PCB Escape Routing and Layer Minimization for Digital Microfluidic Biochips

Jeffrey McDaniel, *Member, IEEE*, Zachary Zimmerman, Daniel Grissom, and Philip Brisk, *Member, IEEE*

***Abstract*—This paper introduces a multiterminal escape routing algorithm for the design of printed circuit boards (PCBs) that control digital microfluidic biochips (DMFBs). The new algorithm extends a negotiated congestion-based single-terminal escape router that has been shown to be superior to previous methods. It relaxes the pin assignment to allow pin groups to be broken up when doing so can reduce the number of PCB layers. Experimental results indicate that the improved method can reduce both the number of PCB layers and average wirelength compared to existing DMFB escape routers.**

***Index Terms*—Design automation, digital microfluidics, escape routing, printed circuit board (PCB).**

I. Introduction

THIS paper presents a multiterminal escape routing algorithm for the design of printed circuit boards (PCBs) for digital microfluidic biochips (DMFBs), and empirically demonstrates that it is more effective than existing escape routers in terms of its ability to reduce the number of PCB layers required to realize a design. Reducing the number of PCB layers, in turn, reduces the cost of the DMFB, which is typically integrated into a software-programmable laboratory-on-a-chip (LoC).

Compared to traditional benchtop chemistry methods, LoCs offer the benefits of miniaturization, automation, and software control; this reduces the overall usage of costly reagents on a per-experiment basis and eliminate many sources of human error. DMFBs have been used for a wide variety of applications including cryopreservation [27], single-cell analysis [29], immunoassays [32], point-of-care diagnostics [21], [25], [26], drug screening [1], and many others.

Manuscript received April 3, 2015; revised July 13, 2015, October 12, 2015, and March 25, 2016; accepted April 9, 2016. Date of publication May 13, 2016; date of current version December 20, 2016. This work was supported in part by the National Science Foundation (NSF) under Grant 1035603, and in part by the Directorate for Computer and Information Science and Engineering under Grant 1035603, Grant 1351115, and Grant 1545097. The work of D. Grissom was supported in part by the NSF Graduate Research Fellowship, and in part by the University of California at Riverside Graduate Division Dissertation Year Fellowship. This paper was recommended by Associate Editor V. Narayanan.

J. McDaniel, Z. Zimmerman, and P. Brisk are with the Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA 92507 USA (e-mail: jmcda001@ucr.edu).

D. Grissom is with the Department of Computer Science, Azusa Pacific University, Azusa, CA 91702 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2016.2568199

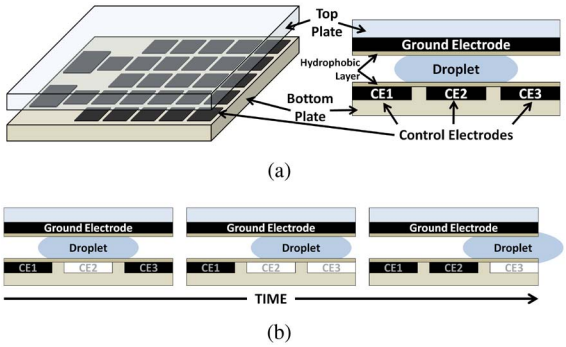

Fig. 1. (a) Left: a DMFB comprises a 2-D array of control electrodes (CEs). Right: a cross sectional view of a droplet sandwiched between a ground electrode (top) and the CE array (bottom). (b) Droplet motion is induced by activating (white) and deactivating (black) CEs in sequence.

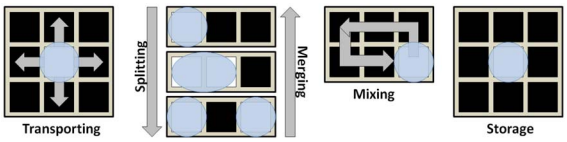

Fig. 2. Basic DMFB instruction set: transport, splitting, merging, mixing, and storage.

As shown in Fig. 1, a DMFB is a 2-D grid of electrodes that offers discrete control over individual droplets of liquid: activating an electrode underneath a droplet holds it in-place; activating adjacent electrodes induces droplet motion through an electrostatic force, a phenomenon referred to as electrowetting on dielectric [28].

Fig. 2 depicts a basic DMFB instruction set. As a DMFB offers abundant spatial parallelism, many such operations can be performed concurrently. In addition to the operations shown in Fig. 2, integrated sensors [6], [23], [33] and external devices (e.g., heaters [17], [40], magnets [29], etc.) can be placed adjacent to prespecified array locations, which add new operational capabilities to the device.

Fig. 3 illustrates the main stages of DMFB synthesis. The input is a biochemical assay, specified as a directed acyclic graph (DAG). The first three stages of the flow schedule, place, and route the DAG onto the device: these topics are beyond the scope of this paper; we refer the interested reader may refer to [3] and [10] for details.

The fourth stage of the synthesis flow is pin-mapping, which can reduce the number of control pins required to address the DMFB while converting the chip from a general-purpose

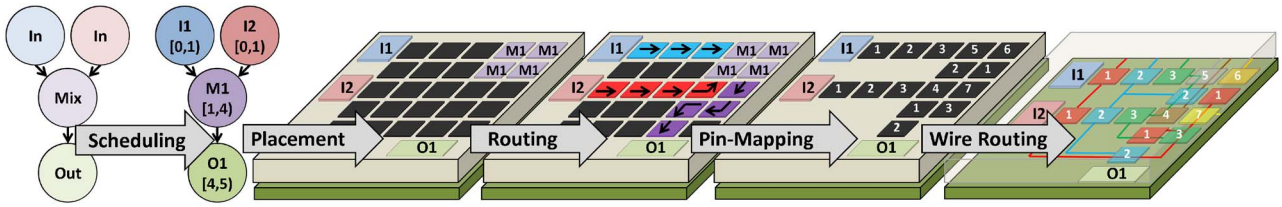

Fig. 3. DMFB synthesis flow: an assay, represented as a DAG is scheduled, operations are then placed on the DMFB surface, according to the schedule, and routes are computed to transport droplets between operation locations and I/O ports on the perimeter of the chip. After synthesis, pin-mapping and wire routing can be performed to reduce the number of external control signals and lay out the PCB that delivers signals to the DMFB.

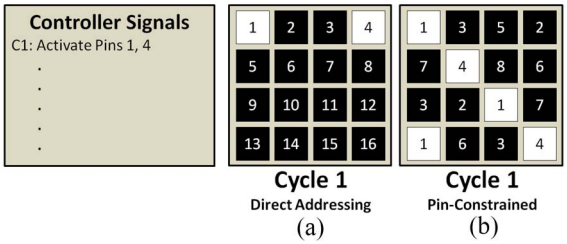

Fig. 4. (a) In a direct-addressing DMFB, each control pin drives one electrode. (b) In a pin-constrained DMFB, control pins may drive one or more electrodes.

programmable device to an application-specific design [41]. This lowers the total device cost by reducing the number of PCB-mounted shift registers required to supply control signals, along with the 2-D PCB area required to mount them [11]. Pin-mapping is optional and is not required to produce a correctly working device.

As shown in Fig. 4, direct-addressing DMFBs, in which each electrode has an independent control pin, are perfectly feasible, although they may come at a higher cost in terms of the number of PCB layers and the number of control signals; pin-constrained DMFBs have been optimized via pin-mapping, allowing multiple electrodes to share the same control line without affecting assay execution [41].

The last stage of the synthesis flow is PCB wire routing, which is the focus of this paper. The wire router determines a connection from a control pin on the perimeter of the chip to each electrode that it drives. As the position of the control pin is not known *a-priori*, the route "escapes" to any location on the perimeter of the chip. On a single-layer PCB, the routes must be disjoint (nonintersecting); if intersections cannot be avoided, the routed nets must be partitioned across multiple layers, as shown in Fig. 5.

In principle, multiple PCB layers may be needed to realize both direct-addressing and pin-constrained DMFBs, although it has always been presumed that the latter requires fewer layers. This paper shows empirically that pathological pin assignments can significantly increase the number of PCB layers; if such pin assignments are avoided, then the number of PCB layers depends primarily on the quality of the escape routing solution, rather than pin assignment decisions. To further reduce the number of PCB layers, this paper introduces a scheme that eliminates pathological pin assignments that may result from overly aggressive pin-mapping decisions. Prior work has shown that the cost of each additional PCB layer is significantly greater than the cost of marginally increasing the pin count [11], which justifies the optimization strategy advocated here.

The escape router introduced in this paper is based on the principle of negotiated congestion [19], [20], [24], [31]. The results reported within demonstrate that negotiated congestion yields fewer PCB layers than prevailing approaches based on maze routing with an integrated rip-up and reroute step [14], [38]. The escape router introduced here can be used independently to route a precomputed pin-mapping solution, or it can replace the escape routing subroutine employed by integrated methods that simultaneously co-optimize pin-mapping and escape routing.

II. Related Work

Prior work on PCB escape routing for DMFBs has been integrated with pin mapping in the context of synthesizing application-specific designs [14]. The application has already been scheduled, placed, and routed, so the electrode activation sequence for a direct addressing chip is known. These algorithms convert the direct addressing DMFB into an application-specific pin-constrained design with fewer control pins, and, presumably, lower cost. Typically, these algorithms target single-layer PCBs and co-optimize the number of control pins with other objectives relating to routability [16] and/or reliability [35], [38], [39]. Power-aware [13] and reliability-aware [12] pin mappers that do not integrate PCB escape routers have also been proposed.

These aforementioned algorithms typically model pin sharing as clique partitioning problem on a compatibility graph (or, equivalently, as a graph coloring problem on a conflict graph) [41]. Each clique (independent set) in the graph represents a potential pin group, i.e., a set of electrodes that can share the same control pin without inadvertently causing errorenous behavior in terms of the electrode actuation sequence. These algorithms incrementally form pin groups one-by-one in accordance with the chosen optimization criteria; in other words, they compute a clique (independent set) from the conflict (compatibility) graph, and remove the corresponding nodes from the graph to form the pin group.

