Synthesis of Multiplexed Biofluidic Microchips

Anton J. Pfeiffer, Tamal Mukherjee, *Member, IEEE*, and Steinar Hauan

***Abstract*—Lab-on-a-chip (LoC) devices are a class of microfluidic chip-based systems that show a great deal of promise for complex chemical and biological sensing and analysis applications. An approach for full-custom LoC design, which leverages optimal design techniques and system-on-a-chip (SoC) physical design methods, is being developed. Both physical design of the chip and microfluidic performance are simultaneously considered to obtain complete LoC layouts. The proposed approach is demonstrated by designing multiplexed capillary electrophoresis (CE) separation microchips. The authors believe that this approach provides a foundation for future extension to LoC devices in which many different complex chemical operations are performed entirely on-chip.**

***Index Terms*—Biochips, lab-on-a-chip design, microfluidic layout.**

I. Introduction

MICROCHIPS represent an advantageous platform for microscale chemical sensors and analytical devices. Microfluidic devices that are fast, accurate, portable, and readily automated can be fabricated inexpensively using methods adapted from the semiconductor industry [1]. This paper explores model-based design of microfluidic channel systems that fall within a broad category known as lab on a chip (LoC) [1]. An LoC is essentially a miniaturized microchip implementation of a macroscale analytical chemistry laboratory. LoCs have already seen a great deal of use within the life-science and biomedical industries for applications in genomics, proteomics, and combinatorial chemistry [1]. An exciting direction for LoC development concerns point-of-care (PoC) medical devices. PoC devices must be both highly compact and capable of efficiently performing complex chemical operations. However, the design of LoC devices is complicated by performance limits arising from complex physiochemical phenomena and difficult layout and integration issues.

Design methods for various LoC components have recently been developed. A partitioning-based approach has been successfully used to layout regular arrays of DNA probes on microchips [2]. Shape optimization methods have been employed to generate high-performance channel geometries [3]. Both heuristic and optimal design methods for capillary electrophoresis (CE) channel systems have been developed [4]. Each of these design methods pertain to subcomponents or subsystems within a complete LoC. In addition, a design methodology for reconfigurable droplet-based microfluidic systems has been investigated [5]. In this architecture, droplets rather than continuous streams of fluid are moved throughout the chip.

Here, we present our work toward automating the design of complete LoC systems. For instance, for DNA analysis, our methodology would allow for the simultaneous design and layout of the channel network that would be needed to sample body fluid, biochemically preprocess the fluid, and bring the analyte to the DNA sensor array. The DNA array itself could be designed simultaneously with the rest of the chip using [2].

Our approach combines system-on-a-chip (SoC) circuit floorplanning and routing techniques with rigorous subsystem design optimization to develop methods for full-custom LoC design. We focus on the design of multiplexed systems in which multiple noninteracting subsystems of similar function are compacted onto a single chip. Multiplexing exploits device level parallelism to increase analysis speed and decrease the required sample size, which provides redundancy, robustness, and potential for combinatorial experiments. This makes multiplexed LoCs applicable to PoC applications.

We demonstrate our approach by designing microchips containing multiplexed CE subsystems. CE is a common on-chip separation technology that is capable of separating complex biological mixtures [1] and is thus relevant for use in PoC devices.

The organization of the paper is as follows. Section II presents the background and design considerations of microchip-based CE. Section III reviews subsystem simulation and optimization. An algorithm for designing and placing subsystems on a chip is introduced in Section IV. Section V presents an approach for connecting subsystems and fluid input/output (I/O) structures on the chip. Results and completed designs are discussed in Section VI. Finally, extensions and future directions are summarized in Section VII.

II. CE on Microchips

A. Background

Fig. 1 shows the major components of a simple chip-based capillary electrophoretic separation system. The chip is composed of the following: 1) an injector where the analyte, or mixture of chemical species to be separated, enters the system; 2) the separation channel where the analyte separates into unique species bands; and 3) the detector where species bands are detected; typically optically or electrically. The detector output or electropherogram is shown in Fig. 1(d), where the

Manuscript received February 15, 2005 revised June 7, 2005. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the U. S. Air Force Research Laboratory under Agreement F30602-01-2-0987 and by the National Science Foundation (NSF) under Award CCR-0325344. This paper was recommended by Associate Editor K. Chakrabarty.

A. J. Pfeiffer and S. Hauan are with the Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: pfeiffer@cmu.edu; hauan@cmu.edu).

T. Mukherjee is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: tamal@ece.cmu.edu).

Digital Object Identifier 10.1109/TCAD.2005.855931

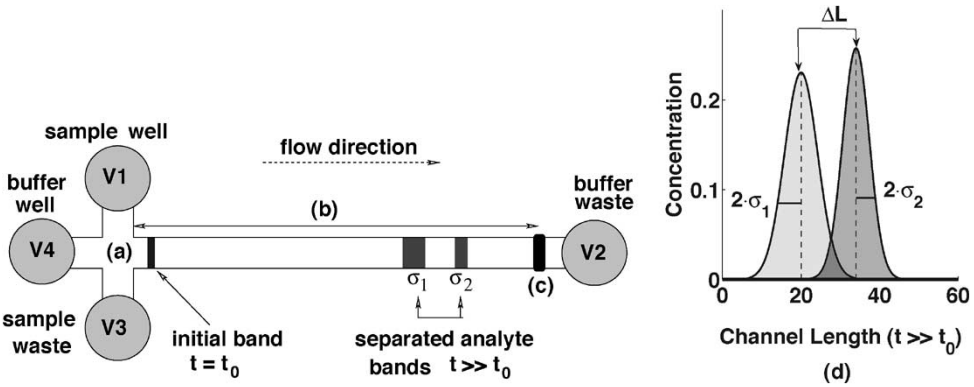


Fig. 1. Simple capillary electrophoresis schematic. (a) Injector. (b) Separation channel length L. (c) Detector. (d) Resulting detector output (electropherogram) showing the two species bands as Gaussian distributions.

species band concentration profiles are represented as Gaussian distributions. The microchannels are filled with a carrier electrolyte known as a buffer solution. Electrodes positioned in the four wells generate an electric field through the buffer that imparts specific velocities to the species within the channel.

The CE separation process has two phases, namely: 1) a loading phase, where a continuous analyte stream is drawn from the sample well to the sample waste well by applying a direct current (dc) potential between V1 and V3; and 2) a dispensing stage, where a band of analyte material is injected into the separation channel by applying a potential between V2 and V4 [6].

Electrophoretic separation occurs because of the differential transport of charged species in the presence of an electric field generated between V2 and V4. As an analyte mixture travels down the separation channel, the species within the mixture separate into bands according to their electrophoretic mobilities μ [7]. However, all the while species bands are traveling down the channel, they are broadening or dispersing due to factors such as diffusion, channel geometry, Joule heating, adsorption, and electromigration [7]. Band broadening not only impedes separation but also results in the dilution or reduction of an analyte band's concentration.

The total dispersion for a given analyte band is quantified in terms of the variance σ^2_{tot} of that band's concentration distribution. The total dispersion can be estimated by summing each of the major contributions to dispersion (1), given as

$$\sigma^2_{\text{tot}} = \sigma^2_{\text{inj}} + \sigma^2_{\text{diff}} + \sigma^2_{\text{geom}} + \sigma^2_{\text{JH}} + \cdots. \tag{1}$$

In (1), the contributions to dispersion from the injector σ^2_{inj} [8], diffusion σ^2_{diff} [9], channel geometry σ^2_{geom} [9], and Joule heating σ^2_{JH} [9] are all functions of the separation channel length L and the applied voltage V.

The measure that describes the degree of separation between a pair of migrating analytes, i and j, is the resolution $\hat{R}_{i,j}$ (2), given as

$$\hat{R}_{i,j} = \frac{\Delta L}{2(\sigma_i + \sigma_j)} = \frac{(\mu_i - \mu_j)L}{2\mu_i(\sigma_i + \sigma_j)}. \tag{2}$$

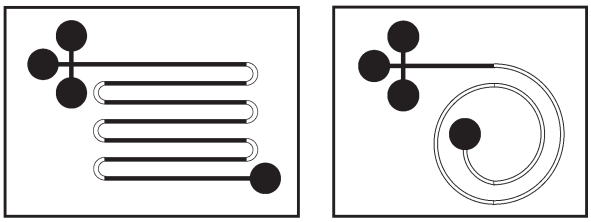
Here, ΔL is the distance between the peaks of the concentration distributions of bands i and j (Fig. 1), where μ_i and μ_j are the species mobilities with $\mu_i > \mu_j$. The denominator of (2) quantifies the average dispersion of bands i and j in terms of the standard deviations of their concentration distributions, σ_i and σ_j. A value of $\hat{R} \geq 1.5$ implies baseline resolution [7], which we use as a lower bound on separation performance in our studies.

B. Design Considerations

The separation channel length L and the applied voltage V are the key design variables that we manipulate to achieve separation. We consider channel width ω to be a fixed variable, since its feasible range is generally small. The design variables interact in complex ways. For example, for constant L, increasing V will typically reduce diffusional dispersion σ^2_{diff}, but may increase the dispersion resulting from channel geometry σ^2_{geom} and bulk fluid heating σ^2_{JH}. Likewise, for constant V, increasing L provides more time for analyte bands to move apart, but causes σ^2_{diff} to increase. In this case, σ^2_{geom} and σ^2_{JH} can become negligible when compared to σ^2_{diff}. Additionally, an analyte band's dispersion is inversely related to its concentration. A band's peak concentration C may drop below the limit of detection for a particular detection technology when dispersion is too great [7].

