \text{Non-overlap: (12), (19)} \\ & \text{Distance (modules): (13)} - \text{(17), (20)} - \text{(24)} \\ & \text{Distance (module-pad): (29)} - \text{(31)} \\ & \text{Variables: } \sigma_{ij}, \alpha_{ij} = -1 \text{ or } 1, \ \forall i < j, \\ & d^x_{ij}, d^y_{ij}, d^x_{ip}, d^y_{ip}, S^x_{ij}, S^y_{ij} \in \mathbb{R}, \ \forall i < j, \\ & w_i, h_i, x_i, y_i \in \mathbb{R}, \ \forall i < j. \end{array}$$

4 Deriving the SDP relaxation

Often when deriving LP relaxations for integer problems, valid equalities or inequalities are added to the models to avoid trivial solutions or strengthen the bounds. When they involve the binary variables, they are called *special ordered sets (SOS) constraints.*

4.1 Special Ordered Sets constraints

We first ensure that our formulation satisfies the following **transitivity property** between the modules: "*suppose that three modules i, j, k are separated in the same direction, then in that direction, if i precedes j and j precedes k, then i precedes k*". Such transitivity constraints were used in [2] for the single row placement problem. Note that:

- The modules i, j, k are separated in the same direction if and only if $\sigma_{ij} = \sigma_{jk} = \sigma_{ik}$.
- The transitivity property in the selected direction is then : $\alpha_{ij} = \alpha_{jk} \Rightarrow \alpha_{ij} = \alpha_{ik}$.

Therefore, our transitivity property is stated as:

$$\left(\begin{array}{c} \sigma_{ij} = \sigma_{jk} = \sigma_{ik} \\ \text{and} \\ \alpha_{ij} = \alpha_{jk} \end{array} \right) \Rightarrow \alpha_{ij} = \alpha_{ik}. \quad (33)$$

Proposition 4.1 *A sufficient condition to have* (33) *is : $\forall i, j, k$ such that $i < j < k$,*

$$(\sigma_{ij} + \sigma_{jk})(\sigma_{ij} + \sigma_{ik})(\alpha_{ij} + \alpha_{jk})(\alpha_{ij} - \alpha_{ik}) = 0. \quad (34)$$

Note the restriction $i < j < k$ introduced.

Proof: For any i, j, k, one of the following cases holds: **(i)** $i < j < k$, **(ii)** $k < i < j$, **(iii)** $j < k < i$, **(iv)** $i < k < j$, **(v)** $j < i < k$, **(vi)** $k < j < i$. From the definition of α_{ij} and σ_{ij}, we have also: $\sigma_{ij} = \sigma_{ji}$ and $\alpha_{ij} = -\alpha_{ji}, \quad \forall i < j$. Case **(i)** is straightforward. For case **(ii)**, we have:

$$\begin{array}{rcl} (34) & \Rightarrow & (1 + \sigma_{ki}\sigma_{kj} + \sigma_{ki}\sigma_{ij} + \sigma_{kj}\sigma_{ij}) \\ & & (1 - \alpha_{ki}\alpha_{kj} + \alpha_{ki}\alpha_{ij} - \alpha_{kj}\alpha_{ij}) = 0, \\ & \Rightarrow & (1 + \sigma_{ik}\sigma_{jk} + \sigma_{ik}\sigma_{ij} + \sigma_{jk}\sigma_{ij}) \\ & & (1 - \alpha_{ik}\alpha_{jk} - \alpha_{ik}\alpha_{ij} + \alpha_{jk}\alpha_{ij}) = 0, \\ & \Rightarrow & (1 + \sigma_{ij}\sigma_{ik} + \sigma_{ij}\sigma_{jk} + \sigma_{ik}\sigma_{jk}) \\ & & (1 - \alpha_{ij}\alpha_{ik} + \alpha_{ij}\alpha_{jk} - \alpha_{ik}\alpha_{jk}) = 0, \\ & \Rightarrow & (33). \end{array}$$

The other cases can be proved in the same way. ■

Note that (34) can be expanded as : $\forall i, j, k, i < j < k$,

$$\begin{array}{rl} 1 = & \alpha_{ij}\alpha_{ik} - \alpha_{ij}\alpha_{jk} + \alpha_{ik}\alpha_{jk} \\ & -\sigma_{ij}\sigma_{ik} - \sigma_{ij}\sigma_{jk} - \sigma_{ik}\sigma_{jk} \\ & +\sigma_{ij}\sigma_{ik}\alpha_{ij}\alpha_{ik} - \sigma_{ij}\sigma_{ik}\alpha_{ij}\alpha_{jk} + \sigma_{ij}\sigma_{ik}\alpha_{ik}\alpha_{jk} \\ & +\sigma_{ij}\sigma_{jk}\alpha_{ij}\alpha_{ik} - \sigma_{ij}\sigma_{jk}\alpha_{ij}\alpha_{jk} + \sigma_{ij}\sigma_{jk}\alpha_{ik}\alpha_{jk} \\ & +\sigma_{ik}\sigma_{jk}\alpha_{ij}\alpha_{ik} - \sigma_{ik}\sigma_{jk}\alpha_{ij}\alpha_{jk} + \sigma_{ik}\sigma_{jk}\alpha_{ik}\alpha_{jk}. \end{array} \quad (35)$$

Because these constraints can be generated, as we will see below, by the products of two sigmas or two alphas, we call them **second order special ordering sets** (**SOS**[2]) constraints.

