

# A Real-Time Digital-Microfluidic Platform for Epigenetics

Mohamed Ibrahim<sup>†</sup>, Craig Boswell<sup>†</sup>, Krishnendu Chakrabarty<sup>†</sup>,  
Kristin Scott<sup>‡</sup>, Miroslav Pajic<sup>†</sup>

<sup>†</sup> Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA

<sup>‡</sup> Department of Molecular Genetics and Microbiology, Duke University, Durham, NC 27710, USA

## ABSTRACT

Advances in digital-microfluidic biochips have led to miniaturized platforms that can implement biomolecular assays. However, these designs are not adequate for running multiple sample pathways because they consider unrealistic static schedules; hence runtime adaptation based on assay outcomes is not supported and only a rigid path of bioassays can be run on the chip. We present a design framework that performs fluidic task assignment, scheduling, and dynamic decision-making for quantitative epigenetics. We first describe our benchtop experimental studies to understand the relevance of chromatin structure on the regulation of gene function and its relationship to biochip design specifications. The proposed method models biochip design in terms of real-time multiprocessor scheduling and utilizes a heuristic algorithm to solve this NP-hard problem. Simulation results show that the proposed algorithm is computationally efficient and it generates effective solutions for multiple sample pathways on a resource-limited biochip. We also present experimental results using an embedded microcontroller as a testbed.

## 1. INTRODUCTION

The increasing level of complexity in biomolecular research motivates the need for comprehensive gene-expression analysis [8]. Digital microfluidics has emerged as an on-chip solution for executing bioassays to support such biomolecular research [18]. Digital-microfluidic biochips (DMFBs) provide a scalable platform based on a two-dimensional array of electrodes on which picoliter droplets can be manipulated. A DMFB can be dynamically reconfigured under program control during the concurrent execution of a set of bioassays. Moreover, today's DMFBs embody cyber-physical integration through on-chip sensors [14] and droplet monitoring using a CCD camera [24]. Therefore, a reconfigurable DMFB platform is well-suited to replace traditional bench-top chemistries, but with significantly smaller sample/reagent volumes and much higher automation.

Recently, DMFBs have been developed to process the complete workflow for gene-expression analysis [22]. This workflow includes bioassays for cell lysis, mRNA isolation and purification, cDNA synthesis, and quantitative polymerase

chain reaction (PCR). However, these platforms were designed for sample-limited analyses, in which on-chip devices were allocated in advance. Therefore, they are not suitable for running down-stream analysis on the “expressed” or “suppressed” genes. Moreover, they lack the versatility needed for conducting complicated quantitative protocols (e.g., epigenetics protocols [27]), which require the running of multiple samples through independent pathways.

In realistic biomolecular protocols involving multiple sample pathways, there is inherent uncertainty about the order of fluidic steps (referred to here as *fluidic tasks*); thus fluidic task assignment and scheduling are significant challenges that have not been addressed thus far. Another drawback associated with current solutions is that they do not cope with heterogeneity in biological processes, which is observed even in samples with genetically similar cells [10]. In today's designs, multiple bioassays are mapped to the chip in an *a priori* manner; thus manual intervention is needed to respond to reaction outcomes. We list below several other limitations of current methods.

- Problems arise when a reaction is terminated too soon or allowed to run too long. For example, the completion time for the preparation of reverse-transcription master mixes depends on the time needed for mixing [19]. On the other hand, droplets that are subject to overly long-lasting reactions (especially in a medium of air) are susceptible to evaporation, which impacts the efficiency of enzymatic reactions and alters protocol outcomes [13]. In previous methods, the reaction time is specified using a microfluidic library, which considers the overly pessimistic worst-case timing of bioassay reactions. Moreover, these methods overlook droplet evaporation.
- Recently proposed error recovery methods make only limited use of spatial reconfiguration to avoid faulty sites [1, 15]. This approach is ineffective for real-time decision-making for concurrent sample pathways, since it does not provide effective resource sharing when the execution flows of the samples are not known *a priori*.
- The coordination among the components of the system controller (i.e, electrical-actuation control, firmware operation, and synthesis of fluidic operations) was overlooked in previous methods, hence the deployment of these methods in a real-time integrated system is not feasible.
- Moreover, while today's methods address droplet manipulation on a chip for basic fluidic operations (e.g., droplet routing [16, 26], cross-contamination avoidance [11], reconfigurability during fluidic operations, and fault tolerance [21]), they have yet to address the challenges associated with non-trivial biology-on-a-chip. A key limitation of prior solutions for biochip synthesis is that they consider a given sequencing graph and a pre-determined

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

CASES'16 October 01-07, 2016, Pittsburgh, PA, USA.

©2016 ACM. ISBN 978-1-4503-4482-1/16/10...\$15.00.

DOI: <http://dx.doi.org/10.1145/2968455.2968516>

sequence of fluidic operations. However, in actual biochemistry protocols, the actual sequence of fluidic operations is not known until intermediate reaction results are available. Therefore, biochip designs based on static scheduling are unrealistic and today’s techniques do not exploit the potential of DMFBs for implementing real-life microbiology applications. Not surprisingly, no practical demonstrations have been reported yet using these design-automation methods.

A new design paradigm was recently introduced in [12] to support non-trivial biology-on-a-chip applications (e.g., quantitative gene-expression analysis). In this work, a dynamic reconfiguration technique is used to spatially map resource specifications of multiple sample pathways to the biochip devices. This technique is embedded in an adaptive framework that facilitates real-time resource sharing among bioassays and reduces protocol completion time without sacrificing the chip’s lifetime. However, a drawback of this work is that it makes spatial reconfiguration decisions at the bioassay level (i.e., “locally”), and it does not capture interactions between multiple sample pathways at the protocol level. In other words, resource sharing among different sample pathways is achieved at a coarse-grained level, thus biochip devices are not efficiently exploited. Furthermore, this work does not consider the upper-bound temporal constraints imposed by the application domain; these constraints may arise due to physical phenomena such as droplet evaporation and deadlines imposed by the target chemistry, e.g., degradation of samples and reagents.

Another conceptual design of control flow was presented in [9]. However, this method is only used to advance the specifications of Biocoder—a programming language for biochemical assays—and to support conditional bioassay execution on DMFBs. It overlooks the underlying realism and spatio-temporal challenges of microbiology-on-a-chip applications when multiple samples are involved.

**Paper Contributions.** We overcome the above drawbacks by presenting a system design and a design-automation method that carries out task assignment and scheduling for cyber-physical DMFBs for quantitative analysis, e.g., the study of alterations in gene expression or cellular phenotype of epigenetics. The proposed method scales efficiently to multiple independent biological samples and supports on-the-fly adaptation under temporal and spatial constraints. The main contributions of this paper are as follows:

- We first present the outcomes of our benchtop experiment to motivate the study of epigenetic regulation in fission yeast. Motivated by the experimental results, we introduce the first layered system design for DMFBs. The experimental outcomes are also used to guide the synthesis of the underlying protocol.
- We map the synthesis problem to real-time multiprocessor scheduling and formulate it as an Integer Programming problem [2]. Since the scheduling is NP-hard, we develop a heuristic for dynamic fluidic task scheduling to respond to protocol-flow decisions. The proposed algorithm provides resource sharing and handles droplet evaporation.
- To promote component-based design in DMFBs [20], the interaction between the proposed algorithm and other system components (actuation and firmware) is demonstrated using an embedded micro-controller board.

**Paper Organization.** The rest of the paper is organized as follows. An introduction to epigenetic regulation and the

layered system design are presented in Section 2. In Section 3, we introduce the system model and associated constraints. Next, an algorithm for task scheduling is presented in Section 4. Finally, results of our experimental evaluation are presented in Section 5 and conclusions are drawn in Section 6.

