Performance Analysis and Optimization of Mixed Asynchronous Synchronous Systems

J. Teich, L. Thiele
Institute TIK
ETH Zurich
CH-8092 Zurich
Switzerland

S. Sriram
Texas Instruments Inc.
Dallas, TX
U.S.A.

M. Martin
Dept. of Mathematics and Computer Science
Martin-Luther University Halle
D-O6120 Halle an der Saale
Germany

April 14, 1997

Abstract

This paper deals with the system-level performance analysis and optimization of a class of digital systems we call mixed asynchronous-synchronous systems. In such a system, each computation module is either *synchronous* or *asynchronous*. The communication between all the modules is assumed to be data-driven. In order to adequately describe the timing of such architectures, we introduce a graph model called MASS which is based on several extensions of *timed marked graphs*. The first extension is that the node set V is partitioned into synchronous and asynchronous nodes. A synchronous node can only fire at ticks of its local module clock. Based on these extensions, we analyze the behavior of MASS, in particular period, periodicity and maximal throughput rate.

Finally, we introduce the optimization problem of assigning appropriate clock phases to synchronous nodes so to maximize the throughput rate of the resulting system. An exact solution as well as a polynomial time algorithm for nearly optimal phase assignment are presented.

1 Introduction

This paper is concerned with the performance analysis and optimization of a class of digital systems we call *mixed asynchronous-synchronous systems.* In such a system, communication and operation of all nodes is data-driven. However, each module may be either *synchronous*, i.e., synchronously clocked or *asynchronous*. The communication between all the modules is assumed to be data-driven.

1.1 Motivation

Today, architectures that consist of a combination of *dedicated* circuits, for example VLSI circuits, and *general purpose* components (memory blocks, A/D and D/A converters, DSPs, etc.) are becoming increasingly attractive and can be built at relatively low costs. Very often, they contain synchronous as well as asynchronous components.

Synchronously clocked circuit modules compute functions by separating stages of combinational logic with latches or registers that are clocked with a globally distributed clock. The advantages of synchronous design styles can be summarized as follows:

- *Maturity of existing CAD design tools,*
- *no circuit overhead due to handshaking or completion signal generation.*

Asynchronous components do not employ a global clock for enforcing system activities. In case they use *handshake signals* to sequence operations, they are called *selftimed* [1]. Those modules are supposed to have the following properties:

- *Average-case instead of worst-case performance*
  In a synchronous system, the maximal clock rate is determined by the slowest computation module.
- *Delay-insensitivity*
  The communication based on a selftimed handshake is reliable due to paradigm of *delay-insensitivity* [2, 3, 4]. Hence, physical design variations may have no influence on the correctness of the circuit. Also, a better technology migration is possible. In synchronous systems, typical problems are clock skew and critical path estimation.
- *Modeling the environment*
  Especially in the case of embedded electronic systems (see, e.g., [5]), the behavior of the computation environment can very often be modelled using asynchronous computation modules, for example, in case of sensors, actuators, and mechanical components.

For a more complete summary of the advantages of asynchronous systems, and for an overview of the state of the art in design methodologies for asynchronous circuit design, we refer to the collection of excellent papers edited in

[6, 7]. However, at a certain point in the design hierarchy, the communication costs for building a selftimed circuit (e.g., for dual-rail coding [1], differential logic [8], handshake logic) are not justified any more by the possible gains.

As a consequence of this discussion, it is desirable to model and design systems which consist of communicating synchronous and asynchronous components. On the one hand, this may be the appropriate model for the (embedded) electronic system and its possibly asynchronous environment. On the other hand, there is an optimum ground for the structure of a system where we do not have a monolithic synchronous system, but a mixture of asynchronous and synchronous design styles. Some of these architectures are also referred to as *globally asynchronous locally synchronous systems* (GALS) ([9],[10]). For a discussion on advantages and disadvantages, see [11]. Aspects of describing such systems can be found in [1], [9], [12], and [10]. Whereas [1] and [9] describe implementation aspects when building globally asynchronous locally synchronous systems, the approach presented in [10] proposes a language oriented approach to the design of globally asynchronous locally synchronous systems. There, the goal is to derive a circuit from the high level language SIGNAL. Finally, the work described in [13] concentrates on synthesizing a synchronous finite state machine from a mixed synchronous/asynchronous state graph with the same behavior.

Example 1 *Fig. 1 shows a typical example of a mixed asynchronous synchronous system. It is a disk drive application where a microcontroller (MC) controls a servo. The microcontroller transfers a position to the servo. As soon as the servo has reached its actual position, it issues a completion signal to the microcontroller. In the meantime, the microcontroller is busy with other tasks (e.g. has to serve other requests). We assume that the microcontroller polls every N cycles of its clock period T for the completion of the servo.*[1] *Obviously, the servo represents an asynchronous system: The servo may start and finish serving a request at time instances which are not multiples of the period $N \cdot T$ of the microcontroller.*

1.2 Goals

Unfortunately, none of the work described above proposes a model that enables the exact timing behavior of a system containing synchronous and asynchronous modules. Here, we are concerned with an exact analysis of performance, e.g., in determining the minimal period (or maximal throughput rate) achievable by such systems. Primarily, our goal is to generate a timing model that satisfies the following requirements:

- *Simplicity and implementation independence*
  In order to be amenable for CAD, the model should be simple and not

[1] We assume polling for either reason that a) there are not enough interrupt inputs or b) that no assumptions may be made on interrupt latencies.

focus on one particular implementation style or design methodology (e.g., circuit delay model).

- *Efficiency*
  For the same reasons as above, we would like to investigate efficient techniques for analysis and optimization, i.e., polynomial time algorithms, if possible.
- *Generality*
  It should be able to analyze and optimize GALS (all nodes are locally synchronous) or purely asynchronous architectures (all nodes are asynchronous nodes) as special cases.

The theory of mixed asynchronous-synchronous systems as introduced here combines aspects of asynchronous as well as synchronous performance analysis. In the realm of synchronous architecture design, Leiserson et al. ([14]) have developed a theory for analysis and optimization of *signal flow graphs* (SFG).

In the domain of asynchronous systems, graph models such as timed Petri net models have been investigated for analysis of performance [15, 16, 17]. In case of deterministic computation times, the corresponding Petri net is decision-free and can be represented by a marked graph [18], for a classification see e.g., [19]. In [20] and [16], it is shown that under certain conditions, systems modeled by marked graphs have an asymptotically periodic behavior and that the minimal period of such a system is given by the maximal cycle mean. A detailed analysis of this class of event systems is contained in [21].

Based on these results, we will introduce a graph model called MASS (mixed asynchronous-synchronous systems) which is an extended timed marked graph model. The first extension is that the node set of a MASS is a partition of asynchronous and synchronous nodes. Also, in contrast to marked graphs in which a computation module can commence its operation if all incoming arcs contain valid data, a synchronous node in an MASS can only start or finish its computation at a tick of its local module clock.

2 Models of computation

2.1 Marked graphs and their unfolding

The main purpose of this section is to introduce some notation and to recall the computational model associated with marked graphs, because this is the basis for the extensions described in this paper. For more details, see e.g., [19, 21] and the references therein.

Definition 1 (Marked Graph) *A timed marked graph $G = (V, A, d, h)$ denotes a directed graph with*

- *nodes* $V = \{v_1, v_2, \cdots, v_{|V|}\}$,
- *arcs* $A = \{a_1, a_2, \cdots, a_{|A|}\}$, *where any arc is an ordered pair of nodes* $a_p = (v_i, v_j)$,
- *a distance function* $d : A \to \mathbf{Z}_{>0}$ *and a weight function* $h : A \to \mathbf{R}_{\geq 0}$. [2] *We use* d_{ij} *and* h_{ij} *as short hand notations for the distance* $d(v_i, v_j)$ *and weight* $h(v_i, v_j)$ *associated with arc* (v_i, v_j), *respectively.*

A node of G represents a computation module. Each arc $a_p = (v_i, v_j)$ represents a collection of data linked to the computation module assigned to v_i and to the computation module assigned to v_j. Thus the computation module v_i places the results of its calculation onto arc a_p and the data on a_p are available as inputs to the computation module v_j. The distances and weights in Definition 1 are interpreted as follows:

- d_{ij} denotes the initial number of data (*tokens*) on arc $a_p = (v_i, v_j)$.
- h_{ij} denotes the holding time of a token on arc (v_i, v_j).

These quantities are related to the token game one can play on timed marked graphs: If a node v_i fires, then the oldest token of each arc ending in v_i is removed and one token enters each arc originating from v_i. The transfer is assumed to take zero time.

It remains to be addressed when such an event can take place. Informally, a node v_j is enabled if the oldest token of each incoming arc (v_i, v_j) has spent at least its holding time h_{ij} in the arc. Note that a node need not fire immediately after it is enabled. The dynamic behavior of a timed marked graph will become more clear by the following formal definitions (Definition 2 and Definition 3).

Many other essentially equivalent notions of marked graphs are possible, see e.g., [20, 19, 21] and the references therein.

Example 2 *As an example, consider the marked graph on the left hand side of Fig. 2. It is characterized by* $V = \{v_1, v_2\}$, $A = \{a_1, a_2, a_3\}$, $a_1 = (v_1, v_1)$, $a_2 = (v_1, v_2)$, $a_3 = (v_2, v_1)$, $d_{11} = 1$, $d_{12} = 0$, $d_{21} = 2$, $h_{11} = 2.5$, $h_{12} = 3.5$ *and* $h_{21} = 2$. *One possible evolution of the marked graph is shown for three time instances. At time* $t = 0$, *all tokens are placed onto the arcs. At time* $t = 2$, *both tokens in arc* (v_2, v_1) *contribute to the enabling of* v_1. *At time* $t = 2.5$ *the node is enabled because the holding time of the token in arc* (v_1, v_1) *is elapsed. Now, a token transfer may take place. If the firing happens at* $t = 2.5$, *the new situation shown in the middle graph results.*

To be more precise about the above model, e.g., with respect to the initial conditions and the properties of the collections, the concept of (event)–unfolding is introduced which represents the set of all evolutions of a timed marked graph

[2] Let **R**, **Z** stand for the set of real and integer numbers, respectively.

and includes information on the corresponding scheduling. Most of the results described in this paper will be derived using the notion of unfolding.

