Multiscale Variation-Aware Techniques for High-Performance Digital Microfluidic Lab-on-a-Chip Component Placement

Chen Liao and Shiyan Hu*, *Senior Member, IEEE*

***Abstract*—The invention of microfluidic lab-on-a-chip alleviates the burden of traditional biochemical laboratory procedures which are often very expensive. Device miniaturization and increasing design complexity have mandated a shift in digital microfluidic lab-on-a-chip design from traditional manual design to computer-aided design (CAD) methodologies. As an important procedure in the lab-on-a-chip layout CAD, the lab-on-a-chip component placement determines the physical location and the starting time of each operation such that the overall completion time is minimized while satisfying nonoverlapping constraint, resource constraint, and scheduling constraint. In this paper, a multiscale variation-aware optimization technique based on integer linear programming is proposed for the lab-on-a-chip component placement. The simulation results demonstrate that without considering variations, our technique always satisfies the design constraints and largely outperforms the state-of-the-art approach, with up to 65.9% reduction in completion time. When considering variations, the variation-unaware design has the average yield of 2%, while our variation-aware technique always satisfies the yield constraint with only 7.7% completion time increase.**

***Index Terms*—Lab-on-a-chip design automation, multiscale optimization, placement, variations.**

I. Introduction

TRADITIONAL biochemical laboratory procedures are often very expensive to perform and the invention of microfluidic lab-on-a-chip alleviates the burden. Lab-on-a-chip integrates miniaturized components for implementing various functions in biochemical analysis into a chip using microfluidic technology [1], [2]. By lab-on-a-chips, biochemical experiments can be performed in a much cheaper way while having higher sensitivity and accuracy in detection. Lab-on-a-chip has been successfully applied to many important biochemical analysis procedures such as DNA analysis and proteomic analysis [3]. In addition, it becomes indispensable for conducting human health related research including clinical diagnostics and drug discovery [4], [5]. For example, lab-on-a-chip has been successfully used in clinical diagnostics on human physiological fluids [6].

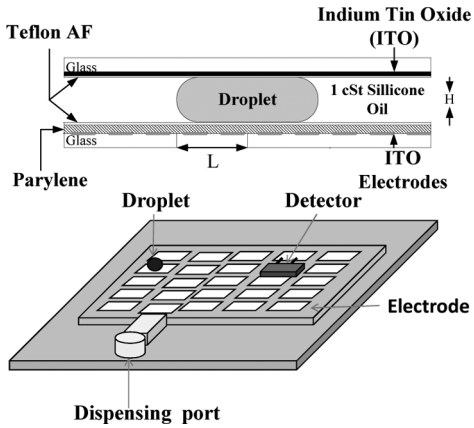


Fig. 1. The schematic of a lab-on-a-chip [2], [8], [9].

The first generation of microfluidic lab-on-a-chip is continuous flow based and consists of permanently etched micropumps, microvalves, and microchannels. In contrast, the prevailing second generation of microfluidic lab-on-a-chip is droplet based, which means that the liquids are manipulated as discrete microdroplets [2]. A few methods to manipulate microdroplets have been designed, such as those with structured surfaces, thermocapillarity, electrochemical effects, and electrostatic actuation [7]. A popular lab-on-a-chip technology is based on electrowetting where each nanoliter droplet is controlled by the electric-field-induced electrohydrodynamic force generated from the programmed electrodes (refer to Fig. 1 [2], [8], [9]). With systematic electrical signal programming, each biochemical operation can be treated as a three dimensional module/cell whose size is decided by the space and the duration the operation needs. Together with the reconfigurability offered in lab-on-a-chips, a three-dimensional module/cell library can be designed [2]. The cell-library-based design methodology enables us to build a large-scale integrated microfluidic lab-on-a-chip efficiently using computer-aided design (CAD) methodologies [2], [10].

In contrast to the manual design, CAD uses the computer technology in the design process to handle device miniaturization and increasing design complexity. In the lab-on-a-chip CAD flow, *physical component placement of digital microfluidic lab-on-a-chip* is a crucial part whose solution may significantly impact the whole flow. Precisely, it determines the physical location and the starting time of each operation such that

Manuscript received August 10, 2009; revised August 06, 2010; accepted February 27, 2011. Date of current version April 27, 2011. *Asterisk indicates corresponding author.*

C. Liao is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: cliao@mtu.edu).

*S. Hu is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton, MI 49931 USA (e-mail: shiyan@mtu.edu).

Digital Object Identifier 10.1109/TNB.2011.2129596

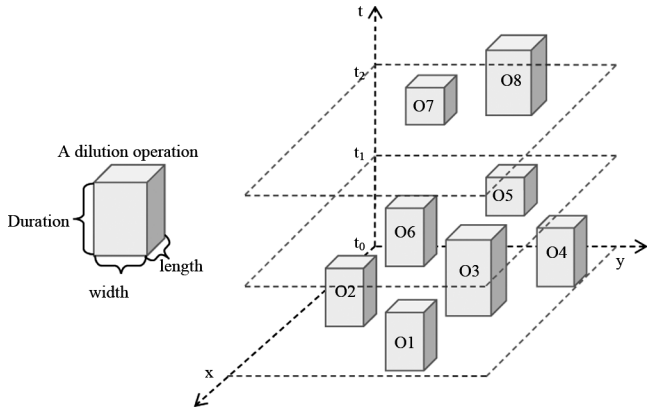


Fig. 2. Illustration of a lab-on-a-chip component placement.

the overall completion time determined by the last operation is minimized (see Fig. 2 for an example). Similar to very large scale integration (VLSI) placement, designing a high-quality module placer is quite challenging in lab-on-a-chip CAD. There are a large multitude of previous works for VLSI placement. For example, [11] proposes a quadratic-based placement algorithm. A partition-based placement approach is designed in [12]. Reference [13] proposes the constraint graph based technique for global placement and the greedy method based technique for the detailed placement. All the above techniques are effective in VLSI placement. One may want to directly migrate the techniques from VLSI placement to lab-on-a-chip component placement. However, there are critical differences. For example, lab-on-a-chip component placement handles resource constraint and scheduling constraint while VLSI placement only considers location optimization. On the other hand, VLSI placement is usually for two dimensions while lab-on-a-chip component placement is for three dimensions due to the reaction time. Thus, simple migration of the techniques from VLSI design to lab-on-a-chip design is difficult to lead to high-quality solutions.

