Placement of Digital Microfluidic Biochips Using the T-tree Formulation *

Ping-Hung Yuh, chia-Lin Yang
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
{r91089, yangc}@csie.ntu.edu.tw

Yao-Wen Chang
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University
Taipei 106, Taiwan
ywchang@cc.ee.ntu.edu.tw

ABSTRACT

Droplet-based microfluidic biochips have recently gained much attention and are expected to revolutionize the biological laboratory procedure. As biochips are adopted for the complex procedures in molecular biology, its complexity is expected to increase due to the need of multiple and concurrent assays on a chip. In this paper, we formulate the placement problem of digital microfluidic biochips with a tree-based topological representation, called *T-tree*. To the best knowledge of the authors, this is the first work that adopts a topological representation to solve the placement problem of digital microfluidic biochips. Experimental results demonstrate that our approach is much more efficient and effective, compared with the previous unified synthesis and placement framework.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithm, Performance, Design

Keywords

Microfluidics, biochip, placement, floorplanning

1. INTRODUCTION

Due to the advances in the microfabrication and microelectromechanical systems, the microfluidic technology has gained much attention recently. The composite microsystems could perform the conventional biological laboratory procedures on a small and integrated system. As a result, the microfluidic biochips are used in several common procedures in molecular biology, such as the clinic diagnosis and the DNA sequence analysis.

Most recently, the second-generation (digital) microfluidic biochips, which are based on the manipulation of the discrete liquid particles (the *droplets*), have been proposed [1]. Each droplet can be independently controlled by the electrohydrodynamic forces generated by an electric field. The field can be generated by an individually accessible electrode. Compared with the first-generation microfluidic biochips that are based on the continuous fluid flow and contain external pressure sources, the digital microfluidic biochips have the advantages of scalability and reconfigurability. The scalability allows biochips to handle large-scale and complex procedures, and the reconfigurability allows biochips to be reconfigured for different levels of hierarchy.

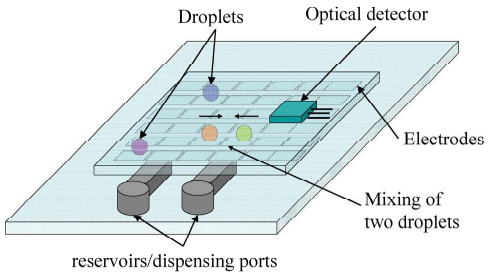

Figure 1: The schematic view of a digital microfluidic biochip.

Figure 1 shows the schematic view of a digital microfluidic biochip based on the principle of electrowetting on dielectric [3]. The basic operations (e.g., mix, dilute, etc.) can be performed anywhere in the 2D microfluidic array because each basic cell has the same architecture. We refer to this type of operations as *reconfigurable operations* in this paper. Besides the 2D microfluidic array, there are other devices in a biochip, such as the on-chip reservoirs/dispensing ports and optical detectors. In contrast to the 2D array, these devices cannot be reconfigured. The operations performed on these devices are referred to as *non-reconfigurable opertaions*.

Due to the reconfigurability, the placement problem of digital microfluidic biochips includes architectural-level synthesis (i.e., scheduling and resource binding) and physical placement. How to simultaneously perform architectural-level synthesis and physical placement is the most challenging issue in this placement problem.

Architectural-level synthesis and physical placement of digital microfluidic biochips have been addressed in the recent literature. For the architectural-level synthesis, Su *et al.* [7] used the sequencing graph to represent the bioassay protocol and proposed an integer linear programming based formulation and two heuristics to solve this problem. Su *et al.* [8] proposed a simulated annealing based algorithm for the physical placement with given scheduled operations. Recently, [9] presented a unified synthesis and placement flow based on parallel recombinative simulated annealing. The synthesis and placement of digital microfluidic biochips are closely related to the operations of dynamically reconfigurable FPGAs (DRFPGAs), which have received much attention recently [2]. Many approaches, such as the graph-theoretic ap-

*This work was partially supported by the National Science Council of Taiwan under Grant No's. NSC 93-2220-E-002-001 and NSC 93-2752-E-002-008-PAE.