## 2. MINIATURIZATION OF EPIGENETIC-REGULATION ANALYSIS

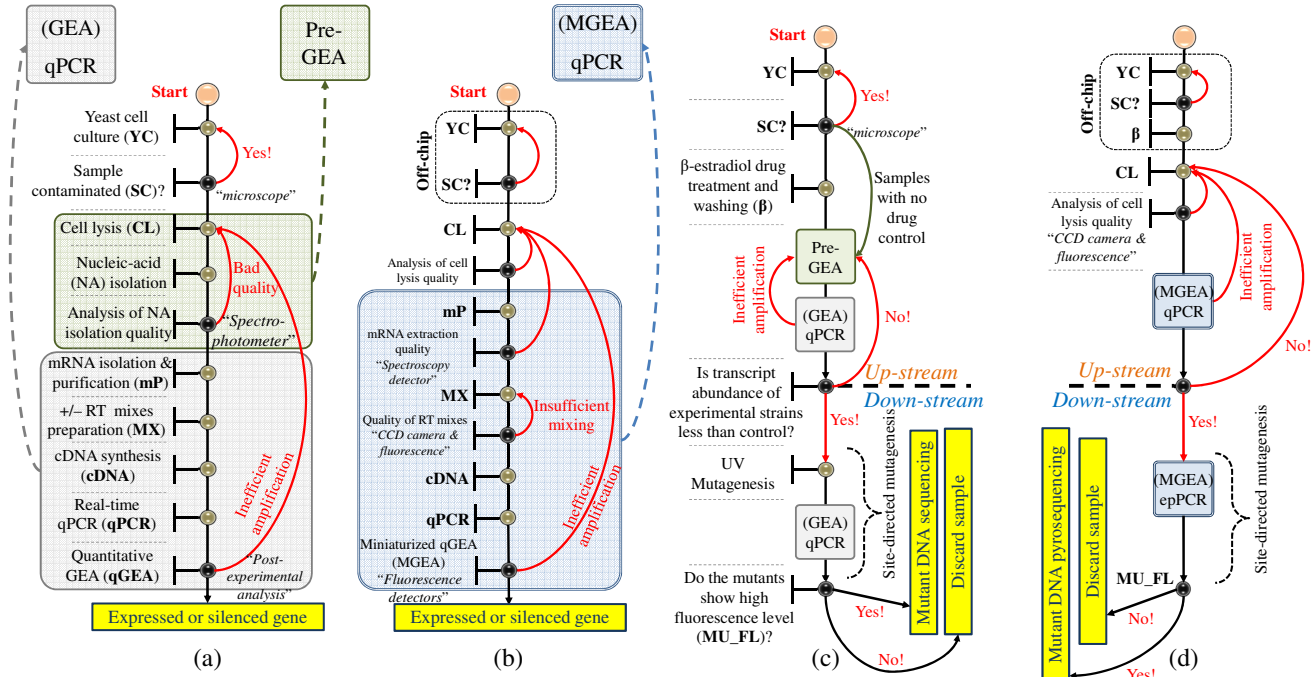
Quantification of the expression level for a gene is among the most important techniques in molecular biology [27]. We performed quantitative gene-expression analyses of a green fluorescent protein (GFP) reporter gene under epigenetic regulation in fission yeast. Using a benchtop setup, control (GFP constitutively expressed) and experimental strains were analyzed by quantitative PCR (qPCR). The steps of this experiment as carried out by us are listed below.

- (1) The samples were first placed in culture medium and grown overnight; then they were observed under a Phase Contrast Microscope to evaluate live cell concentration.
- (2) Cell lysis was performed for each sample to release intracellular contents, e.g., protein, nucleic acid (DNA and RNA), and cell debris. Glass beads were used for cell lysis.
- (3) Nucleic acids (NAs) were isolated and precipitated with 100% ethanol. The NAs were then resuspended in ultra-purified diethylpyrocarbonate (DEPC) water.
- (4) The purity of the isolated NA was assessed by spectrophotometry. Using a “QuantiTect Reverse Transcription (RT)” kit, a DNA-wipe-out (DNase) was added to each sample to eliminate all DNA, leaving the mRNA in the solution. Next, positive and negative RT mixes were prepared.
- (5) Finally, DNA amplification and gene-expression analysis through qPCR were carried out.

Fig. 1(a) shows a flowchart corresponding to the benchtop protocol for our gene-expression analysis experiment [27]; Fig. 1(b) illustrates the miniaturized implementation of this protocol.

One of the important uses of the above quantitative-analysis technique is in epigenetics, which identifies changes in the regulation of gene expression that are not dependent on gene sequence. Often, these changes occur in response to the way the gene is packaged into chromatin in the nucleus. For example, a gene can be unfolded (“expressed”), be completely condensed (“silenced”), or be somewhere in between. Each distinct state is characterized by chromatin modifications that affect gene behaviour [7]. An improved understanding of the *in vivo* cellular and molecular pathways that govern epigenetic changes is needed to define how this process alters gene function and contributes to human disease [27].

Based on our benchtop study of gene-expression analysis, we assessed the relevance of chromatin structure on regulation of the gene function. A second benchtop experiment was carried out to image yeast chromatin samples under a transmission-electron microscope (TEM) [7]. Fig. 2 relates the outcome of the experiment to gene-regulation behaviour based on chromatin structure. With multiple samples and with several causative factors affecting chromatin behaviour, implementing epigenetic-regulation analysis using a benchtop setting is tedious and error-prone; thus this preliminary benchtop study motivates the need to miniaturize epigenetic-regulation analysis. The benchtop study also



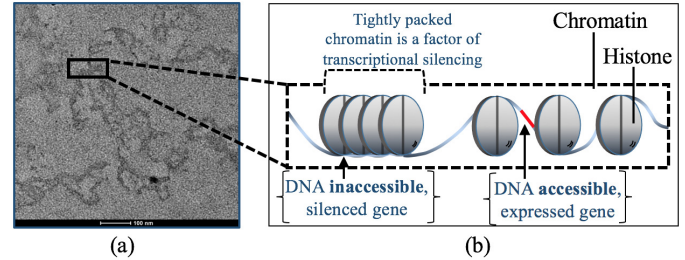
**Figure 1: (a)-(b) Protocol for gene-expression analysis using (a) benchtop setup; (b) DMFBs. (c)-(d) Protocol for epigenetic gene-regulation analysis using (c) benchtop setup; (d) DMFBs.**

provides important guidance on the design of the miniaturized protocol for a DMFB. Fig. 1(c) depicts a flowchart of the benchtop protocol. The protocol consists of two stages: (1) Up-stream stage in which the transcriptional profile (gene expression) of a GFP reporter gene is investigated. Control samples (GFP not under epigenetic/drug control) and experimental strains (under epigenetic control) were analyzed by qPCR. The goal was to explore how chromatin-folding alterations influence gene expression.

(2) Down-stream stage in which novel modifiers of epigenetic gene regulation are identified. Samples are mutagenized, for example by ultraviolet radiation, and cells whose transcriptional activity has been enhanced or suppressed are analyzed further. Following quantitative gene expression analysis, the causative mutation can be identified by whole genome sequencing [3]. The role of the genes in epigenetic processes can be verified by additional studies, including TEM analysis, ChIP, and other assays.

Reliable concurrent manipulation of independent samples requires the incorporation of sample-dependent decision-making into the protocol. We have developed a miniaturized protocol for epigenetic-regulation analysis; see Fig. 1(d). Using DMFBs, both up-stream and down-stream stages are carried out using the protocol for gene-expression analysis, followed by DNA pyrosequencing [5], to identify the sequences of the generated mutations. To provide a successful transformation of the complex epigenetic protocol into a biochip setting, a layered structure of a digital-microfluidic system is deployed.

Fig. 3 illustrates the components required for on-chip implementation of the protocol, and the interactions among them. The control software consists of a system model, embedded in a firmware layer, and a real-time resource assignment and scheduling layer. The protocol is initially synthesized to process the fluids across independent sample pathways. At the

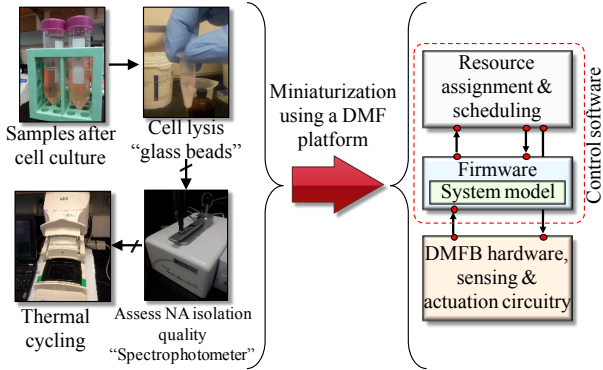


**Figure 2: (a) TEM image of chromatin; (b) correlation with chromatic control of epigenetic gene regulation.**

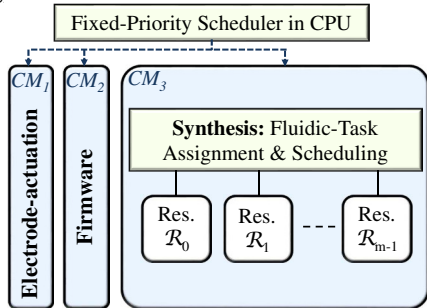
end of a bioassay, a sample is subjected to a detection operation that triggers the start of a decision-making process. The firmware receives the sensor readout, analyzes the data, and produces a decision to the scheduler. Real-time synthesis is then employed to provide the needed actuation sequences to adapt the system to the new situation.