Definition 2 (Unfolding) *The unfolding of a timed marked graph* $G = (V, A, d, h)$ *is an infinite directed graph* $\Sigma(G) = (V_\Sigma, A_\Sigma, h_\Sigma)$ *where*

1. V_Σ *contains nodes* $v_i(k)$ *for all* $v_i \in V$ *and for all integers* $k > -\max\{d_{ij} : v_j \in V$ *such that* $(v_i, v_j) \in A\}$,
2. $A_\Sigma = \{(v_i(k - d_{ij}), v_j(k)) : (v_i, v_j) \in A \wedge k \in \mathbf{Z}_{>0}\}$,
3. *to each arc in* A_Σ *the weight of the corresponding arc in* G, *i.e.,* $h_\Sigma : A_\Sigma \to \mathbf{R}$ *with* $h_\Sigma(v_i(k - d_{ij}), v_j(k)) = h_{ij}$ *for all* $(v_i(k - d_{ij}), v_j(k)) \in A_\Sigma$ *is assigned.*

A node $v_i(k)$ of the unfolding represents the kth firing of node v_i in the marked graph. An arc from $v_i(k)$ to $v_j(l)$ expresses the fact that the lth firing of node v_j can take place only after the kth firing of node v_i. The nodes $v_i(k)$ for $k \leq 0$ represent the initial conditions of the marked graph, i.e., the placement of d_{ij} tokens into the collection corresponding to arc (v_i, v_j).

The scheduling of a timed marked graph that has been introduced informally, can now be described more precisely. An admissible schedule is defined as follows:

Definition 3 (Admissible Schedule) *An admissible schedule function* $\sigma : V_\Sigma \to \mathbf{R}_{\geq 0}$ *satisfies:*

1. $\sigma_i(k) = 0$ *for all* $v_i(k) \in V_\Sigma \ \wedge \ k \leq 0$,
2. $\sigma_j(k) \geq \sigma_i(k - d_{ij}) + h_{ij}$ *for all* $(v_i(k - d_{ij}), v_j(k)) \in A_\Sigma$

where $\sigma(v_i(k))$ *is written* $\sigma_i(k)$.

Here, $\sigma_j(k)$ denotes the time when node v_j fires for the kth time. The first condition in Definition 3 serves to consider the initial timing conditions of the token game. All initial tokens are placed into the collections at time 0.

An example explaining this model of computation is given next.

Example 3 *Consider the marked graph shown on the left hand side of Fig. 2 again. The corresponding unfolding is shown in Fig. 3. Obviously, the graph contains for each node* v_i *of the marked graph an infinite number of nodes* $v_i(k)$ *corresponding to its* k *firings. The weights assigned to the arcs correspond to the holding times* h_{ij}. *If the nodes fire as soon as they are enabled, the following admissible schedule (called free schedule) is obtained:*

$$\begin{aligned} \sigma_1 &= \begin{pmatrix} 0 & 2.5 & 5 & 8 & 10.5 & \dots \end{pmatrix} \\ \sigma_2 &= \begin{pmatrix} 0 & 0 & 6 & 8.5 & 11.5 & 14 & \dots \end{pmatrix} \end{aligned}$$

Therefore, nodes v_1 *and* v_2 *fire at times* $t = 2.5, 5, 8, 10.5, \dots$ *and* $t = 6, 8.5, 11.5, 14, \dots$, *respectively.*

2.2 Maximal–rate schedules of marked graphs

For obvious reasons, one is interested in a schedule of a given marked graph that leads to the maximal throughput rate. This throughput rate expresses the average time interval between two successive firings of any node of maximal–rate schedules. This observation leads to the following definition:

Definition 4 (Maximal–Rate Schedule) *The rate R of an admissible schedule σ is defined as $R(\sigma) = \frac{1}{P(\sigma)}$ with the average period*

$$P(\sigma) = \max \left\{ \lim_{k \to \infty} \frac{\sigma_i(k)}{k} : v_i \in V \right\}$$

The maximal rate R_{max} is defined by $R_{max} = \frac{1}{P_{min}}$ with the minimal average period

$$P_{min} = \min \{ P(\sigma) : \sigma \text{ is an admissible schedule } \}$$

In order to compare the timing of different realizations, we are particularly interested in periodic admissible schedules, because one is very often faced with periodic signals.

Definition 5 (L-Periodic Schedule) *An admissible schedule of a marked graph $G = (V, A, d, h)$ is* L-periodic *with $L \in \mathbf{Z}_{>0}$ and period $P \geq 0$, if for all $v_i \in V$*

$$\sigma_i(k + L) = \sigma_i(k) + LP, \ \forall k \in \mathbf{Z}_{>0} \tag{1}$$

The following theorem can be similarly found in [20] for computation graphs.

Theorem 1 (Maximal Cycle Mean) *A marked graph $G = (V, A, d, h)$ has an admissible periodic schedule with period P iff $\forall v_i \in V \ \exists \tau_i \in \mathbf{R}$, such that*

$$\tau_j - \tau_i \geq h_{ij} - P d_{ij} \ \ \forall (v_i, v_j) \in A. \tag{2}$$

Using the definition of the maximal cycle mean P_{cm} of G with

$$P_{cm} = \max \left\{ \frac{\sum_{(v_i,v_j) \in W} h_{ij}}{\sum_{(v_i,v_j) \in W} d_{ij}} : W \in C(G) \right\} \tag{3}$$

where W contains all arcs in a directed cycle and $C(G)$ contains all simple directed cycles of G the following statements hold:

- *There are admissible 1-periodic schedules for all $P \geq P_{cm}$.*
- *A 1-periodic admissible schedule with $P = P_{cm}$ has minimal period, i.e., $P_{cm} = P_{min}$.*

Proof: The detailed proof is given in [22]. The first equation (2) can be proven by combining the admissible schedule equation $\sigma_j(k) \geq \sigma_i(k - d_{ij}) + h_{ij}$ with (1) and choosing e.g., $\tau_i = \sigma_i(1)$ for all $v_i \in V$. Then the one-periodic schedule

$$\sigma_i(k) = \tau_i + (k-1)P, \ \forall k \in \mathbf{Z}_{>0} \tag{4}$$

is admissible. The proof of the maximal cycle mean (3) uses the dual representation of (2). ∎

P_{cm} is the maximum mean of accumulated path weights h_{ij} and number of tokens d_{ij} for all directed cycles of G. Using the above theorem, the maximal–rate schedule can also be determined by solving the linear program

$$P_{cm} = \min\{P : \tau_j - \tau_i \geq h_{ij} - P d_{ij} \ \forall (v_i, v_j) \in A\} \tag{5}$$

Note that the maximum cycle mean of a marked graph $G = (V, A, d, h)$ can be determined efficiently, see [23],[24]. Feasible potentials τ_i for all $v_i \in V$ can be determined using a shortest path algorithm in time $O(|V||A|)$ [25].

Example 4 *We compute the maximal throughput rate for the marked graph used in Examples 2 and 3. The maximal cycle mean*

$$P_{cm} = \max\left\{\frac{h_{11}}{d_{11}}, \frac{h_{12}+h_{21}}{d_{12}+d_{21}}\right\} = \max\{2.5, 2.75\} = 2.75$$

could have also been determined by solving the linear program (5) which yields $\tau_1 = 0$ *and* $\tau_2 = 3.5$. *Using (1) and (4), an admissible fastest periodic schedule is obtained as*

$$\sigma_1(k) = 2.75(k-1), \ \sigma_2(k) = 3.5 + 2.75(k-1) \quad \forall k \in \mathbf{Z}_{>0}$$

In the following, we will analyze the performance of mixed asynchronous–synchronous systems. But first, we will introduce the required extensions to the timed marked graph model in order to adequately describe such systems.

2.3 Modeling of mixed asynchronous–synchronous systems

We assume that in a mixed asynchronous–synchronous system, a synchronous node can only send or receive data at its local clock ticks. The formal model of MASS to be defined next is an extended marked graph model.

Definition 6 (MASS) *A mixed asynchronous-synchronous system (*MASS*) denotes an extended marked graph* $G = (V, A, d, h, r, p)$. *The set of nodes* V *is partitioned into disjoint subsets* V_A *and* V_S, *corresponding to asynchronous and synchronous nodes, respectively. In addition, the function* $r : V \to \mathbf{N}$ *assigns a* *clock period, and the function $p : V \rightarrow \mathbf{R}$ assigns a clock phase $0 \leq p_i < r_i$ to each node $v_i \in V$.*

In the unfolding $\Sigma(G)$, to each synchronous node $v_i(k) \in V_\Sigma$ there is assigned the clock period r_i and the clock phase p_i.

The following example clarifies the chosen representation of mixed asynchronous–synchronous systems.

Example 5 *Again, the system described in Example 2 considered. Now, we suppose that node v_2 is a synchronous node with the local clock phase $p_2 = 0.1$ and clock period $r_2 = 1$. Hence, the clock period is normalized to 1. Fig. 4 shows the corresponding extended graph.*

The major difference to the asynchronous case is the token game played on the MASS. The time instances $\sigma_j(k)$ when a synchronous node can complete an operation is now constrained as follows:

$$(\sigma_j(k) - p_j) \equiv 0 \bmod r_j \tag{6}$$

This restriction is motivated by the fact that a synchronous module can deliver a value at its local clock ticks only. Therefore, the firing of the node is delayed until the next clock event. For asynchronous nodes, clock period and clock phases play no role. However, by letting $r_i = 1$ and $p_i = 0$ for all $v_i \in V_A$, we can consolidate the description of the behaviour of synchronous and asynchronous nodes.