The algorithms differ in terms of how escape routing is performed. Some of them incrementally compute escape routes for the nets corresponding to each pin group. Others try to compute escape routes for all pin groups at the same time; if a legal escape route for all nets cannot be found, they may break apart some of the pin groups (increasing the number of control pins) and then recompute the escape route, eventually stopping

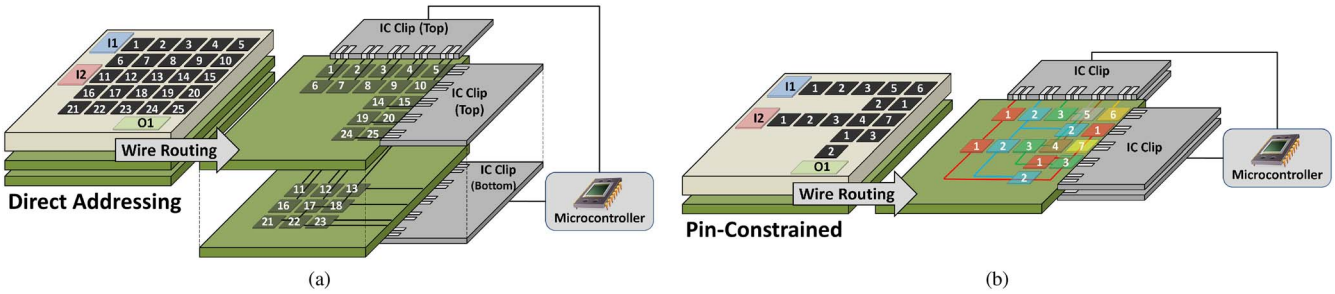

Fig. 5. DMFB typically has one or more PCB layers (green) beneath the electrode array which connect the microcontroller to the control electrodes. (a) Direct addressing DMFB is believed to require many PCB layers, while (b) pin-constrained DMFBs are designed to reduce the number of PCB layers.

when a legal route is found, or the chip design degenerates to a direct-addressing solution.

The aforementioned integrated algorithms typically employ greedy maze routers based on breadth-first search or Lee's algorithm [15] for 2-D grids [16], [35], [38], [39]. Some of the more advanced techniques in this space include: using Lee's algorithm to tether all of the electrodes in a pin group together and then modeling the escape process as a network flow [39] or integer linear program (ILP) [16]; ripping up and rerouting nets that may have contributed to failures [38]; and enhancing Lee's algorithm with an A* cost function [35]. It is also possible to formulate the entire multiterminal escape routing problem as an ILP [4]. Although ILP-based approaches may yield optimal solutions, we eschew their usage because they are unable to tractably scale to handle large problem instances, unless it can be proven that $P = NP$.

The negotiated congestion router presented in this paper repeatedly rips up and reroutes nets while adjusting the associated history and penalty costs in congested areas. Prior work has shown that this approach is more effective than straightforward maze routing [19], [20], [24], [31], and it is far more efficient than ILP-based formulations. The algorithm described in this paper can be run once, after pin mapping, or it can be called repeatedly as a subroutine by any of the integrated algorithms described above.

Our implementation differs from the aforementioned algorithms in one key respect: we attempt to minimize the number of PCB layers as an objective, rather than assuming one available PCB layer as a constraint. This is motivated by the observation that dual-sided PCB technology has a nonlinear cost curve: if n is an odd number, and n- and $(n+1)$-layer PCB has the same cost; the cost increases for $n+2$ layers [11]. A second observation is that cost of adding an extra control pin is much lower than the cost of adding an extra PCB layer.

In response, we include an optional layer minimization step that increases the pin count by breaking apart pin groups when doing so can reduce the number of PCB layers. In many cases, a small increase in the number of control pins can reduce the number of PCB layers, leading to an overall reduction in cost.

III. Multiterminal PCB Escape Routing

This section outlines our multiterminal PCB escape routing algorithm for DMFBs based on the principle of negotiated congestion. The algorithm takes as input the DMFB dimensions, electrode locations, and the preliminary pin assignment. As noted earlier, a pin group is the set of electrodes driven by a common external control pin. The pin assignment can either be direct addressing, in which each electrode is driven by a unique control pin, or pin-constrained, where electrodes share control pins. Each electrode must belong to exactly one pin group, as it can only be driven by one control pin; electrodes that lack a control pin driving them can be removed from the DMFB, as they are not used during assay operation.

Initially, the router tries to find a legal escape route using one PCB layer; if this is not possible, it searches for a multilayer route with the minimum number of layers. In the most straightforward configuration, the algorithm does not modify the pin assignment; a more advanced configuration allows the router to modify the pin assignment opportunistically, when doing so can reduce the number of layers. To obtain a legal solution, each net must be routed on exactly one layer; introducing vias to allow nets to switch PCB layers is not permitted because doing so would degrade signal integrity [2], [22].

A. Problem Definition

The input to the multiterminal escape routing problem is a pin map $M = (P, E)$ and a routing graph G (described in the next section). P is the set of external control pins, and E is a set of electrodes. E is partitioned into $|P|$ subsets (pin groups) such that E_{p_i} contains the electrodes driven by control pin $p_i \in P$. The location of each electrode in the routing graph G is known, but the location of pin p_i on the perimeter of the chip has not yet been determined. An escape route for p_i is a multiterminal routing tree, within the routing graph, that connects all of the electrodes in E_{p_i} to one another and to any vertex on the perimeter of the chip; in the case of a single-terminal net, the routing tree degenerates to a path. A legal escape routing solution for two (or more) nets p_i and p_j such that $i \neq j$ is a set of two (or more) vertex-disjoint trees in the routing graph [4], [5], [14], [19], [37], [38].

In a multilayer PCB, a routing graph G_j is created for layer j. Each net must be routed on one PCB layer, and layer switching is not permitted, as mentioned above; all of the escape routes on the same layer, once again, must be disjoint. Our escape router has the freedom to assign (and reassign) nets to PCB layers; the number of PCB layers can be specified as a

constraint, or the escape router can attempt to minimize the number of PCB layers while ensuring legal routes for all nets.

Once an escape routing solution has been obtained for all nets, the next step is to establish a physical connection from the escape point to the control pin, which may be placed anywhere on the PCB. This problem can be handled by different algorithms [36], and is beyond the scope of this paper.

B. Graph Representation

We employ a planar graph, called the routing resource graph (RRG), to represent the free space on the PCB underneath the DMFB [19]. The orthogonal capacity of the RRG is the number of wires that can pass between two orthogonally adjacent electrodes, and the diagonal capacity is the number of wires that can pass between diagonally adjacent electrodes; in modern PCB technologies, the diagonal capacity is slightly larger than the orthogonal capacity [37]. An RRG tile can be generated for different orthogonal capacities: the orthogonal capacity is determined by the PCB technology (wire diameter and spacing rules) and the electrode dimension; the diagonal capacity is derived from the tile size and its orthogonal capacity. A higher orthogonal capacity can reduce the number of layers needed since more wires are able to pass between the electrodes; our algorithm and RRG can scale to any orthogonal capacity, but through experiment, we found diminishing returns on orthogonal capacities above three [11].

The RRG is a 2-D array of tiles, where each tile itself is a planar sub-RRG, as shown in Fig. 6. Each tile contains a set of edge nodes which are either escape nodes on the perimeter of the chip, or interface nodes to adjacent tiles. The tile also includes internal nodes, which represent the physical portion of the PCB layer available for routing wires. The black nodes represent control electrodes, or the vias which connect to the electrodes; these nodes are sinks for the router. The DMFB (including electrodes) lies on top of the PCB.

Without loss of generality, escape routing physically establishes a connection between a control pin and the electrode(s) that it drives on layer n. This means that a via must be created from the 2-D position of each electrode upward from layer n to connect to the electrode above it. This via creates a physical blockage at the same position as the electrode(s) on all PCB layers above n; thus, it is not possible to route wires through these positions, so the corresponding nodes and all incident edges must be removed from the RRG for each layer above n. Since the via does not extend below layer n, the RRG nodes in the same position as the electrode can be used for routing on all layers below n.

C. Single-Layer, Multiterminal Escape Routing

First, we describe a multiterminal PCB escape routing procedure for a single PCB layer. The pseudocode, shown in Fig. 8, takes the RRG and pin mapping as input. The routing phase (lines 9–11) executes a multiterminal variant of Lee's maze routing algorithm [15] on each pin group p_i. Under the negotiated congestion paradigm, multiple paths corresponding to distinct nets may share RRG nodes and edges [19], [20]. In each subsequent iteration of Lee's algorithm the nets are rerouted, attempting to avoid intersections, until a set of disjoint paths (i.e., a legal solution) is obtained.

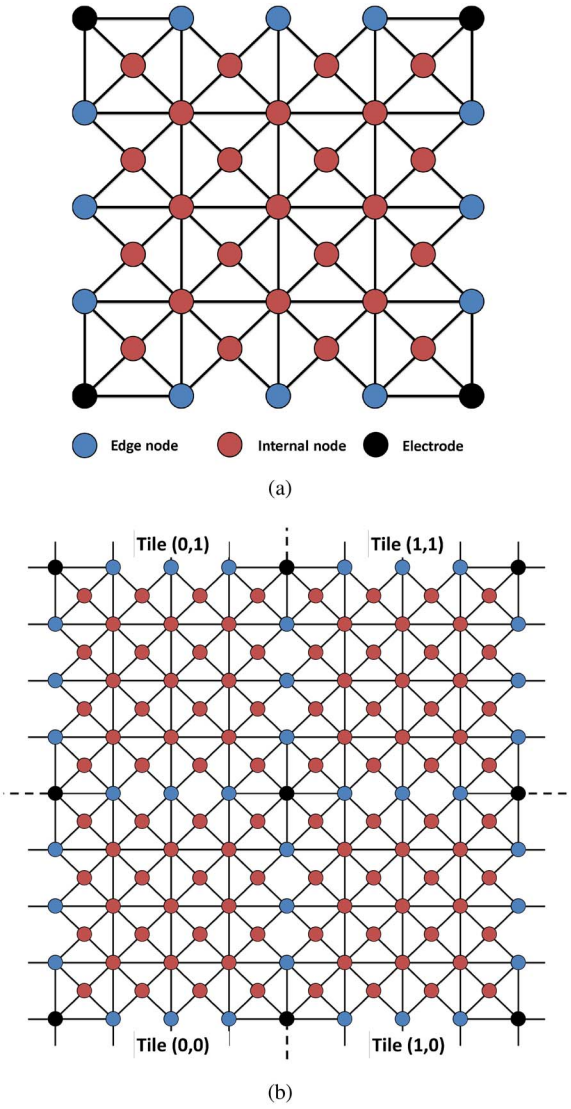

Fig. 6. (a) RRG tile. (b) 2×2 tiled chip. The lines going off-chip in indicate external pins. Black nodes represent electrodes, red nodes represent open space usable for routing, and blue nodes represent the edge of the chip.