In practice, for difficult separations ($\mu_i \approx \mu_j$), the available voltage source becomes a limiting factor and long separation channels are required to achieve baseline resolution. However, long channels can only be made to fit on a microchip by adding turns or bends to the design, which results in a phenomenon called turn induced dispersion [10]. The topology of a design, or interconnectivity of channel sections, can also be shown to have a significant impact on the performance of a design [11]. Thus, minimizing design area and maximizing separation performance are conflicting design goals.

Two common topologies found in the literature are the serpentine and spiral topologies [Fig. 2(a) and (b)]. In our view, channel topologies can be built from a library of channel sections consisting of straights, U-bends, elbows, crosses, tees, and wells as described in [9]. For example, Fig. 2(a) is composed of a cross injector followed by an alternating series of straight (black) and U-bend (white) channel sections. We use the serpentine topology for every subsystem in our designs, because serpentines are compact and allow for easy access to I/O ports in a multiplexed layout.



(a) (b)

Fig. 2. Two common compact channel topologies. (a) Serpentine topology composed of an injection cross with wells (black circles) straight channels (black) and U-bends (white). (b) Spiral topology composed of an injection cross with wells, U-bends, and an elbow.

Our design approach uses accurate parameterized reduced order models of CE that are derived from the partial differential equations that describe the electric field, species transport, and thermal effects within channel sections [9]. These models were verified against experimental designs presented in the literature and against a suite of finite element simulations. They can be connected together to create a flexible CE simulation framework that is amenable to both iterative studies and design optimization.

In LoC devices, the subsystem design and ultimate chip layout are intimately linked due to the influence that channel geometry has on device performance. Therefore, a solution approach that is capable of simultaneously considering both subsystem design and overall system layout is required. Currently, creating multiplexed designs [12] requires a substantial investment of time, manpower, and expertise. Typical design cycles include extensive laboratory experimentation, time-consuming numerical simulations and manual verification processes. The adaptation of SoC techniques for chip layout coupled with an effective methodology for subsystem design has the potential to reduce the length of design cycles to only days and to facilitate work toward innovative applications for LoC technology.

III. CE Subsystem Simulation Optimization

In this section, we review our simulation-based approach for optimally designing single CE subsystem topologies. In Section IV, we extend these concepts to multiplexed LoC design.

Serpentine topologies can be represented as a vector $\mathcal{T} = [L_1^{\text{str}}, L_1^{\text{trn}}, L_2^{\text{str}}, L_2^{\text{trn}}, \ldots]$. The odd elements of $\mathcal{T}$ are straight section lengths L_n^{str} where $n = 1, \ldots, \lceil (1/2)|\mathcal{T}| \rceil$, and the even elements are turn lengths L_m^{trn} along the centerline radius of the channel, where $m = 1, \ldots, \lceil (1/2)(|\mathcal{T}| - 1) \rceil$. Here, we show a functional representation of our general electrophoretic channel simulator (GSIM) and the information relevant to the current problem, written as

$$[X, Y, E, \hat{R}_{i,j}, C_i] = \text{GSIM}(V, \mathcal{T}, \text{props}). \tag{GSIM}$$

The simulator takes in the operating voltage V, the topology instance $\mathcal{T}$, and an object containing the physiochemical properties of the system props. The simulator internally calculates the design bounding box dimensions, X and Y, which are the smallest horizontal and vertical spans that contain the entire topology. The simulator also returns the electric field strength E as well as the separation resolution $\hat{R}_{i,j}$ and peak concentrations C_i for each band at the end of the channel.

$$\begin{aligned}
\min \quad & A_{\mathcal{T}} = X \cdot Y \\
\text{s.t.} \quad & \hat{R}_{i,j} \geq \hat{R}_{spec} && \forall i, j \in props \\
& C_i \geq C_{min} && \forall i \in props \\
& E \leq E_{max} \\
& V \leq V_{max} \\
& X \leq X_{max} \\
& Y \leq Y_{max} \\
& \omega \leq \mathcal{T}_{(2k-1)} \leq X_{max} - 2 \cdot PAD && \forall k = 1 \ldots |\mathcal{T}| \\
& \pi \cdot r_{min} \leq \mathcal{T}_{(2k)} \leq \pi \cdot \frac{Y_{max-2 \cdot PAD-\omega}}{2} \\
& \pi \cdot PAD \leq \mathcal{T}_{(2k)} \\
& X, Y, V, E \geq 0
\end{aligned} \tag{MSA}$$

Fig. 3. Minimum subsystem area formulation.

In previous work [11], we developed a nonlinear program (NLP) formulation to obtain the minimum subsystem area (MSA) of serpentine channel topologies (Fig. 3). In this formulation, the objective is to minimize the area of a given topology $A_{\mathcal{T}}$. GSIM is implicitly a part of MSA. The resolution $\hat{R}_{i,j}$ between species i and j as well as the concentration C_i of every species band must be greater than the specified values, $\hat{R}_{\text{spec}}$ and C_{min}, respectively, to ensure detectability. Furthermore, E and V must be less than the operational specifications, E_{max} and V_{max}, to prevent the possibility of boiling or electrical arcing at the electrodes. The entire design must fit within a bounding box with a horizontal dimension of X_{max} and a vertical dimension of Y_{max}.

Bounds on the lengths of straight sections $\mathcal{T}_{(2k-1)}$ and the lengths of turns $\mathcal{T}_{(2k)}$ are also invoked. The minimum turn radius, $r_{\text{min}} \geq L_n^{\text{trn}}/(\pi\omega)$, where ω is the channel width, is set such that optimal solutions are not found outside the region of model validity. The fabrication method determines the feature spacing PAD of elements on the chip (e.g., spacing between adjacent channels). While MSA is a natural and rigorous formulation of the serpentine design problem, it can be difficult to solve [11].

Using MSA directly for multiplexed layout would rapidly increase both problem size and complexity. Based on previous work, we have simplified GSIM so that it is more amenable to inclusion within a multiplexed design framework [4]. In our simplified simulator (SSIM), the design variables are the bounding box dimensions X and Y, the voltage V, and the topology instance $\mathcal{T}$, which is now a scalar value representing the total number channel sections making up the serpentine. The length of each channel section is no longer allowed to vary independently. Instead, for any given X, Y, and $\mathcal{T}$, a

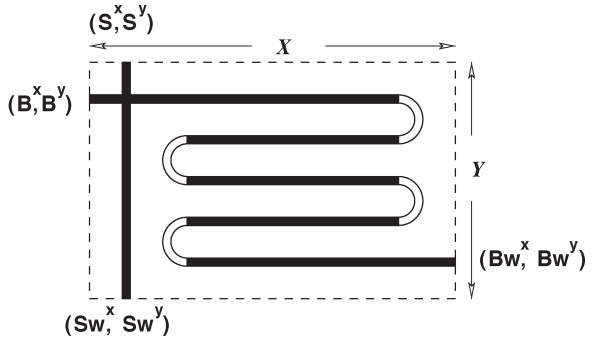


Fig. 4. Schematic of a serpentine subsystem showing dimensions, X and Y, and port locations [(S^x, S^y), etc.].

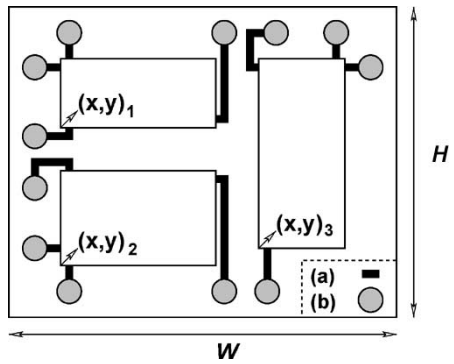


Fig. 5. Three-subsystem-chip layout instance showing chip dimensions, W and H, and subsystem locations, $(x, y)_i$, with (a) auxiliary channels and (b) fluid I/O wells.

completely symmetric serpentine channel layout that uses all of the available area is calculated deterministically. The number of design variables in SSIM is constant and no longer depends on the size of $|\mathcal{T}|$ as it did in GSIM

$$[E, \hat{R}_{i,j}, C_i] = \text{SSIM}(X, Y, V, \mathcal{T}, \text{props}). \quad \text{(SSIM)}$$

Our computational evidence supports replacing GSIM with SSIM, since both simulators yield the same optimal solution when used in our NLP. The only exceptions occur when $\hat{R}_{\text{spec}}$ is set unrealistically low, since many equivalent alternative solutions are then possible.

IV. Multiplexed Floorplanning

SSIM and the constraint set of MSA form the basis for the design of each individual subsystem. For multiplexed systems, the constraints on individual subsystem size, $X_{\max}$ and $Y_{\max}$, are relaxed, since we are concerned with total chip area and not individual subsystem area.

A subsystem schematic is shown in Fig. 4. The coordinate locations of the sample port (S^x, S^y), buffer port (B^x, B^y), sample waste (Sw^x, Sw^y), buffer waste (Bw^x, Bw^y), and subsystem dimensions, X and Y, are labeled. We assume that all subsystems will be similar to the topology shown in Fig. 4, with ports deterministically located one per side. This configuration allows for convenient subsystem I/O and heuristically reduces the potential for routing congestion around each subsystem.

Fig. 5 is an example instance of a design containing three subsystems. Auxiliary channels (a) transport fluid between subsystem ports and particular wells (b). The overall chip design dimensions are denoted by its width W and height H. The wells form the world-to-chip interface and allow fluids to be introduced and removed from the chip. The position of each subsystem and well is defined by an (x, y) coordinate point in its lower left corner. Notice that the wells are relegated to the chip edges. This requirement makes it easier to transport fluids on and off the chip. Figs. 4 and 5 give a schematic description of the simultaneous subsystem design and layout problem.