In spite of wide application of lab-on-a-chips, existing research works on lab-on-a-chip CAD are quite limited. For the lab-on-a-chip component placement, there are only two previous works [9], [14]. Reference [14] proposes a simulate annealing based technique and [9] improves it by the incorporation of T-tree data structure. These algorithms are effective; however, they are not always able to compute a lab-on-a-chip component placement satisfying all design constraints due to that they are based on simulated annealing [9]. On the other hand, the timing of a biochemical reaction is sensitive to variations such as temperature variations, which necessitates the variation-aware lab-on-a-chip component placement. However, no previous work considers this issue. Thus, effective algorithmic techniques to compute high-quality variation-aware lab-on-a-chip component placement satisfying various design constraints are in demand.

In this paper, a multiscale variation-aware optimization technique is proposed for the lab-on-a-chip component placement. Our technique is always able to return high-quality placement and it is the first technique addressing the variation-aware lab-on-a-chip component placement. Our simulation results demonstrate that without considering variations, the proposed technique significantly outperforms the previous state-of-the-art approach [9] and is able to achieve up to 65.9% reduction in completions time. When considering variations, the variation-unaware design has the average yield of 2%, while our variation-aware technique can always compute the designs satisfying the yield constraint with only 7.7% completion time increase. The main contribution of the paper is summarized as follows.

- Without considering variations, the multiscale technique, which consists of the grid coarsening stage and the front-line-based fine-scale tuning stage, is proposed to efficiently compute the solution for the variation-unaware lab-on-a-chip component placement.
- In contrast to the previous simulated annealing-based works, which cannot always return the solution satisfying all constraints, our proposed technique can obtain the nearly optimal solutions while satisfying all the constraints.
- To the best of the authors' knowledge, this is the first work addressing the variation-aware lab-on-a-chip component placement. A novel multiscale variation-aware optimization technique is proposed. Latin Hypercube sampling technique is also integrated in the multiscale technique for efficiency.
- The simulation results on the standard benchmark lab-on-a-chip designs demonstrate that without considering variations, the proposed technique always satisfies the design constraints and it outperforms the state-of-the-art approach [9] with up to 65.9% reduction in completion time.
- When considering variations, the variation-unaware design has the average yield of 2%, while the design by our variation-aware technique can always satisfy the yield constraint with only 7.7% completion time increase.

The rest of the paper is organized as follows. Section II presents the problem formulation and integer linear programming formulation for lab-on-a-chip component placement. Section III introduces our multiscale optimization technique for variation-unaware placement. Section IV describes our multiscale variation-aware technique to efficiently compute the variation-aware placement. Section V presents the simulation results with analysis. A summary of work is given in Section VI.

II. Preliminaries

A. Problem Formulation

A lab-on-a-chip is to manipulate droplets which contain biochemical reactants to perform biochemical reactions/operations. As a basic element in a lab-on-a-chip, each biochemical operation can be treated as a three-dimensional cell/module whose volume is decided by the space ($x-y$ plane) and the duration (t-dimension) the operation needs. For a module m, denoted by $l(m)$ its length, by $w(m)$ its width, and by $h(m)$ its height, which is the time needed to perform the corresponding biochemical operation, respectively. Note that the timing of a biochemical reaction is sensitive to variations such as temperature variations. That is, the variations may impact the height of a module $h(m)$, which results in the *variational height*. In

practice, droplet movements are manipulated at discrete intervals and the biochemical operations are scheduled at discrete time. A lattice is laid onto the solution space, which means that a module can be placed only at a grid point. The lab-on-a-chip component placement is to determine the physical location and the starting time of each operation/module/component.

In the lab-on-a-chip component placement, modules are to be packed such that *scheduling constraint*, *spacing constraint*, and *resource constraint* are satisfied [9]. Scheduling constraint, also called *precedence constraint*, specifies temporal relationship between operations by a sequencing graph. Spacing constraint, also called *nonoverlapping constraint*, ensures that no operations can be performed during the same scheduled time period at the same location. For those modules sharing some resources, they can be scheduled at the same time only if there are enough available resources. This is called resource constraint. Given n modules, the lab-on-a-chip component placement problem targets to compute module placement solution with minimum completion time (determined by the last module) subject to the above constraints. When considering variations such as temperature variations, some modules in the above lab-on-a-chip component placement may overlap due to the change of module size in some operating condition. Given a lab-on-a-chip component placement, we call the placement under an operating condition a sample. Given a large enough number of samples, yield is defined as the ratio of the number of samples not leading to any overlap over the total number of samples. In the variation-aware lab-on-a-chip component placement problem, a yield constraint is given and one targets to compute a placement solution satisfying the yield constraint. Our problem is formulated as follows.