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

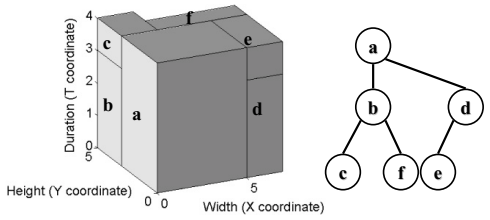

Figure 2: A compacted placement and its corresponding T-tree.

proach [4] and the topological representation based approach [10, 11], have been proposed. Among these approaches, the T-tree [10] representation is the state-of-the-art method for DRFPGAs.

In this paper, we adopt the T-tree topological representation [10] to solve the placement problem of digital microfluidic biochips. Due to the reconfigurability of DRFPGAs, the placement of digital microfluidic biochips is similar to the simultaneous scheduling and placement of DRFPGAs. However, the placement of biochips is more complicated than that of DRFPGAs for two reasons. First, besides reconfigurable operations, biochips also need non-reconfigurable operations. Second, a storage unit is required for two data-dependent operations if they are not scheduled at consecutive time steps. To the best knowledge of the authors, our work is the first to apply a topological representation to solve the placement problem of digital microfluidic biochips. Experimental results show that our algorithm is much more effective and efficient than the previous unified synthesis and placement approach in [9] for *in-virto* diagnostics and colorimetric protein bioassay. For the protein bioassay, our method achieves 19% smaller volume (area times assay completion time) in less CPU time. For the *in-virto* diagnostics, we can meet all design specifications while the previous work cannot.

The remainder of this paper is organized as follows. Section 2 formulates the placement problem of digital microfluidic biochips. Section 3 reviews the T-tree formulation. Section 4 presents the T-tree based formulation for the placement problem. Section 5 describes our algorithm. Section 6 reports the experimental results. Finally, conclusions are given in Section 7.

2. PROBLEM FORMULATION

We give a formal definition for the placement problem of digital microfluidic biochips. There are three inputs to the placement problem. The first one is the sequencing graph $G = \{V, E\}$ that represents the protocol of a bioassay [7], where $V = \{v_1, v_2, \ldots, v_m\}$ represents a set of m assay operations and $E = \{(v_i, v_j), 1 \leq i, j \leq m\}$ denotes the data dependencies between two assay operations; i.e., the *precedence constraints*. We may need a storage unit for each edge in G to store the intermediate data between two operations. Throughout this paper, we use operation and task interchangeably. The second one is the microfluidic module library that contains the basic modules for biochips. Each basic module is characterized by its functionality (i.e., mixing, dilution) and parameters (i.e., width, height, and operation duration). The third one is the design specification, including: (1) the maximum available area and assay completion time and (2) the maximum number of instances for each type of non-reconfigurable resources; i.e., the *resource constraints*.

The goal of our algorithm is to simultaneously perform resource binding, scheduling, and placement with area and assay completion time optimization under the design specification. In this paper, we ignore the time for transporting droplets between different tasks because the movement of droplets is very fast compared with the duration of each task [3, 6]. We also follow [9] to use the segregation cells to wrap each reconfigurable operation and storage unit for the isolation purpose.

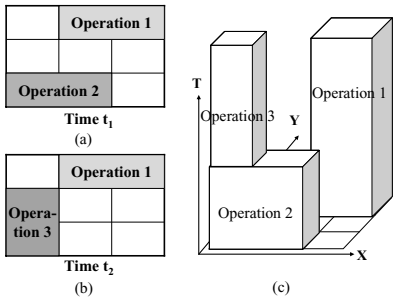

Figure 3: (a) Two operations are executed at time $t1$. (b) At time $t2$, operation 3 starts to execute at the same physical location as operation 2. (c) The 3D modelling of the three operations.