As a result, the definitions of admissible schedules must be extended as follows:

Definition 7 (Admissible Schedule) *An admissible schedule function of a mixed asynchronous–synchronous system is a function $\sigma : V_\Sigma \rightarrow \mathbf{R}_{\geq 0}$ that satisfies $\sigma_i(k) = \sigma_i(k)^* + p_i$, and $h_{ij} = h^*_{ij} + p_j - p_i$ where*

1. $\sigma_j(k)^* = 0$ *for all* $v_j(k) \in V_\Sigma$, $k \leq 0$
2. $\sigma_j(k)^* \geq F_j(\sigma_i(k - d_{ij})^* + h^*_{ij})$ *for all* $(v_i(k - d_{ij}), v_j(k)) \in A_\Sigma$
3. $\sigma_j(k)^* \equiv 0 \bmod r_j \; \forall j : v_j \in V_S$.

The function F_j is given by

$$F_j(a) = \begin{cases} a & : \quad v_j \in V_A \\ \left\lceil \frac{a}{r_j} \right\rceil r_j & : \quad v_j \in V_S \end{cases}$$

In case of an asynchronous node, the same definition as in Definition 3 is obtained because $\sigma_j(k)^* \geq F_j(\sigma_i(k - d_{ij})^* + h^*_{ij})$ is equivalent to $\sigma_j(k) \geq \sigma_i(k - d_{ij}) - p_i + h_{ij} + p_i - p_j$ which yields $\sigma_j(k) \geq \sigma_i(k - d_{ij}) + h_{ij}$.

For synchronous nodes, $\sigma_j(k)$ is divisible by r_j as desired. Moreover, the synchronous node fires at the *next* available clock tick. Consequently, we have the stronger schedule condition

$$\sigma_j(k) \geq p_j + \left\lceil \frac{\sigma_i(k - d_{ij}) + h_{ij} - p_j}{r_j} \right\rceil r_j$$

which is identical to Definition 7.

Example 6 *The same system as in Example 5 is considered. Its unfolding is shown in Fig. 5. To the nodes, there are assigned the earliest possible firing times $\sigma_j(k)$. For example at node $v_2(2)$ it can be seen that the second firing of the synchronous node v_2 happens at time $\sigma_2(2) = 9.1$ as $\sigma_1(2) + 3.5 = 8.5$ has been rounded up to the next integer plus $p_2 = 0.1$.*

Looking at the definition of an unfolding, it is obvious that one possible maximal–rate schedule is obtained if a node fires as soon as it is enabled. This fact is elaborated in the following theorem.

Theorem 2 (Free Schedule) *The following conditions for the free schedule of a MASS are equivalent:*

1. *There is no admissible schedule with smaller firing times $\sigma_j(k)$.*
2. $\sigma_j(k)^* = \max\{0, F_j(\sigma_i(k - d_{ij})^* + h_{ij}^*) : (v_i(k - d_{ij}), v_j(k)) \in A_\Sigma\}$

The free schedule is a maximal–rate schedule according to Definition 4.

Proof: Suppose that for some node $v_j(k)$, we have $\sigma_j(k)^* > \max\{0, F_j(\sigma_i(k - d_{ij})^* + h_{ij}^*) : (v_i(k - d_{ij}), v_j(k)) \in A_\Sigma\}$. Then the firing time $\sigma_j(k)$ can be reduced without violating condition 1,2 or 3 in Definition 7. Now, suppose that there is a schedule with a smaller kth firing time $\sigma_j(k)$ for node v_j. Comparing the second condition with condition 2 in Definition 7 yields that one of the predecessors of $v_j(k)$ must have a smaller firing time, too. Using induction and condition 1 in Definition 7, this leads to a contradiction. The free schedule is a maximal–rate schedule because of Definition 4. ■

Note that the above theorem is closely related to Bellman's shortest path equations. Consequently, the free schedule could be computed for a fixed number of firings by solving a variation of a longest path problem on the unfolding:

1. The nodes of the unfolding are ordered topologically, i.e., if there is an arc from node $v_i(k)$ to node $v_j(l)$, then $v_i(k)$ precedes $v_j(l)$ in the ordering.
2. Then, the node potentials $\sigma_j(k)$ are successively determined according to

$$\sigma_j(k)^* = \max\{0, F_j(\sigma_i(k - d_{ij})^* + h_{ij}^*) : (v_i(k - d_{ij}), v_j(k)) \in A_\Sigma\}$$

3 Maximal–rate schedules of mixed asynchronous–synchronous systems

Now, we will analyze the periodic behavior of MASS graphs and their maximal throughput rates. In this paper, we restrict ourselves to analyzing *single clock rate* MASS ($r_j = 1$ for all $v_j \in V$). Hence, we will often simply denote a MASS graph as $G = (V, A, d, h, p)$.

At first, some bounds on the average period of the maximal–rate schedule determined above will be given. These bounds can be computed efficiently (in polynomial time). In particular, they can be related to the maximal cycle mean of marked graphs with appropriately chosen weights.

Theorem 3 (Rate Bounds) *The minimal average period P_{min} of a MASS $G = (V, A, d, h, p)$ can be bounded by*

$$P_{cm}(\hat{G}) \le P_{min} \le P_{cm}(\tilde{G})$$

where $P_{cm}(\hat{G})$ and $P_{cm}(\tilde{G})$ denote the maximal cycle means of the marked graphs $\hat{G}$ and $\tilde{G}$, respectively, which are defined as follows:

- $\hat{G} = (V, A, d, \hat{h})$ *with*

$$\hat{h} = \begin{cases} h^*_{ij} & : \ v_j \in V_A \\ \lceil h^*_{ij} \rceil & : \ v_i, v_j \in V_S \\ h^*_{ij} & : \ v_i \in V_A \ , \quad v_j \in V_S \end{cases}$$

- $\tilde{G} = (V, A, d, \tilde{h})$ *with*

$$\tilde{h} = \begin{cases} h^*_{ij} & : \ v_j \in V_A \\ \lceil h^*_{ij} \rceil & : \ v_i, v_j \in V_S \\ h^*_{ij} + 1 & : \ v_i \in V_A \ , \quad v_j \in V_S \end{cases}$$

Proof: The firing times of G, $\hat{G}$ and $\tilde{G}$ in the case of free schedules are denoted $\sigma_i(k)$, $\hat{\sigma}_i(k)$ and $\tilde{\sigma}_i(k)$, respectively. Moreover, remember that $\sigma_i(k)^* = \sigma_i(k) - p_i$ and $h^*_{ij} = h_{ij} + p_i - p_j$.

If we show that for free schedules in G, $\hat{G}$ and $\tilde{G}$ the relations $\hat{\sigma}_i(k) \le \sigma_i(k)^* \le \tilde{\sigma}_i(k)$ hold for all nodes $v_i \in V$, $k \in \mathbf{Z}_{>0}$, then $P_{cm}(\hat{G}) \le P_{min} \le P_{cm}(\tilde{G})$ holds as $P_{min}(\hat{G}) = P_{cm}(\hat{G})$ and $P_{min}(\tilde{G}) = P_{cm}(\tilde{G})$.

For all nodes without predecessor we have $\hat{\sigma}_i(k) = \tilde{\sigma}_i(k) = \sigma_i(k)^* = 0$. Consequently, the initial conditions for all unfoldings are identical with $\hat{\sigma}_i(k) = \sigma_i(k)^* = \tilde{\sigma}_i(k)$.

Let us consider an arc $(v_i(k - d_{ij}), v_j(k))$ in the unfoldings of G, $\hat{G}$ and $\tilde{G}$. We will show now that the inequalities implied by such an arc in G:

$$\sigma_j(k)^* - \sigma_i(k - d_{ij})^* \ge r_{ij}$$

for some r_{ij} is less strict in the case of $\hat{G}$:

$$\sigma_j(k)^* - \sigma_i(k-d_{ij})^* \geq r_{ij} - \delta \quad , \qquad \delta \geq 0$$

and stricter in the case of $\tilde{G}$:

$$\sigma_j(k)^* - \sigma_i(k-d_{ij})^* \geq r_{ij} + \gamma \quad , \qquad \gamma \geq 0$$

As this holds for all arcs, $\hat{\sigma}_i(k) \leq \sigma_i(k)^* \leq \tilde{\sigma}_i(k)$ follows because of the monotonicity of the Bellman-type equations for free schedules, see Theorem 2. Let us consider three cases

$v_j \in V_A$: In the unfolding of G we have $\sigma_j(k)^* \geq \sigma_i(k-d_{ij})^* + h^*_{ij}$ which leads to $\sigma_j(k)^* - \sigma_i(k-d_{ij})^* \geq h^*_{ij}$. As $\hat{h}_{ij} = \tilde{h}_{ij} = h^*_{ij}$ in this case, we have $\delta = \gamma = 0$.

$v_i, v_j \in V_S$: In the unfolding of G we have $\sigma_j(k)^* \geq \lceil \sigma_i(k-d_{ij})^* + h^*_{ij} \rceil$. As $\sigma_i(k-d_{ij})^*$ is integral in a free schedule, we have $\sigma_j(k)^* \geq \sigma_i(k-d_{ij})^* + \lceil h^*_{ij} \rceil$ and $\sigma_j(k)^* - \sigma_i(k-d_{ij})^* \geq \lceil h^*_{ij} \rceil$. As $\hat{h}_{ij} = \tilde{h}_{ij} = \lceil h^*_{ij} \rceil$ in this case, we have $\delta = \gamma = 0$.

$v_i \in V_A$, $v_j \in V_S$: In the unfolding of G we have $\sigma_j(k)^* \geq \lceil \sigma_i(k-d_{ij})^* + h^*_{ij} \rceil$ which leads to $\sigma_j(k)^* - \sigma_i(k-d_{ij})^* \geq \lceil \sigma_i(k-d_{ij})^* + h^*_{ij} \rceil - \sigma_i(k-d_{ij})^*$. Consequently, we have $r_{ij} = \lceil \sigma_i(k-d_{ij})^* + h^*_{ij} \rceil - \sigma_i(k-d_{ij})^*$. Now, $r_{ij} - 1 < h^*_{ij} \leq r_{ij}$ and $\hat{h}_{ij} = h^*_{ij}$ which leads to $\hat{h}_{ij} = r_{ij} - \delta$ with $1 > \delta \geq 0$. On the other hand, $\tilde{h}_{ij} = h^*_{ij} + 1$ leads to $\tilde{h}_{ij} = r_{ij} + \gamma$ with $1 \geq \gamma > 0$.