A. Floorplan Problem Definition

We define our floorplan problem as follows: Given a set of subsystems $\mathcal{N}$ and their associated input and output wells $\mathcal{W}$, obtain a planar arrangement $\mathcal{A}$ of subsystems and wells that does not exceed the total chip height H or width W, and where all subsystems with respect to the constraint set in MSA are feasible. The objective is to choose the arrangement of subsystems and wells that is most compact and where fluid I/O wells are positioned on the edge closest to their associated subsystem ports.

Our problem is similar to a standard very large scale integrated (VLSI) floorplan problem in which chip area and wire length are minimized. However, unlike many typical VLSI floorplan formulations, neither the subsystem aspect ratio nor the total area is known *a priori*, since subsystem dimensions are a function of the constraint set shown in MSA. We assume a building-block layout style for our problem as opposed to a grid-point assignment layout style [13], because the dimensions of each subsystem are highly variable. We allow the dimensions of each subsystem to vary so that for a given arrangement $\mathcal{A}$, the optimal values of W and H are obtained.

B. Floorplan Problem Formulation

In previous work, we developed a rigorous floorplanning formulation using a concise modeling technique known as general disjunctive programming (GDP) [14]. Typically, GDPs are translated into mixed integer nonlinear programs (MINLPs) and solved using deterministic optimization approaches that combine gradient-based search methods with branch and bound [15]. However, the floorplanning problem that we have formulated is a rectangle packing problem that has been shown to be NP-hard [16]. In addition, floorplan optimization using branch and bound [17] or conventional mixed integer linear programming (MILP) approaches [18] often have difficulty handling problems of realistic size. Since we are coupling rectangle packing with a nonconvex (nonlinear) physical problem, our problem should be at least as hard as rectangle packing alone. Our computational experience supports this conjecture. Solving our GDP formulation directly using an MINLP solver requires hours of central processing unit (CPU) time for problems of trivial size (i.e., $|\mathcal{N}| \leq 4$). While we consider our GDP formulation to be a rigorous description of the multiplexed CE floorplanning problem, it is apparent that a heuristic is required to obtain good solutions for problems instances of realistic size.

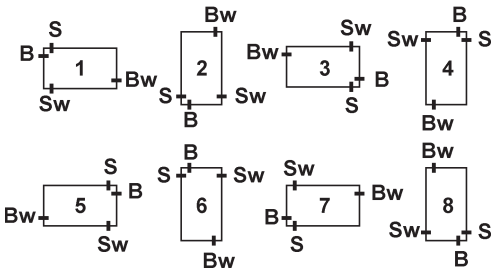


Fig. 6. Eight possible subsystem orientations.

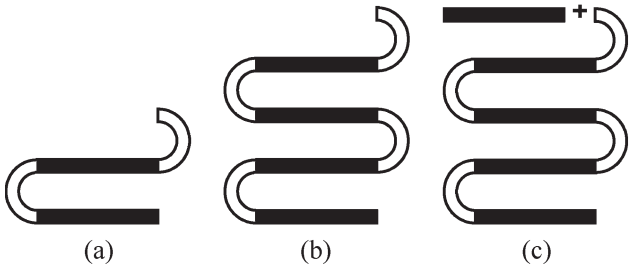


Fig. 7. (a) One serpentine unit ($\tau = 1$). (b) Two serpentine units ($\tau = 2$). (c) Completing the topology with an initial straight section.

In the following sections, we discuss four key parts of our floorplanning problem formulation, namely: 1) subsystem orientation; 2) serpentine topology size; 3) relative subsystem positions; and 4) well placement.

1) Subsystem Orientation: Each subsystem can have one of eight possible orientations (Fig. 6). Different orientations contribute to design compaction. We define a vector $\boldsymbol{Z} = (z_1, z_2, \ldots, z_{|\mathcal{N}|})$, which contains the subsystem orientation labels $z_i \in \{1, \ldots, 8\}$, where i is a unique element of $\mathcal{N}$. Each z_i defines the orientation of subsystem i and allows for the deterministic calculation of that subsystem's port locations. For example, the buffer port location B_i for orientation $z_i = 4$ is $B_i = (B^x, B^y) = (x_i + w_i - \omega - \text{PAD}, y_i + h_i)$. The port locations are a function of the subsystem's location x_i and y_i, width w_i, and height h_i as well as the feature spacing PAD and channel width ω. However, recall that SSIM has no knowledge of subsystem orientation. Thus, (3) must be applied to translate the general subsystem dimensions, X_i and Y_i, used in the simulator to the actual subsystem dimensions w_i and h_i

$$w_i = \begin{cases} X_i, & \text{if } z_i \text{ even} \\ Y_i, & \text{otherwise} \end{cases} \quad h_i = \begin{cases} X_i, & \text{if } z_i \text{ odd} \\ Y_i, & \text{otherwise} \end{cases}. \tag{3}$$

2) Serpentine Topology Size: The types list $\mathcal{T}$ can be reduced to a scalar value τ, indicating the number of serpentine units within a subsystem topology (Fig. 7). Fig. 7(a) and (b) shows a topology with $\tau = 1$ and $\tau = 2$, respectively. The topology is completed by adding an initial straight section to the serpentine [Fig. 7(c)]. We define a vector $\boldsymbol{S} = (\tau_1, \tau_2, \ldots, \tau_{|\mathcal{N}|})$, which contains the number of serpentine units $\tau_i \in \{1, \ldots, \tau^{\text{UB}}\}$ for each subsystem $i \in \mathcal{N}$. Here, τ_i can range from 1 to a user-specified upper bound τ^{UB}. Our experience indicates that a conservative value for τ^{UB} is 15, which corresponds to a serpentine composed of 61 individual channel sections where the number of sections is equal to $4\tau_i + 1$.

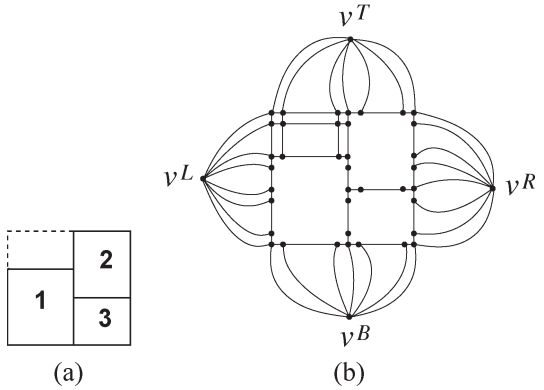


Fig. 8. (a) Compacted arrangement of subsystems. (b) Corresponding well placement graph.

3) Relative Subsystem Positions: We encode the constraints to prevent subsystems from overlapping using a VLSI floorplan representation known as the sequence pair (SP) [16]. SP encodes the {left, right, above, below} spatial relations between objects in the plane into a concise structure. This structure is readily searchable using standard heuristic methods such as simulated annealing (SA) [19].

A SP is composed of two $|\mathcal{N}|$-tuples $\Gamma = (\Gamma_+, \Gamma_-)$, where Γ_+ and Γ_- are ordered lists of the subsystem labels in $\mathcal{N}$. An SP maps to $(|\mathcal{N}|(|\mathcal{N}| - 1)/2)$ linear math programming constraints as shown in (4)

$$\begin{aligned} x_i + w_i \le x_j &\leftarrow (\ldots i \ldots j \ldots, \ldots i \ldots j \ldots) \\ y_i + h_i \le y_j &\leftarrow (\ldots j \ldots i \ldots, \ldots i \ldots j \ldots). \end{aligned} \tag{4}$$

Here, i and j are unique subsystem labels in $\mathcal{N}$. If i appears before j in both Γ_+ and Γ_-, then subsystem i is left of j. Likewise, if i appears after j in Γ_+ and before j in Γ_-, then subsystem i is below j. The SP representation has several attractive features. First, the solution space, although large, is finite, i.e., $\Psi \propto (|\mathcal{N}|!)^2$. Second, the neighborhood of a particular arrangement $\mathcal{A}$ is readily constructible through simple perturbations of Γ. Furthermore, every evaluation of Γ results in a feasible planar arrangement, which is not true for all nonslicing floorplan representations (e.g., CBL [20]). Finally, SP is a general floorplan representation, which means that optimal solutions are not excluded due to assumptions about floorplan structure.

4) Well Placement: Confining the fluid I/O wells to the edges of the chip facilitates fluid transport on and off the chip. We have developed a graph-based approach that places a well on the chip edge closest to its associated port.

For a given arrangement $\mathcal{A}$ of subsystems, we construct an undirected edge-weighted planar graph $G_{\mathcal{A}} = \langle \mathcal{V}, \mathcal{E} \rangle$ by noting that a compacted arrangement of subsystems is essentially a connected planar graph (Fig. 8). The graph we create is similar to a placement graph used in VLSI circuit design [13].

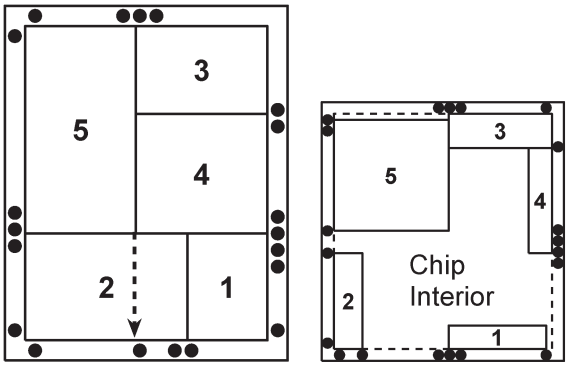


Fig. 9. (a) Poor well placement based on rectilinear model (dotted arrow). (b) Compacted arrangement pulled apart by weighted-sum objective function.