In principle, the real-time synthesis is sufficient to capture the dynamics of fluid-handling operations within multiple sample pathways. However, an integrated system for epigenetics must take into account the impact of other active components: electrode actuation and firmware. In other words, the system scheduler is responsible for the timely coordination between the periodic loading of actuation sequences ( $CM_1$ ), firmware computation ( $CM_2$ ), and synthesis execution ( $CM_3$ ).

Therefore, we represent our digital microfluidic (DMF) platform as a hierarchy of components (Fig. 4);  $CM_1$  triggers a set of *periodic* tasks to stimulate the CPU to transfer actuation sequences from the controller memory to the biochip control pins at the start of every actuation period. The tasks triggered by  $CM_2$  are invoked *sporadically* at every decision-point within the protocol. Note that the durations of tasks generated by  $CM_1$  and  $CM_2$  are fixed and can be



**Figure 3: Proposed layered structure of a DMF platform to miniaturize a quantitative protocol. In this paper, we design the system model and the real-time scheduler.**



**Figure 4: Hierarchical scheduling of DMF system components.**

easily determined using offline simulation.

Our focus in this paper is on the control-software design and optimization, specifically, system modeling and real-time scheduling. A fully integrated hardware/software demo involving a fabricated biochip is a part of ongoing work, and beyond the scope of this paper. In Section 5.3, we present a preliminary demo using a micro-controller and simulation data. In the following two sections, we present details about the system model and real-time scheduling.

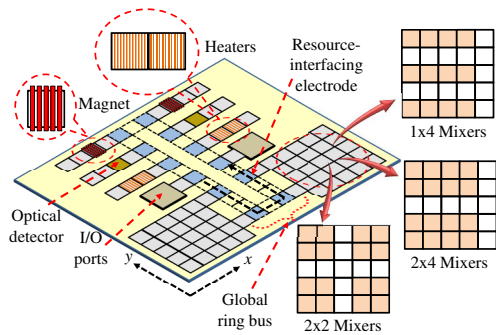
### 3. SYSTEM MODEL

We model the digital microfluidic (DMF) system for gene-regulation analysis in terms of real-time computing systems.

#### 3.1 Biochip Resources

The DMF platform includes three categories of resources: (1) *physical, non-reconfigurable* resources ( $\mathcal{PN}$ ) such as I/O ports, (2) *physical, reconfigurable* resources ( $\mathcal{PR}$ ) such as heaters, detectors, and regions to manipulate magnetic beads, and (3) *virtual, reconfigurable* resources ( $\mathcal{VR}$ ) such as mixers. The set of chip resources  $\mathcal{R}$  is defined as  $\mathcal{R} = \mathcal{PN} \cup \mathcal{PR} \cup \mathcal{VR}$ . Unlike  $\mathcal{VR}$ , the resources in  $\mathcal{PR}$  and  $\mathcal{PN}$  are spatially fixed, but a resource in  $\mathcal{PR}$  can be reconfigured to leverage the electrodes located within its region for sample processing in addition to its original function. For example, a magnet resource can be used either for magnetic-bead snapping or for sample processing, but not both at the same time.

Consequently, a biochip resource  $\mathcal{R}_r \in \mathcal{R}$  is characterized by  $\mathcal{R}_r = (\gamma_r, x_r, y_r)$  where  $\gamma_r$  is the resource type,  $x_r$  and  $y_r$  are the  $x$  and  $y$  coordinates of the resource interface, respectively. The resource interface is represented by an electrode that connects the resource to the global, unidirectional routing



**Figure 5: Resources of a DMF system used for implementing the epigenetic regulation protocol.**

bus. Fig. 5 shows an example of DMF resources. A dedicated dispensing reservoir is used to store a replenishment solution to counter droplet evaporation [13]. The proposed chip layout can implement the epigenetic regulation protocol, and it facilitates the real-time coordination of multiple droplets along the global routing bus.

#### 3.2 Fluidic Operations in Multiple Pathways

In prior work, bioassay operations and the interdependencies among them have been modeled as a directed sequencing graph  $\mathcal{G} = (V, E)$  [25]. A node  $v \in V$  signifies an operation and an edge  $e = (v_1, v_2) \in E$  represents precedence relation between operations  $v_1$  and  $v_2$ , respectively. This model is sufficient to handle synthesis for a single sample pathway, but inadequate for multiple sample pathways.

When multiple sample pathways are involved, modeling the synthesis problem based on real-time system theory enables us to assess system performance and robustness under various conditions and constraints, e.g., through compositionality and schedulability analyses [20]. In analogy with real-time multiprocessor scheduling, we introduce the following key terms to model the synthesis problem:

- *Fluidic-task set* ( $\mathcal{T}$ ): Similar to computational tasks in computer systems, a fluidic task  $\tau_b \in \mathcal{T}$  represents a bioassay in a target protocol. A fluidic subtask  $\tau_b^i$  represents a fluid-handling operation within a bioassay  $\tau_b$ .
- *Fluidic-processor set* ( $\mathcal{R}$ ): A biochip resource is also referred to as a fluidic processor. Hence, a DMFB is comprised of a set of heterogeneous processors corresponding to the chip resources  $\mathcal{R}$ .

In real-time systems, precedence-related subtasks can be assigned and scheduled on dedicated processors, where inter-processor communication induces delays in the release of subsequent subtasks [4]. Similarly, we build on the directed-acyclic graph (DAG) model [6] to capture the characteristics of a quantitative-analysis protocol. This model can then be used to compute task-assignment and scheduling solutions in our DMF system. The details of our task model are described below.

- The protocol consists of a set of bioassays  $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n\}$ . Each bioassay  $\mathcal{B}_b \in \mathcal{B}$  is a fluidic task  $\tau_b$  that is characterized as a 3-tuple  $(\phi_b, G_b, D_b)$ , where  $\phi_b$  is the task release time,  $G_b$  is a DAG, and  $D_b$  is a positive integer representing the relative deadline of the task. DAG  $G_b$ , whose number of nodes will be denoted by  $z_b$ , is specified as  $G_b = (V_b, E_b, P_b, C_b)$ , where  $V_b$  is a set of vertices representing subtasks  $\{\tau_b^1, \tau_b^2, \dots, \tau_b^{z_b}\}$ ,  $E_b \in [0, 1]^{z_b \times z_b}$  is an adjacency matrix that models the directed-edge set of  $G_b$ , and  $P_b \in [0, 1]^{z_b \times z_b}$  is a matrix derived from  $E_b$

that represents precedence relationships between the vertices  $V_b$ ; formally,  $P_b(i, j) = 1$  if and only if subtask  $\tau_b^i$  has to be completed before subtask  $\tau_b^j$  starts. Finally,  $C_b \in \mathbb{N}^{z_b \times z_b \times |\mathcal{R}| \times |\mathcal{R}|}$  represents the *lower-bound* routing-cost matrix between the vertices  $V_b$ , considering all resource combinations used for subtask executions. Specifically, if resources used to execute  $\tau_b^i$  and  $\tau_b^j$  are determined to be  $\mathcal{R}_r$  and  $\mathcal{R}_{r'}$ , respectively,  $C_b(i, j, r, r') = L$  indicates that the lower-bound value for the routing distance from the interfacing electrode of  $\mathcal{R}_r$  to that of  $\mathcal{R}_{r'}$  is equal to  $L$ . Note that  $C_b(i, j, r, r')$  is a function of coordinates  $(x_r, y_r)$  and  $(x_{r'}, y_{r'})$ , and it is calculated based on the unidirectional ring bus shown in Fig. 5. Furthermore, note that  $C_b(i, j, r, r')$  does not have to be equal to  $C_b(i, j, r', r)$ .

- A fluidic subtask  $\tau_b^i \in \{\tau_b^1, \tau_b^2, \dots, \tau_b^{z_b}\}$  is characterized by  $\tau_b^i = (T_b^i, \alpha_b^i, S_b^i, \mathcal{F}_b^i)$ , where: **(i)**  $T_b^i$  is a vector used to specify the times needed by the chip resources to execute subtask  $\tau_b^i$ ; **(ii)**  $\alpha_b^i \in \{0, 1\}^{|\mathcal{R}|}$  is a vector that specifies the subtask assignment to biochip resources  $\mathcal{R}$  (i.e.,  $\alpha_b^i(r) = 1$  if subtask  $\tau_b^i$  is allocated to the resource  $\mathcal{R}_r$ ); **(iii)**  $S_b^i$  represents the start time of the subtask; **(iv)**  $\mathcal{F}_b^i$  represents the end time of the subtask.

