International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

A High Performance Parallel Router for DMFBs

Prasanta Kr. Roy[1], Suman Shaw[2]

[1, 2]Siliguri Institute of Technology, Siliguri – 734009, West Bengal, India

Abstract: ***In this present era, design of digital micro-fluidic biochips (DMFBs) is a challenging area of research interest. The source-target routing is a crucial problem in a DMFB. In this paper, we propose a high performance linear-time based routing algorithm to route multiple droplets at the same time (Latest Arrival Time) with special emphasis on collision avoidance satisfying the underlying constraints. Here we consider a number of droplets with a single source-target pair for each droplet (2-pin network). The major goal is to route all droplets concurrently with optimized latest arrival time and reduced cell utilization. The algorithm is based on Breadth-First-Search [14] with a little modification in order to reduce CPU utilization.***

Keywords: Microfluidics, Biochip, Routing, Algorithm and Complexity

1. Introduction

Recently DMFB technology has gained much attention due to the increasing advancements in micro-fabrication and micro-electromechanical systems (MEMS). It can perform necessary biological tasks (such as clinical diagnosis and DNA sequence analysis) on a single small integrated system (Lab-on-a-Chip [6]) and thus provides design flexibility, higher sensitivity, comparatively smaller size and lower cost.

The underlying technology also provides comparatively reduced sample size, reagent volume and power consumption by micromanipulation of discrete fluid particles (droplets).

The movement of droplets is controlled by the principle of electro-wetting-o-dielectric [10]. In a DMFB, an array of electrodes is placed in between two parallel glass plates where the top plate and the bottom plate is applied a ground voltage and high voltage, respectively. Each parallel plate electrode pair is considered as a unit cell. The movement of a droplet from one cell to its neighboring cell is controlled by applying proper voltage to the bottom plate. The application of voltage may vary according to the need of droplet movement from one electrode to the other, and the overall process is controlled by a processor with pre-determined clock frequency to determine the velocity of movement [12].

Recently, a coplanar digital microfluidic system [7] has been introduced which have no top plate. Because of their digital characteristic, any operation on droplets can be performed with a set of library operations (VLSI standard library). The control of a droplet can be accomplished by applying a sequence of preprogrammed electric signals (Actuation sequences) [5]. Therefore, a *DMFB* can be designed by applying a hierarchical cell-based design methodology. Thus, once strong CAD frameworks are ready, a large scale complex *DMFB* can be designed as done in VLSI. However, CAD research for *DMFB* design has started very recently. A top down design methodology for a *DMFB* is proposed in [12], which mainly consists of architecture level synthesis and geometry-level synthesis. The geometry-level synthesis consists of module placement and droplet routing. Module placement determines the location of each module in order to minimize the chip area or response time. During droplet routing, the transportation path of each droplet is determined to avoid any unexpected mixture satisfying all the design constraints. As in the module placement, a cell can be used to transport different droplets during different time intervals (time-multiplexing), which increases the complexity of routing. One of the major goals of droplet routing is routability as in VLSI [17], while satisfying timing constraint and maximizing fault tolerance [6]. [12] describes the synthesis procedure of DMFB which involves optimization of certain cost functions under some resource constraints .Area optimization as well as resource sharing is essential for easier subsequent phases during placement of cells. Typically, in droplet routing, the optimization of throughput, time and resource utilization is the most crucial aspect.

Section 2 describes the related works based on the literature survey. In section 3, we focus on the preliminary concepts of droplet routing. Section 4 discusses the proposed approach along with an example. Section 5 covers the complexity analysis of the proposed algorithm. In section 6, we summarize the experimental results based on some standard benchmarks and finally section 7 provides the concluding remarks.

2. Related Works

Nowadays, optimization of ever increasing complex design of DMFB has become a critical area of research [11]. Designers are continuously trying to improve DMFB integration in order to optimize the basic goals such as throughput, time and resource utilization.