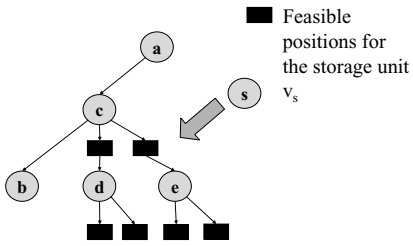

Figure 4: An example of finding the feasible locations for a storage unit in a T-tree. Suppose that v_a and v_b need a storage unit.

3. REVIEW OF THE T-TREE

T-trees [10] are a 3-ary tree, where each node corresponds to a unique task and has at most three children to represent the dimensional relationship among tasks. The T-tree is designed to represent a compacted placements where each task cannot move toward the origin. Figure 2 shows a compacted placement and its corresponding T-tree. The T-tree represents the geometric relationship between two tasks as follows. If node n_j is the left child of node n_i, task v_j is placed adjacently to task v_i on the T^+ direction. If node n_k is the middle child of node n_i, task v_k is placed in the Y^+ direction of task v_i, with the t-coordinate of v_k being equal to that of v_i. If node n_l is the right child of node n_i, task v_l is placed in the X^+ direction of task v_i, with the t- and y-coordinates being equal to those of v_i.

4. T-TREE BASED BIOCHIP PLACEMENT

In this section we first demonstrate that the placement of tasks on a biochip can be treated as the temporal floorplanning problem. Then we describe how we model each type of tasks with the T-tree. Finally we present how to handle the design specification in our algorithm.

Due to the reconfigurability of biochips, the execution of a set of tasks can be viewed as the temporal floorplanning problem as shown in Figure 3. The X and Y dimensions give the area of a biochip, and the T dimension represents the duration of a task. Suppose that both operations 1 and 2 are executed at time $t1$, as shown in Figure 3 (a). Figure 3 (b) shows that at time $t2$, we can perform operation 3 at the same physical location as operation 2 after completing operation 2. The execution of the three operations can be modelled as a set of 3D modules with their widths and heights (X and Y dimensions) representing the physical dimensions occupied by the operations, and its duration (T dimension) being the execution time required for each operation, as shown in Figure 3 (c).

For each task in a sequencing graph, we create a unique node in the T-tree. Recall that there are both reconfigurable and non-

Algorithm: Feasibility Detection (H)
H: a T-tree;
1 Set the durations of all storage units n_s to be zero;
2 **repeat**
3 **if** n_s is not in its feasible position
4 Move n_s to one of its feasible positions;
5 **else if** n_s has its left child
6 Move n_s as the middle or right child of n_k,
7 where n_k is one of n_s's children;
9 **else**
10 Traverse H to obtain the starting time of each task;
11 Determine the duration of all storage units;
12 Check the precedence constraints and reconstruct H;
13 **if** all precedence constraints are satisfied
14 Adjust the number of storage units;
15 **until** the topology of H is fixed
17 **Output:** H with the feasible topology.

Figure 5: Feasibility detection and tree reconstruction.

reconfigurable tasks. For reconfigurable tasks and the detection tasks, since we need to perform this type of tasks in the 2D microfluidic array, we model it as a 3D box. For non-reconfigurable tasks except the detection tasks, since the reservoirs and dispensing ports are outside the 2D microfluidic array as shown in Figure 1, we only need to consider the time aspect for this type of tasks. Therefore, we model it as a 3D line with both its width and height being zero.