Lee's algorithm [15] is used to route nets corresponding to pin groups one at a time. First, the RRG is modified by adding a supersource node which connects to each external pin on the RRG perimeter. This supersource is used as the source of the search, while the set of electrodes $e_j \in E_{p_i}$ are the sinks. A breadth first search is then performed, marking each cell with the iteration it was discovered. Once the first sink (i.e., an electrode $e_j \in E_{p_i}$) is found during the search, the path is obtained by tracing back to the node just before the supersource. This path is added to the current net W_{p_i} corresponding to that electrode group, and will be used as the escape wire for that group. The entire net W_{p_i} is then reinitialized as the set of source nodes. The supersource is removed from the RRG to prevent future iterations from escaping to the edge of the chip; it is reinserted into the RRG when the next net is routed.

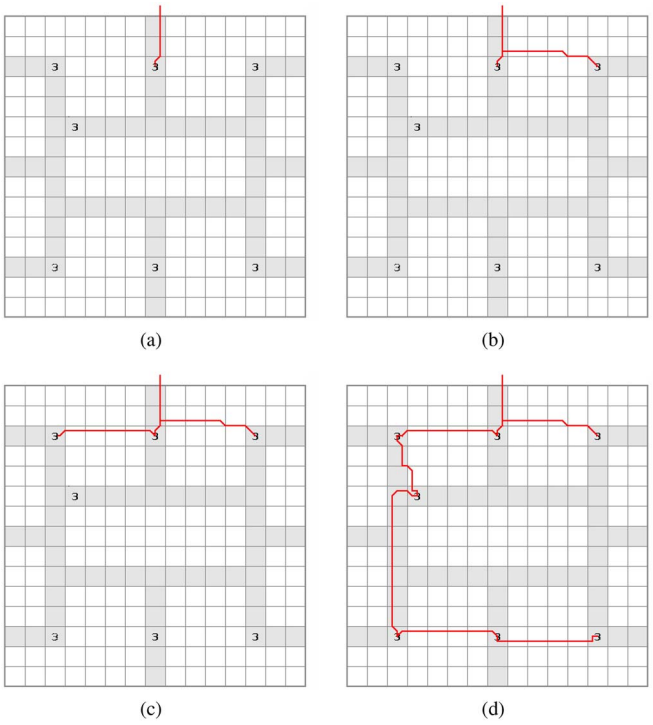

Fig. 7. Routing a single pin group. (a) First link establishes a connection between the supersource and the first electrode discovered by Lee's algorithm. (b) and (c) Second and third links, respectively, establish a multiterminal connection to the second and third electrodes discovered. (d) Process continues until all electrodes in the group are discovered.

Input : $RRG, P :=$ set of pins with unrouted nets
Output : $R :=$ set of pins with routed nets
//$M :=$ max iterations
//$HC :=$ History cost
//$i :=$ iteration
1: $i = 0$
2: $R_{best} \leftarrow \emptyset$
3: **for all** Nodes $x \in RRG$ **do**
4: $x_{HC} := 0$
5: **end for**
6: **repeat**
7: Rip up routes
8: $R_{curr} \leftarrow \emptyset$
9: **for all** $p_i \in P$ **do**
10: $R_{curr} \leftarrow$ LeeMazeRoute(p_i)
11: **end for**
12: **for all** Nodes $x \in R_{curr}$ **do**
13: **if** $x_{occ} > 1$ **then**
14: $x_{HC} = x_{old\_HC} + (x_{occ} - 1)$
15: **end if**
16: **end for** *//int(R):=#intersections*
17: **if** int$(R_{curr}) <$ int(R_{best}) **then**
18: $R_{best} \leftarrow R_{curr}$
19: **end if**
20: i += 1;
21: **until** R_{best} has no intersections **or** $(i >= M)$
22: **if** int$(R_{best}) > 0$ **then**
23: Failed to route
24: **else**
25: **return** R_{best}
26: **end if**

Fig. 8. Pseudocode for the single-layer DMFB escape router.

This breadth first search is repeated until all electrodes (sinks) in the group E_{p_i} are discovered. The source nodes are initialized to contain the stack of current nodes on the net W_{p_i} during each run. Fig. 7 shows Lee's multiterminal escape routing algorithm finding the escape wire off chip, then routing between seven electrodes in the same pin group.

Negotiated congestion allows multiple nets to share the same routing resource(s) during the search; although such routing solutions are illegal, allowing the router to explore them during the search leads to higher quality solutions when and if the search converges. As the search processes negotiated congestion assigns a history cost to nodes that are presently in use in order to reduce the likelihood that another net routed during the current iteration will share the node. After routing all pin groups ($p_i \in P$) the negotiated congestion router increases the history cost of each RRG node x that is shared among multiple nets (lines 12–16 in Fig. 8) using

$$\text{History}_x = \text{old\_history}_x + (\text{occupancy} - 1). \quad (1)$$

A legal routing solution is obtained if all nets are routed without any shared nodes; otherwise, all nets are immediately ripped up and rerouted on the RRG using the updated history costs. Intuitively, as the history costs of congested RRG nodes increases over time, the likelihood that nets continue to route through these nodes is reduced, nudging the overall routing solution toward convergence.

There is no guarantee that negotiated congestion will converge to a legal solution, presuming that one even exists. As such, the algorithm could, presumably, loop indefinitely.

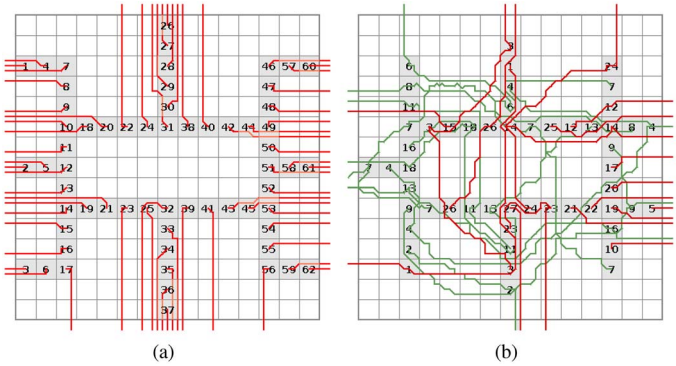

Fig. 9. (a) PCR_DA benchmark routed successfully using one layer. (b) Zhao_Protein benchmark failed to route on one layer: red lines indicate successful routes, and green lines indicate routes that failed due to intersections.

To prevent this from occurring, a maximum number of iterations is established *a-priori*; if a legal route is not achieved after the maximum number of iterations, then the router quits and reports a failure to the user. Based on previous work [20], we let the router iterate 30 times before declaring a failure. Fig. 9(a) and (b), respectively, shows examples of pin assignments that can and cannot be routed on a single PCB layer.

Input : $RRG, P :=$ set of unrouted pins
Input : $E :=$ set of electrodes (sinks)
Output : $L :=$ set of routed PCB layers

```
repeat
    P:=singleLayerRouter(RRG, P)
    l_curr ← ∅
    for all p_i ∈ P do
        if !intersect(p_i, l_curr) then
            l_curr ← p_i
            for all e_j ∈ E_{p_i} do
                create_via(e_j)
            end for
            P ← P − {p_i}
        end if
    end for
    L ← L ∪ l_curr
until P = ∅
return L
```

Fig. 10. Pseudocode for the LNC DMFB escape router. In principle, singleLayerRouter could be any single-layer routing method. The intersection function determines if the current net shares shares any routing resources with other previously routed nets.

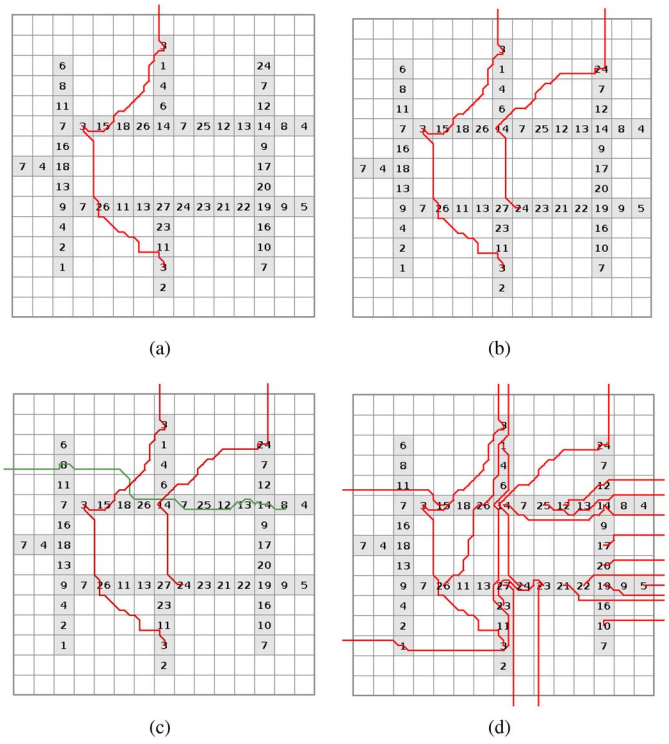

Fig. 11. LNC routing example. (a) Net for Pin 3 routes successfully. (b) Net for Pin 24 routes successfully without intersecting the net for Pin 3. (c) Net for Pin 8 routes unsuccessfully, indicated by the green line, intersecting with the routes computed for Pins 3 and 24; Pin 8 will be removed from the current layer and eventually routed on a subsequent layer. (d) Process continues until no other nets can route successfully on the current layer.

