In Proc. 1994 IEEE Workshop on VLSI Signal Processing

Performance Analysis of Mixed Asynchronous Synchronous Systems

J. Teich, S. Sriram
Department of EECS
University of California at Berkeley
Berkeley, CA 94720
email: teich@hoff.eecs.berkeley.edu

L. Thiele, M. Martin
Institute on Microelectronics
University of Saarland
D-66041 Saarbrücken, Germany
email: thiele@ee.uni-sb.de

Abstract - The paper is concerned with the timing analysis of a class digital systems we call mixed asynchronous–synchronous systems. In such a system, each computation module is either *synchronous* (i.e. clocked) or *asynchronous* (i.e. selftimed). The communication between modules is assumed to be selftimed for all modules. We introduce a graph model called MASS for describing the timing behaviour of such architectures. The graph contains two kinds of nodes, synchronous and asynchronous nodes. The operation model of a MASS is similar to that of a timed marked graph, however, additional schedule constraints are imposed on synchronous nodes: A synchronous node can only fire at ticks of its local module clock. We analyze the behaviour of MASS, in particular period, periodicity and maximal throughput rate.

Introduction

This paper treats the timing analysis of a class of architectures that can be characterized by selftimed communication of either synchronous (clocked) and/or selftimed computation modules. We call such architectures mixed asynchronous–synchronous systems.

Motivation In pure *synchronous systems*, a periodic clock signal is used to control its state. However, in complex digital systems, it is not possible to tie all modules to the same clock for the following reasons:

- Within the system, clocking signals arrive at the computation modules with possibly different delays (*clock skew*).
- The maximal clock rate is determined by the slowest subsystem.

Today, architectures that consist of a combination of *dedicated* circuits for carrying out time critical computations (i.e. VLSI circuits) and *general purpose* components (e.g., memory blocks, A/D and D/A converters, DSPs, etc.) are becoming increasingly attractive. Very often, they contain asynchronous components such as selftimed computation and communication modules for the following reasons:

- Most of existing digital systems have selftimed interfaces.
- Selftimed computation is in general faster than any equivalent synchronous computation.
- Selftimed communication is reliable, e.g., robust, delay–insensitive.

At a certain point in the design hierarchy, however, the communication costs (handshake hardware) for building completely selftimed systems would be intolerable. As a consequence, there is an optimum ground for the structure of a system where we do not have a monolithic synchronous system, but a mixture of synchronous and selftimed components. Some of these architectures are also referred to as *globally asynchronous, locally synchronous systems* (GALS) ([Sha84],[WB93]). For a discussion on advantages and disadvantages of synchronous, asynchronous and mixed styles, see [GJ93]. Aspects of describing such systems can be found in [Sei80], [Sha84] and [WB93]. Whereas [Sei80] and [Sha84] describe implementation aspects, [WB93] describes a language oriented approach to the design of GALS. The generation of the hardware of the asynchronous parts is done using *delay insensitive elements* [Sut89].

Goals Unfortunately, none of the work described above proposes a model that enables the determination of the exact timing behavior of a system containing synchronous and asynchronous modules communicating asynchronously. Here, we are concerned with an exact analysis of the timing of such systems, e.g., in determining the throughput rate, and compare those results to existing results in the domain of selftimed architectures and synchronous architectures. Primarily, our concern is to generate a timing model that satisfies the following requirements:

- *exactness*: The model should mirror the exact timing behaviour of mixed asynchronous–synchronous systems.
- *simplicity*: In order to be amenable for CAD, the model should be as simple as possible and abstract from implementation details.
- *generality*: The model should be general, e.g., it should be able to describe GALS (all nodes are synchronous nodes) or purely selftimed architectures (all nodes are asynchronous nodes) as special cases.

MASS- A graph model for mixed synchronous–asynchronous systems The theory of mixed asynchronous–synchronous systems as introduced here combines and makes use of the theories of both synchronous and selftimed architecture design. In the realm of synchronous architecture design, Leiserson et al. ([LRS83]) have developed a theory for analysis and optimization of synchronous circuits modeled by signal flow graphs. For modeling selftimed communication and computation, Reiter [Rei68] describes a graph model called computation graph. Sometimes, these graphs are also called marked graphs; for a classification see e.g. [Pet81]. In [Rei68], it is shown that under certain conditions, systems modeled by these graphs have an asymptotically periodic behaviour and that the minimal period of such a system is given by the maximal cycle mean. A detailed analysis of this class of discrete event systems is contained in [BCOQ92].