Thus, to satisfy the requirements of the bioassay  $\mathcal{B}_b$ , the following constraints must be satisfied:

$$\forall i, j, b : \tau_b^i, \tau_b^j \in \{\tau_b^1, \dots, \tau_b^{z_b}\}; r, r' : \mathcal{R}_r, \mathcal{R}_{r'} \in \mathcal{R},$$

$$\text{(C1)} \sum_r \alpha_b^i(r) = 1 \text{ \{allocation\};}$$

$$\text{(C2)} \mathcal{F}_b^i \geq S_b^i + T_b^i(r) \cdot \alpha_b^i(r) \text{ \{task duration\};}$$

**(C3)**  $S_b^i - \mathcal{F}_b^j \geq C_b(j, i, r, r') - K(2 - \alpha_b^j(r) - \alpha_b^i(r')) - K(1 - P_b(j, i)) + \mathcal{Q} \cdot \lambda_b^{(j, i)}$  {task precedence and inter-processor communication}, where the constant  $K$  is a large positive constant that linearizes the AND Boolean terms in the task-precedence satisfaction problem, described in Equation (1).

$$(S_b^i - \mathcal{F}_b^j \geq C_b(j, i, r, r')) \text{ if } (\alpha_b^i(r')) \wedge (\alpha_b^j(r)) \wedge (P_b(j, i)) \quad (1)$$

- In order to counter droplet evaporation, replenishment steps are incorporated before the start of every new bioassay [13]. Note that each bioassay  $\mathcal{B}_b$  (composed of a set of subtasks  $\tau_b^i$ ) is characterized by a deadline,  $D_b$ , on its completion time after which a sample must be run through a just-in-time replenishment process. Note that fulfilling a bioassay deadline is a soft real-time requirement. In addition, the need to fulfil all protocol deadlines imposes temporal constraints on our design, since resorting to extra replenishment steps during protocol execution significantly impacts completion time. In other words, if a fluidic task violates its deadline (creating non-zero tardiness), extra replenishment steps are applied to the associated sample pathway to counter droplet evaporation, leading to an increase in completion time.

To address this problem, each subtask  $\tau_b^i$  is also characterized by a Boolean variable  $\Omega_b^i \in \{0, 1\}$ , where  $\Omega_b^i = 1$  indicates that subtask  $\tau_b^i$  is planned to start execution after the deadline  $D_b$ . In addition,  $\pi_b$  represents the start time of a bioassay  $\mathcal{B}_b$  and  $\mathcal{Q}$  models the number of time steps<sup>2</sup> needed to complete sample replenishment. The following additional constraints must be satisfied:

<sup>1</sup> $T_b^i(r) = \infty$  indicates that subtask  $\tau_b^i$  cannot execute on  $\mathcal{R}_r$ .

<sup>2</sup>A time-step refers to the clock period, typically in the range of 0.1 to 1 second [19].

$$\forall i, j, b : \tau_b^i, \tau_b^j \in \{\tau_b^1, \dots, \tau_b^{z_b}\}; r, r' : \mathcal{R}_r, \mathcal{R}_{r'} \in \mathcal{R},$$

$$\text{(C4)} S_b^i \geq \pi_b \text{ \{bioassay start time\};}$$

The value of the variable  $\Omega_b^i$  for every subtask  $\tau_b^i$  can be specified using the difference between the subtask start time  $S_b^i$  and the absolute deadline  $(\pi_b + D_b)$  of the bioassay, as follows:

**(C5)**  $U \cdot \Omega_b^i \geq S_b^i - (\pi_b + D_b)$ ;  $\Omega_b^i \geq 0$ ;  $\Omega_b^i \leq 1$  {tasks beyond deadline}, where  $U$  is a large positive constant that can be used to upper-bound the allowable range of tardiness;

$$\text{(C6)} \mathcal{F}_b^i \leq T_{PF} \text{ \{protocol finish time\};}$$

In addition, the inequality in **(C3)** is modified to incorporate replenishment as follows. Note that just-in-time replenishment is applied before subtask  $\tau_b^i$  only when  $\Omega_b^i = 1$  and  $\Omega_b^j = 0$ , such that  $E_b(j, i) = 1$ . In other words, the bioassay deadline  $D_b$  is violated during or immediately after the execution of  $\tau_b^j$ .

**(C3')**  $S_b^i - \mathcal{F}_b^j \geq C_b(j, i, r, r') - K(2 - \alpha_b^j(r) - \alpha_b^i(r')) - K(1 - P_b(j, i)) + \mathcal{Q} \cdot \lambda_b^{(j, i)}$ ; where  $\lambda_b^{(j, i)}$  is a Boolean variable specified through the expression  $(\lambda_b^{(j, i)} = \neg \Omega_b^j \wedge \Omega_b^i \wedge E_b(j, i))$ , and it can be formulated as in **(C7)**;

**(C7)**  $\lambda_b^{(j, i)} \geq \Omega_b^i - \Omega_b^j + E_b(j, i) - 1$ ;  $\lambda_b^{(j, i)} \leq \Omega_b^i$ ;  $\lambda_b^{(j, i)} \leq 1 - \Omega_b^j$ ;  $\lambda_b^{(j, i)} \leq E_b(j, i)$ ;  $\lambda_b^{(j, i)} \geq 0$  {replenishment requirement}.

- Finally, to achieve mutual exclusion in system resources, each resource  $\mathcal{R}_r \in \mathcal{R}$  is also characterized by a Boolean variable  $\omega_{(i, b)}^r(t)$ , where  $\omega_{(i, b)}^r(t) = 1$  indicates that  $\mathcal{R}_r$  is being utilized by subtask  $\tau_b^i$  at time  $t$ . Therefore, the following constraint must be satisfied:

$$\text{(C8)} \sum_{(i, b)} \omega_{(i, b)}^r(t) = 1 \text{ \{mutual exclusion\};}$$

The value of  $\omega_{(i, b)}^r(t)$  can be specified as follows:

**(C9)**  $\omega_{(i, b)}^r(t) \geq 1 - K(3 - \alpha_b^i(r) - \delta_b^i(t) - \eta_b^i(t))$ ; where  $\delta_b^i(t)$  and  $\eta_b^i(t)$  are Boolean variables that are specified through the following formulation:

$$\text{(C10)} U \cdot \delta_b^i(t) \geq t - S_b^i; \delta_b^i(t) \geq 0; \delta_b^i(t) \leq 1;$$

$$\text{(C11)} U \cdot \eta_b^i(t) \geq \mathcal{F}_b^i - t; \eta_b^i(t) \geq 0; \eta_b^i(t) \leq 1.$$

## 4. TASK ASSIGNMENT AND SCHEDULING

In this section, we present our algorithm for fluidic task assignment and scheduling.

### 4.1 Problem Formulation

**Inputs:** (i) A set of bioassays  $\mathcal{B}$ , where each bioassay  $\mathcal{B}_b \in \mathcal{B}$  is characterized by a task  $\tau_b \in \mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_n\}$ , bioassay deadlines  $D = \{D_1, D_2, \dots, D_n\}$ , and adjacency matrices  $\{E_1, E_2, \dots, E_n\}$ ; (ii) precedence-relationship matrices  $\{P_1, P_2, \dots, P_n\}$ ; (iii) a set of subtasks  $\{\tau_b^1, \tau_b^2, \dots, \tau_b^{z_b}\}$  for each task  $\tau_b \in \mathcal{T}$ ; (iv) the routing-cost matrix  $C_b$  for each task  $\tau_b$ ; (v) the processing-time vector  $T_b^i$  for each subtask  $\tau_b^i$ ; (vi) biochip resources  $\mathcal{R}$ .

**Output:** (i) Assignment of fluidic subtasks to resources  $\alpha_b^i(r)$ ; (ii) start time  $S_b^i$  and finish time  $\mathcal{F}_b^i$  for each subtask  $\tau_b^i$ .