Recently, droplet routing is a crucial issue to enhance the performance for biochip design automation. Several routing mechanisms have been proposed so far. [3] describes a graph coloring approach in which the control of droplet movement over the electrodes was defined by direct addressing of the micro-controller control unit. An acyclic graph was constructed based on the movement time of droplets and the coloring was done based on the parallel routing of droplets. [16] also uses a direct addressing mechanism along with a graph clique model for mapping of the droplet routing problem. The optimal partitioning of the clique model is used to optimize the droplet routing time. An integer linear programming (ILP) based approach was proposed in [9] which explored the use of direct addressing mode in biochip

routing problem. In [4], dynamic reconfigurability of the microfluidic array is exploited during run-time. The proposed method starts with an initial placement technique. During module placement phase, a series of 2-*D* placement configurations is obtained in different time spans. Then appropriate routing paths are determined to complete droplet routing. To find the ultimate solution the given problem was decomposed into several sub-problems based on their initial placement and then each sub-problem was solved sequentially. A high performance droplet routing mechanism was proposed in [2] using a grid based representation. Initially, the proposed algorithm checks for those droplets which can be routed without any obstacle or blockage due to other droplets (routing by bypassibility) and then they are arranged for concurrent routing without considering the blockages. Routing of the remaining droplets is considered in presence of blockage and a concession zone was introduced to ascertain feasibility of the routing. Finally, the optimization phase is done by applying a compaction based algorithm. The network flow based method was proposed in [8]. The proposed method was based on non-intersecting bounding box technique. Firstly the bounding box of each net was determined and then a non-intersecting set of bounding boxes were chosen to route first. The min-cost max-flow algorithm was used for routing of the remaining nets. The concept of stalling was used in the A* search algorithm [1]. The algorithm used a graph representation to differentiate the state of the source-target pairs at different times. Then the A* search algorithm was used to choose the optimal path between source-target pairs. A pin-constraint based biochip design was proposed in [13] to optimize the number of control pins by proper modeling of assays and their operations. This involves scheduling of the bio assays and paths for net routing. [15] also proposed a partition based mechanism for pin constraint based design.

3. The Routing Task

Recently, DMFB routing is a crucial area of research interest due to increasing complexity of biochips. The major goal is to find an optimal path between source(s)-target(s) pair to transport a droplet while satisfying the underlying constraints. The DMFB routing task is very similar to VLSI routing where a set of wires need to be connected under certain design rules. Though DMFB routing differs from VLSI routing in the following manners:

- In DMFB routing, multiple droplets can share the same cell during different time intervals [1, 4] (Time Division Multiplexing).
- Stalling of a droplet, if needed, at a particular cell is allowed in DMFB routing.
- DMFB routing needs 3-D spacing by underlying constraints.

Typically, a droplet routing problem in DMFBs can be configured in terms of a 2-D grid array (Figure 1). Each droplet is associated with a source-target pair. The goal is to route all the droplets concurrently, if feasible, from its source location to its target location. Since multiple droplets are routed in parallel, they may intersect or overlap with each-other. To avoid this undesirable behavior, fluidic constraint rules [2] must be introduced. The movement of droplets is done in time multiplexed manner to optimize their reachability at the target cells. Due to the overall routing task is done in parallel; there can be unwanted mixtures of droplets if the minimum spacing is not maintained between them. Though, in some cases merging of droplets is required (3-terminal nets).

Suppose we have two droplets $D_i(X_i(t),Y_i(t))$ and $D_j(X_j(t),Y_j(t))$ initially at time t. They must not be located adjacent or diagonally adjacent to each-other in order to avoid mixing. Therefore, at any instant of time t, either $|X_i(t) - X_j(t)| \geq 2$ or $|Y_i(t) - Y_j(t)| \geq 2$ must be satisfied. This constraint is known as Fluidic constraint [2].