Performance Driven Variation-Aware Lab-on-a-chip Component Placement: Given a set of three-dimensional modules, each of which is specified by some length, width, and variational height and a fixed die area, to compute a solution for the lab-on-a-chip component placement, i.e., to decide the physical locations of the module with minimum overall completion time such that the nonoverlapping, resource, and scheduling constraints are satisfied and the yield constraint is also satisfied.

B. Integer Linear Programming (ILP) Formulation for Variation-Unaware Lab-on-a-Chip Component Placement

We first introduce the ILP formulation for variation-unaware lab-on-a-chip component placement in this subsection. The variation-aware lab-on-a-chip component placement will be presented in Section IV. In contrast to the previous approach for the lab-on-a-chip component placement [9], which migrates VLSI circuit placement/floorplanning techniques, our technique directly solves the lab-on-a-chip component placement as a constrained three dimensional packing problem. For this, the problem of the lab-on-a-chip component placement is formulated as an integer linear programming (ILP) problem. We associate with each module i and each three-dimensional grid point (x, y, t) a binary variable $a_{x,y,t,i}$, i.e, $a_{x,y,t,i} \in \{0, 1\}$. Each module is indexed by its lower left corner. Thus, $a_{x,y,t,i} = 1$ means that the lower left corner of the ith module is placed at (x, y, t). Given n modules, the objective is to minimize the timing T subject to the following constraints.

The first constraint is to ensure that each module can be placed at only one location. That is,

$$\sum_{(x,y,t)} a_{x,y,t,i} = 1, \quad \forall\, i = 1, 2, \ldots n. \tag{1}$$

The nonoverlapping constraint is handled as follows. For any grid point, at most one module can be placed to cover it, i.e., the corresponding operation is active at the grid point. Formally, a module m_i covers (x, y, t) if its lower left index is in $[x - l(m_i), x] \times [y - w(m_i), y] \times [t - h(m_i), t]$, i.e., any of $a_{x',y',t'}$ is 1 where $x - l(m_i) < x' \leq x$, $y - w(m_i) < y' \leq y$ and $t - h(m_i) < t' \leq t$. Thus,

$$\sum_{\forall \text{ module } i \text{ covers } (x,y,t)} a_{x,y,t,i} \leq 1, \quad \forall\, (x, y, t). \tag{2}$$

To handle the resource constraint, assume that there are υ types of resources and there are $r_j,\ j = 1, 2, \ldots, \upsilon$ available units of resource j. Denote by the nonnegative constant $s_{i_{r_j}}$ the required units of resource j for performing operation i. For all the modules performed at time t, total required resources need to be no greater than the available resources for each resource type. Thus,

$$\sum_{\forall \text{ module } i \text{ covers } t} s_{i_{r_j}} \cdot a_{x,y,t,i} \leq r_j, \quad \forall\, t,\ j, \tag{3}$$

where a module covers a time t if the corresponding biochemical operation is active at t. A module m_i covers t if the t-coordinate of its lower left index is in $[t - h(m_i), t]$. To handle precedence/scheduling constraint, if module i needs to be performed before module j as specified in the sequencing graph, we have

$$a_{x,y,t,j} + \sum_{\forall\, (x',y',t') \text{ with } t' \geq t} a_{x',y',t',i} \leq 1, \quad \forall\, (x, y, t). \tag{4}$$

Note that the above t' needs to be iterated to T (subject to the following boundary issues), which is the maximum completion time. Further, note that the boundary issue needs to be handled. An operation with the corresponding module m cannot be started at time later than $T - h(m)$ since all operations need to be completed by T. This means that all t for module m in the above formulations can be up to $T - h(m)$. x and y for module m are similarly handled. For the simplicity of presentation, assume that all boundary issues have been taken care of in the following formulation.

The objective of the linear program (LP) is to minimize T. Since one does not know T and the precedence/scheduling constraints cannot be formulated without it, the above mathematical program is cast to a decision problem rather than an optimization problem. The complete *decision integer linear pro-*

gramming formulation is shown as follows, assuming that the boundary issues have been taken care of for simplicity:

$$\begin{aligned}
\min \quad & 1 \\
s.t. \quad & \sum_{(x,y,t)} a_{x,y,t,i} = 1, \quad \forall i = 1,2,\ldots,n \\
& \sum_{\forall \text{ module } i \text{ covers } (x,y,t)} a_{x,y,t,i} \leq 1, \quad \forall (x,y,t) \\
& \sum_{\forall \text{ module } i \text{ covers } t} s_{i_{r_j}} \cdot a_{x,y,t,i} \leq r_j, \quad \forall t,j \\
& a_{x,y,t,j} + \sum_{\forall (x',y',t') \text{ with } t' \geq t} a_{x',y',t',i} \leq 1, \quad \forall (x,y,t) \\
& a_{x,y,t,i} \in \{0,1\}. \qquad (5)
\end{aligned}$$

Suppose that there is an approach to solve this integer linear program which will be presented soon. The optimal completion time, denoted by T^*, can be then found by performing a binary search within the upper bound, denoted by $\bar{T}$ and lower bound, denoted by $\underline{T}$. Each time, $T = \bar{T} + \underline{T}/2$ is used to formulate the above decision linear program. Subsequently, either the lower bound or the upper bound will be set to T according to the decision result. That is, if the integer linear program is not feasible, the lower bound $\underline{T}$ is set to T. Otherwise, the upper bound $\bar{T}$ is set to T. The above process is iterated until the ratio between $\bar{T}$ and $\underline{T}$ is smaller than a user specified parameter. Denote by B the ratio between the initial upper and lower bounds. The above technique needs $O(\log B)$ iterations.