**Objective:** Reduce the number of tardy tasks to avoid repetition of droplet-replenishment procedures.



### Algorithm 1: TASK ASSIGNMENT AND SCHEDULING

```

1: procedure MAIN( $\mathcal{T}, \rho, \sigma, D, C, T, \mathcal{R}, E, P, \Upsilon$ )
2:    $Sol \leftarrow \emptyset; T_c \leftarrow \infty; vl \leftarrow \infty; util_{min} \leftarrow \infty;$ 
3:    $iter \leftarrow 0; \alpha_{prev} \leftarrow \emptyset;$ 
4:   while  $iter \leq \Upsilon$  do
5:      $iter \leftarrow 0;$ 
6:      $\alpha \leftarrow$  PROCESSOR_ASSIGN( $\mathcal{T}, \mathcal{R}, C, \alpha_{prev}$ );
7:      $T_{PF}, S_b^i, \mathcal{F}_b^i \leftarrow$  COORDINATE( $\mathcal{T}, C, T, D, \alpha, \mathcal{R}, E, P, \Upsilon$ );
8:      $v_l \leftarrow$  CAL_VIOLATIONS( $D, S_b^i, \mathcal{F}_b^i$ );
9:      $util \leftarrow \rho \cdot T_{PF} + \sigma \cdot v_l;$ 
10:    if  $util < util_{min}$  then
11:       $util_{min} \leftarrow util;$ 
12:       $Sol \leftarrow$  GET_RESULTS( $T_{PF}, S_b^i, \mathcal{F}_b^i$ );
13:    end if
14:     $\alpha_{prev} \leftarrow \alpha_{prev} \cup \alpha; iter \leftarrow iter + 1;$ 
15:  end while
16:  return  $Sol$ ;
17: end procedure
18: procedure COORDINATE( $\mathcal{T}, C, T, D, \alpha, \mathcal{R}, E, P, \Upsilon$ )
19:   $t \leftarrow 0;$ 
20:  while true do
21:     $RQ \leftarrow$  PUSH_READY_TASK_SORT( $\tau, P, C$ );
22:    if  $t == D_b$  AND TASK_UNFIN( $\tau_b, E$ ) then
23:      REPLENISH( $\tau_b, \mathcal{Q}$ );
24:    end if
25:     $S_b^i, \mathcal{F}_b^i \leftarrow$  SCHEDULE_IF_POSSIBLE( $RQ, T, t, \alpha, \mathcal{R}$ );
26:     $t \leftarrow t + 1;$ 
27:    if all  $\mathcal{T}$  scheduled then break;
28:  end while
29:  return  $\{T_{PF}, S_b^i, \mathcal{F}_b^i\}$ ;
30: end procedure

```

## 4.2 Heuristic Algorithm

The scheduling problem addressed here is mapped to the problem of scheduling DAG tasks on heterogeneous multiprocessors. Recently, schedulability of DAG tasks in uniform multiprocessors has been investigated and it has been shown that this problem is NP-hard [6, 23]. Not much is known about schedulability of DAG tasks in heterogeneous multiprocessors, but the problem is likely to be computationally intractable [4]. We handle this problem as described below.

The pseudo-code for the algorithm is described in Algorithm 1. The algorithm expects a user-specified input that characterizes the upper-bound ( $\Upsilon$ ) on the number of iterations (Line 4) to terminate the optimization process. In addition,  $\rho$  and  $\sigma$  are integers that are used as weighting factors for the utility function (Line 9), which is used to determine the goodness of a solution.

At every iteration, the algorithm performs task assignment and scheduling (Lines 6-7); subtasks are randomly assigned to the fluidic processors according to the required resources (Line 6), whereas a scheduling algorithm is used for real-time scheduling (Lines 18-30). The algorithm evaluates the utility of the produced solution based on the number of bioassay-deadline violations ( $v_l$ ) and the total completion time ( $T_{PF}$ ) for the protocol (Lines 8-9). The solution with the best utility (Lines 10-13) is finally selected. We introduce two policies to guide the behaviour of the scheduler (Line 21) below for updating the ready subtask queue  $RQ$ . Note that a scheduling policy must preserve precedence relationships among subtasks. In addition, a scheduling decision taken by a policy must take into consideration the droplet-routing cost (inter-processor communication cost).

1. *First-Come-First-Served (FCFS)*: A static policy in which the ready subtasks are prioritized based on their resource-request time. Note that the resource-access time for a subtask  $\tau_b^i$  depends on the finish time of the preceding subtasks.

This static approach is oblivious to bioassay deadlines.

2. *Least-Progression-First (LPF)*: A dynamic policy in which the task that belongs to the least-progressing bioassay is selected first. The *least-progressing bioassay* is a bioassay that is most likely to miss its deadline and its tasks urgently need to be advanced. This policy is similar to the Least-Laxity-First policy [17]. Quantifying the progression of bioassay execution is performed through a utility function  $f(D_b, t, s, n) = (D_b - t)(1 - \frac{s}{n})$ , where  $D_b$  is the bioassay deadline,  $t$  is the elapsed time since the bioassay has started,  $s$  is the number of time steps completed by this bioassay, and  $n$  is the summation of the number of time steps for all the bioassay operations. Note that the least-progressing bioassay has the lowest utility value.

We are given a set  $RQ$  of ready subtasks. We determine the processing time, the start time, and the finish time of every subtask in  $RQ$  and ensure that a subtask can get hold of the pre-assigned resource at time  $t$  (Line 25). Note that a task  $\tau_b$  that is not completed by the deadline is suspended until sample replenishment is carried out (Lines 22-24). Based on the scheduling choices, the synthesis algorithm (using the one-pass algorithm in [28]) is invoked to generate the actuation sequences (Line 12).

The scheduling scheme developed in this work is based on a timewheel that is controlled by an entity known as the coordinator and a priority queue, that is used to enforce the policy. The sequence of actions involved in our scheduling scheme is illustrated in Fig. 6.

## 4.3 Scheduling-Policy Analysis

We analyze the scheduling policies explained above using the example in Fig. 7. We consider two cases: (i) Case A: a case where a limited-resource chip is given; (ii) Case B: a case where an unlimited-resource chip is given. In Case A, we consider a biochip with a single mixer  $M$ , a single optical detector  $D$ , and a single waste reservoir  $W$  that is used to discard droplets. In both Case A and Case B, we consider a dedicated reservoir for each dispensing operation; i.e., unique resources  $A_b, O_b$ , and  $H_b$  for each task  $\tau_b$ . The task set consists of three tasks  $\tau_1, \tau_2$ , and  $\tau_3$ , which have release times  $\phi_1 = \phi_2 = \phi_3 = 0$ . The DAG representation for these tasks is shown in Fig. 7. The letters inside the nodes indicate the assigned resources. Also, the numbers above the nodes (shown in black) represent the worst-case processing time resulting from task assignment, and the numbers shown in red represent the droplet-routing cost. We consider that

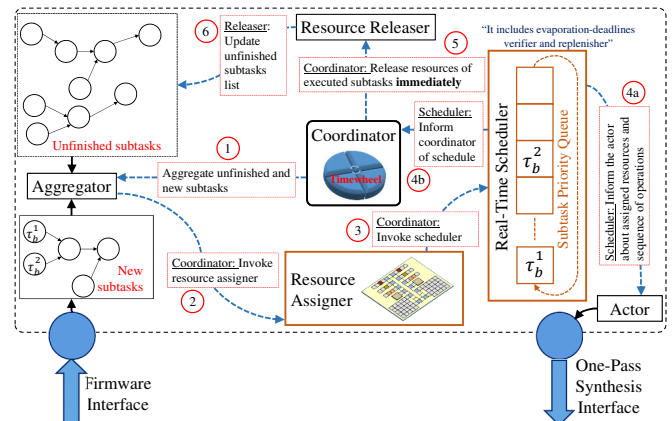
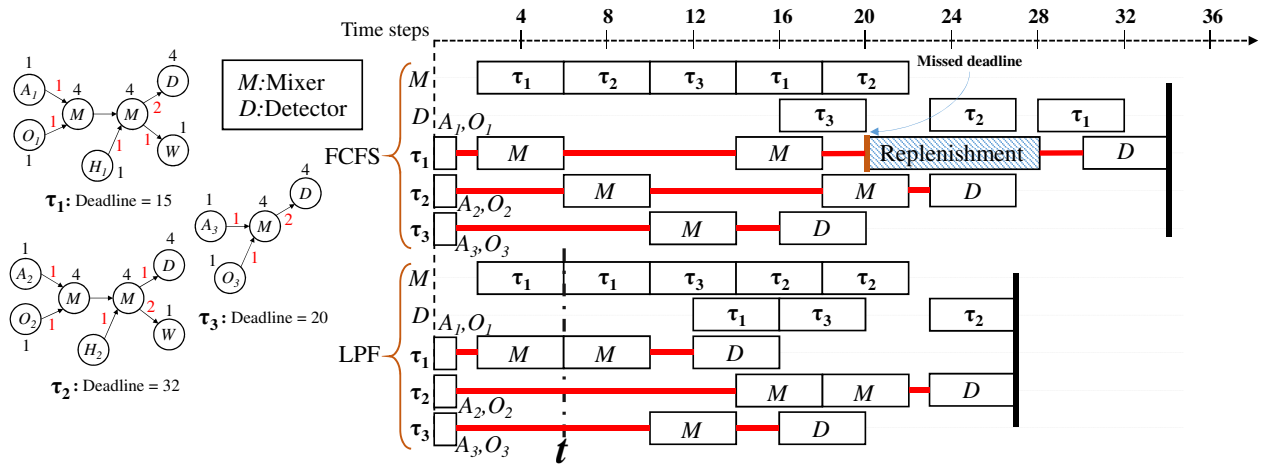


Figure 6: The sequence of actions in the proposed real-time scheduling scheme.