- Static constraint: $|X_i(t) - X_j(t)| \geq 2$ or $|Y_i(t) - Y_j(t)| \geq 2$
- Dynamic constraint: $|X_i(t+1) - X_j(t)| \geq 2$ or $|Y_i(t+1) - Y_j(t)| \geq 2$ Or $|X_i(t) - X_j(t+1)| \geq 2$ or $|Y_i(t) - Y_j(t+1)| \geq 2$

4. Proposed Approach

All the methods discussed in Section 2 can perform concurrent routing only for those droplets whose paths are clear (i.e. no blockage can occur between source-target routing paths). The routing for remaining droplets can be done in a sequential manner. In this paper, we propose a parallel routing approach for all droplets (whether blocked or cleared). Considering a EWOD model, we summarize our proposed algorithm as follows;

- Concurrent routing of all droplets satisfying the fluidic constraints.
- Finding the most significant path in order to optimize the latest arrival time.
- Assignment of priority, by means of their distance function (Manhattan Distance), to each droplet so that in case of a collision, the higher priority droplet can be routed first and the lower priority droplet is in waiting (stalling of droplet) state until the path is clear.
- Backtracking also can be done when more than one same priority droplets cause a collision.

4.1. The Algorithm

- **Before Routing Starts:**

1. Set-up floor [x, y].
2. If blocks present then mention location of blocked cells.
3. For each cell C_{ij} in the Floor add reference of its adjacent cell (i.e. C.NORTH, C.SOUTH, C.EAST and C.WEST) in it. If any of the adjacent cells are blocked or absent then mark its reference as absent.
4. Place droplets on the floor at its respective source position sequentially. If any droplet is found to being placed at the same/adjacent position of any other droplet, register that droplet as failed. Don't place that Droplet.
5. Nearby droplets are clustered/grouped using k-mean algorithm [19]. Each Droplet is assigned priority based on the number of droplets in its cluster.
6. For each droplet find a path using Modified_BFS ( ) from its source to target location. If no path can be found then droplet cannot be routed. Register that droplet as failed and kill it.

7. Thread priority is assigned in decreasing order with respect to the path length of the droplet.
8. For each droplet, if it is alive then start routing using Path_Route( ).
9. Wait till all the droplets reaches its target then print "routing complete".

- ***Path_Route( ):***

Let us assume the pre-calculated path is stored in an array "step" of length not more than x*y.

1. *Initialize i ← 0 , count ← 1, count1 ← 1*
2. *Until droplet position not equal to target repeat*
   - *2.1. If Move_Ahead() returns successful then wait for the next clock signal and then go back to 2.*
   - *2.2. Else if returns unsuccessful then check around the position step[i+1]. Assign t ← priority of the lowest priority droplet responsible for the collision around step[i+1].*
   - *2.3. If this droplet priority is less than t then*
     - *2.3.1. Check if i > 0 and if this droplet can go 1 step backwards then move the droplet backwards 1 step using Move_Back( ). count1 ← count1+1.*
     - *2.3.2. Else try to make an alternate path. If alternate path cannot be found the move back and try to find alternate path again. Loop 2.3.2 until alternate path is found.*
   - *2.4. Else if droplet priority is equal to t then*
     - *2.4.1. Assign tt ← priority of the lowest path length droplet around step[i+1].*
     - *2.4.2. If this droplet path length is less than tt then*
       - *2.4.2.1. Check if i > 0 and if this droplet can go 1 step backwards then move the droplet backwards 1 step. count1 ← count1+1.*
       - *2.4.2.2. Else try to make an alternate path. If alternate path cannot be found the move back and try to find alternate path again. Loop 2.4.2.2. until alternate path is found.*
     - *2.4.3. Else if droplet path length is equal to tt then*
       - *2.4.3.1. If this droplet has the lowest droplet ID among all the droplets around step[i+1] then*
         - *2.4.3.1.1. Check if i > 0 and if this droplet can go 1 step backwards then move the droplet backwards 1 step. count1 ← count1+1.*
         - *2.4.3.1.2. Else try to make an alternate path. If alternate path cannot be found the move back and try to find alternate path again. Loop 2.4.3.1.2. until alternate path is found.*
       - *2.4.3.2. Else keep droplet at step[i] and count ← count+1.*
       - *2.4.3.3. End if*
     - *2.4.4. Else keep droplet at step[i] and count ← count+1.*
     - *2.4.5. End If*
   - *2.5. Else keep droplet at step[i] and count ← count+1.*
   - *2.6. End If*
   - *2.7. If count is divisible by 4 then*
     - *2.7.1. Try to make an alternate path. If alternate path cannot be found the move back and try to find alternate path again. Loop 2.6.1. until alternate path is found.*
     - *2.7.2. Increment count by one.*
   - *2.8. Else if count1 is divisible by 4 then*
     - *2.8.1. Try to make an Make_Alternate_Path( ). If alternate path cannot be found then move back and try to find alternate path again at next clock. Loop 2.7.1. until alternate path is found.*
     - *2.8.2. Increment count1 by 1*
   - *2.9. End If*
3. *End Loop*
4. *If droplet has reached target position then register droplet as reached and pick up droplet from floor.*
5. *End*