D. Multilayer Multiterminal Escape Routing

If a single-layer escape router fails, it is still possible to produce a legal and usable escape routing solution using multiple PCB layers. Fig. 10 presents pseudocode for an algorithm that we call layered negotiated congestion (LNC). LNC generalizes our single-layer routing algorithm so that it can route multiple layers.

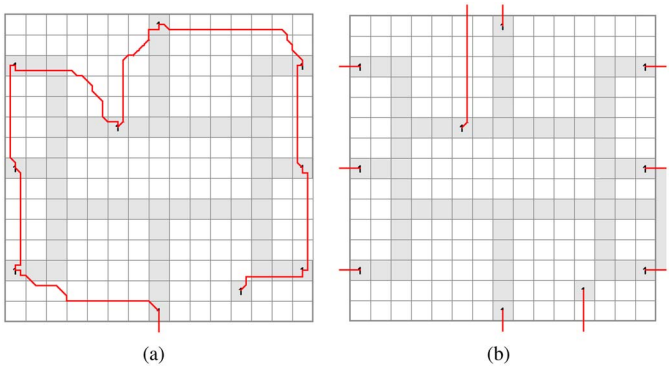

Fig. 12. (a) Pin 1 routed using only one external control signal, the wire wraps around the entire chip; at most three more control pins will be able to route into internal portion of the current layer. (b) Electrodes are now allowed to route directly off chip instead, resulting in 9 additional pins, but more nets are now able to route on this layer, reducing total number of layers.

If the single-layer negotiated congestion algorithm fails (i.e., multiple nets share a node after 30 iterations), routed nets are processed one-by-one in-order; through experimentation we discovered that ordering the nets does not significantly change the results. If the current net shares one or more node with a net that has been previously processed and has been routed successfully, then it is removed from the routing solution for the current layer; otherwise, it is left in-place. Fig. 11 shows an example of this process.

Next, LNC generates the RRG for the subsequent layer, blocking off all RRG nodes corresponding to positions where vias have already been instantiated, and routes all of the remaining nets, i.e., those that were not successfully routed on the first layer. This process repeats until all nets are routed. Eventual convergence is guaranteed since the first net is guaranteed to route on an empty layer.

E. Layer Minimization

The LNC escape routing method works well in general, however, it does have some drawbacks. One example is where a single control pin shares a large number of electrodes that span the perimeter of the chip, as shown in Fig. 12(a). Regardless of the location of the escape pin, this route will block many other nets from being able to route on the current layer. This can significantly reduce the number of routes per layer, and unnecessarily increase the number of PCB layers.

To address this concern, we propose layer minimized negotiated congestion (LMNC) to reduce the likelihood that adverse routes occur by relaxing the pin grouping when doing so it likely to reduce the number of PCB layers. For example, Fig. 12(b) shows each electrode in the same pin group routed off-chip; although this increases the number of control pins, doing so reduces congestion on the current layer.

LNC initiates the route for each net at the supersource node until it finds the first electrode $e_j \in E_{p_i}$ (sink) in the pin group; it then traces back a path from e_j to the supersource, and uses that path as the set of sources for the remainder of the search, which ensures that routed net W_i escapes exactly once.

Input : $S :=$ Supersource node
Input : $E_{p_i} :=$ Unrouted electrodes
Output : $W :=$ set of routed nets
1: $wavefront :=$ Priority queue of cells to search *//Sorted by cost*
2: $W \leftarrow \emptyset$
3: $wavefront.push(S)$
4: **repeat**
5: **repeat**
6: $cell_{curr} \leftarrow wavefront.pop()$
7: **for all** $cell_i$ adjacent to $cell_{curr}$ **do**
8: $update\_cost(cell_i)$
9: $wavefront.push(cell_i)$
10: **end for**
11: **until** $cell_{curr} = e_k \in E_{p_i}$ for some i
12: $w_{new} =$ traceBack(e_k)
13: **if** intersect($w_{new}, w_j \in W$) **then**
14: $w_j \leftarrow w_j \cup w_{new}$
15: **else**
16: $W \leftarrow W \cup w_{new}$
17: **end if**
18: $wavefront \leftarrow \emptyset$
19: $wavefront.push(W)$
20: $wavefront.push(S)$
21: $E_{p_i} \leftarrow E_{p_i} - \{e_k\}$
22: **until** $E_{p_i} = \emptyset$
23: **return** W

Fig. 13. Lee's algorithm [15], modified for usage with LMNC escape routing. Line 20 facilitates layer minimization by allowing the wavefront expansion to emanate from the supersource during each iteration.

LMNC relaxes this constraint, allowing electrode group E_{p_i} to be partitioned into multiple groups, each of which is driven by a new control pin.

As shown in Fig. 13, line 20, LMNC does not remove the supersource node from the set of sources. At each step, the wavefront expands the supersource node as well as the set of sinks that have been discovered thus far. If a new sink e_j is discovered by expansion from an existing net, then it is added to that net; however, if e_j is discovered by expansion from the supersource node, then e_j is added to a new electrode group $E_{p_{i_j}}$ that is driven by a new control pin p_{i_j} with a new (partially) routed net W_{i_j} that escapes from e_j to the perimeter of the chip.

The wavefront expansion then continues from all partially routed nets $W_{i_1}, W_{i_2}, \ldots, W_{i_j}$, as well as the supersource. When all sinks are discovered, the resulting nets are disjoint (except for the supersource node), i.e., $W_{i_j} \cap W_{i_k} = \emptyset$ when $k \neq j$, but drive all electrodes in the pin group E_{p_i}. In other words, $E_{p_i} = \bigcup_j E_{p_{i_j}}$ and $W_i = \bigcup_j W_{i_j}$.

F. Detour Reduction

The LNC and LMNC algorithms repeatedly call Lee's algorithm [15] to compute routes for each net, while adjusting the history and penalty costs to reduce the likelihood of intersections occurring on regions of the chip that were highly congested in previous iterations. For any given layer of the chip, the routes that were chosen may have detours around unused portions of the chip in order to avoid occupying cells with high costs. Sometimes, these detours lead to unnecessary increases in wirelength. The following two sections introduce extensions to LNC and LMNC that attempt to limit the occurrence of unnecessary detours and reduce wirelength.

1) Limited-Turn Maze Expansion: Lee's algorithm does not specify the order in which to expand cells at the wavefront. The first extension, which borrows ideas from prior work by Rubin [31], favors expansion in the current direction of the search. Rubin's algorithm was originally applied to a 2-D grid; our implementation, which we call limited-turn maze expansion (LTME) is modified to account for the diagonal edges in the routing graph. The priority for node expansion under LTME is as follows.

1) The cell which is in exactly the same direction.
2) The cells which are diagonally in the same general direction.
3) The cells which are perpendicular to the current direction.
4) Finally, the cells which are diagonally in the opposite direction.

For example, if the current direction is north (N), then the highest priority is to expand the cell in the N direction; the second highest priorities are to expand the two cells in the NE and NW directions; the third highest priorities are to expand the two cells in the E and W directions; and the lowest priorities are to expand the two cells in the SE and SW directions.

The basic premise is that favoring the current direction of expansion will minimize the number of unnecessary detours (i.e., those that are not required to route around occupied routing resources).

2) Post-Processing: The second extension is a post-processing (PP) step that rips up and reroutes each net, one-at-a-time, while leaving the other routed nets in-place. If Lee's algorithm is used, then the rerouted net may have a route that is shorter and/or has fewer turns, as the history cost that has accumulated during negotiated congestion are now ignored; if LTME is used, then the resulting route is likely to have fewer detours as well. A legal route is guaranteed to exist since the route that was originally ripped up was legal.

For LMNC, we start with the original pin groups (not the ones that have been split apart by the LMNC process). We post-process each layer one-at-a-time, and do not move nets (or subnets) between layers. First we try to route the entire pin group using the LNC router; if we are lucky, this route succeeds without creating extra pin groups, and can lead to a pin count reduction compared to the initial LMNC result. If the LNC route fails, we reroute the net using the LMNC router; similarly, this route may reduce the pin-expansion compared to the initial LMNC result.

IV. Experimental Results

A. Baseline Router

Our baseline escape router is based on a subroutine which is part of an integrated pin-mapper and wire router targeting single-layer PCBs [38]; when a single-layer escape routing solution cannot be found, the pin-mapping solution is relaxed

and the algorithm reroutes some of the nets. The escape routing subroutine is based on Lee's algorithm [15] coupled with a rip-up and reroute PP step. Our implementation of this escape router uses a process akin to LNC to extend single layer routes to multiple layers as described in Fig. 10. Henceforth, we refer to this escape router as the Naïve method. Our experiments compare the Naïve method with the LNC and LMNC escape routers introduced in this paper; we also evaluate the runtime and solution quality of the negotiated congestion router compared to the Naïve method as subroutine for integrated algorithms that simultaneously optimize the pin-mapping and escape routing solutions.

B. Experimental Setup

All experiments reported here were performed on a Dell Optiplex 580 PC with an AMD Phenom II X2 B53 Processor, and 4 GB of memory running 32-bit Ubuntu 14.04 LTS.

The first experiment compares LNC and LMNC to the Naïve method on several precomputed pin-mappings reported in previous papers [11], [18], [41]. We assume an orthogonal capacity of three and a diagonal capacity of six in all experiments [11].

A second experiment evaluates the impact of detour avoidance techniques, as discussed in Section IV-D, on the same benchmark set.