■

Example 7 *Consider the MASS on the left side of Fig. 6 with an asynchronous node v_1 and a synchronous node v_2. In the middle, respectively on the right hand side of Fig. 6, the associated marked graphs $\tilde{G}$ and $\hat{G}$ are shown. The maximal cycle means of the associated marked graphs which determine the bounds for P_{min} are $P_{cm}(\tilde{G}) = 3.25$ and $P_{cm}(\hat{G}) = 3.1$, respectively. Without considering the unfolding of G, we can say according to Theorem 3 that $3.1 \leq P_{min}(G) \leq 3.25$. By the determination of a maximal–rate schedule for G using Theorem 2, we get $P_{min}(G) = 28/9$.*

As a result of the above theorem, we have $P_{cm}(\hat{G}) = P_{min} = P_{cm}(\tilde{G})$ for the subclass of MASS that do not have arcs from asynchronous to synchronous nodes because $\hat{G} = \tilde{G}$. Note that this subclass of MASS includes *asynchronous systems* ($V = V_A$) and GALS (globally asynchronous locally synchronous systems) ($V = V_S$) as special cases.

It turns out that one can analyze the behavior of this subclass in much more detail. In particular, we are interested in the determination of a maximal–rate periodic schedule in this case. We already know that the corresponding period is $P_{cm}(\hat{G})$ where $\hat{G} = (V, A, d, \lceil h^* \rceil)$. As a simple consequence of the proof given for Theorem 3 , one can even compare the free schedules of both graphs directly.

Corollary 1 *The free schedule of a MASS $G = (V, A, d, h, p)$ containing no arcs from asynchronous to synchronous nodes is identical to that of the corresponding marked graph $\hat{G}$ defined in Theorem 3. In particular we have $\sigma_i(k) = \hat{\sigma}_i(k) + p_i$ for all $v_i(k) \in V_\Sigma$ where $\sigma_i(k)$ and $\hat{\sigma}_i(k)$ denote the firing times of node i in G and $\hat{G}$ in a free schedule, respectively.*

Proof: The proof is based on the fact that according to the case $v_i, v_j \in V_S$ and $v_j \in V_A$, the inequalities which determine the firing times in G and $\hat{G}$ are identical, i.e., $\delta = \gamma = 0$. Therefore, we have $\sigma_i(k)^* = \sigma_i(k) - p_i = \hat{\sigma}_i(k)$. ■

Finally, the following theorem and the corresponding constructive proof lead to periodic maximal–rate admissible schedules for a subclass of mixed asynchronous–synchronous systems.

Theorem 4 *Given is a MASS G containing no arcs from asynchronous to synchronous nodes. Then any admissible schedule of $\hat{G}$ as defined in Theorem 3 leads to an admissible schedule for G with*

$$\sigma_i(k) = \begin{cases} \lceil \hat{\sigma}_i(k) \rceil + p_i & : \quad v_i \in V_S \\ \hat{\sigma}_i(k) + p_i + 1 & : \quad v_i \in V_A \end{cases}$$

with the same computation rate. If G contains synchronous nodes only, then there is an L-periodic maximal–rate schedule of G where L denotes the number of tokens in the simple cycle which determines the maximal cycle mean of $\hat{G}$. [3]

Proof: An admissible schedule for $\hat{G}$ satisfies $\hat{\sigma}_j(k) \geq \hat{\sigma}_i(k - d_{ij}) + \hat{h}_{ij}$. Note that we have defined $\sigma_i(k)^* = \sigma_i(k) - p_i$ in Definition 7.

For $v_i, v_j \in V_A$ we have $\hat{h}_{ij} = h^*_{ij}$ which leads to $\hat{\sigma}_j(k) \geq \hat{\sigma}_i(k - d_{ij}) + h^*_{ij}$. With $\hat{\sigma}_j(k) = \sigma_j(k)^* - 1$ and $\hat{\sigma}_i(k - d_{ij}) = \sigma_i(k - d_{ij})^* - 1$ we get $\sigma_j(k)^* \geq \sigma_i(k - d_{ij})^* + h^*_{ij}$ which is admissible for G.

For $v_j \in V_A$, $v_i \in V_S$ we have $\hat{h}_{ij} = h^*_{ij}$ which again leads to $\hat{\sigma}_j(k) \geq \hat{\sigma}_i(k - d_{ij}) + h^*_{ij}$. With $\hat{\sigma}_j(k) = \sigma_j(k)^* - 1$ and $\sigma_i(k - d_{ij})^* = \lceil \hat{\sigma}_i(k - d_{ij}) \rceil$ we get $\sigma_j(k)^* \geq \hat{\sigma}_i(k - d_{ij})^* + h^*_{ij} + 1 \geq h^*_{ij} + \sigma_i(k - d_{ij})^*$. Therefore, $\sigma_j(k)^* \geq \sigma_i(k - d_{ij})^* + h^*_{ij}$ which is admissible for G.

For $v_i, v_j \in V_S$ we have $\hat{h}_{ij} = \lceil h^*_{ij} \rceil$ which leads to $\hat{\sigma}_j(k) \geq \hat{\sigma}_i(k - d_{ij}) + \lceil h^*_{ij} \rceil$ and also $\lceil \hat{\sigma}_j(k) \rceil \geq \lceil \hat{\sigma}_i(k - d_{ij}) + \lceil h^*_{ij} \rceil \rceil = \lceil \hat{\sigma}_i(k - d_{ij}) \rceil + \lceil h^*_{ij} \rceil$. With $\sigma_j(k)^* = \lceil \hat{\sigma}_j(k) \rceil$ and $\sigma_i(k - d_{ij})^* = \lceil \hat{\sigma}_i(k - d_{ij}) \rceil$ we get $\sigma_j(k)^* \geq \sigma_i(k - d_{ij})^* + \lceil h^*_{ij} \rceil = \lceil \sigma_i(k - d_{ij})^* + h^*_{ij} \rceil$ which is admissible for G.

---

[3] We assume for simplicity, that this cycle is unique. However, it can be easily shown that the result can be extended to the case where the critical cycle is not unique. Let the numbers of tokens in different critical cycles have different values L_i. Then, the solution has a periodicity at most $lcm(L_i)$ over all critical cycles.

Now we will prove the periodicity part. First, a 1-periodic maximal–rate admissible schedule for $\hat{G}$ is constructed, see section 2.2. This leads to

$$\hat{\sigma}_i(k+1) = \hat{\sigma}_i(k) + P_{cm}(\hat{G})$$

According to Theorem 1, $P_{cm}(\hat{G}) = \frac{H}{L}$ where L is the number of tokens and H is the sum of weights in the critical cycle. Note that $H, L \in \mathbf{Z}$ as $\hat{h}_{ij}, d_{ij} \in \mathbf{Z}$. Therefore, we can write

$$\hat{\sigma}_i(k+L) = \hat{\sigma}_i(k) + H$$

With $\sigma_i(k)^* = \lceil \hat{\sigma}_i(k) \rceil$ we have

$$\sigma_i(k+L)^* = \sigma_i(k)^* + H$$

which clearly is an L-periodic schedule. A remark: The above result can be extended to the case where G contains asynchronous nodes. The construction of a maximal–rate periodic schedule is again possible. In this case one may use the fact that there are no paths from asynchronous nodes to synchronous nodes. Therefore, both parts of G can be dealt with independently. ■

Example 8 *Given the MASS of Fig. 7 with two synchronous nodes v_1 and v_2, clock phases $p_1 = 0.1$ and $p_2 = 0.6$ and holding times $h_{12} = 1.9$ and $h_{21} = 1.7$, respectively. If we determine the transformed arc weights, we get $h^*_{12} = 1.4$ and $h^*_{21} = 2.2$. Then we have $\hat{h}_{12} = 2$ and $\hat{h}_{21} = 3$, which can be seen in the associated marked graph $\hat{G}$ shown on the right hand side of Fig. 7. We obtain $P_{min}(\hat{G}) = 2.5$. Considering the unfolding with corresponding firing times, we can see that there is a 2-periodic schedule for G according to Theorem 4. For example, we obtain the following schedule:*

$$\sigma_1(2k) = 0.1 + 5k, \ \sigma_1(2k+1) = 3.1 + 5k,$$

$$\sigma_2(2k) = 0.6 + 5k, \ \sigma_2(2k+1) = 2.6 + 5k \ \ \forall k \in \mathbf{Z}_{>0}$$

4 Optimal phase assignment

Thus far, we know from Theorem 3 that the average period P_{min} of a MASS G can be bounded by $P_{cm}(\hat{G}) \leq P_{min} \leq P_{cm}(\tilde{G})$. The bounds on the minimal average period strongly depend on the values $\hat{h}$ and $\tilde{h}$ of $\hat{G}$ and $\tilde{G}$, respectively. By definition, $h^*_{ij} = h_{ij} + p_i - p_j$. Hence, the minimal average period depends on the given node phase assignment.

Let us suppose we have the freedom to adjust the clock phases of synchronous nodes of a given MASS. Then we might be able to find a combination of phases leading to another MASS having a smaller average period. Hence, the objective of this chapter is to investigate the effect of the phases of synchronous nodes on the minimal average period of topologically equivalent MASS graphs.