- ***Make_Alternate_Path( ) :***

1. Make copy of main floor excluding the droplets but keeping the blocks as virtualfloor.
2. Find all the droplets around step[i+1] and mark all the cells around those droplets including the position of the droplet as blocked.
3. For each cell C_{ij} in virtualfloor add reference of its adjacent cell (i.e. C.NORTH, C.SOUTH, C.EAST and C.WEST) in it. If any of the adjacent cells are absent or blocked then mark its reference as absent.
4. Try to make a path from the current position to the target using BFS and overwrite the pre-calculated path from step[i] to target with the new found path around the collision causing higher priority droplet.
5. If the alternate path is successfully discovered then discard the duplicate floor and return successful.
6. Else discard the duplicate floor and return unsuccessful.
7. End

- ***Move_Ahead( ) :***

1. Check if step[i+1] is available i.e. no droplet is present around step[i+1] excluding itself.
2. If available then place droplet at that position, increment i and return successful.
3. Else return unsuccessful.
4. End If
5. End

- ***Move_Back( ) :***

1. Check if step[i-1] is available i.e. no droplet is present around step[i-1]excluding itself.
2. If available then place droplet at position step[i-1], decrement i and return successful.
3. Else return unsuccessful.
4. End If
5. End

- ***Modified_BFS( ):***

1. Initialise Queue ← Ø
2. N ← x*y (total number of cells)
3. Initialise all cells as white. Set d← ∞ and π ← NIL
4. Set colour of source as grey. d← 0 & π← NIL
5. Enqueue ← source cell
6. Loop till Queue is not empty
   6.1. cell ← dequeue( )
   6.2. for all the cells v present adjacent to cell(i.e. north, east, west, south), if v.colour is white then
      6.2.1. set v.colour ← grey
      6.2.2. set v.d ← cell.d + 1
      6.2.3. v.π ← cell
      6.2.4. Enqueue ← v
   6.3. cell.colour ← black
7. end loop
8. Make the path using Print-path( ) algorithm and return successful. If "path not found" then return unsuccessful.
9. End

- ***Print-path( ):***

1. if target cell.π = NIL then return "path not found"
2. Initialize stack top = -1
3. Loop until cell.π = NIL
   3.1. Push into stack ←cell
   3.2. cell←cell.π
4. end loop
5. loop until stack is not empty
   5.1. m←m+1
   5.2. step[m]←stack.pop
6. end loop
7. assign the rest of the array step as NIL.
8. return successful

Hence, all the droplets are routed in parallel and the maximum timestamp (Latest Arrival Time) and overall cells utilization has been considerably reduced.

4.2. An Example

Considering a 8×8 grid with 3 source-target pairs (2-pin net) for 3 droplets (Figure 1), the proposed algorithm works as follows;
S1 source position (3,0) target position (2,7) clustered with itself due to k-mean algorithm hence drop priority = 1 and path length = 8
S2 source position (2,7) target position (2,0) clustered with itself due to k-mean algorithm hence drop priority = 1 and path length = 7
S3 source position (6,3) target position (0,5) clustered with itself due to k-mean algorithm hence drop priority = 1 and travel priority = 6
They choose the optimal path to their target and routing starts. The color associated with each source pair reflects the routing path of that droplet towards its target.