This process can be accelerated through performing the following logarithmic scale binary search proposed in [15] within the upper and lower bounds of T^*. For this, each time, $T = \sqrt{\bar{T} \cdot \underline{T}}$ is used to formulate the decision linear program. The decision problem is then solved and either the lower bound or the upper bound will be set to T as above. It can be shown that in this way, the number of iterations is reduced to $O(\log \log B)$ as follows. After solving each decision problem, if the upper bound is reduced to T, the ratio between the new upper and lower bounds will be $T/\underline{T} = \sqrt{\bar{T} \cdot \underline{T}}/\underline{T} = \sqrt{\bar{T}/\underline{T}}$. If the lower bound is increased to T, the new ratio will be $\bar{T}/T = \bar{T}/\sqrt{\bar{T} \cdot \underline{T}} = \sqrt{\bar{T}/\underline{T}}$. In either case, the ratio of new upper and lower bounds is reduced to $\sqrt{B}$. After the next iteration, the ratio will be reduced to $B^{1/4}$. In general, one can prove that after i oracle queries, the ratio between upper and lower bounds is $B^{1/2^i}$. Clearly, for this ratio to be below a ratio b, it needs $O(\log(\log B/\log b))$ iterations. For any constant ratio b, the above logarithmic scale binary search needs to perform $O(\log \log B)$ iterations. In our simulations, since T needs to be an integer by recalling that each module can only be placed in discrete position (i.e., at grid point), instead of using a fixed ratio b, we perform the logarithmic scale binary search until $\bar{T} - \underline{T} \leq 1$. As demonstrated in the simulation results, the logarithmic scale solution search is efficient.

The classic sequential rounding proposed in [16] is adopted to solve our ILP. One first relaxes the integer constraint $a_{x,y,t,i} \in \{0,1\}$ in (5) to be $a_{x,y,t,i} \in [0,1]$. The resulting linear program, called *relaxed linear program*, does not have any integer constraint. It is well known that in practice, the linear programming problem can be solved in quadratic time in terms of the number of variables and constraints. The solution for the relaxed linear program will be rounded to integers in an iterative manner.

In each iteration, a linear program is solved. In the solution, all $a_{x,y,t,i} > 1 - \delta$ are fixed to 1, where δ is a user specified parameter. If no new variable is fixed, the one with the smallest rounding error will be fixed to 1. These can be accomplished by adding some additional constraints $a_{x,y,t,i} = 1$ to (5). The new linear program will then be solved. Thus, each time, at least one variable is rounded. The above process is repeated until all variables have the integer values. This guarantees that the rounding procedure will compute an integer solution. This process is integrated into the logarithmic solution search as described above to solve the ILP.

III. Multiscale Optimization

Given a large lab-on-a-chip, the linear program may contain many variables and constraints, which degrades the efficiency in computation. This motivates us to explore the multiscale optimization technique for the problem of the lab-on-a-chip component placement. This technique consists of grid coarsening for speedup and fine-scale tuning for completion time improvement. Note that similar multiscale techniques are popular in VLSI placement, however, our algorithm is different from them.

A. Grid Coarsening

In the grid coarsening stage, the grid size is first coarsened by a factor of ρ. That is, the three-dimensional modules are only allowed to be placed at grid location which is a multiple of ρ. For example, setting $\rho = 4$ along t-dimension means that each module can only be placed at a coordinate of 0, 4, 8, … along t-dimension (refer to Fig. 3). In this way, the number of variables will be significantly reduced. For example, there is no $a_{x,y,1,i}, a_{x,y,2,i}, a_{x,y,3,i}$ in the linear program when setting $\rho = 4$. One can similarly set ρ along the x, y dimensions. As a result, the number of constraints and the complexity of the linear program will be significantly reduced. Note that there is no such restriction on T. Together with the fact that modules can have various heights, T does not need to be a multiple of ρ. Solving the lab-on-a-chip component placement at a coarse scale introduces speedup but degrades the solution quality. Varying ρ, different trade-off between runtime and solution quality can be obtained.

B. Fine-Scale Tuning

To recover the solution quality loss, the following fine-scale tuning technique is proposed. For simplicity of presentation, let us just focus on the completion time reduction along t-dimension and other dimensions could be handled similarly. After obtaining the integer linear program solution at a coarse scale, we will attempt to move each module downward along t-dimension. Consider an example in Fig. 3 which shows an integer linear program solution with $\rho = 4$. The modules starting at $t = 4$ can be certainly pushed downward to reduce the completion time. In contrast to only moving the modules along t-dimension, in our algorithm, more solution space will be explored. Take module 5 as an example. Suppose that its lower left corner is placed at (1, 2, 4), i.e., $a_{1,2,4,5} = 1$. A new linear program

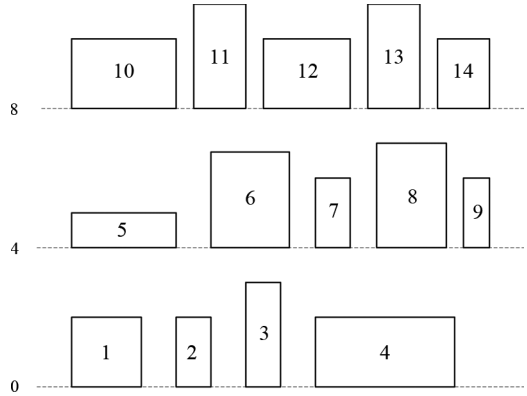


Fig. 3. An example to illustrate fine-scale tuning where 0, 4, 8 refer to the time.

will be formulated as follows. All the binary variables corresponding to the module 5 are limited to those $a_{x,y,t',5}$ such that (x, y) spans the lab-on-a-chip base area and t' only spans over $\{1, 2, 3, 4\}$, i.e., at a fine scale. Recall that in the previous linear problem in (5), t in $a_{x,y,t,5}$ spans over the whole t-dimension ($\leq T$ in the decision LP) when grid coarsening is not applied. With grid coarsening, t can only take the values of $0, 4, 8, \ldots, T$ to save the runtime. In contrast, in the fine-scale tuning stage, one can afford to set t as every possible integer value since it is restricted to a small local region around the solution of the coarsened integer linear program. The whole strategy makes sense since the coarsened integer linear program solution should be a good starting point for further tuning.