4.2 SDP relaxation matrix

In (25)-(28), (35), some products of our binary showed up. Therefore, we need an SDP relaxation *matrix* in which all these products of binary variables appear. As a result, the relaxation matrix contains at least second order liftings of the binary variables. We introduce the following vector of $1 + 2\text{tn} + 2\text{ttn}$ binary variables called bin:

$$\begin{aligned}\text{bin} = \quad & (1, \sigma_{12}, \ldots, \sigma_{N-1,N}, \alpha_{12}, \ldots, \alpha_{N-1,N}, \sigma_{12}\sigma_{13}, \\ & \ldots, \sigma_{N-2,N}\sigma_{N-1,N}, \alpha_{12}\alpha_{13}, \ldots, \alpha_{N-2,N}\alpha_{N-1,N})^T\end{aligned}$$

where $\text{tn} = \begin{pmatrix} N \\ 2 \end{pmatrix}$, $\text{ttn} = \begin{pmatrix} \text{tn} \\ 2 \end{pmatrix}$. This vector contains all our binary variables and their required products. Then, the classical *rank-one* SDP formulation matrix is:

$$\text{bin} \cdot \text{bin}^T \succeq 0. \tag{36}$$

4.3 Symmetry breaking and valid inequalities

We use the so-called *p-q symmetry breaking method* of [12] by adding the constraints to our problem: $\sigma_{kl} = 1; \alpha_{kl} = 1; y_k \leq y_l$, where the pair $\{k, l\}$ is the one with the largest connection cost. Then the $B2$ valid inequalities (of [12]) are: $d^x_{ij} \geq \frac{1}{4}(w_i^{\min} + w_j^{\min}) \cdot (1 + \sigma_{ij}), d^y_{ij} \geq \frac{1}{4}(h_i^{\min} + h_j^{\min}) \cdot (1 - \sigma_{ij})$. The last valid inequalities are used to enforce centroid separation. We call them S valid inequalities:

$$\begin{aligned}
x_i + \tfrac{1}{2}w_i &\leq x_j - \tfrac{1}{2}w_j + \tfrac{1}{2}\left(3 - \sigma_{ij} - \alpha_{ij} - \sigma_{ij}\alpha_{ij}\right) w_F, \\
x_j + \tfrac{1}{2}w_j &\leq x_i - \tfrac{1}{2}w_i + \tfrac{1}{2}\left(3 - \sigma_{ij} + \alpha_{ij} + \sigma_{ij}\alpha_{ij}\right) w_F, \\
y_i + \tfrac{1}{2}h_i &\leq y_j - \tfrac{1}{2}h_j + \tfrac{1}{2}\left(3 + \sigma_{ij} - \alpha_{ij} + \sigma_{ij}\alpha_{ij}\right) h_F, \\
y_j + \tfrac{1}{2}h_j &\leq y_i - \tfrac{1}{2}h_i + \tfrac{1}{2}\left(3 + \sigma_{ij} + \alpha_{ij} - \sigma_{ij}\alpha_{ij}\right) h_F.
\end{aligned}$$

5 Computational results

We applied the SDP relaxation to three problems from the MCNC benchmark: apte, xerox and hp. To solve the SDP relaxation, we used the public domain software CSDP [4] on a 2.0 GHz Dual Opteron with 16Gb of RAM. We compared the bounds we have obtained with the best known solutions.

Table 1. Relaxation bounds

Circuit	Ratio	Bounds	CPU (sec.)
apte	2	3135.9	891
	3	3021.6	789
	5	2918.5	848
	8	2848.8	815
	10	2434.0	793
xerox	2	2434.0	1665
	3	2051.0	2290
	5	1539.8	2930
	8	1217.1	2835
	10	1153.1	2721
hp	2	976.98	8156
	3	893.25	8230
	5	822.11	8294
	8	783.80	7840
	10	773.23	7855

Table 2. SDP bounds vs. solution, $\beta_i = 10$

Circuit	Bounds	Anjos-Vannelli		Kuh-Murata	
		Solution	Gap (%)	Solution	Gap (%)
apte	2847.7	5205.4	45.3	4353.5	34.6
xerox	1153.1	6538	82.4	4976.5	76.8
hp	773.23	2101.2	63.2	1779.8	56.6

5.1 SDP relaxation bounds

We first solved the SDP relaxation for each of our test problems and the lower bounds obtained are reported in Table 1. These bounds are compared in Table 2 with solutions to the problems computed using the Anjos-Vannelli [3] formulation for the floorplanning problem and with solutions obtained by Kuh-Murata [11]. In particular, we evaluate the relative gap between the computed solution and our bound using the following formula: $\frac{\text{solution}-\text{bound}}{\text{solution}}$. We make the following observations about these results:

- **First non-trivial bounds.** Our SDP relaxation provides non-trivial lower bounds on the optimal value of the problem. This is the first time such lower bounds have been computed, and they provide new information to evaluate current and future floorplans.