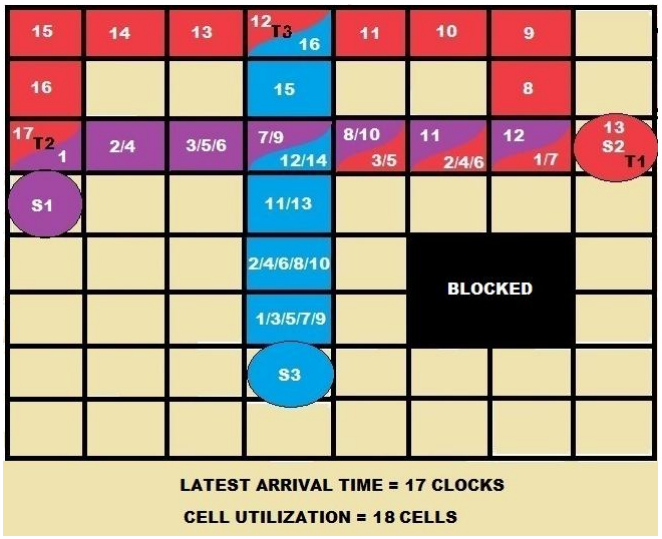

Figure1: Routing of a 2-pin net considering 3 source-target pairs

Step 1, 2: no collision occurs, all droplets moves forward one step at a time.
Step 3: S1, S2 progresses and S3 moves back due to lower travel priority .S3 registers its count1 to 1.
Step 4: Collision is avoided between S1 and S2 and both move back at that time instant S1 and S2 registers its count1 to 1. S3 moves forward.
Step 5: S1, S2 progresses and S3 moves back due to lower travel priority .S3 registers its count1 to 2.
Step 6: Due to higher travel priority S1 stays at its position while S2 moves back 1 step. S2 register its count1 to 2. S1 register count to 1.S3 moves to next step due to availability.
Step 7: Next cell is available for S1 hence moves forward. S2 and S3 next step is unavailable hence moves back. Register count1 to 3 and 3 respectively. S2 and S3 creates an alternate path. Since S1 at that time instant is not blocking S3's next path, S3 could not locate any nearby droplets from its step[i+1] and hence nothing to avoid and recalculates the same path and path is unchanged. Whereas for S2, S3 was responsible for making S2's next path unavailable. So S2 rewrites its path and makes an alternate path avoiding S1. Count1 is set to 0 for both S2 and S3.
Step 8: S1 moves forward to its next step as it is available. S2 moves to the next step as per the newly calculated and rewritten path. S3 moves to next step.
Step 9: next step for S1 is unavailable at that moment as S1 has higher travel priority S1 moves first hence S1 moves back 1 step and count1 for S1 is 2. S2 moves to next step. S3 next step is unavailable hence S3 moves back and registers count1 as 1.
Step 10, 11, 12: S1, S2, S3 moves forward.
Step 13: S1 reaches target. S2 moves forward. S3's next step is unavailable hence moves backwards. count1 is set to 2 for S3.
Step 14, 15, 16: S2, S3 moves forward. S3 reaches Target.
Step 17: S2 reaches target.

Now we can calculate the latest arrival time and overall cells utilization as follows;

Latest Arrival Time: Maximum {S1 to T1, S2 to T2, S3 to T3}
=Maximum {13, 17, 16} = 17
Cells Utilization: {Total cells used by all routing paths -

Total number of pins to represent all Source-target pairs}
= {23 - 5} = 18

5. Complexity Analysis

5.1. Complexity at Routing Time

1. As path length 'm' (Manhattan distance) for any droplet from source to target will be less than n. Hence for our best case when no collision occurs and droplet at each clock will move to its next step and reach its target. Complexity will be O(m).
2. Suppose if collision occurs the droplet will decide to stall or move back or find another path depending on the conditions. If it tries to find a new path, our modified BFS algorithm takes O(n) time to make path from source cell to all cells in the grid. Hence, at run time complexity will be O(m*n).

5.2. Modified_BFS

1. Instead of using a graph matrix we are using the source as the root and all the adjacent cells as its child, and so on.
2. For the grid each cell is enqueued only once due to greying. Hence complexity = O(n).
3. Since path cannot be greater than n hence printing the path will take m < n iterations. Hence complexity is O(m).
4. Therefore, complexity is O(m)+O(n) = O(m+n).