We describe how we model the storage units. We create a node n_s for each storage unit v_s. Since v_s holds the intermediate data between tasks v_i and v_j, v_s must satisfy the *storage constraint*. The storage constraint states that the starting time of v_s must be equal to the ending time of v_i, and the ending time of v_s must be equal to the starting time of v_j. Figure 4 illustrates how to find the feasible locations for n_s in a T-tree while satisfying the storage constraint. Suppose we want to find the feasible locations for n_s. Recall that if n_j is the left child of n_i, the starting time of v_j is the ending time of v_i. Otherwise, the starting time of v_j is the same as the starting time of v_i. Thus, based on the structure of T-tree, the starting time of v_c in Figure 4 is the same as the ending time of v_a, and the starting times of v_b, v_c, and v_d are the same. Based on above observation, the feasible locations of n_s are the middle or the right child of nodes n_b, n_c, or n_d, as the black boxes shown in Figure 4. After placing n_s in its feasible location, we set the ending time of v_s as the starting time of v_b. Note that the duration of v_s is not fixed; it varies based on the starting time of v_b.

The design specification describes the maximum available area and the assay completion time as well as the resource constraints. We model the limit on the maximum available area and assay completion time as the *fixed-cube constraint*. The fixed-cube constraint states that a 3D floorplan must be within a 3D cube. To handle the resource constraints, we introduce the concept of the *virtual* precedence constraints. If two tasks are bound to the same resource, such as the same dispensing port, these two tasks cannot be executed at the same time. Therefore, we add an additional edge between these two tasks in the sequencing graph to satisfy the resource constraint. Note that there is no storage unit requirement in these additional edges.

5. THE FLOORPLANNING ALGORITHM

In this section, We describe our T-tree based floorplanning algorithm for digital microfluidic biochips. Our algorithm is based on the simulated annealing (SA) method [5]. Given a feasible T-tree, we perturb it to obtain another feasible T-tree through a set of pre-defined SA operations. After perturbation, we perform the feasibility detection and tree reconstruction to obtain a feasible topology with respect to the precedence constraints and storage constraints. Finally, a packing procedure is invoked to evaluate the solution quality.

5.1 Cost Function

Our goal is to optimize the biochip area and assay completion time under the design specification. Therefore, the cost function Φ used in our algorithm is given by

$$\Phi = \alpha V/V_{norm} + \beta S/s_{norm} + \gamma M, \quad (1)$$

where V is the volume of the 3D floorplan, S is the total number of storage units required, V_{norm} is the normalized volume, S_{norm} is the normalized total number of storage units, and M is the penalty term for fixed-cube constraint. α, β, and γ are user-specified constants. M is defined as

$$M = \frac{max(W_f - W_p, 0) \times W_f}{N_w^2} + \frac{max(H_f - H_p, 0) \times H_f}{N_h^2} + \frac{max(T_f - T_p, 0) \times T_f}{N_t^2}, \quad (2)$$

where N_w (N_h, N_t) is the normalized width (height, assay completion time), W_p (H_p, T_p) and W_f (H_f, T_f) denote the width (height, assay completion time) of the design specification and a 3D floorplan, respectively. Since we must pack all modules into a pre-defined 3D cube, we penalize the excessive width, height, and completion time in the cost function. The rationale behind M is that when SA minimizes the cost function, it automatically minimizes the penalty term. Thus, we can automatically satisfy the fixed-cube constraint.

5.2 Perturbation

The original SA operations defined in [10] contain Move, Swap, and Rotation. For the placement of digital microfluidic biochips, we introduce a new type of SA operations, called *Rebind*. Rebind is to bind a task to another functional resource. Note that we need to change the virtual precedence constraints after rebinding a non-reconfigurable task.

Besides adding the penalty term in the cost function, we bias the Move operation based on the probability of violating the fixed-cube constraint in each dimension. Let k_w (k_h, k_t) be the number of floorplans whose width (height, completion time) exceeds the user-specified width (height, completion time) in the last n iterations. In this paper, we set n equal 500. We bias the selection of the destination of the Move operation based on the values k_w/n, k_h/n, and k_t/n. For example, a larger k_w/n implies that it is more difficult to fit the floorplans to the 3D cube in the X direction. Therefore, we should try to place tasks along the Y or T direction to satisfy the fixed-cube constraint.