We construct the graph by first generating a set of preliminary nodes by taking the Cartesian product of the bounding box dimensions, subsystem locations, and port locations (5)

$$\text{nodes} = \{0, W, x_i, S_i^x, Sw_i^x, B_i^x, Bw_i^x \quad \forall i \in \mathcal{N}\} \times \{0, H, y_i, S_i^y, Sw_i^y, B_i^y, Bw_i^y \quad \forall i \in \mathcal{N}\}. \quad (5)$$

A set of preliminary edges is formed by connecting each node with its nearest neighbor in the x and y directions. The graph's vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ are generated by removing all the nodes and associated edges that reside within subsystem boundaries. Finally, four supernodes, v^L, v^T, v^R, and v^B, are added to the vertex set. The supernodes are connected to the leftmost, topmost, rightmost, and bottommost nodes of $G_{\mathcal{A}}$ as shown in Fig. 8(b).

The weights assigned to edges within the interior of $G_{\mathcal{A}}$ are simply the distances in the x and y directions between connected adjacent nodes. The edges connecting supernodes are weighted based on the maximum bounding box dimension. This prevents short-circuit paths through the graph during the well placement procedure. Each well is placed on one of the four sides of $\mathcal{A}$ as follows:

1) Construct $G_{\mathcal{A}}$ and initialize sets: left = right = top = bottom = $\emptyset$.
2) Select a port vertex $p \in v_p$, where $v_p \subset \mathcal{V}$ is the set of ports.
3) Determine the shortest path between p and supernodes v^L, v^T, v^R, and v^B. Define path lengths as d^L, d^T, d^R, and d^B.
4) Find $d_{\min}(p) = \min\{d^L, d^T, d^R, d^B\}$ and assign p to left if $d_{\min}(p) = d^L$, to top if $d_{\min}(p) = d^T$, to right if $d_{\min}(p) = d^R$, or to bottom if $d_{\min}(p) = d^B$.
5) Order wells in left or right based on the y-coordinate of its port.
6) Order wells in top or bottom based on the x-coordinate of its port.
7) Obtain a total routing length estimate, $\phi = \sum_{p \in \mathcal{P}} d_{\min}(p)$, where $\mathcal{P}$ is the set of shortest routes between ports and wells in $G_{\mathcal{A}}$ (note that $|\mathcal{P}| = |\mathcal{W}| = 4|\mathcal{N}|$).

In step 3), we apply a shortest path algorithm to efficiently search $G_{\mathcal{A}}$. Steps 5) and 6) can be performed using an efficient sorting method.

Our solution methodology allows us to avoid two fundamental problems that would result from a more simplistic formulation. The first possible problem is illustrated in Fig. 9(a). Here, wells are placed using a standard rectilinear distance metric. However, the shortest rectilinear distance between a port and a well, indicated by the dotted arrow, may pass through a subsystem and is, therefore, a poor estimate of the true distance between a port and the chip edge. The distance estimates produced using $G_{\mathcal{A}}$ are far more accurate, because they account for the fact that routes must go around subsystems.

A second problem can arise if a common VLSI floorplanning objective function of the form: $\alpha_1 A + \alpha_2 \text{WL}$, is used, where A is chip area, WL is wire length, and α_1 and α_2 are weighting factors. This objective works well for typical circuit design problems, because circuit elements are highly interconnected within the chip interior and a wide variety of α_1 and α_2 values will produce satisfactory results. Since we connect single ports within the chip's interior to single wells on the chip's edges, overweighting α_2 pulls a compact arrangement apart and results in a large amount of unused space within the chip interior [Fig. 9(b)]. Since we generally do not know good values for α_1 and α_2 *a priori*, we first compact the subsystems and lock their positions. Then we place the wells, thereby decoupling the continuous variables of the problem while retaining the global nature of the combinatoric variables Γ, $\boldsymbol{Z}$, and $\boldsymbol{S}$.

C. Floorplan Solution Method

Fig. 10 is a flowchart illustrating our simultaneous design and layout approach. The main idea is to use a probabilistic search heuristic such as SA to deal with the combinatorial aspects of the problem and an efficient gradient-based method for the remaining continuous-space problem. We have chosen this hybrid approach, because SA has been shown to be an effective search method for difficult combinatorial problems, but is generally inferior to gradient-based methods on well-posed continuous-space problems [21].

In our approach, the combinatorial states Γ, $\boldsymbol{Z}$, and $\boldsymbol{S}$ are instantiated by the SA algorithm resulting in an NLP. A new NLP is dynamically constructed for each new problem instance. This approach is conceptually similar to floorplanning methods meant to handle soft blocks (i.e., circuit modules with variable dimensions), where the resulting instantiated problem is an linear programming (LP) [22] or a convex program [23]. In our case, we are not only concerned with floorplanning, but also with the performance of each subsystem on the chip, where subsystem performance is a strong function of available chip area. The resulting NLP is nonconvex, which results in convergence issues and increased computational complexity that are not present in linear or convex formulations. We combat these problems using an intuitive initialization procedure and a penalty function constraint-relaxation approach.

1) Stage 1—Instantiation: The algorithm in Fig. 10 begins by obtaining the relevant subsystem and chip design specifications. The search heuristic proposes an SP, $(\Gamma_+; \Gamma_-)$, an

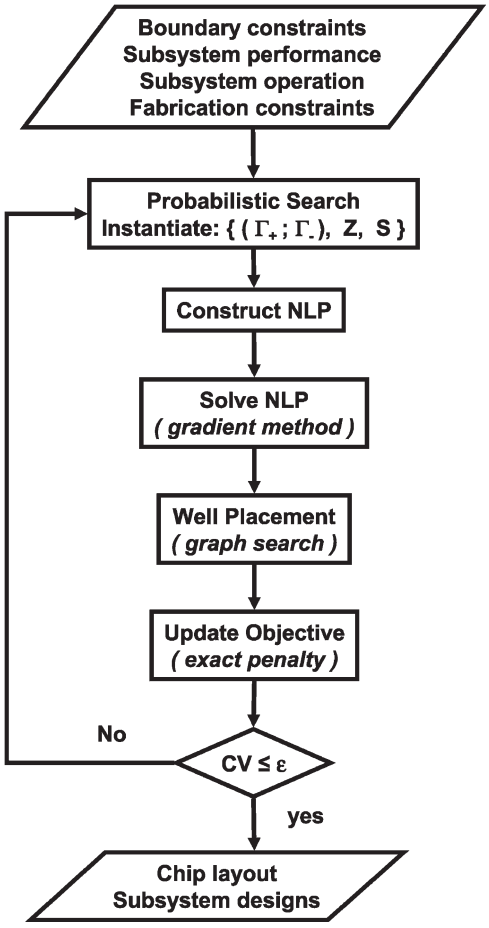


Fig. 10. Flowchart for the floorplanning algorithm.

instance of the block orientation vector $\boldsymbol{Z} = (z_1, \ldots, z_{|\mathcal{N}|})$ and an instance of the subsystem topology vector $\boldsymbol{S} = (\tau_1, \ldots, \tau_{|\mathcal{N}|})$.

For example, suppose that we want to design a chip containing five subsystems. The SA instantiates an SP of $\Gamma = (\langle 5,3,4,2,1\rangle, \langle 2,5,1,4,2\rangle)$, a block orientation vector $\boldsymbol{Z} = (8,2,5,5,1)$, and a topology size vector $\boldsymbol{S} = (4,5,6,4,7)$. The relative position of each subsystem is defined by Γ, however, the dimensions of each subsystem and the resulting compact arrangement have not yet been determined. The instantiated values are then passed to the remaining stages in the procedure.

2) Stage 2—NLP Initialization: We have developed a tailored initialization procedure to aid in the convergence of our NLP. Our initialization procedure generates an upper bound on the chip design area by minimizing the size of each individual subsystem to produce a set of fixed-sized rectangles. The $|\mathcal{N}|$ fixed-size rectangles are generated by minimizing $(X_i + Y_i)$ for each subsystem subject to SSIM and the MSA constraint set using a standard sequential quadratic programming method. The z_i's from Stage 1 along with (3) are used to translate X_i and Y_i to the appropriate subsystem dimensions, w_i and h_i. When the dimensions of each subsystem are known, the (x, y) positions of each subsystem are determined by evaluating the Γ from stage 1 using an efficient SP decoding algorithm [24]. The dimensions and positions are provided as initial values to the compaction NLP described in the following section.

$$
\begin{aligned}
\min \quad & \varepsilon^R + \varepsilon^T \\
\text{s.t.} \quad & [E_i, \hat{R}_i, C_i, \nabla E_i, \nabla \hat{R}_i, \nabla C_i] = \mathrm{SSIM}(X_i, Y_i, V_i, \boldsymbol{\tau}_i, props_i) \\
& w_i = \mathrm{mod}(z_i, 2) \cdot X_i + \neg\, \mathrm{mod}(z_i, 2) \cdot Y_i \\
& h_i = \neg\, \mathrm{mod}(z_i, 2) \cdot X_i + \mathrm{mod}(z_i, 2) \cdot Y_i \\
& x_i + w_i \leq x_j \leftarrow (\Gamma_+; \Gamma_-) \\
& y_i + h_i \leq y_j \leftarrow (\Gamma_+; \Gamma_-) \\
& \varepsilon^R \geq x_i + w_i \\
& \varepsilon^T \geq y_i + h_i \\
& \hat{R}_i \geq \hat{R}_{spec} \\
& C_i \geq C_{min} \\
& E_i \leq E_{max} \\
& V_i \leq V_{max} \\
& x_i, y_i, w_i, h_i, X_i, Y_i, V_i, E_i \geq 0 \quad \forall i \in \mathcal{N}
\end{aligned}
\tag{MCA}
$$

Fig. 11. Minimum chip area formulation.