Example 9 *Given the MASS of Fig. 7 with two synchronous nodes v_1 and v_2, and holding times $h_{12} = 1.9$, and $h_{21} = 1.7$, respectively. The average periods for different combinations of clock phases can be computed as follows:*

1. $p_1 = 0.5$, $p_2 = 0.5 \Rightarrow h^*_{12} = 1.9$, $\hat{h}_{12} = \lceil 1.9 \rceil = 2$, $h^*_{21} = 1.7$, $\hat{h}_{21} = \lceil 1.7 \rceil = 2 \Rightarrow P_{cm}(\hat{G}) = P_{min} = 2$.
2. $p_1 = 0.1, p_2 = 0.6 \Rightarrow P_{cm}(\hat{G}) = P_{min} = 2.5$.
3. $p_1 = 0.2, p_2 = 0.0 \Rightarrow P_{cm}(\hat{G}) = P_{min} = 2.5$.

Fig. 8 is a density plot of the period P_{min} in dependence of p_1 (x-axis) and p_2 (y-axis). For all combinations of clock phases $p_1, p_2 \in [0, 1)$, it turns out that only two different periods exist.

This simple example shows that the average period may vary due to different assigned clock phases. Here, we therefore would like to consider the *optimal phase assignment problem*. Before we introduce the optimization problem, we define when a MASS G is called *phase optimal*.

Definition 8 *Given the set of MASS $M = \{G = (V, A, d, h, p) \, : \, 0 \le p_i < 1\}$. We define*

$$P_{opt} := \min_{G \in M} P_{min}(G)$$

and call P_{opt} the optimal average period *of the set of MASS M. A MASS $G \in M$ is said to be* phase optimal, *if $P_{min}(G) = P_{opt}$.*

In other words, a MASS G is called phase optimal if there exists no MASS G' with same topology and same weight and distance functions, but different phase assignment p and smaller average period. In this section, we are going to address the following questions:

- How can we find the minimal period P_{opt} in case the clock phases of all synchronous nodes are freely adjustable?
- Sometimes, some clock phases are fixed, others are not. What is the best adjustment of the remaining clock phases then?

We will answer these questions by proposing a simple optimization procedure. First, we introduce an exact procedure for finding an optimal phase if the MASS contains no asynchronous nodes. In the subsequent section, we will consider the general case. We will introduce a polynomial time procedure for finding a MASS $G \in M$ such that $P_{min}(G)$ satisfies $P_{min}(G) \le P_{opt} + 1$.

4.1 Exact phase optimization

From Theorem 1, we know that when given a MASS containing no arcs from asynchronous to synchronous nodes, then the free schedule is identical to that of the corresponding marked graph given in Theorem 3. Furthermore, Theorem 4 states how an admissible schedule for $\hat{G}$ can be related to an admissible schedule for G with the same computation rate. Based on these results, we are going to analyze the influence of the clock phases on the period of $\hat{G}$.

Theorem 5 *Given a set of MASS M containing synchronous nodes only, an optimal phase MASS $G \in M$ may be found by solving the following MILP:*

$$\begin{array}{l} \text{minimize } P \\ \text{subject to} \\ \quad \tau_j - \tau_i + P d_{ij} \geq \tilde{h}_{ij} \geq h_{ij} + p_i - p_j, \quad \tilde{h}_{ij} \in \mathbf{Z}_{>0} \quad \forall (v_i, v_j) \in A \\ \quad 0 \leq p_i \leq 1 \quad \forall v_i \in V \end{array} \tag{7}$$

Let $P = \frac{H}{L}$, $H, L \in \mathbf{Z}$ and H, L coprime, be an optimal solution of (7) and p an optimal phase. Then the minimal average period of the corresponding MASS is $P_{min} = P = P_{opt}$. Furthermore, the schedule $\sigma_i(k+L) = \sigma_i(k) + H$ is an L-periodic maximal-rate admissible schedule for G where

$$\sigma_i(k) = p_i + \left\lceil \tau_i + (k-1)\frac{H}{L} \right\rceil \quad \forall 1 \leq k \leq L \tag{8}$$

Proof: Let the phases of a given MASS be fixed. Then we are able to determine P_{min} by determining the minimal average period of the marked graph $\hat{G}$ by solving the linear program (see (5))

$$P_{min} = \min\{P : \tau_j - \tau_i \geq \lceil h^*_{ij} \rceil - P d_{ij} \ \forall (v_i, v_j) \in A\}$$

Using $h^*_{ij} = h_{ij} + p_i - p_j$ and replacing $\tilde{h}_{ij} = \lceil h^*_{ij} \rceil$ by $\tilde{h}_{ij} \in \mathbf{Z}$ and $\tilde{h}_{ij} \geq h^*_{ij}$, the final MILP in (7) is obtained.

The MILP has $2|V|+1$ rational variables (vectors τ, p and P) and $|A|$ integer variables $\tilde{h}_{ij}$. The proof that (8) is a periodic admissible schedule is almost trivial. From (4) in the proof of Theorem 1, we know that $\hat{\sigma}_i(k) = \tau_i + (k-1)\frac{H}{L}$, $k \in \mathbf{Z}_{>0}$ is a maximal-rate periodic schedule for $\hat{G}$. We apply the transformation described in Theorem 4 which transforms this admissible schedule for $\hat{G}$ to obtain the following admissible schedule for G with the same computation rate

$$\sigma_i(k) = \lceil \hat{\sigma}_i(k) \rceil + p_i = \left\lceil \tau_i + (k-1)\frac{H}{L} \right\rceil + p_i \quad \forall k \in \mathbf{Z}_{>0}$$

■

Example 10 *Given the MASS G in Example 9 and Fig. 7. Solving the corresponding MILP in (7) provides the solution $P = 2$. From (7), we obtain three different solution polytopes $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ corresponding to the solutions $\tilde{h}_{12} = 3, \tilde{h}_{21} = 1$ for $\mathcal{P}_1$, $\tilde{h}_{12} = 2, \tilde{h}_{21} = 2$ for $\mathcal{P}_2$, and $\tilde{h}_{12} = 1, \tilde{h}_{21} = 3$ for $\mathcal{P}_3$. The polytopes are given as follows:*

$$\begin{aligned}
\mathcal{P}_1 &: \tau_2 - \tau_1 = 1, 0.7 \leq p_1 - p_2 \leq 1.1, 0 \leq p_1, p_2, \leq 1 \\
\mathcal{P}_2 &: \tau_2 - \tau_1 = 0, -0.1 \leq p_2 - p_1 \leq 0.3, 0 \leq p_1, p_2, \leq 1 \\
\mathcal{P}_3 &: \tau_2 - \tau_1 = 1, 0.9 \leq p_2 - p_1 \leq 1.3, 0 \leq p_1, p_2, \leq 1
\end{aligned} \tag{9}$$

For example, for $\mathcal{P}_2$, the periodic admissible schedule $\hat{\sigma}$ of $\hat{G}$ satisfies:

$$\hat{\sigma}_1(k) = \hat{\sigma}_2(k) = \tau + 2k - 2 \tag{10}$$

for some parameter $\tau \in \mathbf{R}$. Fixing this arbitrary time shift to $\tau = 2$, we obtain admissible schedules σ for G corresponding to $\mathcal{P}_2$ as:

$$\begin{aligned}
\sigma_1(k) &= \lceil 2k \rceil + p_1 = 2k + p_1 \\
\sigma_2(k) &= \lceil 2k \rceil + p_2 = 2k + p_2
\end{aligned}$$

with $-0.1 \leq p_2 - p_1 \leq 0.3$, $0 \leq p_1, p_2 \leq 1$ and period $P = 2$.

Note that the MILP in Theorem 5 can be changed in a simple manner to account for other constraints concerning p_i, e.g., by simply adding the corresponding inequalities to (7).

4.2 A polynomial time procedure for nearly optimal phase assignment

In this section we give an efficient procedure to find a phase assignment p for a set of MASS M such that the corresponding MASS $G \in M$ has a period $P_{min}(G)$ at most one unit greater than P_{opt}.

Definition 9 *Given the MASS $G = (V, A, d, h, p)$, define $G^A = (V, A, d, h)$ to be a marked graph in which all nodes are assumed to be asynchronous, i.e., $F_j(a) = a$ for all $v_i \in V$ in Definition 7.*

Since the nodes in G^A may start computation at arbitrary time instances, $P_{cm}(G^A) \leq P_{opt}$. In other words, a lower bound on P_{opt} is $P_{cm}(G^A)$, the maximal cycle mean of G^A.

Lemma 1 *Let σ^A be an admissible 1-periodic schedule for G^A with period $\lceil P_{cm}(G^A) \rceil$ and $\sigma_i^A(k) = \tau_i + (k-1)\lceil P_{cm}(G^A) \rceil$. Then σ^A is a 1-periodic admissible schedule for the MASS G with the phase assignment $p_i = \tau_i - \lfloor \tau_i \rfloor \ \forall v_i \in V_S$ (and $p_i = 0 \ \forall v_i \in V_A$).*

Proof: A schedule as required by Lemma 1 can be determined using (4) and (5) in time $O(|A||V|)$. Using

$$\sigma_i^A(k) = p_i + \lfloor \tau_i \rfloor + (k-1)\lceil P_{cm}(G^A) \rceil \quad \forall v_i \in V_S$$

one obtains $\sigma_i^A(k) - p_i \in \mathbf{Z}$ for all $v_i \in V_S$. Additionally,

$$\sigma_j^A(k) \geq \sigma_i^A(k - d_{ij}) + h_{ij} \quad \forall (v_i, v_j) \in A$$

because $\sigma_i^A(k)$ is an admissible schedule for G^A. As a result, we have

$$\sigma_j^A(k) - p_j \geq \sigma_i^A(k - d_{ij}) - p_i + (h_{ij} + p_i - p_j) \quad \forall v_j \in V$$

Moreover, for $v_j \in V_S$ one obtains

$$\sigma_j^A(k) - p_j \geq \lceil \sigma_i^A(k - d_{ij}) - p_i + (h_{ij} + p_i - p_j) \rceil$$

because $\sigma_j^A(k) - p_j \in \mathbf{Z}$ for all $v_j \in V_S$. Thus, by Definition 7, σ^A is a one-periodic admissible schedule for G with the phase assignments $p_i = \tau_i - \lfloor \tau_i \rfloor \ \forall v_i \in V_S$. ■

Theorem 6 *The phase assignment p as described in Lemma 1 leads to a MASS G whose minimal average period satisfies $P_{min}(G) \leq P_{opt} + 1$.*

Proof: From Lemma 1 we know, that the phase assignment p leads to a MASS G with $P_{min}(G) \leq \lceil P_{cm}(G^A) \rceil$. Moreover, we have $P_{cm}(G^A) \leq P_{opt}$ which leads to $P_{cm}(G^A) - P_{opt} \leq 0$ and $\lceil P_{cm}(G^A) \rceil - P_{opt} \leq 1$. As a result, we have $P_{min}(G) - P_{opt} \leq 1$. The entire procedure to determine p takes $O(|V||A|)$ time according to (4) and (5). ■

The whole procedure is explained in the following example.