The third experiment uses three pin-mappers [12], [13], [41] and one integrated pin-mapping and escape routing algorithm [39] to compare LNC and LMNC to the Naïve method, when used as a subroutine.

The fourth and final experiment considers a more efficient implementation of the integrated algorithm, which uses the Naïve method as an escape routing subroutine, but then applies LNC and LMNC as PP steps.

Escape routing alone does not determine the cost of a PCB; the exact PCB cost cannot be known until the PCB is fully laid out, including both the number of layers and 2-D planar dimensions. The PCB will have other IC components, such as a microcontroller (to interface to a host PC) [10] and shift registers when the number of DMFB control pins exceeds the supply of microcontroller outputs [11]. Prior work has shown that the number of PCB layers has a much greater impact on cost than the number of control pins [11].

C. Experiments on Known Pin-Mapping Solutions

All benchmarks used in this experiment are either direct addressing pin assignments, or optimized pin assignments taken from previously published papers. The benchmarks labeled Zhao_XXX were taken from [41] and those labeled Luo_XXX from [18]; these benchmarks do not use every electrode in the 2-D grid. The benchmarks labeled XXX_DA impose a direct addressing scheme on the pattern of used electrodes from the preceding references. The benchmark FPPC 12×15 is taken from [11]. The benchmarks IA_XXX are individually addressing chips, which are similar to direct addressing chips, but use all electrodes.

1) Number of PCB Layers and Pin Count: Table I reports the number of PCB layers obtained by all three algorithms.

TABLE I
NUMBER OF (PCB) LAYERS AND (EXTERNAL CONTROL) PINS OBTAINED BY THE NAÏVE METHOD, LNC, AND LMNC

		Naïve	LNC	LMNC	
Benchmark	Pins	Layers	Layers	Layers	Pins[1]
Zhao_PCR	14	7	4	3	35
Luo_PCR	22	12	5	3	43
PCR_DA	62	1	1	1	62
Zhao_Protein	27	11	3	3	43
Luo_Protein	21	9	4	3	33
Protein_DA	54	1	1	1	54
Zhao_InVitro	25	9	4	3	47
Luo_InVitro	21	12	5	3	44
InVitro_DA	59	1	1	1	59
Zhao_Multi	32	12	5	3	48
Luo_Multi	27	17	6	3	57
Multi_DA	81	2	1	1	81
FPPC 12x15	33	10	4	3	72
IA 10x10	100	2	1	1	100
IA 15x15	225	3	2	2	225
IA 15x19	285	3	2	2	285
IA 30x30	900	7	5	5	900

[1]LMNC adds additional pins in order to reduce the layer count.

The Naïve method and LNC to not alter the pin-mapping, while LMNC does; the last column shows the resulting pin count for LMNC. Table I shows that LNC and LMNC achieve far fewer PCB layers than the Naïve method. LNC and LMNC achieve identical results for all eight direct and individually addressable chips in Table I, as there is no opportunity for LMNC to further increase the pin count. LMNC achieved fewer PCB layers than LNC for eight of the nine remaining pin-constrained chips.

Fig. 17 shows the Zhao_Protein benchmark routed using the Naïve method, requiring seven PCB layers. Meanwhile, LNC required four PCB layers, as shown in Fig. 18, and LMNC required just three layers, as shown in Fig. 19.

Altogether, the results reported in Table I clearly establish the algorithmic superiority of negotiated congestion compared to the Naïve method. On average, LNC reduced the number of layers by 42.59%, with a maximum savings of 72.73%, while LMNC was able to further reduce the number of layers by 19.29% with a maximum improvement of 50%.

2) Wirelength: Table II reports the average and total wirelength obtained by the Naïve method, LNC, and LMNC. Compared to the Naïve method, LNC reduced total and average wirelength, indicating an improvement in routability. Compared to LNC, LMNC increased total wirelength, but reduced average wirelength per control pin. The increase in total wirelength occurs because each additional control pin creates a new net that must escape; the reduction in average wirelength per net suggests improvements in routability due to LMNC's ability to intelligent split apart pin groups. Once again, LNC and LMNC obtain identical solutions for direct and individually addressable chips.

3) Runtime: Fig. 14 reports the runtimes of LNC and LMNC normalized to the runtime of the Naïve method. The Naïve method, which calls Lee's algorithm once per PCB layer, ran considerably faster than LNC, which may iterate up

TABLE II
AVERAGE AND TOTAL WIRELENGTH OBTAINED BY THE NAÏVE METHOD, LNC, AND LMNC

		Naïve		LNC		LMNC		
Benchmark	Pins	Total	Avg.	Total	Avg.	Pins[1]	Total	Avg.
Zhao_PCR	14	5500	392.86	5066	361.86	35	7032	200.91
Luo_PCR	22	7228	328.55	6840	310.91	46	8386	195.02
PCR_DA	62	9924	160.06	9904	159.74	62	9904	159.74
Zhao_Protein	27	7348	272.15	6704	248.30	36	7478	173.91
Luo_Protein	21	7014	334.00	6262	298.19	33	7314	221.64
Protein_DA	54	9028	167.19	8900	164.81	54	8900	164.81
Zhao_InVitro	25	6932	277.28	6662	266.48	45	8120	172.77
Luo_InVitro	21	7660	364.76	6002	285.81	34	6878	202.29
InVitro_DA	59	9210	156.10	9006	152.64	59	9006	152.64
Zhao_Multi	32	9024	282.00	8418	263.06	49	9240	192.50
Luo_Multi	27	9084	336.44	8778	325.11	57	9944	174.46
Multi_DA	81	13226	163.28	12858	158.74	81	12858	158.74
FPPC 12x15	33	9314	282.24	8556	259.27	72	11458	159.14
IA 10x10	100	10892	108.92	10964	109.64	100	10964	109.64
IA 15x15	225	35350	157.11	35882	159.48	225	35882	159.48
IA 15x19	285	50384	176.79	49954	175.28	285	49954	175.28
IA 30x30	900	269794	299.77	268012	297.79	900	268012	297.79

[1]LMNC adds additional pins in order to reduce the layer count.

TABLE III
TOTAL AND AVERAGE WIRELENGTH, NUMBER OF (PCB) LAYERS, AND NUMBER OF (EXTERNAL CONTROL) PINS OBTAINED BY LMNC USING DIFFERENT DETOUR MINIMIZATION TECHNIQUES

		LMNC				LMNC + PP				LMNC + LTME				LMNC + LTME + PP			
Benchmarks	Pins	Total	Avg.	Layers	Pins[1]	Total	Avg.	Layers	Pins[1]	Total	Avg.	Layers	Pins[1]	Total	Avg.	Layers	Pins[1]
Zhao_PCR	14	7032	200.91	3	35	6958	193.28	3	36	7278	196.7	3	37	7228	195.35	3	37
Luo_PCR	22	8386	195.02	3	43	8730	194	3	45	7720	197.95	3	39	7696	197.33	3	39
PCR_DA	62	9904	159.74	1	62	9908	159.81	1	62	9872	159.23	1	62	9884	159.42	1	62
Zhao_Protein	27	7478	173.91	3	43	7570	199.21	3	38	7252	213.29	3	34	7272	213.88	3	34
Luo_Protein	21	7314	221.64	3	33	6858	190.5	3	36	7118	187.32	2	38	7088	186.53	2	38
Protein_DA	54	8900	164.81	1	54	8916	165.11	1	54	8832	163.56	1	54	8884	164.52	1	54
Zhao_InVitro	25	8120	172.77	3	47	7972	177.16	3	45	7124	182.67	2	39	7172	183.9	2	39
Luo_InVitro	21	6878	156.32	3	44	7232	195.46	3	37	6648	179.68	3	37	6534	176.59	3	37
InVitro_DA	59	9006	152.64	1	59	8946	151.63	1	59	9158	155.22	1	59	9018	152.85	1	59
Zhao_Multi	32	9240	192.5	3	48	9074	181.48	3	50	9442	185.14	3	51	9048	177.41	3	51
Luo_Multi	27	9944	174.46	3	57	10386	182.21	3	57	10154	169.23	3	60	10154	169.23	3	60
Multi_DA	81	12858	158.74	1	81	12882	159.04	1	81	12856	158.72	1	81	12854	158.69	1	81
Total		105060	2123.46	28	606	105432	2148.89	28	600	103454	2148.71	26	591	102832	2135.7	26	591

[1]LMNC adds external control pins to reduce the number of PCB layers

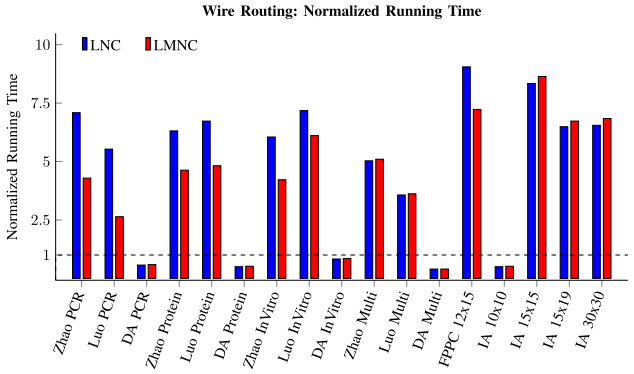

Fig. 14. Runtimes of LNC and LMNC normalized to that of the Naïve method.

to thirty times [20]. LMNC, however, runs faster than LNC because the reduction in layers due to intelligent pin group splitting leads to faster convergence with fewer calls to the negotiated congestion router. Since PCB layout is performed offline, we consider the runtime overhead of LNC and LMNC to be tolerable.

D. Detour Avoidance Experiments

Table III reports the effectivness of the LTME and PP detour minimization techniques, as described in Section III-F, when integrated with LMNC. Results are reported for LMNC in isolation, LMNC with LTME and PP individually, and LMNC with both LTME and PP. Table III reports the initial pin count of each benchmark, the resulting pin count and number of PCB layers after minimization, and the average and total wirelength; the last row reports summated results across all of the benchmarks. The best overall results were obtained running LMNC with both LTME and PP.