- **Size of the gaps.** Some of the gaps in Table 2 are relatively large. This is partially explained by the fact that we are comparing our bounds to best known solutions, and those two quantities can be improved. Indeed, we recall that our bound is computed using second order liftings, where in theory $N(N-1)$ order liftings are needed to obtain the best global lower bound. This means that our lower bounds can be improved by using a larger

Table 3. Global bounds vs. solution, $\beta_i = 10$

Circuit	Bounds	k	Anjos-Vannelli		Kuh-Murata	
			Solution	Gap (%)	Solution	Gap (%)
apte	3119.1	5	5205.4	40.1	4353.5	28.3
xerox	1153.1	4	6538	82.3	4976.5	76.8
hp	781.38	3	2101.2	62.8	1779.8	56.1

relaxation matrix. An alternative way to improve the bounds is described in the next section.

5.2 Global lower bounds

In order to improve the lower bounds obtained, we start a branch & bound tree, perform a four-way branching on both σ and α for a given i, j, solve the 4^k nodes of a certain level k and report the lower bound obtained. The results are reported in Table 3.

- **Improved bounds.** Table 3 shows that the lower bounds can be significantly improved by making a slice in a branch & bound tree. Here again, the level of the slice highly influences the quality of the bounds. There is a trade-off between the computation time needed and the level of the slice.

- **Time issues.** Evidently, the depth of the level solved to obtain improved global lower bounds has an impact on the computation time needed. As one can see from Table 1, solving a relaxation of *hp* requires more than two hours of computation. This is the reason why we only solved level 3 of a branch & bound tree. Solving a deeper level should lead us to better bounds, but will surely be time consuming. Improving the computation time by either using other solvers or implementing an interior-point algorithm totally adapted to the problem is a direction for future research.

6 Conclusion

We have presented a new mixed-integer and semidefinite programming formulation for the VLSI macrocell floorplanning problem. The formulation models exactly the different constraints of the problem. The SDP relaxation of this formulation gives the first non-trivial lower bounds on the optimal value for the small MCNC benchmark problems. However, the computation time of these relaxations is very large. We are working on several extensions of this work: solving larger MCNC benchmark problems, improving the bounds and SDP solvers, deriving rounding schemes.

Acknowledgment

The authors thank C. Luo for providing the layouts obtained using the Anjos-Vannelli approach.

References

[1] S. N. Adya and I. L. Markov. Fixed-outline floorplanning : Enabling hierarchical design. *IEEE Trans. on VLSI Systems.*, 11(6):1120–1135, 2003.

[2] M. F. Anjos, A. Kennings, and A. Vannelli. A semidefinite optimization approach for the single-row layout problem with unequal dimensions. Technical report. To appear in *Discrete Optimization*.

[3] M. F. Anjos and A. Vannelli. A new mathematical-programming framework for facility-layout design. Technical report. To appear in *INFORMS Journal on Computing*.

[4] B. Borchers. CSDP, a C library for semidefinite programming. *Optim. Methods Softw.*, 11/12(1-4):613–623, 1999.

[5] I. Castillo and T. Westerlund. An ε-accurate model for optimal unequal-area block layout design. *Computers & Operations Research*, 32(3):429–447, 2005.

[6] A. B. Kahng. Classical floorplanning harmful? In *ISPD '00: Proceedings of the 2000 international symposium on Physical design*, pages 207–213, 2000.

[7] K. Krishnan and T. Terlaky. Interior point and semidefinite approaches in combinatorial optimization. Technical report. to appear in the GERAD 25th anniversary volume on *Graph Theory and Combinatorial Optimization*, edited by D. Avis, A. Hertz, and O. Marcotte.

[8] J. B. Lasserre. Semidefinite programming vs. LP relaxations for polynomial programming. *Math. Oper. Res.*, 27(2):347–360, 2002.

[9] M. Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Math. Oper. Res.*, 28(3):470–496, 2003.

[10] M. Laurent and F. Rendl. Semidefinite programming and integer programming. Technical report. To appear in the *Handbook on Discrete Optimization*.

[11] H. Murata and E. S. Kuh. Sequence-pair based placement method for hard/soft/pre-placed modules. In *ISPD '98: Proceedings of the 1998 international symposium on Physical design*, pages 167–172, 1998.

[12] H. D. Sherali, B. M. P. Fraticelli, and R. D. Meller. Enhanced model formulations for optimal facility layout. *Oper. Res.*, 51(4):629–644, 2003.

[13] H. Wolkowicz and M. F. Anjos. Semidefinite programming for discrete optimization and matrix completion problems. *Discrete Appl. Math.*, 123(1-3):513–577, 2002.

[14] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of semidefinite programming*. International Series in Operations Research & Management Science, 27. 2000.