**Figure 7: Illustration of scheduling fluidic tasks using FCFS and LPF policies with a limited-biochip setting. Red lines indicate inter-processor communication costs (in time steps).**

the replenishment steps are carried out using dedicated resources in both Case A and Case B, and the time needed to complete droplet replenishment is 8 time steps.

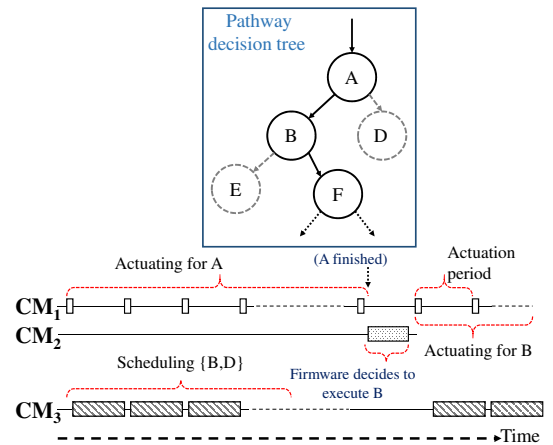
In Fig. 7, we demonstrate the timeline of the scheduling output for FCFS and LPF when Case A is considered. The results show that LPF outperforms FCFS for a limited-biochip setting. The reason is that LPF dynamically adapts the priorities of tasks based on their progression. For example, at time  $t$  (shown in Fig. 7), all tasks are competing for  $M$ . The utility value  $f$  of  $\tau_1$ ,  $\tau_2$ , and  $\tau_3$  at this time are 5.625, 22.75, and 11.2, respectively. As a result,  $\tau_1$ , with the least utility, is selected to process upon  $M$  starting at  $t$ . The completion times for FCFS and LPF are 34 and 27, respectively.

Nevertheless, we expect that both approaches converge to an equivalent lower-bound completion time when we increase the number of on-chip resources. Given the same set of tasks and considering Case B, the number of biochip resources is  $\sum_{b=1}^n z_b = 7 + 7 + 4 = 18$ ; i.e., there is a dedicated resource for each subtask  $\tau_b^i$ . In this case, it is easy to demonstrate that task scheduling depends only on the timing characteristics of a given task set, and that the computed completion time represents the lower-bound, based on a specific task assignment. We introduce the following definition that aids in our explanation [6]:

*Definition 1.* A chain in a DAG task  $\tau_b$  is the sequence of vertices  $V_b^1, V_b^2, \dots, V_b^l$  that forms a DAG  $G_b$  such that  $(V_b^j, V_b^{j+1})$  is an edge in  $G_b$ . The length of this chain is specified after task assignment and it is the sum of the processing times of all its vertices and the linking edges:  $T_b^1(r_1) + \sum_{j=2}^l T_b^j(r_j) + C_b(j-1, j, r_{i-1}, r_j)$ , where  $T_b^j(r_j)$  is the processing time incurred by resource  $\mathcal{R}_{r_j}$  and  $C_b(j-1, j, r_{i-1}, r_j)$  is the cost of the edge  $(V_b^{j-1}, V_b^j)$ . We denote by  $len(G_b)$  the length of the longest chain in  $G_b$ .

In Case B, the completion times obtained by FCFS and LPF are equal and they can be defined using the following equation:  $T_{PF}^* = \max_{\forall b: G_b} len(G_b)$ . According the task set given<sup>3</sup> in Fig. 7,  $T_{PF}^* = \max(17, 17, 12) = 17$ , which is the lower-bound completion time for FCFS and LPF when the number of resources are increased; this finding is corroborated using simulations in Section 5.1.

<sup>3</sup>Consider a unit routing cost between the two  $M$  operations in  $\tau_1$  and  $\tau_2$ .

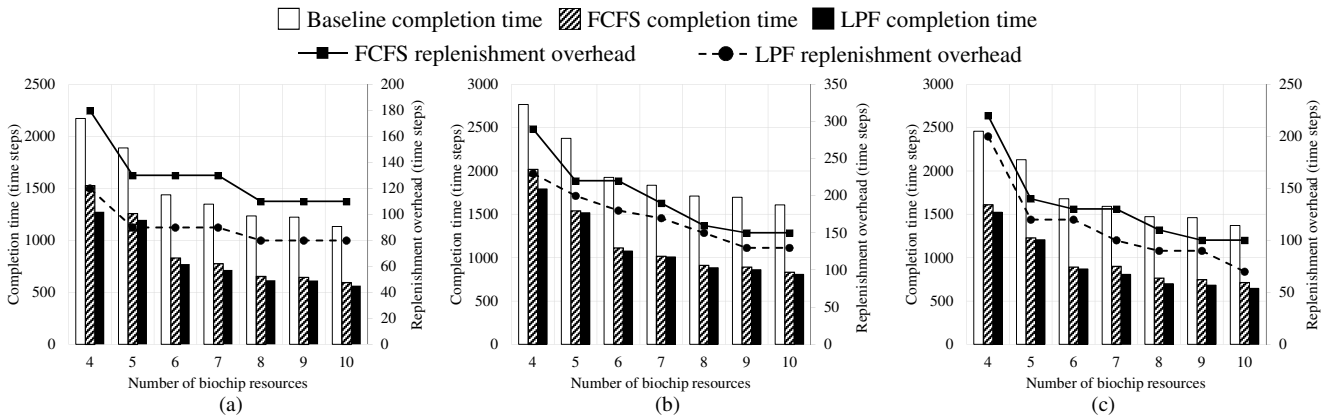


**Figure 8: Lookahead-based execution for a sample pathway. The progression of the sample is modeled as a decision tree.**

The worst-case time complexity of the above algorithm is  $O(\Upsilon.n.|\mathcal{R}|)$  when FCFS is used and  $O(\Upsilon.n^2.|\mathcal{R}|)$  when LPF is used. The algorithm is invoked whenever there is a decision that has been taken, necessitating a change in the task assignments and schedules for the sample pathways. To make use of the hierarchical system structure in Fig. 4, the execution of the algorithm needs to be interleaved with electrode actuation and firmware computation. An approach for employing this scheme is to run the task-assignment and scheduling algorithm in a *lookahead* manner. The progression of a sample pathway through a sequence of bioassays, based on the flow decisions, is modeled as a decision tree. While  $CM_1$  executes a bioassay at the  $q^{th}$  level,  $CM_3$  concurrently carries out task assignment and scheduling considering all choices at the  $(q+1)^{th}$  level. Subsequently, the appropriate assignments and schedules are selected based on the detection results obtained from  $CM_2$ . Note that  $CM_3$  must complete computation of the  $(q+1)^{th}$  level before  $CM_1$  finishes the execution at the  $q^{th}$  level. Fig. 8 depicts the timeline of a lookahead-based execution for a pathway.

## 5. SIMULATION RESULTS AND EXPERIMENTAL DEMONSTRATION

We implemented the proposed heuristic algorithm using C+++. The set of bioassays of the quantitative gene-regulation protocol (described in Section 2) were used as a benchmark. A



**Figure 9: Completion times and replenishment overhead for three task-scheduling schemes—baseline, FCFS, and LPF—running (a) short homogeneous pathways, (b) long homogeneous pathways, (c) heterogeneous pathways.**

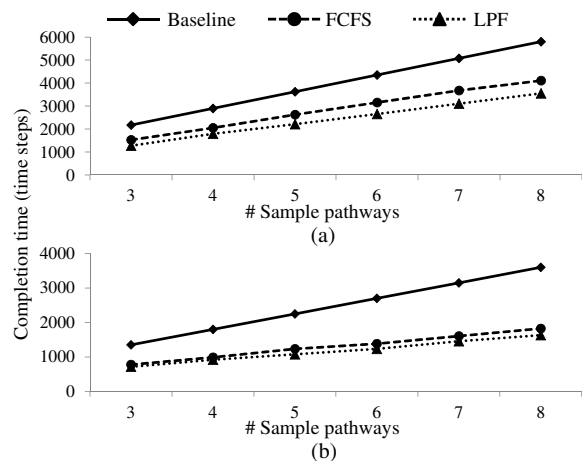
comprehensive analysis of the algorithm is performed based on three groups of evaluations: (1) evaluation of scheduling policies using simulation-based analysis; (2) comparison with previous resource-allocation techniques; (3) real-time experimental demonstration using an embedded micro-controller. Simulation-based assessment was carried out using an Intel Core i7, 3 GHz CPU with 16 GB RAM.