The fine-scale tuning technique is formally described as follows. Define the *i-front-line module set* as a set of modules with starting time at $i \cdot \rho$ in the coarsened integer linear program solution. The 0-front-line module set is the set $\{m_1, m_2, m_3, m_4\}$ for the example shown in Fig. 3. Given the i-front-line module set, define the *i-upward module set* as the union of all q-front-line module set where $q \geq i$. For example, the 1-upward module set consists of all the modules except those in the 0-front-line module set (which have the starting time at 0). It is the set $\{m_5, m_6, \ldots, m_{14}\}$ for the example shown in Fig. 3. Similarly, define the *i-downward module set* as the union of all q-front-line module set where $q < i$. For example, the 1-downward module set consists of the modules in the 0-front-line module set (which have the starting time at 0). It is the set $\{m_1, m_2, m_3, m_4\}$ for the example shown in Fig. 3. Clearly, the union of any ith upward and downward module set forms the set containing all modules.

Starting with the 1-downward module set, for every module m_i in the set, fix it as in the coarsened integer program solution. In Fig. 3, this means that all $a_{x,y,0,i}$ for $i = 1, 2, 3, 4$ are fixed to the values in the solution of the coarsened integer program. Denote by $t_c(m)$ the current t of module m in an integer linear program solution. Thus, $a_{x,y,t_c(m),m} = 1$ for some x, y. For each module m_i in the 1-upward module set, the variables are constructed to span all x, y while t is restricted to $\{t_c(m) - \rho + 1, t_c(m) - \rho + 2, \ldots, t_c(m)\}$. We then formulate all the nonoverlapping, resource, and scheduling constraints over these variables. The resulting integer linear program will be solved using sequential rounding technique described in Section II-B (without grid coarsening since it is the fine-scale tuning stage and the computation is affordable in local region). Note that the new decision program is feasible when the completion time is set to the same T as the previous coarsened integer linear program. This is the case since one can assign the values of a according to the solution of the previous coarsened integer linear program. The purpose of the tuning is to reduce T. For this, a loop is formed and each time T is decremented until the integer program becomes infeasible. For example, one possible fine-scale tuning solution for 1-upward module set of Fig. 3 is shown as Fig. 4. One can see that the locations of some modules are moved downward and the completion time is reduced.

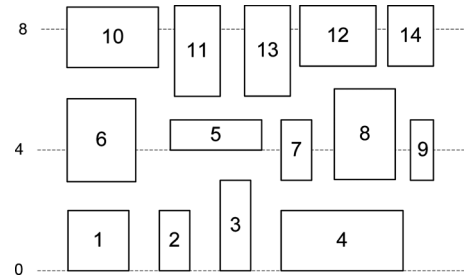


Fig. 4. A possible fine-scale tuning solution for 1-upward module set of Fig. 3, where 0, 4, 8 refer to the time. Assume that all constraints are satisfied.

After this, we will proceed to the 2-downward module set and 2-upward module set. Basically, fix the module location for the second downward set and seek the improvement over the upward second module set. This process is repeated until the current best solution reaches the last front-line. Our simulation results indicate the completion time can be significantly reduced by this fine-scale tuning technique. Refer to Section V for the details.

IV. Variation-Aware Placement Design

In reality, the biochemical operations are quite sensitive to the variations. For example, the environment with different temperature results in the uncertain duration of operations which refers to the variational height of the modules. With variational heights of modules, it becomes a significant challenge to design a high-performance microfluidic lab-on-a-chip. Fig. 5 shows an example of variation-unaware solution for the lab-on-a-chip component placement without any overlap, which is illustrated in two dimension for simplicity. With the variational height of module 2, an overlap between module 2 and module 5 could be introduced, which means that the design is sensitive to variations. It is clear that the quality of microfluidic lab-on-a-chip design is highly dependent on variations. However, no previous works on lab-on-a-chip design consider the variation-aware lab-on-a-chip component placement. It is desirable to design an effective technique for the problem.

A. Variation-Aware Optimization

A novel multiscale variation-aware optimization technique is proposed based on the multiscale technique in Section III. We will first introduce the yield evaluation technique and then the variation-aware optimization technique.

Recall that a placement under an operating condition is called a sample. Given a large enough number of samples, yield is defined as the ratio of the number of samples not leading to any

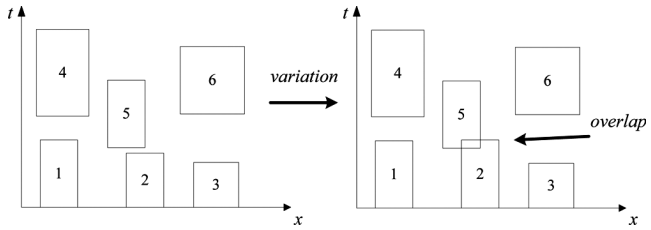


Fig. 5. Illustration of the overlap which is due to variations in a variation-unaware lab-on-a-chip component placement.

overlap over the total number of samples. Monte Carlo simulations are performed to evaluate the yield [17]. For this, a large number (e.g., 10 000) of samples could be generated according to the probabilistic distributions on variational heights. However, it degrades the efficiency. To address this, Latin Hypercube sampling technique, which is first proposed in [18], is integrated in the multiscale variation-aware placement technique. It is a popular technique in improving the simulation efficiency and it has been successfully used in a few VLSI CAD problems such as [19]. The advantage of Latin Hypercube sampling is that one can use only a few samples to approximate the simulation results with a large number of samples. This significantly improves the efficiency in computation. We refer the interested readers to [18], [19] for the details of Latin Hypercube sampling technique.