3) Stage 3—NLP Construction: The minimum chip area (MCA) NLP is dynamically constructed using Γ, $\boldsymbol{Z}$, and $\boldsymbol{S}$ by simultaneously invoking the simulator SSIM and MSA constraint set for every subsystem (Fig. 11). We use automatic differentiation [25] to generate the gradients $(\Delta E_i, \Delta \hat{R}_i, \Delta C_i)$ with respect to X_i, Y_i, and V_i. Equation (3) is used to map the general X_i and Y_i dimensions used in the simulator to the corresponding w_i and h_i for each subsystem. Equation (4) is used to map Γ to a set of relative position constraints. Here, we show the complete formulation of MCA where the objective is to minimize the design width ε^R and height ε^T. Decoding Γ for our five subsystem example results in four horizontal constraints and six vertical constraints. Notice that all of the nonlinear constraints in MCA are embedded within the simulator.

The NLP is solved using a standard general reduced gradient method starting from the initial values generated in stage 2. The solution of the NLP results in an optimally compacted design for a given Γ, $\boldsymbol{Z}$, and $\boldsymbol{S}$. This compacted design is then passed to the well placement stage.

4) Stage 4—Well Placement: Each subsystem is associated with four fluid I/O wells that must be placed on the edges of the chip. For our five subsystem example, each of the 20 fluid I/O wells are assigned to the left, right, top, or bottom sets by applying the well placement algorithm described in Section IV-B-4 to the compacted design obtained in stage 3. In addition, the approximate total routing cost ϕ is also obtained.

Since wells occupy space on the chip, the design dimensions ε^R and ε^T must be updated. The width of the design is updated as shown in (6)

$$
\varepsilon^R = \varepsilon^R + \begin{cases} 2\left(d + 2\mathrm{PAD} + \frac{\omega}{2}\right), & \text{if left} \wedge \text{right} \neq 0 \\ d + 3\mathrm{PAD} + \omega, & \text{elseif left} \veebar \text{right} \neq 0 \\ 2\left(\mathrm{PAD} + \frac{\omega}{2}\right), & \text{otherwise} \end{cases}. \tag{6}
$$

Here, d is the well diameter, PAD is the feature spacing, and ω is the channel width. Equation (6) has three conditions; the first is applied when wells exist on both the left and right sides of a design, the second when wells occupy only the left or right side of a design, and the third when wells occupy neither the left nor right side of a design. The design height is updated similarly, replacing right with top and left with bottom. In each case, ε^R or ε^T is adjusted to satisfy the well placement and feature spacing requirements.

5) Stage 5—Objective and Boundary Constraints: Once the approximate total routing cost ϕ and the updated ε^R and ε^T have been obtained, they are used to construct the objective function that is evaluated by the SA. The only remaining task is to enforce the boundary constraints that are necessary to confine a design to the available chip area.

We have chosen not to incorporate the boundary constraints in the compaction NLP discussed in stage 3, and instead implement them using an exact penalty method [21]. This approach substantially reduces the likelihood that the NLP will return with an infeasible result. Therefore, we can evaluate the progress of our algorithm using information that would otherwise not exist.

Equation (7) shows the objective function evaluated by the SA

$$\begin{aligned} &\min\ \varepsilon^R + \varepsilon^T + \phi + P(\varepsilon^R, \varepsilon^T) \\ &P(\varepsilon^R, \varepsilon^T) = \rho\left(\max(\varepsilon^R - W, 0) + \max(\varepsilon^T - H, 0)\right) \\ &\qquad \rho = \max(W, H). \end{aligned} \tag{7}$$

The first and second terms in (7) are the updated chip dimensions and the total routing length estimate. The third term, $P(\varepsilon^R, \varepsilon^T)$, is the penalty term that enforces the boundary constraints. A penalty is incurred in the objective when either ε^R is greater than the available chip width W or when ε^T is greater than the available chip height H. The penalty term is weighted by a scalar penalty parameter ρ, which we define as the largest possible chip dimension. An attractive feature of the exact penalty method is that for a sufficiently large (finite) value of ρ, the optimal solution of the penalized problem will equal the solution of the original constrained problem. Practically speaking, this means that if ρ is initially set high enough, the boundary constraints will be met at the optimal solution. This is in contrast to other penalty methods where ρ is iteratively increased to reduce constraint violation. Our experience indicates that $\rho = \max(W, H)$ is large enough to ensure good convergence. Any nonsmoothness created by the penalty function is of little consequence, since (7) is evaluated by SA, which does not require gradient information.

V. Routing of Auxiliary Microfluidic Channels

The goal of the routing stage is to take a compacted arrangement generated by our floorplanning algorithm and to determine the placement or precise positions of subsystems and auxiliary channels that connect ports to wells. The auxiliary channels are kept as short and straight as possible to reduce fabrication costs and to allow for more convenient fluid loading and dispensing. In addition, short straight channels reduce dispersion effects [9] and concentration nonuniformities [3] as well as the operating voltage required to drive the chip.

A. Routing Problem Description

We define our routing problem as follows: Given a compact arrangement $\mathcal{A}$ of subsystems, find a routing solution that consists of the set of planar paths $\mathcal{P}$ through $\mathcal{A}$ such that each port is connected to a single well and that the total length and number of bends in $\mathcal{P}$ is minimized.

The routing of auxiliary channels is similar to wire routing in VLSI circuit design; however, there are several complicating features. Most importantly, LoC devices are generally fabricated in a single layer to reduce fabrication costs and complexity [26]. This means that all auxiliary channels must be routed in a planar fashion and cannot be routed above or below a subsystem. Furthermore, the assumptions that channels can feed through a subsystem or that ports may move along a subsystem edge [13] do not apply in our formulation. Additionally, auxiliary channels occupy significant space on the chip, and, therefore, the exact placement of routes is critical to the quality of the overall design.

The single-layer routing problem has been investigated in the VLSI literature [13]. However, much of this work is not directly relevant to our problem, because it either pertains to global, multiterminal, or detailed routing in bounded regions (e.g., channel routing). In global routing, the paths that wires must follow around obstacles on the chip are determined in a general way. A subsequent detailed routing stage places the wires precisely. Furthermore, most VLSI routing algorithms construct multiterminal routes, since single wires must often connect several terminals on the chip. Furthermore, we are interested in finding routing solutions through the whole chip rather than in small bounded regions of the chip. Finally, many general VLSI routing algorithms employ a sequential approach to construct a routing solution. This approach lacks feasibility guarantees and can become artificially overconstrained when poor routing choices are made early in the procedure.

We are currently developing a routing procedure that is specifically designed to solve our two-terminal single-layer detailed routing problem. In addition, we directly address length and bend minimization. In our approach, we hope to exploit the network integrality property of the minimum-cost network flow (MCNF) formulation to find routing solutions in a simultaneous fashion using LP. Large-scale MCNF problems can be solved efficiently using the simplex method or tailored algorithms [27]. Furthermore, LP can quickly provide rigorous feasibility guarantees. However, the bend-minimization constraints discussed in Section V-B-1 break the MCNF structure, thus, forcing us to use an integer programming (IP) approach. While many different IP formulations are possible to solve our routing problem, it is unclear which formulation will perform the best in all cases, since there are nontrivial tradeoffs between the number of constraints, the type of constraints, and the number of variables.

We discuss our formulation, which is based loosely on the concept of MCNF, in the following section. We have

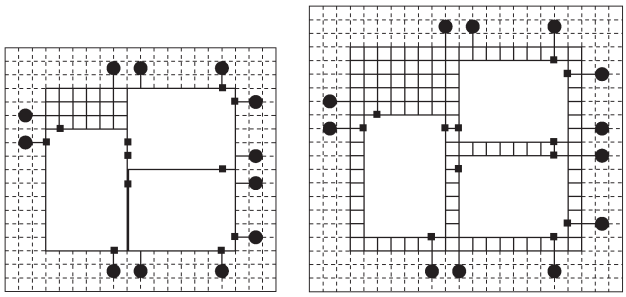


Fig. 12. (a) Initially embedded arrangement. (b) Arrangement after two expansions. Black circles represent wells, black squares represent ports, and solid lines represent available routing paths.

successfully used this formulation to obtain good solutions for several small compact test cases and are currently exploring algorithmic improvements that will allow us to deal with larger test cases.

B. Routing Problem Formulation

Our routing procedure is based on embedding a compacted arrangement of subsystems into a grid graph $G_{\rm grid} = \langle \mathcal{V}, \mathcal{E} \rangle$ as shown for a simple three-subsystem example in Fig. 12(a). The black circles and squares in Fig. 12 represent wells and ports, respectively, and the solid lines represent the complete set of all possible routing paths. The dashed lines are shown to illustrate valid positions for wells. Adjacent vertices, v_i and $v_j \in \mathcal{V}$, are spaced such that $(v_i, v_j) = \text{PAD} + \omega, \forall (v_i, v_j) \in \mathcal{E}$, which represents the center-to-center distance between neighboring channels. All paths in $\mathcal{P}$ are implicitly feasible with respect to PAD and channel width ω. Every edge in $G_{\rm grid}$ is, therefore, of equal length and every vertex is equally spaced. $G_{\rm grid}$ is searched for a routing solution using our IP formulation. The action that is taken when no routing solution is found is discussed in Section V-C.