### 5.1 Evaluation of Scheduling Policies

Since most of the previous design-automation methods have not considered multiple sample pathways, we derive a baseline in which the protocol is executed for each sample separately. Therefore, we evaluate the performance for three task-scheduling schemes: (1) the baseline; (2) FCFS; (3) LPF. The metrics of comparison include: (1) the total completion time for the protocol (including replenishment time); (2) the time overhead incurred by replenishment procedures due to deadline violation; all measured in time steps. The results were obtained for various chip sizes, which are represented in terms of the number of biochip resources (e.g., heaters, mixers, and detectors); see Fig. 5. The deadlines for bioassays were obtained using offline simulation—each bioassay was simulated for various chip sizes and then the longest completion time was considered as a deadline.

We consider three samples being concurrently subjected to fluidic operations. The samples are S1 (GFP gene-targeted sample), S2 (YFP gene-targeted sample, and S3 (actin gene-targeted sample). We consider three different cases in terms of the sample pathways: (1) Short homogeneous pathways (Case I): a case where all three samples follow the same shortest pathway (12 bioassays in each pathway); (2) Long homogeneous pathways (Case II): a case where all three samples follow the same long pathway (16 bioassays in each pathway); (3) Heterogeneous pathways (Case III): a case where these samples are different (the three pathways are comprised of 12, 14, and 16 bioassays, respectively).

Fig. 9 compares the three scheduling schemes in terms of completion times for the three cases. Although the baseline scheme does not provide resource sharing (hence no deadline violation occurs), it leads to the highest completion times for all chip sizes. We also observe that LPF provides lower completion time compared to FCFS for tight resource constraints. This result corroborates our analysis in Section 4.3, in which we demonstrated that LPF is a deadline-driven approach and it achieves less tardiness, compared to FCFS.



**Figure 10: Scalability of the scheduling schemes with a varying number of homogeneous pathways; (a) using a biochip with four resources, (b) using a biochip with seven resources.**

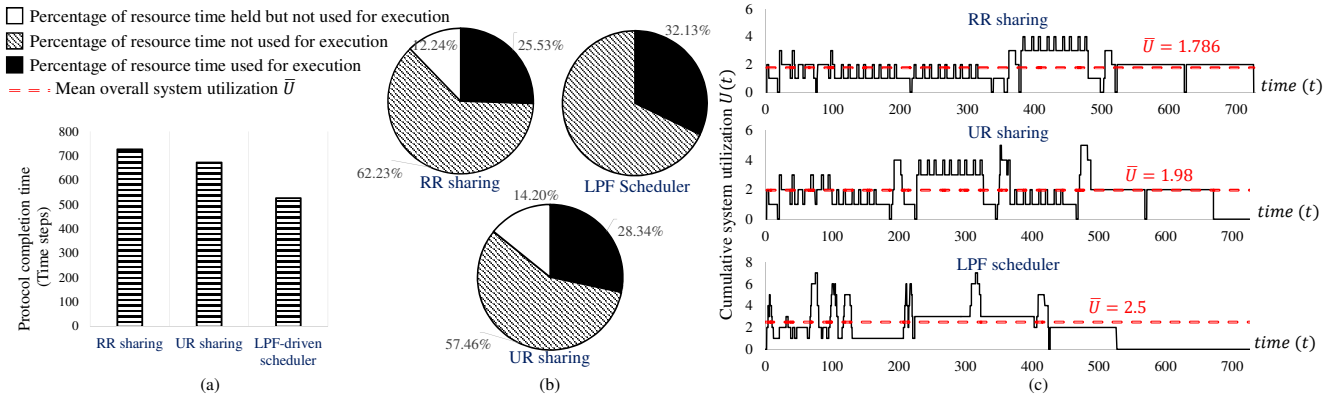
We also estimate the overhead incurred due to replenishment when FCFS and LPF are used. Without loss of generality, we assume that a replenishment procedure takes 10 time steps in the worst case. It is obvious that LPF incurs less replenishment overhead for all chip sizes, as shown in Fig. 9. As a result, compared to FCFS, the priority scheme of the LPF scheduler is more effective in countering droplet evaporation while the protocol completion time is not significantly increased. Also, it is noteworthy to mention that the difference in protocol completion time between LPF and FCFS gradually decreases when the number of biochip resources is increased; see Fig. 9. This finding also adheres to our analytical study performed in Section 4.3.

Next, we explore scalability of the three scheduling schemes. We analyze the completion time as the number of homogeneous pathways is varied; see Fig. 10. We observe that, as expected, the baseline scheme does not scale with the number of samples, since its completion time increases considerably when the number of samples is increased.

### 5.2 Comparison with Recent Work on Resource Allocation [12]

We next compare the completion time and resource utilization of our real-time scheduling approach (using LPF) with





**Figure 11: Comparison between three resource-management schemes: (i) restricted resource sharing (RR) [12]; (ii) unrestricted resource sharing [12]; (iii) LPF-based real-time scheduler (proposed). Comparison is based on: (a) protocol completion time, (b) percentage of time effectively used for execution, (c)  $U(t)$  and  $\bar{U}$ .**

resource-allocation techniques proposed in [12]. This work introduced two schemes to derive upper and lower bounds on protocol completion time considering resource sharing. These schemes are: (i) restricted resource sharing (RR): a scheme that fully restricts the reconfigurability of shared resources among bioassays; (ii) unrestricted resource sharing (UR): a scheme that does not consider any restrictions on resource sharing.

For comparison, we use a simulation environment similar to that in [12], in which a gene-expression analysis protocol is used. The minimum resource requirements as well as fluid-handling operations are re-used from [12]. We utilize three short homogeneous sample pathways (as in [12]), but the conclusions of this simulation holds also for other cases. The biochip architecture consists of seven resources: two mixers, two optical detectors, one magnet, one heater, and one CCD camera. The utilization profile of a resource  $\mathcal{R}_r$  over the course of the protocol execution is denoted by  $u_r(t)$ , where  $u_r(t) = 1$  indicates that resource  $\mathcal{R}_r$  is used for execution during the time step  $[t, t + 1]$ , whereas  $u_r(t) = 0$  indicates that the resource is idle during the same time step. The summation of the utilization profiles of all chip resources is referred to as *cumulative system utilization*  $U(t)$ ; thus  $U(t) = \sum_{r=1}^7 u_r(t)$ . This function gives an indication of how efficiently chip resources are used by a resource-allocation scheme over time. We use this function to derive the mean overall system utilization  $\bar{U}$  to compare system utilization of resource-allocation schemes using numerical values. The parameter  $\bar{U}$  is calculated as follows:

$$\bar{U} = \frac{\int_0^{T_{PF}} U(t).dt}{T_{PF}} \quad (2)$$

where  $T_{PF}$  is the protocol completion time. An upper-bound for  $\bar{U}$  is equal to the number of chip resources; i.e.,  $\bar{U} \in [0, 7]$ . Note that higher  $\bar{U}$  signifies better resource utilization. Although our real-time scheduler takes into consideration droplet-routing overhead, we ignore this cost in our comparison.

Fig. 11(a) compares the three resource-management schemes based on the completion time for gene-expression analysis. It is obvious that the proposed real-time scheduler, i.e., the LPF scheduler, achieves the shortest completion time, compared to the resource-allocation schemes in [12]. As expected, RR sharing leads to the worst-case completion time

due to the restrictions imposed on resource sharing.

Next, we analyze the resource utilization that results from the resource-management schemes, based on two metrics: (1) percentage of resource time used effectively by protocol execution; (2)  $\bar{U}$ . The results for the former metric is illustrated in Fig. 11(b). Using the LPF scheduler, 32.13% of the overall resource time is effectively used to execute protocol fluid-handling operations, whereas only 25.53% and 28.34% of the overall resource time are effectively used by RR sharing and UR sharing, respectively. Note that resource allocation in [12] is carried out at the bioassay level; hence a set of resources might be reserved to execute a bioassay, but not used until the associated fluid-handling operations are invoked. As a result, both RR sharing and UR sharing incur a time cost due to reserving, but not using, chip resources; see Fig. 11(b).