In the variation-unaware linear programming formulation, i.e., (5), the heights of modules are all fixed. While in variation-aware optimization, duration following Gaussian distribution is introduced to our linear programming formulation. To solve it, our approach is motivated from [20] which formulates and solves the robust Knapsack and Portfolio problems using uncertain data-based linear programming. In (5), the height $h(m)$ of each module m is replaced by $h(m) + \beta \times \hat{h}(m)$, where $\hat{h}(m)$ is defined as the *variational range* according to Gaussian distribution and β is a parameter to control the variation. Varying β, different placement solutions will be obtained. For example, when β is set to 1, the placement solution is the worst-case design and when β is set to 0, the placement solution is variation-unaware design. Denote by $\tilde{h}(m)$ the variational height, i.e., $\tilde{h}(m) = h(m) + \beta \times \hat{h}(m)$. Note that after this transformation, each $\tilde{h}(m)$ is a constant for any fixed β. To explore the best trade-off between the yield and completion time, in our simulations, the search of β (from 0.1 to 1.0 with a step size of 0.1) is performed.

With a fixed β, one can solve the integer linear program in (5). Based on this solution, the multiscale placement optimization technique is enhanced to consider the variations for yield improvement. Recall that in fine-scale tuning, when proceeding to i-front-line, the technique perturbs the current placement to compute a new placement solution for the $(i + 1)$-upward module set in the small local region. In contrast to directly using this new placement solution, the yield of this solution will be computed. If the yield is greater than the yield constraint, this solution will be taken. Otherwise, the original solution will be kept. The algorithm will proceed to the next front-line in a similar way. As the example shown in Fig. 6, the yield constraint is set to 99%. A fine-scale tuning solution for 1-upward module-set is computed from a grid-coarsening solution. The

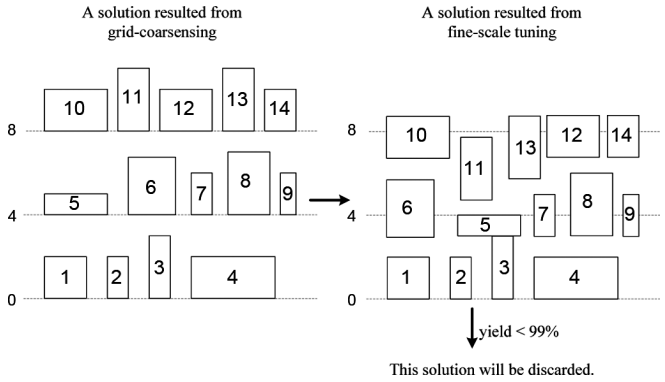


Fig. 6. An example to illustrate the variation-aware multiscale technique.

yield of the new solution is less than 99%. Thus, this solution is discarded and the original one is kept. The algorithm will proceed to the 2-upward module-set in a similar way.

V. Simulation Results

In the simulations, the proposed multiscale integer linear programming based lab-on-a-chip component placement algorithm is implemented in C++ and is tested on a computer with 2.5 GHz CPU and 4 GB main memory. We conduct the simulation on a set of standard lab-on-a-chip benchmarks used in [9] and compare our new algorithm without considering variations to [9]. We also conduct the simulation to compare our variation-aware optimization with the variation-unaware optimization. In our simulation, the yield constraint is set to 99% and Gaussian variation is assumed with the variational range $\hat{h}$ set to 0.1 which is equal to 3σ in Gaussian distribution. Note that our proposed technique could also handle other distributions. Techniques used in comparisons are summarized as follows.

- NEW: the proposed multiscale variation-unaware optimization technique.
- NEW w/o Fine-Tune: NEW except that the fine-scale tuning is turned off.
- NEW w/ standard binary solution search: NEW with standard binary solution search instead of logarithmic scale solution search.
- NEW w/ variations: the proposed multiscale variation-aware optimization technique.

A. Variation-Unaware Design

We first compare our algorithm with the previous work [9]. The results are summarized in Table I. Since our algorithm is a multiscale-based technique, it is interesting to investigate the performance when the fine-scale tuning is turned off, i.e., NEW w/o Fine-Tune. We make the following observations.

- The previous work [9] cannot always return a solution satisfying the chip area constraint due to the nature of simulated annealing. It is the case for the benchmark design vitro1-4. The best design we can get after 20 runs of [9] needs the lab-on-a-chip area of 7×6 with completion time 96. This violates the area constraint of 6×6. Some other results we can get are $9 \times 8 \times 118$ and $12 \times 6 \times 72$, which violate the area constraint even more.
- NEW is always able to meet the design constraints. Our solution is much better than the previous work. On average,

TABLE I
COMPARISON OF NEW AND THE PREVIOUS WORK [9]. TIMING REFERS TO THE OVERALL COMPLETION TIME OF THE LAST MODULE IN THE SOLUTION OF THE LAB-ON-A-CHIP COMPONENT PLACEMENT. CPU REFERS TO THE RUNTIME IN SECONDS. TIMING REDUCTION REFERS TO THE COMPLETION TIME IMPROVEMENT OF NEW WHICH IS COMPUTED BY COMPARING TO THE PREVIOUS WORK [9]