Detour minimization minimally impacted the number of PCB layers. LMNC + PP yielded the same number of PCB layers as LMNC in isolation for each benchmark; LMNC + LTME and LMNC + LTME + PP reduced the number of PCB layers compared to LMNC by one for two benchmarks, Luo_Protein and Zhao_InVitro. This suggests that LTME offers a substantial, yet limited, improvement in routability.

LMNC + PP reduced the summated pin count from 606 to 600 compared to LMNC, while both LMNC + LTME and LMNC + LTME + PP reduced the summated pin count

TABLE IV
NUMBER OF (PCB) LAYERS AND (EXTERNAL CONTROL) PINS OBTAINED BY THE NAÏVE METHOD, LNC, AND LMNC WHEN USED AS ESCAPE ROUTING SUBROUTINES BY INTEGRATED PIN-MAPPERS WIRE ROUTERS

		Clique [41]				Power-Aware [13]				Reliability-Aware [12]				Switching-Aware [39]				Routability-Aware (ACER) [16]			
		Naïve	LNC	LMNC		Naïve	LNC	LMNC		Naïve	LNC	LMNC		Naïve	LNC	LMNC		Naïve	LNC	LMNC	
Benchmarks	Pins	Layers	Layers	Layers	Pins[1]	Layers	Layers	Layers	Pins[1]	Layers	Layers	Layers	Pins[1]	Layers	Layers	Layers	Pins[1]	Layers	Layers	Layers	Pins[1]
PCR	18	5	5	4	42	6	5	4	39	7	6	5	44	6	5	4	42	7	5	3	44
In Vitro 4x4	64	10	10	5	146	11	10	5	157	11	12	4	149	10	9	5	130	10	9	5	132
Protein	71	10	9	5	134	9	11	5	121	10	12	6	136	9	11	5	148	12	9	4	133
Protein Split 5	90	10	10	5	127	12	11	5	160	10	10	5	150	9	10	5	163	7	7	3	191
CoDos	38	7	7	4	65	6	6	4	79	6	6	4	47	8	6	4	59	7	6	4	62
Gorma 23/256	21	4	4	3	38	5	4	3	33	5	4	3	38	5	4	3	39	6	5	3	37
BIN 13/128	35	7	7	4	75	7	7	4	78	8	8	4	66	7	7	5	68	7	8	4	77
Remia 191/1024	41	6	5	4	59	8	8	5	78	7	7	5	67	6	5	4	64	7	7	4	87
Remia 641/1024	16	4	4	3	34	5	4	3	26	4	4	3	39	5	4	3	30	5	3	3	31
Remia 850/1024	19	4	4	3	35	4	4	3	30	5	4	3	34	4	4	3	29	4	4	3	31

[1]LMNC adds external control pins to reduce the number of PCB layers.

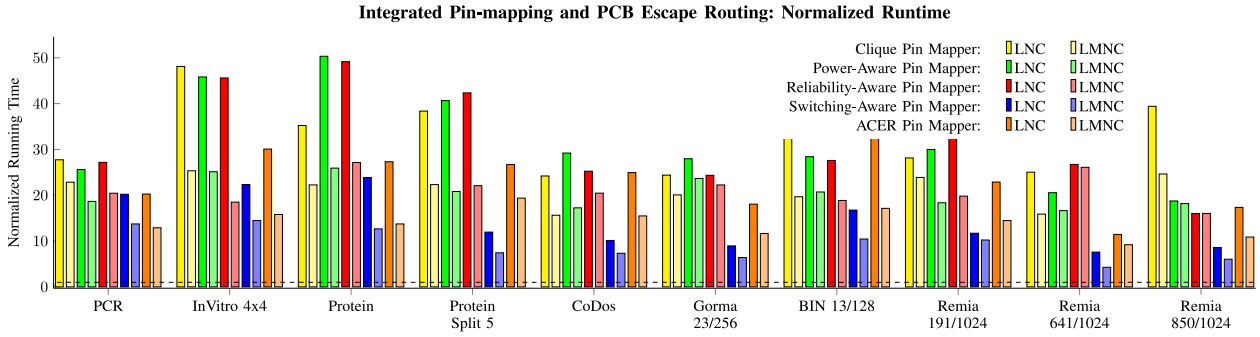

Fig. 15. Runtimes of different integrated pin-mapping and wire routing algorithms using LNC and LMNC as escape routing subroutines, normalized to the runtime of the same algorithms using the Naïve method.

further, to 591; all three detour minimization configurations increased the pin count for some of the benchmarks while reducing it for others; in aggregation, the results seemed favorable. Among the two benchmarks where detour minimization reduced the number of PCB layers, the pin count increased for Luo_Protein and decreased for Zhao_InVitro.

LMNC + PP had the highest summated wirelength, larger than LMNC; LMNC + LTME reduced the summated wirelength compared to LMNC, while LMNC + LTME + PP achieved the lowest summated wirelength. For some benchmarks, the introduction of detour reduction techniques increased total wirelength, while other benchmarks exhibited the opposite effect. Thus, there is no clear, uniform trend, although the summated results show that LMNC + LTME + PP achieves the lowest summated wirelengths.

In summary, the results reported in Table III suggest that detour minimization techniques are effective, and that the best results can be obtained by using LMNC with both LTME and PP; however, for any given benchmark, the impact of detour minimization may turn out to be unfavorable. In practice, all four algorithmic configurations should be evaluated and the best result selected.

E. Escape Routing as Subroutine

1) Experiment Details: This section compares the performance of the Naïve method, LNC, and LMNC as subroutines for use within larger algorithmic frameworks that simultaneously co-optimize pin-mapping with escape routing to produce low-cost application-specific pin-constrained DMFBs. Reference [10] summarizes our implementation of

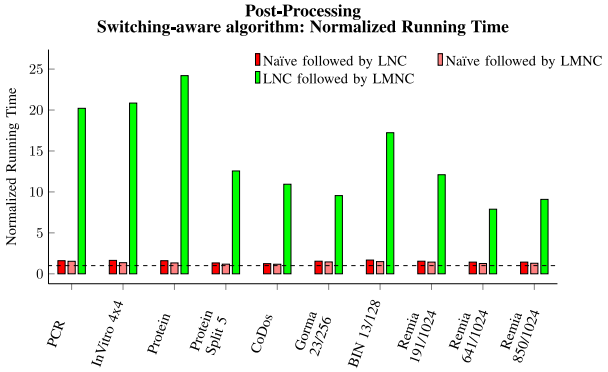

Fig. 16. Runtimes for the switching-aware [39] algorithm using the Naïve method as an escape routing subroutine followed by LNC and LMNC for PP, and using LNC as an escape routing subroutine followed by LMNC for PP. The runtimes are normalized to the runtime of the Naïve method sans PP.

these frameworks. We compare the performance of these escape routing subroutines in the context of five different optimization strategies.

The first four algorithms perform pin mapping upfront with PCB escape routing as a PP step: the first clique-based pin mapper [41]; and then power-aware [13], reliability-aware [12] and routability-aware (ACER) [16] pin-mappers.

The fifth algorithm is a switching-aware pin-mapper [39] which integrates PCB escape routing with the pin-mapping process; pin sharing decisions may be undone if the number of intermediate PCB layers grows too high, similar in principle to LMNC. The pin-mapping part of the algorithm limits the amount of switching activity at each electrode, which mitigates the device-level problem of contact-angle reduction.

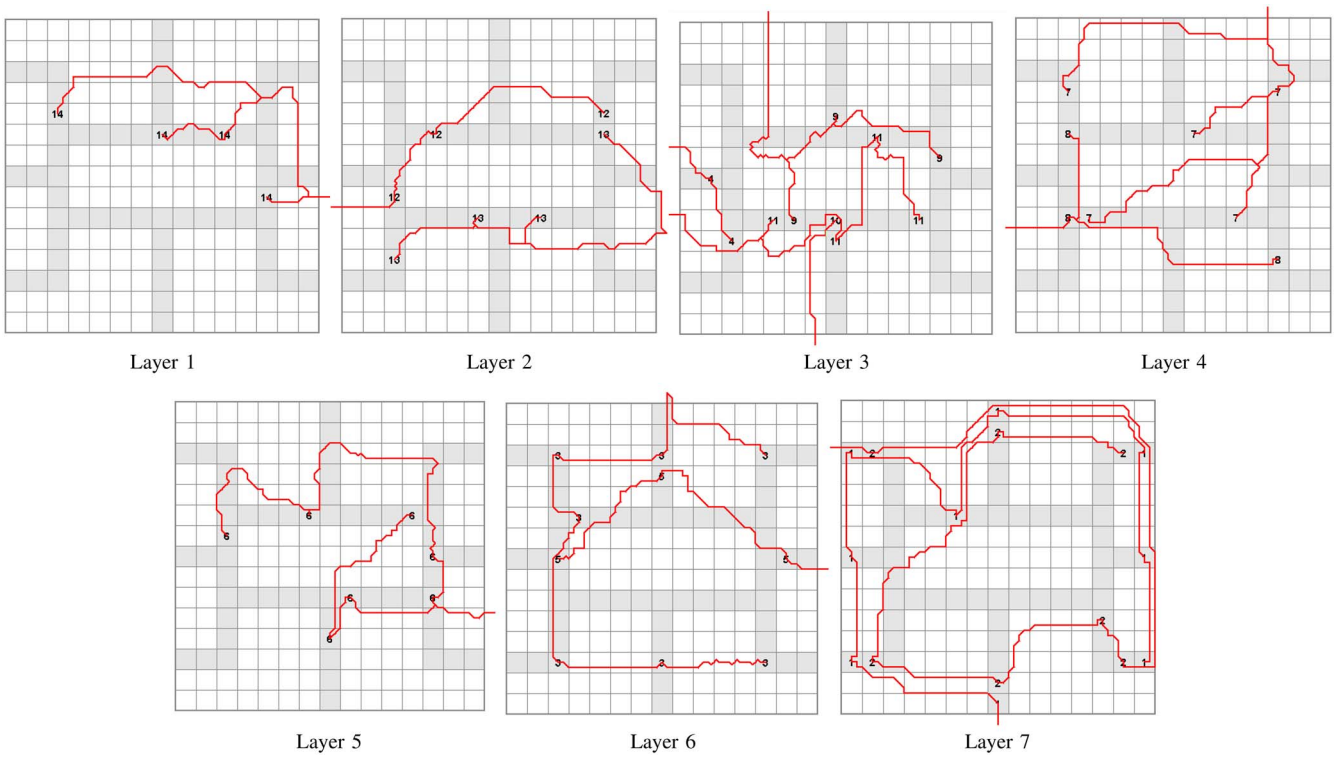

Fig. 17. Naïve method [38] routes the Zhao_PCR benchmark using seven PCB layers.