6. Experimental Results

We applied the proposed algorithm on Benchmark Suite 2 [2]. The layouts of the source-target pairs along with the obstacles (Blockages) for two test cases are shown in Figure 2 and Figure 3. Different colors associated with different sources (S_i) are used for reflecting the routing paths towards their respective targets (T_i). It is found that all the droplets are routed concurrently from their source to destination without any failure. Also the Latest Arrival Time and the overall cells utilization are marginally reduced. However, in some cases the overall electrode usage is increased due to backtracking and detouring but with a significant reduction in Latest Arrival Time.

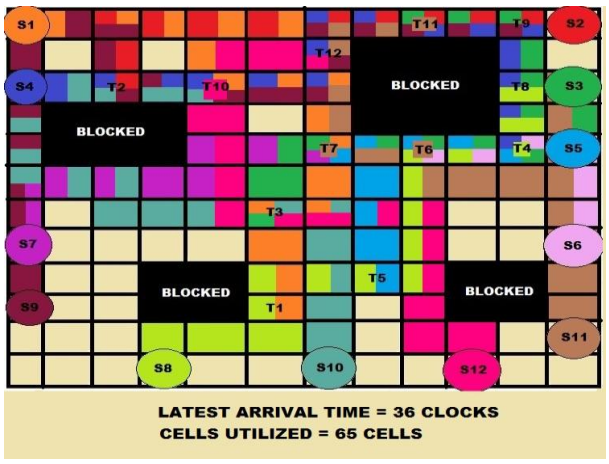

Figure 2: Layout for Test 1 along with the routing path of each droplet (Grid size = 12×12; Number of droplets = 12)

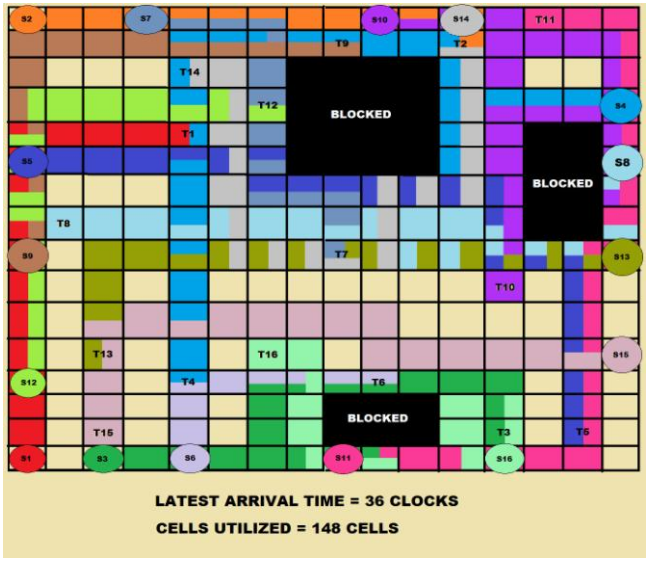

Figure 3: Layout for Test 6 along with the routing path of each droplet (Grid size = 16×16; Number of droplets = 16)

7. Conclusion

The proposed algorithm reveals the parallel routing task with a marginal improvement in Latest Arrival Time and overall electrodes utilization. The algorithm is applied on Benchmark Suite 2 (2-pin nets) and the empirical results are quite encouraging. The algorithm also can be applied on 3-pin or multi-pin nets (source-target pairs).