Based on these results, we will now define a graph model called MASS (mixed asynchronous–synchronous system) that is basically an extended marked graph with two kinds of nodes: synchronous and asynchronous nodes. Whereas the firing rule for asynchronous nodes is similar to nodes in marked graphs (a node representing a computation module can commence its operation if all incoming arcs contain valid data), a synchronous node in a MASS can only start or finish its computation at a tick of its local module clock. A MASS is therefore basically a marked graph with a modified model of computation.

We assume that the reader is familiar with the notion of a marked graph and its model of computation. For more details, see [Pet81], [BCOQ92] and references therein.

Definition 1 (Mixed Asynchronous–Synchronous System) *A mixed asynchronous–synchronous system (*MASS*) denotes an extended marked graph* $G = (V, A, d, h, p)$ *with*

- *nodes* $V = \{v_1, v_2, \cdots, v_{|V|}\}$ *representing the set of computation modules.* V *is partitioned into disjoint subsets* V_A *and* V_S*, corresponding to asynchronous and synchronous nodes, respectively.*
- *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_p *represents a communication (FIFO-like) queue between modules* v_i *and* v_j*.*
- *a distance function* $d : A \rightarrow \mathbf{Z}_{\geq 0}$*, which assigns the number of initial tokens in the queue associated with each arc, and*
- *a weight function* $h : A \rightarrow \mathbf{R}_{\geq 0}$ *which denotes the holding time of a token in the queue associated with each arc. In addition,*
- *the function* $p : V \rightarrow \mathbf{R}$ *assigns a clock phase* $0 \leq p_i < 1$ *to each synchronous node* $v_i \in V_S$*. For* $v_i \in V_A$*,* $p_i = 0$ *holds.*

Hence, in this paper, we restrict ourselves to a common clock period for all synchronous nodes. Its value is normalized to 1. With each synchronous node v_j there is associated a local clock phase p_j, i.e. the local clock is delayed with respect to the global one by p_j. As a consequence, the time instances σ_j when a synchronous node can complete an operation is constrained as follows:

$$\sigma_j - p_j \in \mathbf{Z}$$

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. In order to simplify the notation, each asynchronous node is assigned the clock phase $p_i = 0$.

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

Example 1 *Fig. 1 shows a MASS with two nodes. The initial tokens are represented by dots, the weights are the numbers printed near the arcs. Suppose that node v_2 is a synchronous node with the local clock phase $p_2 = 0.1$. By definition, we have $p_1 = 0$ as v_1 is an asynchronous node.*

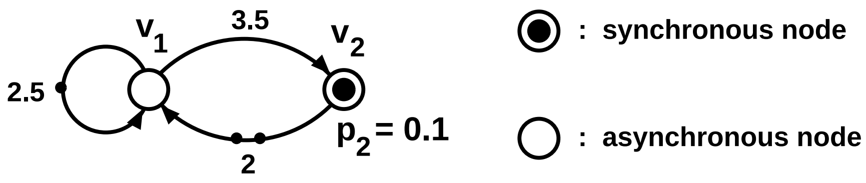

Figure 1: Example of a mixed asynchronous–synchronous system

In order to understand the scheduling of nodes in the evolution of firing events, we introduce the notion of unfolding.

Definition 2 (Unfolding) *The unfolding of a MASS $G = (V, A, d, h, p)$ is an infinite directed graph $\Sigma(G) = (V_\Sigma, A_\Sigma, h_\Sigma, p_\Sigma)$ 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 $(v_i, v_j) \in A\}$* [1]
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 there is associated 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 $(v_i(k - d_{ij}), v_j(k)) \in A_\Sigma$.*
4. *to each node $v_i(k) \in V_\Sigma$ there is associated the clock phase p_i of the corresponding node in G, i.e. $p_\Sigma : V_\Sigma \to \mathbf{R}$ with $p_\Sigma(v_i(k)) = p_i$ for all $v_i(k) \in V_\Sigma$.*

A node $v_i(k)$ of the unfolding represents the kth firing of node v_i in the MASS. An edge from $v_i(k)$ to $v_j(l)$ denotes 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 initialization of the marked graph, i.e. the placement of d_{ij} tokens into the queue corresponding to arc (v_i, v_j).