The escape route is computed and recomputed incrementally each time two pin groups are merged.

The algorithms are run on a set of assays summarized in [10]. Each assay is run through the synthesis flow targeting a 15 × 19 DMFB using identical schedulers, placers: list scheduling [8], [34] was used for all assays, except for "Protein Split 5," in which case only path scheduling [9] could find a legal solution; a virtual topology for placement [7]; and a maze router to compute droplet routes [7], [30]. Pin-mapping and wire routing were performed on the electrode activation sequence produced by the droplet router.

2) PCB Layers and Pin Count: Table IV compares the number of PCB layers and the number of external control pins count obtained by the Naïve method, LNC, and LMNC as escape routing subroutines. LNC yielded marginal reductions in the number of PCB layers compared to the Naïve method in most, but not all, cases, whereas, LMNC reduced the number of PCB layers in all cases, achieving as much as a 67% reduction; in two cases (Protein for the power-aware pin-mapper, and BIN 13/128 for ACER), LNC increased the number of PCB layers compared to the Naïve method. LMNC achieved its PCB layer reduction by increasing the pin count. As noted previously, the number of PCB layers has a greater impact on PCB cost than pin count, thus, the tradeoff here is considered to be favorable.

3) Runtime: Fig. 15 reports the runtime of LNC and LMNC, normalized to the Naïve method for the five pin mappers discussed previously. As expected, negotiated congestion causes LNC and LMNC to run slower than the Naïve method. This overhead is exacerbated for integrated pin-mappers that repeatedly call an escape routing subroutine. That being said, since pin-mapping and PCB layout (including, but not limited to, escape routing) are performed offline, we believe that the reduced layer counts justify the increase in runtime.

TABLE V
NUMBER OF (PCB) LAYERS AND (EXTERNAL CONTROL) PINS OBTAINED BY THE SWITCHING-AWARE PIN-MAPPER/ROUTER USING THE NAÏVE METHOD AS AN ESCAPE ROUTING SUBROUTINE AND PP USING LNC AND LMNC

		None	LNC	LMNC	
Benchmark	Pins	Layers	Layers	Layers	Pins[1]
PCR	22	6	5	4	42
InVitro 4x4	82	10	9	5	133
Protein	75	9	11	5	165
Protein Split 5	95	9	10	5	163
CoDos	47	8	6	4	59
Gorma 23/256	22	5	4	3	39
BIN 13/128	41	7	7	5	68
Remia 191/1024	46	6	5	4	64
Remia 641/1024	18	5	4	3	30
Remia 850/1024	20	4	4	3	29

[1]LMNC adds external control pins to reduce the number of PCB layers.

F. Post-Processing Experiments

The switching-aware pin-mapper repeatedly calls a PCB escape routing algorithm as a subroutine [39]; this yielded a two-hour runtime for LNC, and just under an hour for LMNC, compared to just ten minutes for the Naïve method. To counteract the long run times required for this algorithm we ran a separate set of experiments that used the Naïve method as

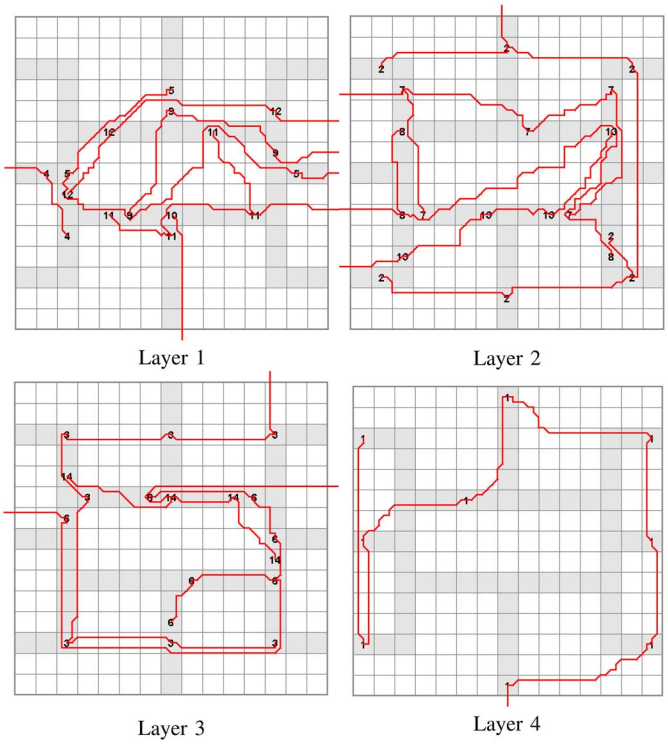

Fig. 18. LNC routes the Zhao_PCR benchmark using four PCB layers.

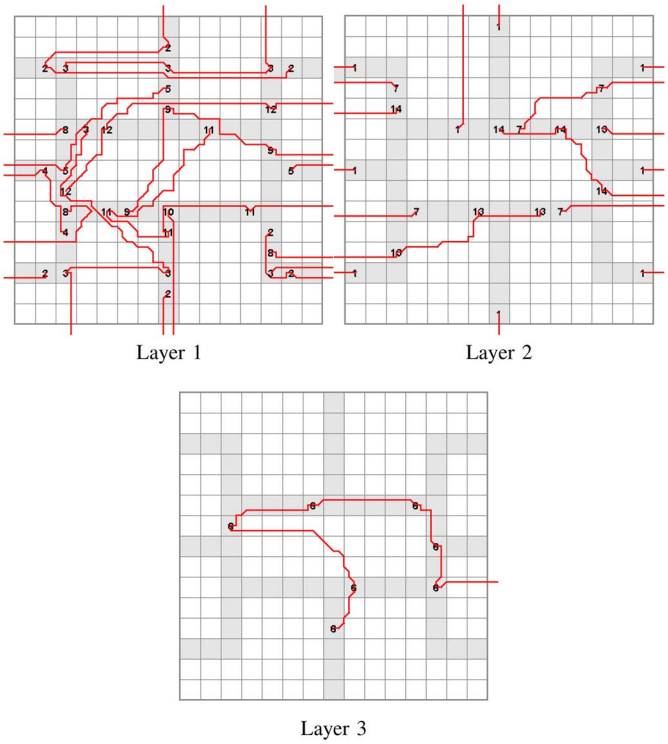

Fig. 19. LMNC routes the Zhao_PCR benchmark using three PCB layers, but increases the number of external control pins from 14 to 35.

the integrated subroutine. As a PP step, we then ran LNC and LMNC to try to reduce the number of PCB layers while running significantly faster. Table V and Fig. 16 report the result of these experiments.

Using LNC as a PP step was marginally effective, reducing the number of PCB layers by as many as two, while increasing the number of PCB layers by one in two cases; meanwhile, PP using LMNC reduced the number of PCB layers in all cases, with a maximum reduction of 50%. We performed a similar experiment using LNC as the integrated escape router with LMNC for PP, but this approach failed to yield further improvements; this result is not reported in Table V.

Fig. 16 reports the normalized runtimes of the switching-aware router and different escape routing and PP configurations. Using LNC as an escape-routing subroutine is significantly slower than using the Naïve method, regardless of which PP algorithm is chosen. Using the Naïve router with LNC and LMNC for PP increase the algorithmic runtime by 1.68× and 1.53× respectively; when LNC is used as the escape routing subroutine and LMNC for PP, the runtime increases by as much as 20.85×, which is untenable and mostly ineffective, as mentioned above.

V. Conclusion

Negotiated congestion was shown to be a useful an effective approach to the multiterminal escape routing problem for PCB-mounted DMFBs. An escape routing algorithm based on negotiated congestion can be run in standalone mode to route a PCB for a given chip, or it can be called as a subroutine by integrated algorithms that simultaneously co-optimize pin-mapping with PCB escape routing. By allowing the algorithm to break apart previously computed pin groups at the escape routing stage the LMNC variant of the proposed algorithm was able to reduce the number of PCB layers, which directly affects the cost of the system.

Escape routing alone creates the portion of the PCB underneath the DMFB; however, it does not solve other pertinent issues, including the placement of other components (e.g., microcontrollers, shift registers, etc.) on the PCB, nor does it instantiate connections between these components and the external control pins. Future work will attempt to automate these tasks, yielding fully functional and workable layouts for PCB-mounted DMFBs.

Acknowledgment

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] S. H. Au, M. D. Chamberlain, S. Mahesh, M. V. Sefton, and A. R. Wheeler, "Hepatic organoids for microfluidic drug screening," *Lab Chip*, vol. 14, no. 17, pp. 3290–3299, Sep. 2014.

[2] D. Brooks, *Signal Integrity Issues and Printed Circuit Board Design*. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

[3] K. Chakrabarty, "Design automation and test solutions for digital microfluidic biochips," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 1, pp. 4–17, Jan. 2010.

[4] J.-W. Chang, S.-H. Yeh, T.-W. Huang, and T.-Y. Ho, "An ILP-based routing algorithm for pin-constrained EWOD chips with obstacle avoidance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 11, pp. 1655–1667, Nov. 2013.