Testcase		Previous work [9]		NEW w/o Fine-Tune			NEW			
Name	Area Constraint	Timing	CPU(s)	Timing	CPU(s)	Timing Reduction	Timing	Timing Reduction	CPU(s)	Yield
vitro1-1	9×9	80	36.8	76	137.7	5.0%	56	30.0%	190.0	0.5%
vitro1-2	8×8	100	20.8	69	95.8	31.0%	54	46.0%	135.5	0%
vitro1-3	7×7	107	31.2	69	209.0	36.7%	54	49.5%	261.0	0%
vitro1-4	6×6	N/A	N/A	69	81.8	N/A	54	N/A	109.8	0%
vitro2-1	8×8	78	18.4	47	94.5	39.7%	42	46.2%	101.1	2%
vitro2-2	7×7	64	13.6	45	54.9	29.7%	42	34.4%	65.5	1.5%
vitro2-3	6×6	129	6.8	48	78.0	62.8%	44	65.9%	73.6	1.0%
vitro3-1	7×7	64	9.6	45	22.6	29.7%	44	31.3%	27.0	4.5%
vitro3-2	6×6	63	14.6	47	29.4	35.4%	44	30.2%	34.2	10.5%
vitro3-3	5×5	112	12.6	45	14.6	59.8%	42	62.5%	17.7	0%
Average								44.0%		2.0%

the completion time is reduced by 44%. For the design vitro2-3, 65.9% completion time reduction is achieved.

- The multiscale technique in NEW consists of grid coarsening and fine-scale tuning. Note that T is not rounded to the multiple of ρ in the linear programming formulation. To choose ρ, extensive simulations are performed and the following parameter setting gives the best trade-off between solution quality and runtime. We coarsen the time dimension by a factor of $\rho = 8$ on t for the first 4 testcases and $\rho = 4$ on t for the last 6 testcases.
- Fine-scale tuning can significantly improve the solution quality. Comparing NEW and NEW w/o Fine-Tune, one can achieve up to $1 - 56/76 = 26.3\%$ completion time improvement.
- It can be seen that [9] saves some runtime over NEW; however, its solution quality is much worse. As mentioned above, the completion time by NEW can be about half of that of [9] in some testcases. Current lab-on-a-chip design technology is still in the early stage without very large scale integration. It is the time to compute high-quality design solutions for promoting the technology advances in lab-on-a-chip as much as possible. That is, the solution quality, rather than efficiency, is of the top importance. With regard to the proposed technique, the obtained large improvement in solution quality clearly outweighs the runtime slowdown.

NEW contains many advanced techniques including logarithmic scale solution search proposed in [15]. It is interesting to investigate the performance of our algorithm (with $T = \sqrt{\bar{T} \cdot \underline{T}}$ compared to the traditional binary search (with $T = \bar{T} + \underline{T}/2$). The results are summarized in Table II. Note that the solution could be changed compared to NEW since each time the approximation result to the decision integer linear program may be different. However, one can see that the solution quality difference is slight, but the runtime difference is quite significant. For example, the logarithmic solution search runs up to $543.5/135.5 = 4.0\times$ faster for the design vitro1-2.

TABLE II
THE RESULTS OF NEW W/ STANDARD BINARY SOLUTION SEARCH

Testcase		NEW w/ standard binary solution search	
Name	Area Constraint	Timing	CPU(s)
vitro1-1	9×9	54	291.6
vitro1-2	8×8	56	543.5
vitro2-1	8×8	42	219.8
vitro2-2	7×7	44	93.2
vitro2-3	6×6	44	79.8
vitro3-1	7×7	44	35.6
vitro3-2	6×6	44	34.3
vitro3-3	5×5	42	17.8

TABLE III
THE RESULTS OF VARIATION-AWARE OPTIMIZATIONS. TIMING INCREASE IS COMPUTED THROUGH COMPARING THE COMPLETION TIME OF NEW AND NEW W/ VARIATIONS

Testcase		Variation aware design			
Name	Area Constraint	Timing	Timing Increase	CPU(s)	Yield
vitro1-1	9×9	58	3.6%	2889.2	100%
vitro1-2	8×8	58	7.1%	1933.2	100%
vitro1-3	7×7	60	11.1%	1544.2	100%
vitro1-4	6×6	61	13.0%	1955.3	100%
vitro2-1	8×8	45	7.1%	1690.9	100%
vitro2-2	7×7	47	11.9%	1130.0	100%
vitro2-3	6×6	47	6.8%	931.7	100%
vitro3-1	7×7	45	2.3%	523.4	100%
vitro3-2	6×6	45	2.3%	528.3	100%
vitro3-3	5×5	47	11.9%	275.7	100%
Average			7.7%		100%

B. Variation-Aware Design

We compare the variation-aware placement with the variation-unaware placement. To explore the best trade-off between the yield and completion time, a search for the best solution satisfying the yield constraint is applied by varying the parameter β. The yield constraint is set to 99%. The results with best β are summarized in Table III, where the timing increase is computed as the ratio of the completion time of NEW w/ variations over that of NEW minus 1. From Tables I and III, we make the following observations.

- For all testcases, the yields of the variation-unaware design computed by NEW are very small. For example, the yield of testcase of vitro3-3 is 0 and the largest yield is 10.5% for testcase of 3-2. The average yield is only 2%. When considering Gaussian variations, for each module m with an original height of $h(m)$, the variational height has a possibility of 1/2 to be greater than $h(m)$, which could lead to overlap if the computed variation-unaware placement is quite compact (i.e., with small amount of empty space). A rough analysis would then show that the possibility of an overlap (when considering variations in a variation-unaware placement solution) could reach $1 - (1/2)^n$ for n modules. Although this number would not be accurate, one

can get the sense on why the yield is very small. This also demonstrates that without considering variations, it is difficult to design a high-performance lab-on-a-chip.