As one of our results, admissible schedules for MASS must satisfy the following conditions:

Definition 3 (Admissible Schedule) *An admissible schedule function of a mixed asynchronous–synchronous system is a function $\sigma : V_\Sigma \to \mathbf{R}_{\geq 0}$ that satisfies*

[1]We are using the simpler notation d_{ij} for denoting $d((v_i, v_j))$. Similarly, h_{ij} is used for denoting $h((v_i, v_j))$.

1. $\sigma_i(k)^* \geq 0$ *for all* $v_i(k) \in V_\Sigma$,

2. $\sigma_j(k)^* \geq F_j(\sigma_i(k-d_{ij})^* + h^*_{ij})$ *for all* $(v_i(k-d_{ij}), v_j(k)) \in A_\Sigma$

where $F_j(a) = a$ *if* $v_j \in V_A$, $F_j(a) = \lceil a \rceil$ *if* $v_j \in V_S$, *and* $\sigma_i(k)^* = \sigma_i(k) - p_i$, $h^*_{ij} = h_{ij} + p_i - p_j$.

Example 2 *Consider again the MASS in Example 1. Its unfolding is shown in Fig. 2. To the nodes, there are associated the earliest possible firing times* $\sigma_j(k)$. *For example, by looking at node* $v_2(2)$ *it can be seen that the second firing of the synchronous node* v_2 *occurs at time* $\sigma_2(2) = 9.1$ *as* $\sigma_1(2) + 3.5 = 8.5$ *has been rounded up to the next integer plus* $p_2 = 0.1$.

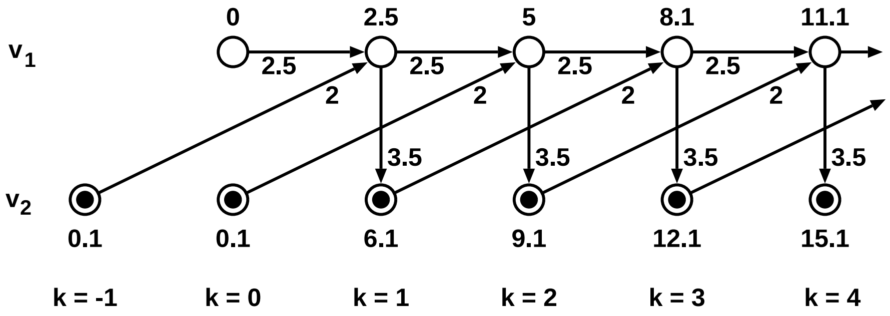

Figure 2: Unfolding of the system shown in Fig. 1

In the following, we are interested in the periodic behavior of a MASS and in the determination of its *maximal throughput rate*. Recall that the (throughput) rate R of an admissible schedule σ of a marked graph G is defined by $R(\sigma) = 1/P(\sigma)$ with the *average period* $P(\sigma) = \max\{\lim_{k\to\infty} \sigma_i(k)/k : v_i \in V\}$ and that the *maximal rate* is given by $R_{max} = 1/P_{min}$ where $P_{min} = \min\{P(\sigma) :$ *σ is an admissible schedule*$\}$. A *maximal–rate schedule* σ satisfies $P(\sigma) = P_{min}$. P_{min} is often referred to as the *minimal average period.* Furthermore, $P_{min} = P_{cm}$ where P_{cm} is called the *maximal cycle mean* of a marked graph G and is defined by $P_{cm} = \max\{\frac{\sum_{(v_i,v_j)\in W} h_{ij}}{\sum_{(v_i,v_j)\in W} d_{ij}} : W \in C(G)\}$. Here, W contains all arcs in a directed cycle and $C(G)$ contains all simple directed cycles of G. Keeping these definitions and results for marked graphs in mind, we will determine the maximal throughput rates for MASS. Looking at the definition of unfolding, it is obvious that one possible maximal–rate schedule is obtained if each node fires as soon as it is enabled. This fact is elaborated in the following Theorem.