1) Integer Programming Formulation: Since our IP formulation is based on MCNF, we construct a directed network $G_{\rm net} = \langle \mathcal{V}, \mathcal{E} \rangle$, which is formed by converting each undirected edge of $G_{\rm grid}$ into a forward-directed edge and a backward-directed edge. We arbitrarily assume that each port belongs to the set of sources, $s \subset \mathcal{V}$, and has a supply of 1, and that each well belongs to the set of sinks, $t \subset \mathcal{V}$, and has a demand of 1.

When an edge $(v_i, v_j) \in \mathcal{E}$ of $G_{\rm net}$ is occupied by a routing path, a binary variable $x_{(v_i,v_j)}$ representing the flow along that edge is equal to 1. Otherwise, if the edge is unoccupied, the flow is equal to 0. We then invoke continuity, planarity, and cycle elimination constraints for each $v_i \in \mathcal{V}$ to enforce feasible paths through $G_{\rm net}$ between ports and wells

$$\sum_{(v_i,v_j)\in\mathcal{E}} x_{(v_i,v_j)} - \sum_{(v_j,v_i)\in\mathcal{E}} x_{(v_j,v_i)} = \begin{cases} 1, & \text{if } v_i \in s \\ -1, & \text{if } v_i \in t \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

Equation (8) states that the flow into a node must equal the flow out of that node so that a feasible solution consists of an unbroken path from a particular port in s to a particular well in t. Equation (9) is used to enforce routing planarity

$$\sum_{(v_i,v_j)\in\mathcal{E}} x_{(v_i,v_j)} + \sum_{(v_j,v_i)\in\mathcal{E}} x_{(v_j,v_i)} \leq 2. \tag{9}$$

In formal terms, (9) states that the degree of every node in $G_{\rm net}$ must not be greater than 2. However, coupled with (8), it states that there can be at most one flow into and out of a given node. Equation (10) is used to eliminate the existence of simple cycles between two adjacent nodes in $G_{\rm net}$

$$x_{(v_i,v_j)} + x_{(v_j,v_i)} \leq 1. \tag{10}$$

Recall that every undirected edge of $G_{\rm grid}$ is represented by a pair of directed edges in $G_{\rm net}$. Equation (10) is used to enforce that cycling does not occur for a pair of edges (v_i, v_j) and (v_j, v_i). At the optimum, (10) is technically not required, since we are searching for minimum cost (shortest) paths. We include it to prevent cycles from existing if the search is terminated prematurely.

In general, there will exist a degenerate set of minimum-length paths between a particular port and a particular well. We can determine the straightest path between a port and well by essentially counting up the number of bends in each possible path and then choosing the path with the fewest bends. Bend-counting constraints are derived by examining connected vertex triplets v_i, v_j, and v_k. Using logic propositions $\mathbf{P}_{(v_i,v_j)}$, $\mathbf{P}_{(v_j,v_k)}$, and $\mathbf{P}_{(v_i,v_k)}$ to indicate if a bend exists between v_i and v_k [27], [28]

$$\left(\mathbf{P}_{(v_i,v_j)} \wedge \mathbf{P}_{(v_j,v_k)} \rightarrow \mathbf{P}_{(v_i,v_k)}\right) \vee \left(\mathbf{P}_{(v_j,v_i)} \wedge \mathbf{P}_{(v_k,v_j)} \rightarrow \mathbf{P}_{(v_i,v_k)}\right). \tag{11}$$

In (11), $\mathbf{P}_{(v_i,v_k)}$ will be true if there is a flow along path (v_i, v_j, v_k) or (v_k, v_j, v_i). An equivalent math programming constraint, shown in (12), can be derived from (11)

$$x_{(v_i,v_j)} + x_{(v_j,v_k)} + x_{(v_j,v_i)} + x_{(v_k,v_j)} - 1 \leq y_{(v_i,v_k)} \quad \text{where} \quad y_{(v_i,v_k)} \geq 0. \tag{12}$$

In (12), $y_{(v_i,v_k)}$ will equal 1 when a path through (v_i, v_j, v_k) or (v_k, v_j, v_i) exists, otherwise, $y_{(v_i,v_k)}$ equals 0.

We have created a general method for deriving bend-counting equations like the one shown in (12) based on the structure of the adjacency matrix $\boldsymbol{A}$ corresponding to $G_{\rm net}$. $\boldsymbol{A}$ is a square symmetric matrix that defines the complete connectivity of $G_{\rm net}$. We decompose $\boldsymbol{A}$ into a horizontal connectivity adjacency matrix $\boldsymbol{A}^{\rm hor}$ and a vertical connectivity adjacency matrix $\boldsymbol{A}^{\rm ver}$, such that $\boldsymbol{A} = \boldsymbol{A}^{\rm hor} + \boldsymbol{A}^{\rm ver}$.

Our algorithm for generating the set of bend-counting constraints for a given $G_{\rm net}$ is shown in the following.

1) Initialize: $i = 1$, $\boldsymbol{n} = \boldsymbol{m} = \emptyset$.
2) Scan row i of $\boldsymbol{A}^{\rm hor}$ and $\boldsymbol{A}^{\rm ver}$ for nonzero entries.
3) Store the indices of the nonzero entries in $\boldsymbol{n}$ from $\boldsymbol{A}^{\rm hor}$ and in $\boldsymbol{m}$ from $\boldsymbol{A}^{\rm ver}$.
4) Construct the equations: $x_{(v_i,v_n)} + x_{(v_i,v_m)} + x_{(v_n,v_i)} + x_{(v_m,v_i)} - 1 \leq y_{(v_n,v_m)} \forall v_n \in \boldsymbol{n}, \forall v_m \in \boldsymbol{m}$.

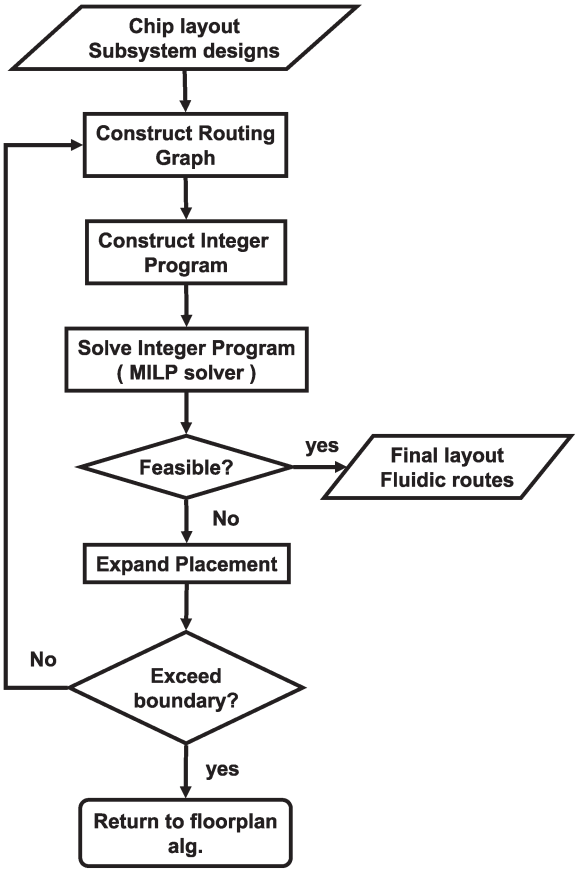


Fig. 13. Flowchart for the routing algorithm.

5) Increment: $i = i + 1$, set: $\boldsymbol{n} = \boldsymbol{m} = \emptyset$, and go to step 2). Stop when all rows of $\boldsymbol{A}^{\mathrm{hor}}$ and $\boldsymbol{A}^{\mathrm{ver}}$ have been scanned.

The constraints shown in step 4) are added to (8)–(10) to generate an IP that is capable of minimizing both channel length and the number of bends.

Equation (13) shows the objective function that minimizes channel length and reduces the number of bends in a routing solution

$$\min \sum_{(v_i,v_j)\in\mathcal{E}} x_{(v_i,v_j)} + \sum_{(v_j,v_i)\in\mathcal{E}} x_{(v_j,v_i)} + \sum_{(v_j,v_k)\in\mathcal{B}} y_{(v_j,v_k)}. \tag{13}$$

The first two terms in (13) determine the shortest path, and the final term is a summation over the set of all bends $\mathcal{B}$ that exist for a particular routing solution $\mathcal{P}$. The third term effectively penalizes routes that contain bends.

C. Routing Solution Method

Fig. 13 shows a flowchart of our procedure for routing compacted arrangements. First, an initial arrangement is legalized so that it can be embedded in G_{grid}. Next, we construct G_{net} from G_{grid} and use it to generate the associated IP. The IP is solved using a standard commercial solver [29]. If the IP is feasible, then the IP solver searches for an optimal routing. If an optimal routing or a routing that meets a user-specified tolerance is found, then the algorithm terminates successfully. If the IP is infeasible, the design is expanded to open up new routing paths and is searched again. If the expanded design exceeds the available chip area, the floorplan algorithm (Section IV-C) must be queried for a new arrangement that may have better routing characteristics. Two of the key steps of our routing procedure are discussed in the following sections.