Example 11 *Consider the MASS shown in Fig. 7 again. We obtain $P_{cm}(G^A)$ = 1.8 and $\lceil P_{cm}(G^A) \rceil = 2$. Thus, we can use Lemma 1 to determine a phase assignment p_1, p_2 such that the resulting MASS G satisfies $P_{min}(G) \leq 2$. The following steps lead to a phase assignment according to Lemma 1 and Theorem 6:*

- *We obtain the following 1-periodic schedule for G^A:*

$$\begin{aligned} \sigma_1^A(k) &= \tau_1 + (k-1)2 \\ \sigma_2^A(k) &= \tau_2 + (k-1)2 \end{aligned}$$

  where $-0.1 \leq \tau_2 - \tau_1 \leq 0.3$ using (2).

- *Then we have*

$$\begin{aligned} p_1 &= \tau_1 - \lfloor \tau_1 \rfloor = 0 \\ p_2 &= \tau_2 - \lfloor \tau_2 \rfloor = 0 \end{aligned}$$

  from Lemma 1.

- *As can be seen, the same solution as in Example 10 is obtained. For example, with* $\lceil\tau_2\rceil = \lceil\tau_1\rceil$, *we have* $p_2 - p_1 = \tau_2 - \tau_1 - \lceil\tau_2\rceil + \lceil\tau_1\lceil = \tau_2 - \tau_1$ *which yields*

$$\sigma_1(k) = 2k + p_1, \quad \sigma_2(k) = 2k + p_2$$

with $0.1 \leq p_2 - p_1 \leq 0.3$, $0 \leq p_1, p_2, \leq 1$. *With* $\lceil\tau_2\rceil = \lceil\tau_1\rceil \pm 1$, *the other solutions in Example 10 are obtained also.*

5 Modeling Issues of Implementation

Finally, we would like to show how the MASS model may be used for representing relevant aspects of the design of mixed asynchronous–synchronous systems.

5.1 Modeling of finite queue sizes

It is possible to model a data-driven communication involving a queue of limited size. In this case, the node set V represents the set of computation modules, and the arc set A represents unidirectional communication channels with FIFO-like buffering capabilities. On the circuit level, the selftimed communication may be realized using a handshake protocol (e.g., two- [26] or four-phase [1]). On the system level, one could imagine a communication mechanism similar to that of the Kahn process network model [27] (blocking reads). The holding times h_{ij} therefore model the raw *computation time* c_j of the receiving node v_j plus a *communication time* w_{ij}, therefore $h_{ij} = c_j + w_{ij}$. We will now show how the MASS model is able to model finite buffering capacities.

Assume that a buffer is of length $q \in \mathbf{N}$, then the sending node must be blocked from initiating a new send if the buffer contains q tokens. This could be modeled by a reverse arc a_{ji} in the given graph G with $d_{ji} = q - d_{ij}$ initial tokens and holding time $h_{ji} = w_{ij}$, that is, the computation time of the reverse arc becomes the communication time.

Example 12 *Fig. 9a models the following situation: There is a data-driven communication from asynchronous node* v_1 *to asynchronous node* v_2 *with a buffer size of* $q = 3$. *A node cannot fire until all its outputs contain at least one free buffer location. Let the number of initial tokens in the buffer between* v_1 *and* v_2 *be* $d_{12} = 2$ *and the raw computation time of node* v_2 *be* $c_2 = 2.0$. *Because of the self loop with* $h_{22} = 2.0$, *module* v_2 *is not pipelined, i.e., a new token cannot be processed before the previous one has been transferred. Furthermore, let the time for communicating a token through the buffer (= time for signaling a free space) be* $w_{ij} = 0.2$. *Therefore,* $h_{ij} = c_j + w_{ij} = 2.2$. *On the reverse arc, the holding time* h_{ji} *is used to model the communication time for transmitting the availability of a space, for example* $d_{ji} = 1$ *and* $h_{ji} = 0.1$.

5.2 Computing holding times and clock periods

In this section we show how to derive holding times h_{ij} and clock periods r_j from a typical specification of a mixed asynchronous-synchronous system. In a MASS, a synchronous node v_j represents a synchronous subsystem, e.g., a synchronous finite state machine or a pipelined digital filter. On the system-level, the characteristics of such a synchronous module may be given by its *latency* (time between arrival of inputs until the corresponding outputs are available in number of clock cycles) of L_j clock cycles and its absolute clock period T_j in time units, e.g., nanoseconds (ns).

Example 13 *Consider again Example 1 and the corresponding Fig. 1. Let us suppose that the microcontroller (MC) is connected to a host as depicted in Fig. 10a. Consequently, we may model the host and the microcontroller by two synchronous nodes* v_1 *and* v_2*, see Fig. 10b. Let the clock periods of these modules be* $T_1 = 100ns$*,* $T_2 = 50ns$*, the computation time of module* v_1 *be* $L_1 = 2$ *clock cycles, and let* $L_2 = 3$ *denote the number of clock cycles needed for module* v_2 *to carry out a computation. Hence, the computation time for module* v_1 *is* $2 \times 100ns = 200ns$*, the computation time of module* v_2 *is* $3 \times 50ns = 150ns$*. The modules are connected together and communicate in a data-driven manner as shown in the MASS graph in Fig. 10b. Finally, let the absolute communication time for each data transfer take* $W_{12}^{abs} = W_{21}^{abs} = 20ns$ *on each communication link and let the system startup at time* 0 *with clock phase* $\phi_2 = 0$ *and* $\phi_1 = 25ns$*. Such a delay may be caused by a simple delay chain. We now explain how to compute computation times, clock periods, and communication times in the MASS model.*

The conversion of performance metrics of a given specification into the MASS model can be described by the following procedure:

1. Compute the *reference clock period*
   $T_{ref} := \gcd(T_i : v_i \in V_S)$;
2. Let $r_i = T_i / T_{ref}\ \forall v_i \in V_S$;
3. Let $p_i = \phi_i / T_{ref}\ \forall v_i \in V_S$;
4. Compute the *holding times*
   $h_{ij} = (W_{ij} + L_j \times T_j)/T_{ref}$ of each communication arc $(v_i, v_j) \in A$;

Example 14 *Consider the system as specified in Example 13. With* $T_{ref} = \gcd(T_1, T_2) = 50ns$*, we obtain* $r_1 = T_1/T_{ref} = 100/50 = 2$*,* $r_2 = T_2/T_{ref} = 50/50 = 1$*. The relative phases are obtained as* $p_1 = 0.5$*,* $p_2 = 0$*. Finally, the holding times are evaluated using* $h_{12}^{abs} = 20ns + 3 \times 50ns = 170ns$*, and* $h_{21}^{abs} = 20ns + 2 \times 100ns = 220ns$ *to* $h_{12} = 170/50 = 3.4$ *and* $h_{21} = 220/50 = 4.4$*. The resulting MASS graph is shown in Fig. 10b. Details on the communication between the microcontroller and the servo are not given.*

This computation shows us how to include the performance figures of synchronous subsystems in our system level MASS graph. Hence, we can use tools like retiming and pipelining [14] to change the numbers L_j and T_j, respectively, and study their influence on the system level by simulating the corresponding MASS graphs. Therefore, our model allows synchronous subsystems to be optimized and developed separately and the effects to be evaluated efficiently.

5.3 Simulation of synchronous systems

The MASS model can be used for modeling synchronous systems. Starting point is a given synchronous signal flow graph consisting of combinational modules and their interconnections which may contain synchronous registers. For the sake of simplicity, we assume only one clock phase to clock all of the registers (i.e., single clock, monophase, edge-triggered). There are many ways of modeling such a system in the MASS model. For example, if only the sequence of operations is of concern, one may simply replace the registers by initial tokens and the combinational modules by asynchronous nodes of a marked graph.

Another possibility is the modeling shown in Figs. 9b,c : The synchronous registers are replaced by synchronous nodes as shown in Fig. 9b. The combinational modules are modeled using asynchronous nodes whose holding times are the delays of the corresponding combinational modules, see Fig. 9c. The timing corresponds to that of the synchronous system if input tokens are continuously available and if all accumulated delay times between registers are smaller than the clock period.

The next possibility takes into account the global clock generation of a synchronous system. An extra clock node is responsible for the simultaneous transfer of tokens to *all* synchronous nodes each of which models one register. The connection between the clock node and the synchronous nodes is bidirectional and corresponds to an interconnection with a queue length of one. This prevents the accumulation of clock tokens and guarantees that if the longest delay path between two registers (synchronous nodes) is larger than the clock period, the period of the whole system is slowed down correspondingly. The modeling of the registers is shown in Fig. 9d where combinational modules are modeled as in Fig. 9c.

5.4 Interfacing

Finally, some remarks concerning interfacing asynchronous and/or synchronous subsystems are presented. The MASS shown in Fig. 9e models the situation where node v_1 belongs to an asynchronous (or synchronous) subsystem whereas node v_2 models the 'first' register of a synchronous subsystem. This input register accepts tokens only at integral time instances. The holding time h represents the sum of the communication time and the processing time of the synchronous node v_2, e.g., for executing the communication protocol. If the asynchronous part is faster than the synchronous one, tokens will accumulate in the queue (v_1, v_2). This can be avoided by using finite length queues, as in Fig. 9a. If the synchronous part is slower, the synchronous node v_2 does not process a token every clock tick, but all other nodes in the synchronous subsystem do. This behavior can be implemented in hardware by sending a token–bit in addition to the data which signals that a register contains valid data. The clock operates continuously. In essence, the operation of an asynchronous system is simulated by the synchronous one.