5.3 Feasibility Detection and Tree Reconstruction

After perturbation, we perform feasibility detection and tree reconstruction to satisfy all precedence constraints and storage constraints. We enhance the feasibility detection and the iterative tree reconstruction process in [10] with the consideration of the storage constraints. After obtaining a feasible topology of a T-tree, we invoke the packing procedure to determine the physical locations for all tasks.

Given a T-tree H, we first check if every storage unit is in its feasible position. If a storage unit n_s is not in its feasible position, we move n_s to one of its feasible positions. Note that since we modify the topology of H during the tree reconstruction, the duration of each storage unit may change. To simplify our algorithm, we thus restrict every storage unit not to have its left child. By doing so, the starting time of a task will not be affected by any storage unit during the tree reconstruction. If n_s has a left child, we shall move n_s as the middle or the right child of n_k, where n_k is one of n_s's children.

Once all storage units are in their feasible positions, we traverse H to obtain the starting time for each task. Next, we check the precedence constraints and reconstruct H if necessary, based on the method proposed in [10]. If all precedence constraints are satisfied, we adjust the number of storage units by deleting an unused one or inserting a new one. The reason why we choose to adjust the number of storage units during the tree reconstruction

is that deleting or inserting a storage unit will change the topology of H, and the resulting topology may not be feasible. Thus, we choose to adjust the number of storage units during (instead after) the feasibility detection and tree reconstruction. When the topology of H is not changed, which means that all precedence constraints and storage constraints are satisfied, the process terminates. Figure 5 summaries the feasibility detection and tree reconstruction process.

Operation	Area	Duration (s)
Generation	on-chip	2
Mix (plasma)	2x4-array	3
	2x3-array	6
	2x2-array	10
	1x4-linear array	5
Mix (Serum)	2x4-array	2
	2x3-array	4
	2x2-array	6
	x4-linear array	3
Mix (Urine)	2x4-array	3
	2x3-array	5
	2x2-array	8
	1x4-linear array	4
Mix (Saliva)	2x4-array	4
	2x3-array	8
	2x2-array	12
	1x4-linear array	6
Opt (glucose)	LED+Photodiode	10
Opt (lactate)	LED+Photodiode	8
Opt (pyruvate)	LED+Photodiode	12
Opt (glutamate)	LED+Photodiode	10
Storage	1x1 cell	N/A

Table 1: The microfluidic module library for the in-vitro diagnostics.

Bioassay	Design Spec.	[9]		T-tree	
		Volume	CPU time	Volume	CPU time
Protein	10x10x400	9x10x400	300	10x10x270	89
	10x10x360	10x10x342	225	10x10x262	119
	11x11x320	(8x13x269)*	208	11x11x238	66
Avg.		1.19	2.67	1.0	1.0
in_vitro	10x10x100	9x11x99	64	9x8x66	6
	8x8x120	(9x9x130)*	104	8x7x68	12
	7x7x140	(9x10x105)*	92	6x7x89	15
Avg.		2.42	7.88	1.0	1.0

Table 2: The experimental result. ()*: the result cannot meet the design specification.

6. EXPERIMENTAL RESULTS

Our algorithm was implemented in the C++ programming language and ran on a 1.2 GHz SUN Blade 2000 with 8 GB memory. We also implemented the unified synthesis and placement algorithm proposed in [9]. For all experiments, we set $\alpha = 1/16.5$, $\beta = 0.5/16.5$, and $\gamma = 15/16.5$.

We evaluated our placement algorithm with two bioassays: the colorimetric protein assay [6] and the multiplexed *in-vitro* diagnostics [7]. For the colorimetric protein assay, we applied the same design specification and used the same microfluidic module library as [9]. For the multiplexed *in-vitro* diagnostics, we used the same design specification as [7]. However, since [7] did not specify the width and height of each operation, we randomly generated them. Table 1 shows the microfluidic module library used for the multiplexed *in-vitro* diagnostics.