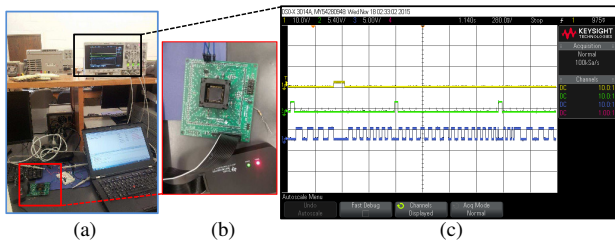
Finally, results involving the parameter  $\bar{U}$  for resource utilization are shown in Fig. 11(c). The proposed LPF scheduler achieves the best cumulative system utilization ( $\bar{U} = 2.5$ ), compared to RR sharing ( $\bar{U} = 1.786$ ) and UR sharing ( $\bar{U} = 1.98$ ), respectively. These results indicate that the proposed real-time scheduler is more cost-effective and it is applicable for tighter resource-budget cases.

### 5.3 Experimental Demonstration

We next demonstrate the application of hierarchical scheduling using a commercial off-the-shelf micro-controller (TI 16-bit MSP430 100-pin target board; see Fig. 12(b)). An oscilloscope is used to probe signals generated from the components  $CM_1$ ,  $CM_2$ , and  $CM_3$ —the experimental setup is shown in Fig. 12(a). The generated waveform is shown in Fig. 12(c), where the blue signal indicates the synthesized time steps based on the task scheduler ( $CM_3$ ), the green signal represents the actuation pulses ( $CM_1$ ), and the yellow pulse reflects firmware computation ( $CM_2$ ) based on a detection operation.

Our experimental target was to execute gene-expression analysis using two sample pathways. For testing and verification purposes, we generate a hypothetical, but feasible, stream of data to mimic the data transfer between the hardware and the control software. This data was obtained by simulating the gene-expression analysis protocol with two sample pathways.

A timer-interrupt module was utilized to trigger periodic



**Figure 12: Experimental demonstration of hierarchical scheduling.** (a) Experimental setup. (b) The 16-bit MSP430 100-pin target board. (c) Probed signals from  $CM_1$  (green),  $CM_2$  (yellow), and  $CM_3$  (blue).

actuation tasks while the task scheduler was running. This mechanism provides flexible tuning for the actuation clock depending on the developed application [19]. The firmware tasks were also realized through a set of interrupt service routines (ISRs) and invoked after each (simulated) detection operation. The selection of an ISR depends on the type of detection method assumed, e.g., CCD-camera monitoring of cell culture or fluorescence detection of amplified nucleic acid.

The outcome of the experiment matches the lookahead timing model described in Section 4.2. This experiment lays the foundations for developing and testing the key hardware/software co-design components for real-time DMFBs that can be used in realistic microbiology protocols.

## 6. CONCLUSION

We have introduced a design-automation method for a cyber-physical DMFB that performs quantitative epigenetic gene-regulation analysis. The design is based on real-time multiprocessor scheduling; it provides dynamic adaptation to protocol decisions under spatio-temporal constraints. We have also presented a hierarchical structure to illustrate the coordination between the synthesis of multiple pathways, electrode actuation, and firmware computation. A heuristic algorithm has been presented for the NP-hard task-scheduling problem. An experimental demonstration has been provided using a micro-controller board to show the interaction between the scheduling algorithm and other components.

Our ongoing work is focused on the integration of a fabricated biochip with the proposed design methodology. The work presented in this paper will be an important component of such an integrated hardware/software demonstration of a quantitative-analysis protocol.

## 7. REFERENCES

- [1] M. Alistar, P. Pop, and J. Madsen. Redundancy optimization for error recovery in digital microfluidic biochips. *Design Automation for Embedded Systems*, 19(1-2):129–159, 2015.
- [2] R. Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.
- [3] C. L. Araya et al. Whole-genome sequencing of a laboratory evolved yeast strain. *BMC genomics*, 11(1):88, 2010.
- [4] S. Baruah, M. Bertogna, and G. Buttazzo. *Multiprocessor Scheduling for Real-Time Systems*. Springer, 2015.
- [5] D. J. Boles et al. Droplet-based pyrosequencing using digital microfluidics. *Analytical chemistry*, 83(22):8439–8447, 2011.
- [6] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese. Feasibility analysis in the sporadic DAG task model. In *Proc. IEEE Euromicro Conference on Real-Time Systems*, pages 225–233, 2013.
- [7] C. R. Brown et al. Linking stochastic fluctuations in chromatin structure and gene expression. *PLoS Biol*, 11(8):e1001621, 2013.
- [8] A. Citri et al. Comprehensive qPCR profiling of gene expression in single neuronal cells. *Nature Protocols*, 7(1):118–127, 2012.
- [9] D. Grissom, C. Curtis, and P. Brisk. Interpreting assays with control flow on digital microfluidic biochips. *ACM J. Emerg. Tech. Com.*, 10(3):24, 2014.
- [10] S. Huang. Non-genetic heterogeneity of cells in development: more than just noise. *Development*, 136(23):3853–3862, 2009.
- [11] T.-W. Huang, C.-H. Lin, and T.-Y. Ho. A contamination aware droplet routing algorithm for the synthesis of digital microfluidic biochips. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 29(11):1682–1695, 2010.
- [12] M. Ibrahim, K. Chakrabarty, and K. Scott. Integrated and real-time quantitative analysis using cyberphysical digital-microfluidic biochips. In *Proc. ACM/IEEE Design Automation and Test in Europe*, pages 630–635, 2016.
- [13] M. J. Jebrail et al. A solvent replenishment solution for managing evaporation of biochemical reactions in air-matrix digital microfluidics devices. *Lab on a Chip*, 15(1):151–158, 2015.
- [14] N. M. Jokerst et al. Progress in chip-scale photonic sensing. *IEEE Trans. Biomed. Circuits Syst.*, 3(4):202–211, 2009.
- [15] Y. Luo, K. Chakrabarty, and T.-Y. Ho. Error recovery in cyberphysical digital microfluidic biochips. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 32(1):59–72, 2013.
- [16] E. Maftai, P. Pop, and J. Madsen. Module-based synthesis of digital microfluidic biochips with droplet-aware operation execution. *ACM J. Emerg. Tech. Com.*, 9(1):2, 2013.
- [17] A. Mok. *Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment*. PhD thesis, Massachusetts Institute of Technology, 1983.
- [18] H. Norian et al. An integrated CMOS quantitative polymerase-chain-reaction lab-on-chip for point-of-care diagnostics. *Lab on a Chip*, 14(20):4076–4084, 2014.
- [19] P. Paik, V. K. Pamula, and R. B. Fair. Rapid droplet mixers for digital microfluidic systems. *Lab on a Chip*, 3(4):253–259, 2003.
- [20] L. T. Phan, I. Lee, and O. Sokolsky. Compositional analysis of multi-mode systems. In *Euromicro Conference on Real-Time Systems*, pages 197–206, 2010.
- [21] P. Pop, M. Alistar, E. Stuart, and J. Madsen. *Design Methodology for Digital Microfluidic Biochips*. Springer, 2016.
- [22] A. Rival et al. An EWOD-based microfluidic chip for single-cell isolation, mRNA purification and subsequent multiplex qPCR. *Lab on a Chip*, 14(19):3739–3749, 2014.
- [23] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu, and C. D. Gill. Parallel real-time scheduling of DAGs. *IEEE Trans. Parallel Distrib. Syst.*, 25(12):3242–3252, 2014.
- [24] Y.-J. Shin and J.-B. Lee. Machine vision for digital microfluidics. *Review of Scientific Instruments*, 81(1):014302:1–7, 2010.
- [25] F. Su and K. Chakrabarty. Architectural-level synthesis of digital microfluidics-based biochips. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 223–228, 2004.
- [26] F. Su, W. Hwang, and K. Chakrabarty. Droplet routing in the synthesis of digital microfluidic biochips. In *Proc. ACM/IEEE Design Automation and Test in Europe*, volume 1, pages 1–6, 2006.
- [27] B. S. Wheeler et al. Uncoupling of genomic and epigenetic signals in the maintenance and inheritance of heterochromatin domains in fission yeast. *Genetics*, 190(2):549–557, 2012.
- [28] R. Wille et al. Scalable one-pass synthesis for digital microfluidic biochips. *IEEE Des. Test*, 32(6):41–50, 2015.