- For all testcases, our variation-aware design can satisfy the yield constraint. It demonstrates that these designs are significantly insensitive to variations, which means that the NEW w/ variations is effective. On the other hand, the completion time of the design computed by NEW is smaller than that of NEW w/ variations. However, the great improvement of the yield outweighs the increase of completion time.

VI. Conclusion

In this paper, a variation-aware optimization technique is proposed for the lab-on-a-chip component placement. As a large number of variables and constraints significantly impact the efficiency in computation, the multiscale technique including the grid coarsening and fine-scale tuning is explored. Meanwhile, since the quality of microfluidic lab-on-a-chip design is highly dependent on the variations, this paper designs the variation-aware technique for lab-on-a-chip component placement using Latin Hypercube sampling based Monte Carlo simulation. The simulation results on standard benchmark designs demonstrate that the proposed technique is effective and outperform the state-of-the-art work, with up to 65.9% reduction in completion time. In addition, our variation-aware technique can always satisfy the yield constraint with small degradation in completion time.

References

[1] T. Mukherjee, "Design automation issues for biofluidic microchips," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2005.

[2] F. Su, K. Chakrabarty, and R. Fair, "Microfluidics-based biochips: Technology issues, implementation platforms and design automation challenges," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, pp. 211–223, 2006.

[3] I. E. D. Chip [Online]. Available: http://www.infineon.com

[4] Advances in Lab-on-Chip and Microfluidics Technologies for Drug Discovery and Clinical Diagnostics Technical Insights, Frost & Sullivan, 2007 [Online]. Available: http://www.researchandmarkets.com/reports/455960/advances_in_lab_on_chip_and_microfluidics

[5] V. Srinivasan, V. Pamula, M. Pollack, and R. Fair, "Clinical diagnostics on human whole blood, plasma, serum, urine, saliva, sweat and tears on a digital microfluidic platform," in *Proc. Micro Total Anal. Syst. (μTAS)*, 2003.

[6] V. Srinivasan, V. Pamula, and R. Fair, "An integrated digital microfluidic lab-on-a-chip for clinical diagostics on human physiological fluids," *Lab Chip*, vol. 4, pp. 310–315, 2004.

[7] M. Pollack, R. Fair, and A. Shenderov, "Electrowetting-based actuation of liquid droplets for microfluidic applications," *Appl. Phys. Lett.*, vol. 77, pp. 1725–1726, 2000.

[8] K. Bohringer, "Modelling and controlling parallel tasks in droplet-based microfluidic systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, pp. 329–339, 2006.

[9] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang, "Placement of digital microfluidic biochips using the t-tree formulation," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2006, pp. 931–934.

[10] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *ACM J. Emerging Technol. Comput. Syst.*, vol. 3, no. 4, 2008, Art. no. 16.

[11] N. Viswanathan and C. Chu, "Fastplace: Efficient analytical placement using cell spreading, iterative local refinement and a hybrid net model," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 722–733, 2005.

[12] P. Maidee, C. Ababei, and K. Bazargan, "Timing-driven partitioning-based placement for island style FPGAs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 3, pp. 395–406, 2005.

[13] J. Cong and M. Xie, "A robust mixed-size legalization and detailed placement algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 8, pp. 1349–1362, 2008.

[14] F. Su and K. Chakrabarty, "Module placement for fault-tolerant microfluidics-based biochips," *ACM Trans. Design Autom. Electron. Syst.*, vol. 11, pp. 682–710, 2006.

[15] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math. Oper. Res.*, vol. 17, no. 1, pp. 36–42, 1992.

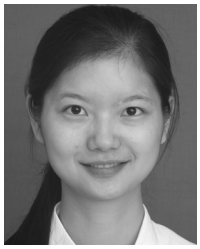
[16] S. Shah, A. Srivastava, D. Sharma, D. Sylvester, D. Blaauw, and V. Zolotov, "Discrete VT assignment and gate sizing using a self-snapping continuous formulation," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2005, pp. 705–712.

[17] G. Fishman, *Monte Carlo: Concepts, Algorithms and Applications*. New York: Springer, 1995.

[18] M. Mckay, R. Beckman, and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[19] S. Hu and J. Hu, "Unified adaptivity optimization of clock and logic signals," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2007, pp. 125–133.

[20] D. Bertsimas and M. Sim, "The price of robustness," *Oper. Res.*, vol. 52, no. 1, pp. 35–53, 2004.



Chen Liao received the M.S. degree from Michigan Technological University, Houghton, in 2010. She is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, Michigan Technological University.

Her research interests are in the area of computer-aided design of VLSI circuits and combinatorial optimizations.

Ms. Liao received the A. Richard Newton Graduate Scholarship from DAC 2009.

Shiyan Hu (SM'10) received the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2008.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at Michigan Technological University, Houghton, where he serves as the Director of Michigan Tech VLSI CAD Research Lab. He was a Visiting Professor with the IBM Austin Research Lab during Summer 2010. He has published over 50 journal and conference papers. His research interests are primarily in VLSI computer-aided design including nanoscale interconnect optimizations, low power optimizations, and design for manufacturability.

Prof. Hu received the Best Paper Award Nomination from ICCAD 2009. He has served as the Technical Program Committee (TPC) members for a number of conferences such as ICCAD, ISPD, ISQED, ISVLSI, SOCC, ICM, and ISCAS.