[5] J.-W. Chang, S.-H. Yeh, T.-W. Huang, and T.-Y. Ho, "Integrated fluidic-chip co-design methodology for digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 2, pp. 216–227, Feb. 2013.
[6] K. Choi *et al.*, "Automated digital microfluidic platform for magnetic-particle-based immunoassays with optimization by design of experiments," *Anal. Chem.*, vol. 85, no. 20, pp. 9638–9646, 2013.
[7] D. Grissom and P. Brisk, "Fast online synthesis of digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 356–369, Mar. 2014.
[8] D. T. Grissom and P. Brisk, "Fast online synthesis of generally programmable digital microfluidic biochips," in *Proc. 10th Int. Conf. Hardw./Softw. Codesign Syst. Synthesis (CODES+ISSS)*, Tampere, Finland, Oct. 2012, pp. 413–422.
[9] D. T. Grissom and P. Brisk, "Path scheduling on digital microfluidic biochips," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 26–35.
[10] D. T. Grissom *et al.*, "An open-source compiler and PCB synthesis tool for digital microfluidic biochips," *Integr., VLSI J.*, vol. 51, pp. 169–193, Sep. 2015.
[11] D. T. Grissom, J. McDaniel, and P. Brisk, "A low-cost field-programmable pin-constrained digital microfluidic biochip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 11, pp. 1657–1670, Nov. 2014.
[12] T.-W. Huang, T.-Y. Ho, and K. Chakrabarty, "Reliability-oriented broadcast electrode-addressing for pin-constrained digital microfluidic biochips," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2011, pp. 448–455.
[13] T.-W. Huang, S. Hong-Yan, and T.-Y. Ho, "Progressive network-flow based power-aware broadcast addressing for pin-constrained digital microfluidic biochips," in *Proc. 48th Design Autom. Conf., (DAC)*, San Diego, CA, USA, Jun. 2011, pp. 741–746.
[14] T.-W. Huang, S.-Y. Yeh, and T.-Y. Ho, "A network-flow based pin-count aware routing algorithm for broadcast-addressing EWOD chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1786–1799, Dec. 2011.
[15] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Trans. Electron. Comput.*, vol. EC-10, no. 3, pp. 1389–1401, 1959.
[16] S. S.-Y. Liu, C.-H. Chang, H.-M. Chen, and T.-Y. Ho, "ACER: An agglomerative clustering based electrode addressing and routing algorithm for pin-constrained EWOD chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 9, pp. 1316–1327, Sep. 2014.
[17] Y. Luo, B. B. Bhattacharya, and T.-Y. Ho, "Design and optimization of a cyberphysical digital-microfluidic biochip for the polymerase chain reaction," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 1, pp. 29–42, Jan. 2015.
[18] Y. Luo and K. Chakrabarty, "Design of pin-constrained general-purpose digital microfluidic biochips," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 18–25.
[19] Q. Ma, T. Yan, and M. D. Wong, "A negotiated congestion based router for simultaneous escape routing," in *Proc. 11th Int. Symp. Quality Electron. Design (ISQED)*, San Jose, CA, USA, Mar. 2010, pp. 606–610.
[20] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. 3rd Int. ACM Symp. Field-Programmable Gate Arrays (FPGA)*, Napa Valley, CA, USA, 1995, pp. 111–117.
[21] N. Mei, B. Seale, A. H. C. Ng, A. R. Wheeler, and R. Oleschuk, "Digital micro fluidic platform for human plasma protein depletion," *Anal. Chem.*, vol. 86, pp. 8466–8472, Aug. 2014.
[22] K. Mitzner, *Complete PCB Design Using OrCAD Capture and PCB Editor*. Burlington, MA, USA: Newnes, 2009.
[23] M. A. Murran and H. Najjaran, "Capacitance-based droplet position estimator for digital microfluidic devices," *Lab Chip*, vol. 12, no. 11, pp. 2053–2059, May 2012.
[24] R. Nair, "A simple yet effective technique for global wiring," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 6, no. 2, pp. 165–172, Mar. 1987.
[25] A. H. C. Ng *et al.*, "Digital microfluidic platform for the detection of rubella infection and immunity: A proof of concept," *Clin. Chem.*, vol. 61, no. 2, pp. 420–429, Feb. 2015.
[26] H. Norian, R. M. Field, I. Kymissis, and K. L. Shepard, "An integrated CMOS quantitative-polymerase-chain-reaction lab-on-chip for point-of-care diagnostics," *Lab Chip*, vol. 14, no. 20, pp. 4076–4084, Aug. 2014.
[27] S. Park, P. A. L. Wijethunga, H. Moon, and B. Han, "On-chip characterization of cryoprotective agent mixtures using an EWOD-based digital microfluidic device," *Lab Chip*, vol. 11, no. 13, pp. 2212–2221, Jul. 2011.
[28] M. G. Pollack, A. D. Shenderov, and R. B. Fair, "Electrowetting-based actuation of droplets for integrated microfluidics," *Lab Chip*, vol. 2, no. 2, pp. 96–101, May 2002.
[29] A. Rival *et al.*, "An EWOD-based microfluidic chip for single-cell isolation, mRNA purification and subsequent multiplex qPCR," *Lab Chip*, vol. 14, no. 19, pp. 3739–3749, Oct. 2014.
[30] P. Roy, H. Rahaman, and P. Dasgupta, "A novel droplet routing algorithm for digital microfluidic biochips," in *Proc. 20th ACM Great Lakes Symp. VLSI*, Providence, RI, USA, May 2010, pp. 441–446.
[31] F. Rubin, "An iterative technique for printed wire routing," in *Proc. 11th Design Autom. Workshop (DAC)*, Denver, CO, USA, 1974, pp. 308–313.
[32] M. H. Shamsi, K. Choi, A. H. C. Ng, and A. R. Wheeler, "A digital microfluidic electrochemical immunoassay," *Lab Chip*, vol. 14, no. 3, pp. 547–554, Feb. 2014.
[33] S. C. C. Shih, I. Barbulovic-Nad, X. Yang, R. Fobel, and A. R. Wheeler, "Digital microfluidics with impedance sensing for integrated cell culture and analysis," *Biosens. Bioelectron.*, vol. 42, pp. 314–320, Apr. 2013.
[34] F. Su and K. Chakrabarty, "High-level synthesis of digital microfluidic biochips," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 3, no. 4, 2008, Art. no. 1.
[35] Q. Wang, W. He, H. Yao, T. Ho, and Y. Cai, "Svm-based routability-driven chip-level design for voltage-aware pin-constrained EWOD chips," in *Proc. Int. Symp. Phys. Design (ISPD)*, Monterey, CA, USA, Mar./Apr. 2015, pp. 49–56.
[36] T. Yan and M. D. Wong, "BSG-route: A length-matching router for general topology," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, Nov. 2008, pp. 499–505.
[37] T. Yan and M. D. Wong, "Correctly modeling the diagonal capacity in escape routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 2, pp. 285–293, Feb. 2012.
[38] S.-H. Yeh, J.-W. Chang, T.-W. Huang, S.-T. Yu, and T.-Y. Ho, "Voltage-aware chip-level design for reliability-driven pin-constrained EWOD chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 9, pp. 1302–1315, Sep. 2014.
[39] S.-T. Yu, S.-H. Yeh, and T.-Y. Ho, "Reliability-driven chip-level design for high-frequency digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 529–539, Apr. 2015.
[40] X. Zeng *et al.*, "Recoverable electrowetting-on-dielectric device in chemiluminescence enzymatic detector," *Jpn. J. Applied Phys.*, vol. 53, no. 6, Jun. 2014, Art. no. 060304.
[41] Y. Zhao, T. Xu, and K. Chakrabarty, "Broadcast electrode-addressing and scheduling methods for pin-constrained digital microfluidic biochips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 7, pp. 986–999, Jul. 2011.

Jeffrey McDaniel (M'15) received the B.S. degree in computer science and pure mathematics and the M.S. degree in computer science from the University of California at Riverside, Riverside, CA, USA, in 2011 and 2014, respectively, where he is currently pursuing the Ph.D. degree in computer science.

He has published five conference papers and two journals. His current research interests include languages, synthesis, and hardware interfacing for continuous-flow microfluidic biochips.

Zachary Zimmerman received the B.S. degree in computer science from the University of California at Riverside, Riverside, CA, USA, in 2015, where he is currently pursuing the M.S. degree in computer science.

He has published one journal paper. His current research interests include digital microfluidics, machine learning, and information retrieval.

Daniel Grissom received the B.S. degree in computer engineering from the University of Cincinnati, Cincinnati, OH, USA, in 2008, and the M.S. and Ph.D. degrees in computer science from the University of California at Riverside, Riverside, CA, USA, in 2011 and 2014, respectively.

He is currently an Assistant Professor with Azusa Pacific University, Azusa, CA, USA. He has published ten conference papers, four journals, and filed one provisional patent. His current research interests include languages, synthesis, and hardware interfacing for digital microfluidic biochips.

Dr. Grissom was a recipient of the NSF Graduate Research Fellowship Award in 2010.

Philip Brisk (M'09–SM'16) received the B.S., M.S., and Ph.D. degrees from University of California at Los Angeles, Los Angeles, CA, USA, in 2002, 2003, and 2006, respectively, all in computer science.

From 2006 to 2009, he was a Post-Doctoral Scholar with the Processor Architecture Laboratory, School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. He is currently an Associate Professor with the Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA, USA. His current research interests include field-programmable gate arrays, compilers, and design automation and architecture for application-specific processors.

Dr. Brisk was a recipient of the Best Paper Award at CASES, 2007 and International Conference on Field Programmable Logic and Applications (FPL) 2009. He is or has been a member of the program committees of several international conferences and workshops, including Design Automation Conference, Asia and South Pacific Design Automation Conference, Design, Automation, and Test in Europe, VLSI-SoC, FPL, FPT, and others. He has been the General Co-Chair of the IEEE International Symposium on Industrial Embedded Systems 2009, IEEE Symposium on Application Specific Processors 2010, and International Workshop on Logic and Synthesis 2011, and participated in the organizing committee of many other conferences and symposia.