1) Floorplan Legalization: Before an arrangement can be placed or embedded into G_{grid}, it must properly conform to the minimum allowable feature spacing requirement. If a subsystem's dimensions are not integer multiples of PAD $+ \omega$, it cannot be properly embedded into the routing grid. Our floorplan algorithm does not guarantee that an arrangement will be embeddable. To account for this, we expand each subsystem in the x and y directions to the nearest integer multiple of PAD $+ \omega$. For example, w_i is updated as follows: $w_i = \lceil w_i/(\mathrm{PAD} + \omega)\rceil(\mathrm{PAD} + \omega)$. h_i is treated in a similar fashion. The current Γ value of the floorplan is used to maintain the relative position of each subsystem. This procedure has a negligible effect on the performance and operation of each subsystem, since the size perturbation represents a relatively tiny fraction of the overall subsystem's size (typically $<1.0\%$).

The wells on the chip's edges are perturbed in a similar fashion so that they too conform to valid grid points. As with other features on the chip, well edges are constrained to be no less than PAD apart. After legalization, port, well, and subsystem locations conform to vertices in G_{grid}. Therefore, when a routing solution is found, it represents the exact location of auxiliary channels on the chip.

2) Floorplan Expansion: For a given G_{net}, the solution to the IP formulation will either yield a feasible routing solution, or it will declare the current routing problem infeasible. If the IP is infeasible, then we are guaranteed that no routing solution exists for the current instance of G_{net}.

We have developed a procedure that iteratively augments G_{grid} by expanding the arrangement to increase the number of possible routing paths in the grid. Each expansion adds one additional routing path around each subsystem. Fig. 12(b) shows a compacted arrangement that has undergone its first expansion.

An arrangement is expanded by first conceptually expanding each subsystem's width and height by $\gamma(\mathrm{PAD} + \omega)$. We define the arrangement expansion number γ as the number of times the arrangement is expanded. The arrangement's SP is used to maintain the relative positions of each subsystem. Finally, each subsystem is shifted right and up from the origin by $\gamma(\mathrm{PAD} + \omega)$. After each expansion, the arrangement is replaced or reembedded and a new G_{net} is generated. Expansions continue until either a feasible routing solution is obtained or until the placed design has become too large. Typically, arrangements require fewer than five expansions to achieve a feasible routing solution.

VI. Multiplex Synthesis Results

Here, we discuss some of the key features and algorithmic behavior of the floorplanning and routing algorithms. The results are encouraging with respect to both computation time and solution quality, since our implementations have not yet been optimized. However, these results also indicate important areas for improvement and future study.

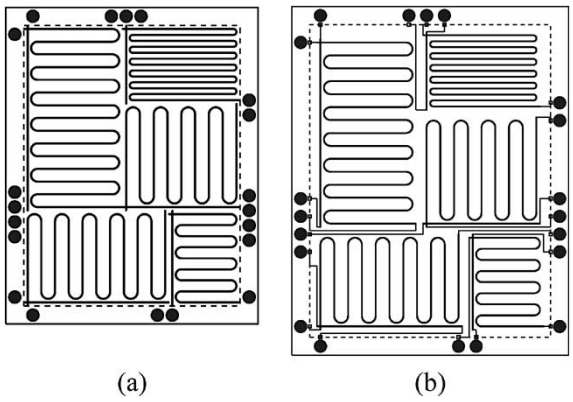


Fig. 14. Five-subsystem example. (a) Compacted arrangement. (b) Final design (routed).

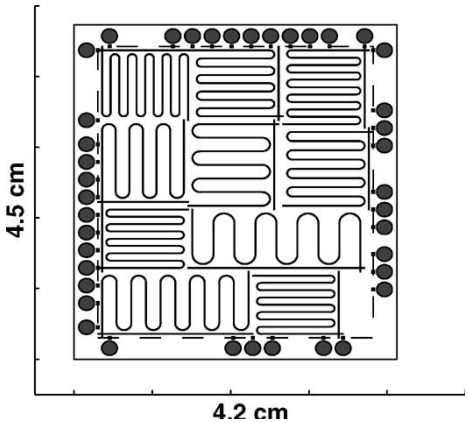


Fig. 15. Compact arrangement of ten subsystems.

The result of our floorplanning algorithm for our five subsystem example is shown in Fig. 14(a). This result was obtained in approximately 5 min of CPU time [30]. At this point, the design is compact, and all constraints are completely satisfied. The result of the routing algorithm for our five subsystem example is shown in Fig. 14(b). This design was routed in approximately 3 min of CPU time [30]. At this point, our design could be fabricated.

A. Floorplan Results

Fig. 15 shows a placed and compacted design including well positions for a multiplexed chip containing ten subsystems. This design was produced in approximately 15 min of CPU time [30].

It appears that 80%–90% of the computation time used by our floorplanning algorithm is spent solving the compaction NLP. Fig. 16 shows the average change in objective value (dotted line) and the average function evaluation time (dash-dotted line) needed to solve the compaction NLP versus the number of subsystems in a given design. We created our test cases by first choosing typical physical property values, operating conditions, and performance specifications for each subsystem. Each data point represents the average of 20 randomly generated instances. One standard deviation about the mean is also shown. As expected, the standard deviations of both

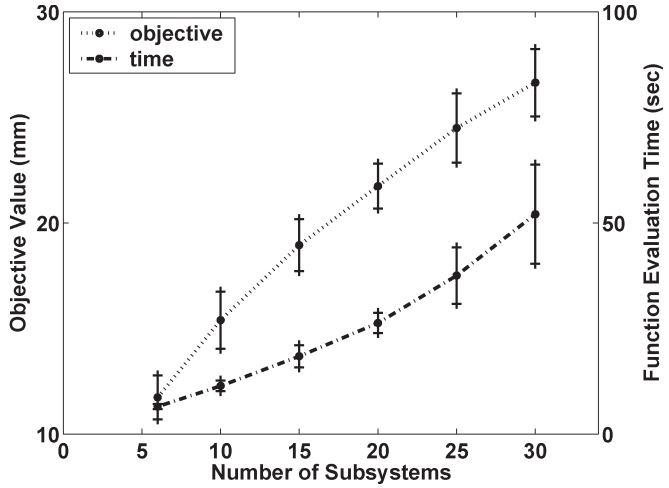


Fig. 16. Scaling of NLP compaction subproblem in the floorplanning algorithm per size of design instance. Tests were performed on a standard personal computer (PC) [31].

time and objective value increase as the number subsystems increase. This is because the solution space grows in proportion to $(|\mathcal{N}|!)^2 \cdot 8^{|\mathcal{N}|}$.

Although the worst case time complexity of typical NLP solvers is exponential, it appears that for the size problems we are currently investigating, the average-case time complexity is not yet dominated by exponential behavior. This can be attributed to the success of our initialization procedure, which helps the NLP solver converge to locally optimal solutions in reasonable time. We have produced multiplexed serpentine designs containing up to 30 subsystems in minutes to hours. As far as we know, these designs are significantly larger than any multiplexed serpentine designs presented in the literature. However, it is apparent that as problem size continues to increase, the computation time required by our algorithm will become prohibitive. We are currently investigating methods that will allow us to solve larger scale problems within a reasonable amount of time.

B. Microchannel Routing Results

Fig. 17(a) shows a minimum length routing for a simple five-subsystem floorplan. The thick lines are the auxiliary channels connecting ports to wells. This floorplan has been expanded two times (i.e., $\gamma = 2$). This means that there are at least three possible routing paths around each subsystem (each subsystem edge and one path in between). Notice that since there are many degenerate equal-length paths between ports and wells, the routing solution contains many bends. Fig. 17(b) shows a routing solution for the same example when both the channel length and the number of bends have been minimized. Typically, both the total channel length and the port-well assignment remain unaltered between the minimum length routing and the minimum length and bend routing (although this behavior is not guaranteed). This is due to the fact that bend minimization represents only a small fraction of the overall objective value. In general, it does not cause rerouting to occur.

While bend minimization helps to reduce the number of nonunique solutions, it increases the optimality gap and makes

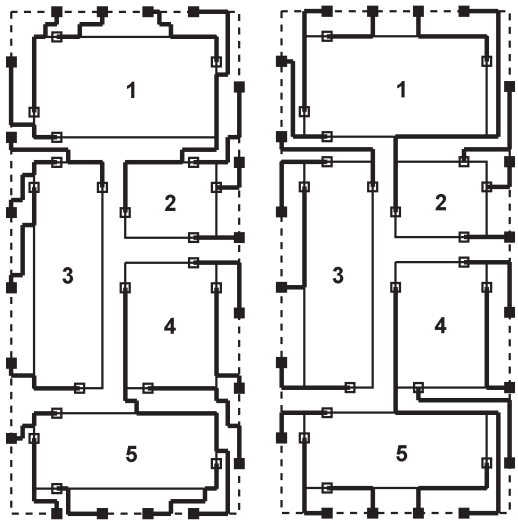


Fig. 17. Example routing. (a) Minimum length only. (b) Minimum length and number of bends.

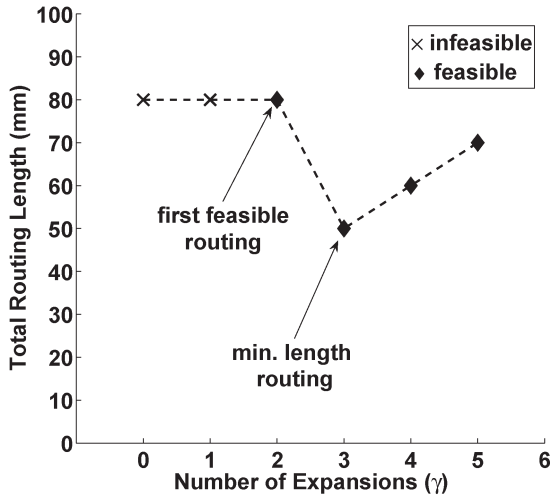


Fig. 18. Total routing length versus number expansions.

the problem more computationally difficult. In our experiments, a standard IP solver [29] can typically prove a proposed routing grid infeasible in less than 30 s. Therefore, the decision to expand the grid is made efficiently. When bend minimization is added to our problem, obtaining a proven optimal solution often becomes computationally prohibitive. Furthermore, no routing computation–time correlation can be drawn based on the number of subsystems in an arrangement, since the routing problem scales with the number of grid edges and not the number of subsystems. Currently, we cope with these issues by allowing routing solutions to have a small relative optimality gap. Despite these complications, we have obtained reasonable quality solutions is under 15 min of CPU time [30].