Table 2 summaries the result of the two bioassays. Column 2 lists the design specifications. For each bioassay, we applied three different design specifications. We reported the resulting volume (area times assay completion time) and CPU time. For the first bioassay, our algorithm can meet all three design specifications

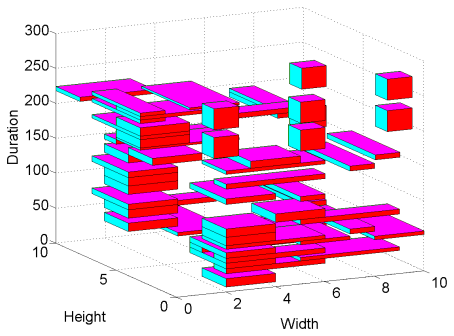

Figure 6: A 3D view of the placement result.

while the work [9] cannot. More importantly, [9] required on average 19% larger volume and 2.67X more CPU time than our algorithm. For the second bioassay, our algorithm can still satisfy all design specifications while the work [9] cannot. For the first design specification that [9] can satisfy, we can obtain a solution with smaller volume (9x8x66 vs. 9x11x99) in less CPU time (6 sec vs. 64 sec). The results show the efficiency and effectiveness of our algorithm with different bioassays and design specifications. Figure 6 shows the placement result of the colorimetric protein assay with the 10x10x400 design specification. For simplicity, we only show the reconfigurable and detection operations.

7. CONCLUSION

We have applied the T-tree representation to the placement problem of digital microfluidic biochips. To our best knowledge, this is the first work that adopts a topological representation for the placement problem of digital microfluidic biochips. We have also presented our placement algorithm based on the T-tree representation. We have shown the efficiency and the effectiveness of our algorithm. The future work includes more sophisticated method for handling the storage units and the issues of fault tolerance and defect tolerance.

8. REFERENCES

[1] http://www.tutorgig.com/encyclopedia.

[2] K. Bazargan, R. Kastner, and M. Sarrafzadeh. Fast template placement for recofigurable computing systems. *IEEE Design and Test of computers*, 17:68–83, 2000.

[3] R. B. Fair, V. Srinivasan, H. Ren, P. Paik, V. Pamula, and M. Pollack. Electrowetting-based on-chip sample processing for integrated microfluidics. In *Proc. IEDM*, pages 32.5.1–32.5.4, 2003.

[4] S. P. Fekete, E. Kóhler, and J. Teich. Optimal fpga module placement with temporal precedence constraints. In *Proc. DATE*, pages 658–665, May 2001.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[6] V. Srinivasan, V. Pamula, P. Paik, and R. Fair. Protein stamping for maldi mass spectrometry using an electrowetting-based microfluidic platform. In *Proc. SPIE*, pages 26–32, 2004.

[7] F. Su and K. Chakrabarty. Architectural-level synthesis of digital microfluidics-based biochips. In *Proc. ICCAD*, pages 223–228, Nov. 2004.

[8] F. Su and K. Chakrabarty. Design of fault-tolerant and dynamically-reconfigurable microfluidic biochips. In *Proc. DATE*, pages 1202–1207, March 2005.

[9] F. Su and K. Chakrabarty. Unified high-level synthesis and module placement for defect-tolerant microfluidic biochips. In *Proc. DAC*, pages 825–830, June 2005.

[10] P.-H. Yuh, C.-L. Yang, and Y.-W. Chang. Temporal floorplanning using the t-tree formulation. In *Proc. ICCAD*, pages 300–305, Nov. 2004.

[11] P.-H. Yuh, C.-L. Yang, Y.-W. Chang, and H.-L. Chang. Temporal floorplanning using 3d-subtcg. In *Proc. ASPDAC*, pages 725–730, Jan. 2004.