References

[1] Boahringer. K. F. —"Modeling and controlling parallel tasks in droplet based microfluidic systems". *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, 25(2):334 - 344, February 2006.
[2] Cho Minsik and Pan. David Z. —"A high-performance droplet routing algorithm for digital microfluidic biochips". *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, 27(10):406-419, October 2008.
[3] Akella S. Griffith E. J. and Goldberg. M. K. —"Performance characterization of a reconfigurable planar-array digital microfluidic system". *IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems*, 25(10):340-352, February 2006.
[4] Hwang William,Su Fei and Chakrabarty Krishnendu. —"Droplet routing in the synthesis of digital microfluidic biochips." In *Proceedings of Design Automation and Test in Europe*, 2006.
[5] Zeng X., Liu L.Wu, J ,Yue. R.— *Droplets Actuating Chip Based On EWOD"*. Springer-Verlog, 2007.
[6] Mukherjee. T. —"Design automation issues for biofluidic microchips". In *Proceedings of International Conference on Computer Aided Design*, page 463 - 470, November 2005.
[7] Pollack M.G, Paik P,Pamula V.K and Chakrabarty K. —"Coplanar digital microfluidics using standard printed circuit board Processes".*In Proceedings of in MicroTAS*, 2005.
[8] Yang C.L, Yuh P.H. and Chang. Y.W.—"Bioroute: A networkflow based routing algorithm for digital

microfluidic biochips". In *Proceedings of IEEE/ACM International Conference of Computer Aided Design*, pages 752-757, 2007.

[9] Yang Chia-Lin,Chang Yao-Wen,Yuh Ping-Hung, Sapatnekar Sachin. —"A progressive-ilp based routing algorithm for Crossreferencing biochips". In *Proceedings of Design Automation Conference*, pages 284 - 289, June 2008.

[10] Ren H., Paik P, Pamula V. K, Fair R. B., Srinivasan V. and Pollack.M.G. —"Electrowetting-based on-chip sample processing for integrated microfluidics". In *Proceedings of IEEE International Electron Devices Meeting (IEDM)*, pages 32.5.1-32.5.4, 2003.

[11] Tailor T.D.,Ivanov V.,Evans R. D., Griffin P. B., Srinivasan,V.Pamula V. K., Pollack M. G. Fair R. B. , Khlystov A.and Zhou J.. —"Chemical and biological applications of digital- microfluidic devices". *IEEE Design and Test for Computers*, 24:10-24, 2007.

[12] Su Fei and Chakrabarty Krishnendu. —"Architectural-Level synthesis of digital microfluidics-based biochips". In *Proceedings of IEEE International Conference on CAD*, pages 223-228,2004

[13] Su F, Xu T. , Hwang W. and Chakrabarty. K. —"Automated design of pin-constrained digital microfluidic biochips under droplet-interference constraints". *ACM Journal on Emerging Technologies in Computing Systems*, 3(14), 2007.

[14] Rivest Robert L., Cormen Thomas H, Leiserson. Charles E. —"*Introduction to Algorithms"*. MIT, 1990.

[15] Xu T. and Chakrabarty. K. —"Droplet-trace-based array partitioning and a pin assignment algorithm for the automated design of digital microfluidic biochips". In *Proceedings of IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, pages 112-117, 2006.

[16] Xu T. and Chakrabarty. K. —"A cross-referencing-based droplet manipulation method for high-throughput and pin-constrained digital microfluidic arrays". In *Proceedings of Design Automation and Test in Europe*, pages 552 - 557, April 2007.

[17] Xu T. and Chakrabarty. K. —"Integrated droplet routing in the synthesis of microfluidic biochips". In *Proceedings of Design Automation Conference*, pages 948 - 953, 2007.

[18] Paik Philip. Y., Pamula Vamsee K. and Chakrabarty Krishnendu,— "Adaptive Cooling of Integrated Circuits Using Digital Microfluidics". *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, vol. 16, no. 4, April 2008.

[19] Han Jiawei, Kamber Micheline, Pei Jian —"Data Mining-Concepts and Techniques". Elsevier Inc. 2012.

Author Profile

Prasanta Kr. **Roy** received his M. Tech. degree in VLSI Design from Bengal Engineering and Science University, Shibpur (Currently known as IIEST, Shibpur) in 2010. Nowadays he is working as an Assistant Professor at Siliguri Institute of Technology in the department of Computer Science and Engineering. He is a lifetime member of "Institution of Engineers, India".

Suman Shaw is a student of Bachelor of Technology in Computer Science and Engineering, at Siliguri Institute of Technology. His areas of interest are Algorithms and design.