Flow-based Microfluidic Technology

Microvalves are fabricated using multi-layer soft lithography with two layers of flexible polymer, polydimethylsiloxane (PDMS), mounted on top of a rigid substrate, typically a glass slide. The two logical layers of these devices are the “flow layer,” which transports biological fluids of an assay, and the “control layer,” which provides actuation capabilities. A microvalve forms where a control channel crosses a flow channel.

Netlist Processing

Here, a programmable mVLSI LoC for mixing and storing liquids [5] that was designed manually is converted to an mVLSI netlist representation of the LoC. Initially each component is represented by a single point. Those points are then arranged into a straight line planar embedding.

This straight line planar embedding of those points represents the initial coordinates of each component. Those points are then expanded to become components of the correct size and shape. This expansion causes collisions between the components and an illegal mVLSI layout.

The layout is then further expanded using one of many reclamation algorithms to correct for the component collisions. Once no more collisions are present in the system, a final pin-to-pin routing between each component is found and the flow-layer is finalized.

Comparison Algorithms

Baseline Expansion (BaseEx) refers to the component expansion method proposed by McDaniel et al. [8]

Shift Expansion (ShiftEx) processes components one-by-one using Circular Propagation. Let ci be the component currently being expanded and cj be the closest component to ci that is occupied within ci’s expanded region. ShiftEx tries to find the smallest scaling multipliers dx in the x-direction and dy in the y-direction that can move cj out of the expanded region; however, the shift is applied to all not-yet-expanded components ck (including cj ) yielding shifted coordinates (xk +dx(xk −xj),yk +dy(yk −yj)). This process repeats until all components are expanded without overlap. ShiftEx tends to pack components tightly in the upper-left corner of the chip.

Scaled Expansion (ScaleEx) also processes components one-by-one using Circular Propagation. ScaleEx uses a brute-force approach to find a pair of scale factors, dx in the x-direction and dy in the y-direction to apply to the (x,y)-coordinates of all components; applying the scale factors ensures that overlap does not occur when all components are expanded. For each component c, the scaled coordinate is (d.cx, d.cy). ScaleEx maintains the relative positions of all components corresponding to the initial planar embedding.

Comparison Algorithms

Baseline Expansion (BaseEx) refers to the component expansion method proposed by McDaniel et al. [8]

Shift Expansion (ShiftEx) processes components one-by-one using Circular Propagation. Let ci be the component currently being expanded and cj be the closest component to ci that is occupied within ci’s expanded region. ShiftEx tries to find the smallest scaling multipliers dx in the x-direction and dy in the y-direction that can move cj out of the expanded region; however, the shift is applied to all not-yet-expanded components ck (including cj ) yielding shifted coordinates (xk +dx(xk −xj),yk +dy(yk −yj)). This process repeats until all components are expanded without overlap. ShiftEx tends to pack components tightly in the upper-left corner of the chip.

Scaled Expansion (ScaleEx) also processes components one-by-one using Circular Propagation. ScaleEx uses a brute-force approach to find a pair of scale factors, dx in the x-direction and dy in the y-direction to apply to the (x,y)-coordinates of all components; applying the scale factors ensures that overlap does not occur when all components are expanded. For each component c, the scaled coordinate is (d.cx, d.cy). ScaleEx maintains the relative positions of all components corresponding to the initial planar embedding.

Diagonal Component Expansion (DICE-U & DICE)

After initial coordinates are found from a straight line planar embedding, for every component with an intersection DICE calculates a Δx and Δy based on the maximum length and width required to guarantee every component will be shifted out for the intersecting space. This generally causes the components to be spread across the diagonal axis of the chip, leaving lots of open routing space. When constrained, the system is reduced to only the diagonal axis and the routes that cross the boundary are shortened across the boundary.