Fig. 18 shows the typical operation of our routing algorithm for the five-subsystem example shown in Fig. 14(b). It illustrates the tradeoff between minimum channel length and minimum design area. Fig. 18 depicts how iteratively expanding a design influences the total auxiliary channel length. For expansion numbers of $\gamma = 0$ and $\gamma = 1$, no feasible routing can be found. A feasible routing is obtained at $\gamma = 2$.

Normally, we would terminate the algorithm after optimizing the first feasible routing. However, if the design is expanded a third, fourth, and fifth time, we notice that we achieve what appears to be a globally minimal length routing at $\gamma = 3$. This occurs, because for low γ's, the routing grid is highly constrained. Therefore, the first feasible routing solution will often contain long channels. As γ increases, more possible routing paths are created and shorter routes become available. Eventually, continuing to increase γ will result in longer channels, simply because the design is now larger and the channels must traverse a longer distance to connect ports to wells.

The minimum length solution at $\gamma = 3$ is also significant, because this solution most closely approaches the solution obtained by the well placement algorithm described earlier in Section IV-B-4. Recall that in the well placement algorithm, route continuity is enforced, but route planarity is not enforced. The well placement algorithm is, in fact, a lower bound relaxation of our IP routing formulation. We are currently investigating how to utilize this information to help us more quickly discover globally minimal routing solutions.

VII. Conclusion and Future Work

We have presented a design automation methodology that is capable of generating full-custom multiplexed LoCs for CE applications in only minutes to hours. We have adapted and combined SoC circuit design techniques with optimal design methods to perform LoC design and layout. We are currently investigating methods for parallelizing our floorplanning algorithm to handle larger problem instances. We are also investigating ways to reduce the number of edges in routing grids while still maintaining a high level of connectivity. We believe that our experience with multiplexed design provides us with a tool set that can be extended to handle multifunctional chips that contain many complex chemical operations.

Acknowledgment

The authors thank the members of the SYNBIOSYS group at Carnegie Mellon University.

References

[1] D. R. Reyes, D. Iossifidis, A. Auroux, and A. Manz, "Micro total analysis systems—1: Introduction, theory and technology," *Anal. Chem.*, vol. 74, no. 12, pp. 2623–2634, Jun. 2002.

[2] A. B. Kahng, I. Măndoiu, S. Reda, X. Xu, and A. Z. Zelikovsky, "Evaluation of placement techniques for DNA probe array layout," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2003, pp. 262–269.

[3] B. Mohammadi, I. Molho, and J. G. Santiago, "Design of minimal dispersion fluidic channels in a CAD-free framework," in *Center Turbulence Research, Proc. Summer Program*, Stanford, CA, 2000, pp. 49–62.

[4] A. J. Pfeiffer, J. D. Siirola, and S. Hauan, "Optimal design of microscale separation systems using distributed agents," in *Proc. Foundations Computer-Aided Process Design (FOCAPD)*, Princeton, NJ, 2004, pp. 381–384.

[5] F. Su and K. Chakrabarty, "Architectural-level synthesis of digital microfluidics-based biochips," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2004, pp. 223–228.

[6] S. V. Ermakov, S. C. Jacobson, and J. M. Ramsey, "Computer simulations of electrokinetic injection techniques in microfluidic devices," *Anal. Chem.*, vol. 72, no. 15, pp. 3512–3517, Jun. 2000.

[7] *High-Performance Capillary Electrophoresis*, M. G. Khaledi, Ed. New York: Wiley, 1998.
[8] R. S. Magargle, J. F. Hoburg, and T. Mukherjee, "Microfluidic injector models based on neural networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. Special Issue on Biochips*, vol. 25, no. 2, pp. 373–380, Feb. 2006.
[9] Y. Wang, Q. Lin, and T. Mukherjee, "Composable modeling and simulation of electrokinetic lab-on-a-chips," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. Special Issue on Biochips*, vol. 25, no. 2, pp. 258–273, Feb. 2006.
[10] C. T. Culbertson, S. C. Jacobson, and J. M. Ramsey, "Dispersion sources for compact geometries on microchips," *Anal. Chem.*, vol. 70, no. 18, pp. 3781–3789, Sep. 1998.
[11] A. J. Pfeiffer, T. Mukherjee, and S. Hauan, "Design and optimization of compact microscale electrophoretic separation systems," *Ind. Eng. Chem. Res.*, vol. 43, no. 14, pp. 3539–3553, Jul. 2004.
[12] C. A. Emrich, H. Tian, I. L. Medintz, and R. A. Mathies, "Microfabricated 384-lane capillary array electrophoresis bioanalyzer for ultrahigh-throughput genetic analysis," *Anal. Chem.*, vol. 74, no. 19, pp. 5076–5083, Oct. 2002.
[13] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*. New York: McGraw-Hill, 1996.
[14] A. J. Pfeiffer, T. Mukherjee, and S. Hauan, "Simultaneous design and placement of multiplexed chemical processing systems on microchips," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2004, pp. 229–236.
[15] S. Lee and I. E. Grossmann, "New algorithms for nonlinear generalized disjunctive programming," *Comput. Chem. Eng.*, vol. 24, no. 9–10, pp. 2125–2141, 2000.
[16] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence pair," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 15, no. 12, pp. 1518–1524, Dec. 1996.
[17] H. Onodera, Y. Taniguchi, and K. Tamaru, "Branch-and-bound placement for building block layout," in *Proc. 28th ACM/IEEE Design Automation Conf.*, San Francisco, CA, 1991, pp. 433–439.
[18] S. Sutanthavibul, E. Shragowitz, and J. B. Rosen, "An analytical approach to floorplan design and optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 10, no. 6, pp. 761–769, Jun. 1991.
[19] S. Kirkpatrick, C. D. Galett, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 13, 1983.
[20] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of non-slicing floorplan," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, 2000, pp. 8–12.
[21] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design: Modeling and Computation*. Cambridge, U.K.: Cambridge Univ. Press, 2000.
[22] J.-G. Kim and Y.-D. Kim, "A linear programming-based algorithm for floorplanning in VLSI design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 5, pp. 584–592, May 2003.
[23] H. Murata and E. S. Kuh, "Sequence-pair based placement method for hard/soft/preplaced modules," in *Proc. Int. Symp. Physical Design (ISPD)*, Monterey, CA, 1996, pp. 167–172.
[24] X. Tang, R. Tian, and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," in *Proc. Design Automation and Test Europe (DATE)*, Paris, France, 2000, pp. 106–111.
[25] Tropics Research Team (INRIA). (2004, Nov.), *Tapenade*. [Online]. Available: http://www.inria.fr
[26] LioniX. (2004, Nov.), *LioniX MEMS Foundry Services*. [Online]. Available: http://www.lionixbv.nl
[27] H. P. Williams, *Model Building in Mathematical Programming*, 4th ed. New York: Wiley, 1999.
[28] G. Nam, K. Sakallah, and R. Rutenbar, "A boolean satisfiability-based incremental rerouting approach with application to FPGAs," in *Proc. Design Automation and Test Europe (DATE)*, Munich, Germany, 2001, pp. 560–565.
[29] ILOG Inc., (2004, Nov.) , *ILOG CPLEX 9.0*. [Online]. Available: http://www.ilog.com/products/cplex
[30] Intel Pentium 4 CPU, @ 2GHz, 1Gb RAM.
[31] Intel Pentium 3 CPU, @ 1GHz, 1Gb RAM.

Anton J. Pfeiffer received the B.S. degree in chemical engineering from the University of South Florida, Tampa, in 2001. He is currently working toward the Ph.D. degree in chemical engineering at Carnegie Mellon University, Pittsburgh, PA.

His research interests include computer-aided design and analysis of complex systems, design automation techniques, and large-scale nonlinear and combinatorial optimization.

Tamal Mukherjee (S'89–M'95) received the B.S., M.S., and Ph.D. degrees from Carnegie Mellon University, Pittsburgh, PA, in 1987, 1990, and 1995, respectively, all in electrical and computer engineering.

He is currently a Research Professor in the Electrical and Computer Engineering Department at Carnegie Mellon University. His research interests include automating the design of analog circuits and microfluidic and microelectromechanical systems. His current work focuses on developing computer-aided design methodologies and techniques for integrated microelectromechanical systems. He is involved in modeling, simulation, extraction, and synthesis of mixed domain microsystems. He is also active in the use of these techniques for the design of radio frequency (RF) circuits whose performance is enhanced by micromachining.

Steinar Hauan received the M.Sc. degree from the Norwegian Institute of Technology, Trondheim, Norway, in 1993, and the Ph.D. degree from the Norwegian University of Science and Technology, Trondheim, Norway, in 1998, both in chemical engineering.

He is currently an Associate Professor of Chemical Engineering at Carnegie Mellon University, Pittsburgh, PA. His research interests are in the area of computer-aided design and analysis of complex chemical systems with special focus in the areas of process intensification, distributed optimization, acoustic microelectromechanical system (MEMS) sensors, and lab-on-a-chip systems.