Another possibility is shown in Fig. 9f. Here, the concept of a global clock node is used, which is bidirectionally connected to all synchronous nodes of the synchronous subsystem. The incoming token is split into one that represents the data and another one that serves to enable the clock signal. In a circuit implementation, this corresponds to *stopping the clock* or *gating the clock* of the synchronous subsystem if no input data is present.

If the output of the synchronous subsystem is connected to an asynchronous subsystem, similar situations occur.

6 Conclusions and Future Research

Of further interest may be the concatenation of high-level synthesis tools for synchronous circuits (see [28] for an overview), and asynchronous circuits (see papers edited in [6, 7]), and investigate the influence of local node optimizations on the system-level performance, given e.g., by a MASS graph. The MASS model may be useful also in the domain of *interface synthesis*, which synergistically combines asynchronous and synchronous design styles. Due to the proven efficiency of the presented analysis algorithms, we believe that the MASS model is an interesting model for system-level CAD tools.

The first author would like to thank the DFG for the grant (TE163/4-1) that supported his work at UC Berkeley. S. Sriram was supported by SRC under grant 94-DC-008. Thanks also to Edward Lee for many helpful comments on the subject of this paper, and to the Ptolemy project for supplemental support.

References

[1] C. E. Seitz. *System Timing*, pages 219–262. Introduction to VLSI Systems, C. A. Mead and L. A. Conway eds. Addison-Wesley, Reading, MA, 1980.

[2] C. E. Molnar, T. P. Fang, and F. U. Rosenberger. Synthesis of Delay-Insensitive Modules. In *Proc. 1985 Chapel Hill Conference on Advanced Research in VLSI*, pages 67–86, 1985.

[3] A. J. Martin. Programming in VLSI: From Communicating Processes to Delay-Sensitive Circuits. In C. A. R. Hoare, editor, *UT Year of Program-*

ming Institute on Concurrent Programming, pages 1–64. Addison-Wesley, 1989.

[4] E. Brunvand and R. F. Sproull. Translating Concurrent Programs into Delay-Insensitive circuits. In *Prof. of ICCAD*, pages 262–265, 1989.

[5] D. Gajski, F. Vahid, S. Naranyan, and J. Gong. *Specification and Design of Embedded Systems*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[6] G. Gopalakrishnan and E. Brunvand. Special Issue on asynchronous design. *INTEGRATION, the VLSI journal*, 15(3), 1993.

[7] T. H. Meng and S. Malik. Special Issue: Asynchronous circuit design for VLSI signal processing. *Journal of VLSI Signal Processing*, 7(1-2), 1994.

[8] T. H. Meng, R. W. Brodersen, and D. G. Messerschmitt. Automatic synthesis of asynchronous circuits from high-level specifications. *IEEE Transactions on Computer-Aided Design*, 8(11):1185–1205, 1989.

[9] D. M. Shapiro. *Globally Asynchronous, Locally Synchronous Systems*. PhD thesis, Stanford University, 1984.

[10] K. Wolinski and M. Belhadj. Vers la synthese automatique de programmes signal. Technical Report 746, IRISA, Rennes, France, 1993.

[11] G. Gopalakrishnan and L. Josephson. Towards amalgating the synchronous and asynchronous styles. In *TAU'93, ACM Workshop on Timing Issues in the specification and synthesis of digital systems*, page paper 10, Malente, Germany, 1993.

[12] P. A. Subrahmanyam. Automated synthesis of mixed-mode (asynchronous and synchronous) systems. *AT&T Technical Journal*, 70(1):111–24, 1991.

[13] T. Wu, T. Tien, A. Wu, and Y. Lin. A synthesis method for mixed synchronous/asynchronous behavior. In *Proc. EDAC*, pages 277–281, 1994.

[14] C.E. Leiserson, F.M. Rose, and J.B. Saxe. Optimizing synchronous circuitry by retiming. In *Proc. Third Caltech Conf. on VLSI*, pages 87–116, 1983.

[15] C. Ramachandani. Analysis of asynchronous concurrent systems by Petri nets. Technical Report Project MAC, TR-120, MIT, Cambridge, MA, 1974.

[16] C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transactions on Software Engineering*, SE-6(5):440–449, 1980.

[17] S.M. Burns. *Performance Analysis and Optimization of Asynchronous Circuits*. PhD thesis, CIT, 1990.

[18] F. Commoner and A.W. Holt. Marked directed graphs. *Journal of Computer and System Sciences*, 5:511–523, 1971.

[19] J.L. Peterson. *Petri Net Theory and Modeling of Systems*. Prentice Hall, Reading, Mass., 1981.

[20] R. Reiter. Scheduling parallel computations. *J. of the ACM*, 15:590–599, 1968.

[21] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. John Wiley, Sons, New York, 1992.

[22] G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot. A Linear-System-Theoretic View of Discrete-Event Processes and Its Use for Performance Evaluation in Manufacturing. *IEEE Transactions on Automatic Control*, AC-30, No. 3:210–220, March 1985.

[23] E.L. Lawler. *Combinatorial optimization: Networks and matroids*. Holt, Rinehardt, and Winston, 1976.

[24] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[25] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. McGraw-Hill, Cambridge, MA, 1990.

[26] I. E. Sutherland. Micropipelines. *Communication of the ACM*, 32:720–738, 1989.

[27] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. of the IFIP Congress 74*, North Holland, 1974.

[28] R. A. Walker and R. Camposano. *A Survey of High-Level Synthesis Systems*. Kluwer, Norwell, MA, 1991.

Symbol	Explanation
$\mathbf{R}, \mathbf{Q}, \mathbf{Z}, \mathbf{N}$	the sets of real, rational, integers, and natural numbers
G	directed graph
V	node set with elements $v_i \in V$ and cardinality $\|V\|$
A	arc set with elements $(v_i, v_j) \in A$ and cardinality $\|A\|$
d	distance function $d : A \to \mathbf{Z}_{\geq 0}$: $d_{ij} = d(v_i, v_j)$
h	weight function $h : A \to \mathbf{R}_{>0}$: $h_{ij} = h(v_i, v_j)$
$G(V, A, d, h)$	characterization of a (timed) marked graph
$\Sigma(G)$	unfolding of graph G: $\Sigma(G) = (V_\Sigma, A_\Sigma, h_\Sigma$
$v_i(k)$	node in $\Sigma(G)$ that represents the kth firing of node $v_i \in V$.
σ	function $V_\Sigma \to \mathbf{R}_{\geq 0}$
$P(\sigma)$	average period of a schedule σ
$R(\sigma)$	rate of a schedule σ
P_{min}	minimal average period
R_{max}	maximal rate
$P_{cm}(G)$	maximal cycle mean of G
$G(V, A, d, h, r, p)$	characterization of a MASS with $V = V_A \cup V_S$,
	$V_A \cap V_S = \emptyset$, and
V_A	set of asynchronous nodes
V_S	set of synchronous nodes
r	clock periods of nodes, i.e., $r : V \to \mathbf{N}$
p	phase of nodes, i.e., $p : V \to \mathbf{R}$
$\sigma_i(k)^*$	$\sigma_i(k) - p_i$
h^*_{ij}	$h_{ij} + p_i - p_j$

Table 1: List of symbols

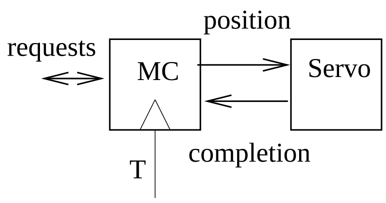

Figure 1: Example of a mixed asynchronous-synchronous system: A disk-drive application where a microcontroller controls a servo.

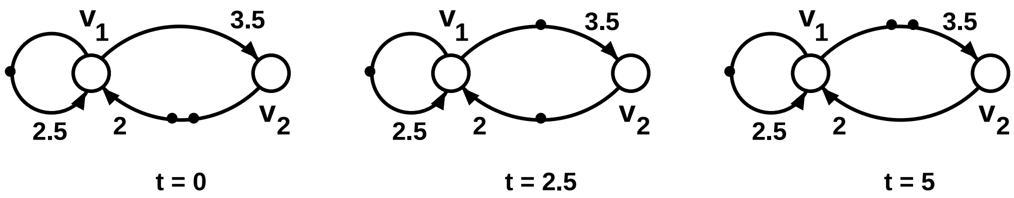

Figure 2: Example of a timed marked graph and its evolution

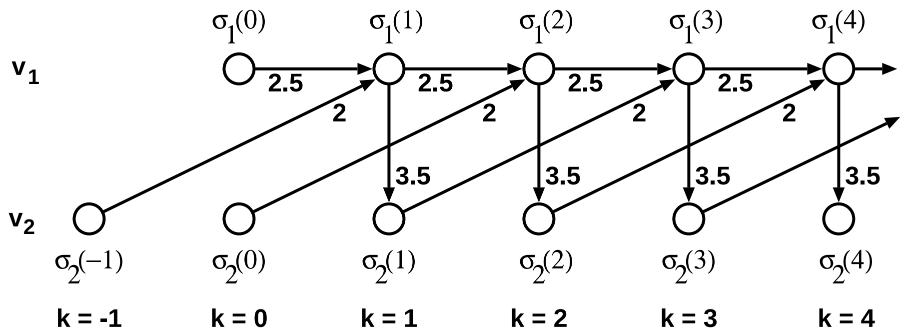

Figure 3: Unfolding of the marked graph shown in Fig. 1

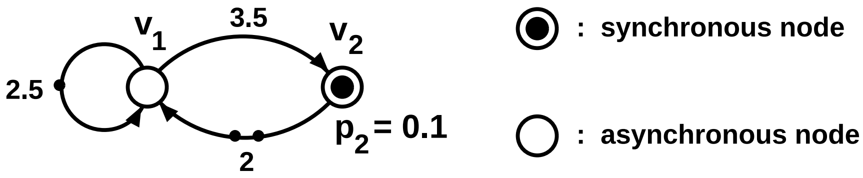

Figure 4: Example of a mixed asynchronous–synchronous system

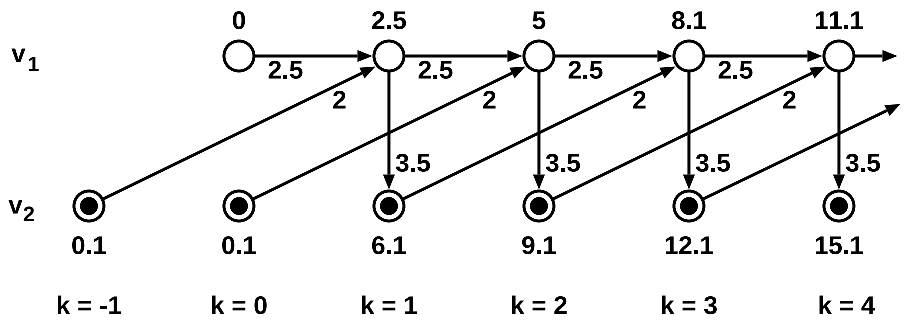

Figure 5: Unfolding of the system shown in Fig. 3

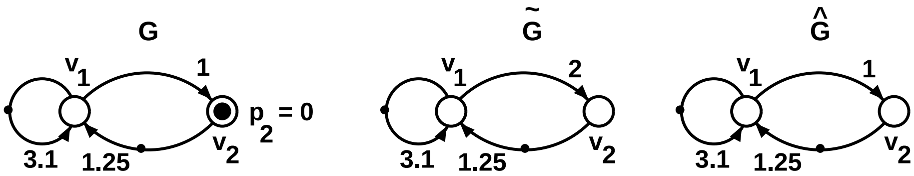

Figure 6: Example of a MASS and associated marked graphs $\tilde{G}$ and $\hat{G}$

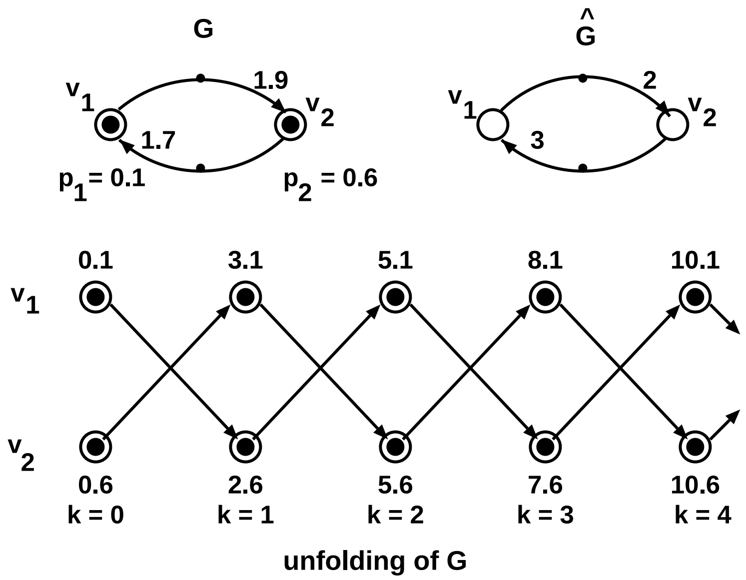

Figure 7: MASS G, associated marked graph $\hat{G}$ and unfolding of G

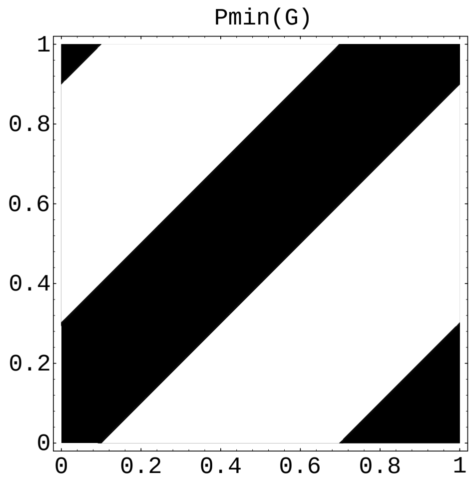

Figure 8: Density plot P_{min} in dependence of p_1 (horizontal axis) and p_2 (vertical axis). $P_{min}(G) = 2$ is shown in black, $P_{min}(G) = 2.5$ is shown in white.

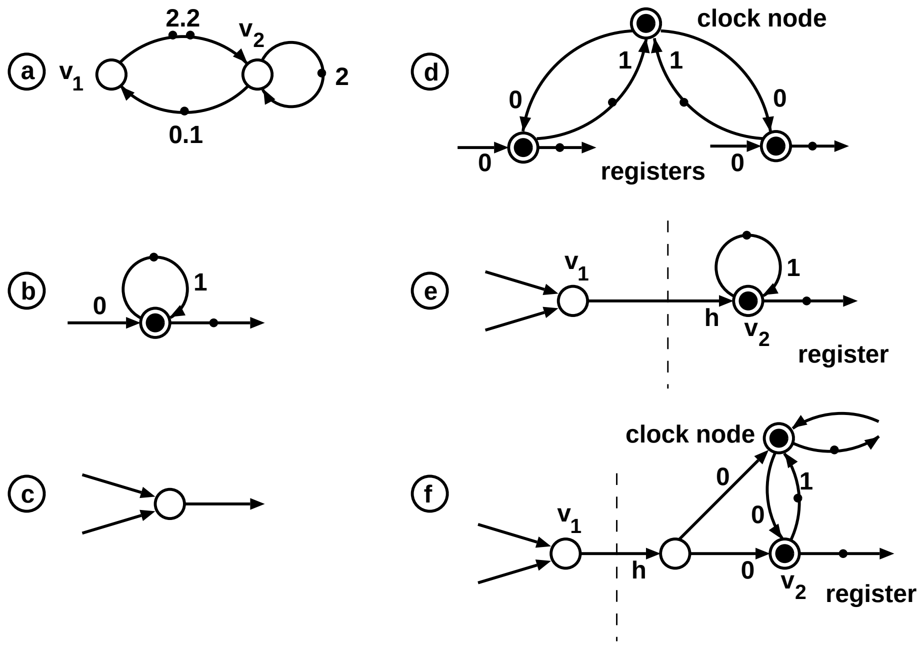

Figure 9: Modeling mixed asynchronous–synchronous systems

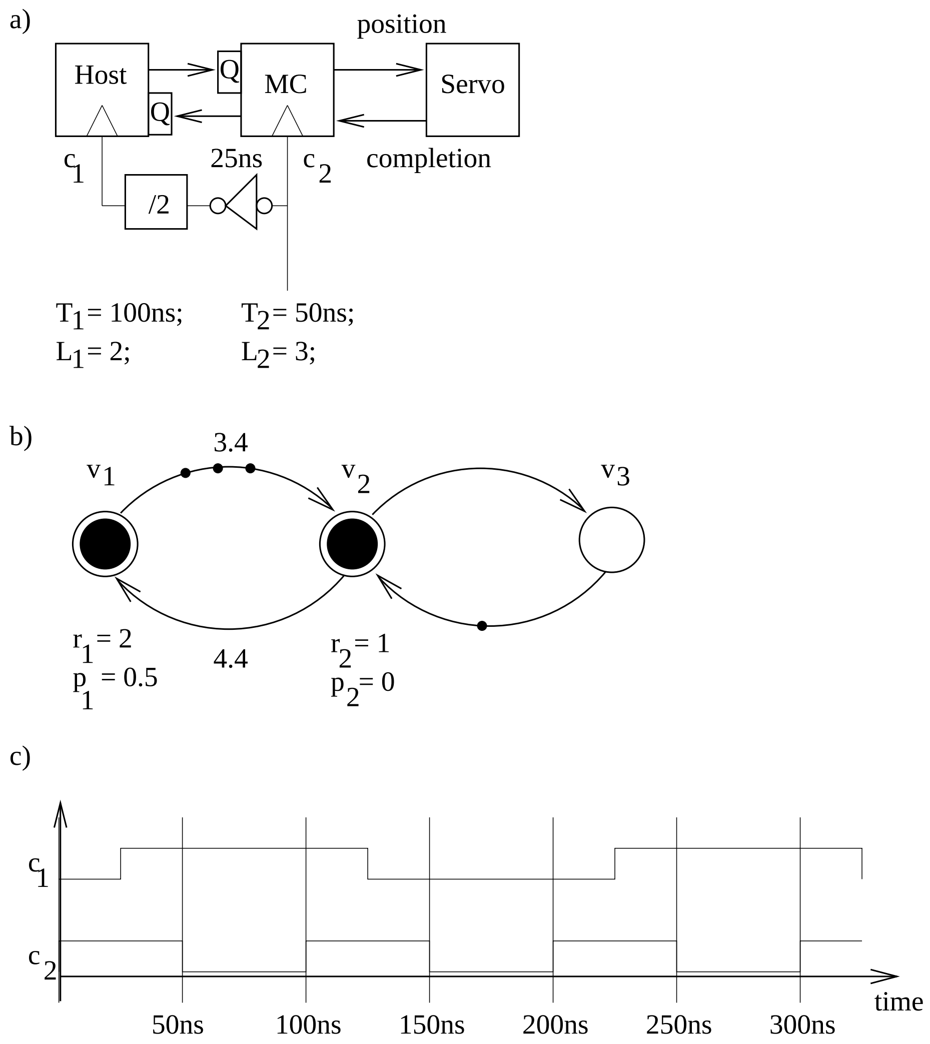

Figure 10: Conversion of an architectural model of a mixed asynchronous-synchronous system (a) with nodes of multiple clock rates (c) into a MASS (b).

Contents

List of Figures

List of Tables