Theorem 1 (Free Schedule) *The following conditions for the free schedule of a MASS are equivalent:*

1. *There is no admissible schedule with smaller firing times* $\sigma_i(k)$.
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.

Proof: Let us suppose that there is a schedule with a smaller kth firing time $\sigma_i(k)$ for node v_i. Comparing the second condition with condition 2. in Definition 3 yields that one of the predecessors of $v_i(k)$ must have a smaller firing time, too. Using induction and condition 1. in Definition 3, this leads to a contradiction. Now, suppose that for some node $v_i(k)$, 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 $\sigma_i(k)$ can be reduced without violating condition 1. or 2. in Definition 3. The free schedule has the maximal rate. ■

Note that the above theorem is closely related to Bellman's shortest path equations. Consequently, the free schedule can be computed by solving a variation of a longest path algorithm on the unfolding:

1. The nodes of the unfolding are ordered topologically, i.e. if there is an edge from node $v_i(k)$ to node $v_j(l)$, than $v_i(k)$ precedes $v_j(l)$ in the ordering.
2. Then, the node potentials $\sigma_j(k)$ are successively determined according to $\sigma_j(k)^* = \max\{0, F_j(\sigma_i(k-d_{ij})^* + h^*_{ij}) : (v_i(k-d_{ij}), v_j(k)) \in A_\Sigma\}$.

Now, bounds on the average period of the above determined maximal–rate schedule will be given which can be related to the maximal cycle mean of marked graphs. Therefore, the bounds may be computed in time $O(|V|\,|A|)$ using a modification of Karp's algorithm described in [Kar78].

Theorem 2 (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*

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

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

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

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 $\hat{\sigma}_i(k) \leq \sigma_i(k)^* \leq \tilde{\sigma}_i(k)$ hold for all nodes $v_i \in V$, $k \in \mathbf{Z}_{>0}$, then $P_{cm}(\hat{G}) \leq P_{min} \leq P_{cm}(\tilde{G})$ 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 edge $(v_i(k - d_{ij}), v_j(k))$ in the unfoldings of G, $\hat{G}$ and $\tilde{G}$. We will show now that the inequality implied by such an edge in G

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

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 1. 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 (see Definition 3), 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 3 *We will now consider the MASS on the left hand side of Fig. 3 with an asynchronous node v_1 and a synchronous node v_2. In the middle, resp. on the right hand side of Fig. 3, we have the associated marked graphs $\tilde{G}$ and*

$\hat{G}$. 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 2 that $3.1 \leq P_{min}(G) \leq 3.25$. By the determination of a maximal–rate schedule for G using Theorem 1, we get $P_{min}(G) = 28/9$.

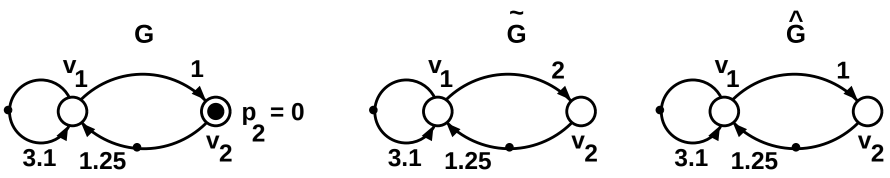

Figure 3: Example of a MASS and associated marked graphs $\tilde{G}$ and $\hat{G}$

From the above theorem it should be obvious that if a MASS contains no arcs from asynchronous nodes to synchronous nodes, we have $P_{cm}(\hat{G}) = P_{min} = P_{cm}(\tilde{G})$ because $\hat{G} = \tilde{G}$.

It turns out that one can analyze the behavior of MASS which contain synchronous nodes only 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 the above theorem, we 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 2. 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 $\sigma_i(k)^* = \sigma_i(k) - p_i = \hat{\sigma}_i(k)$. ■

The following theorem states how an admissible schedule for $\hat{G}$ leads to an admissible schedule for G.

Theorem 3 *Given is a MASS G containing no arcs from asynchronous to synchronous nodes. Then any admissible schedule for $\hat{G}$ as defined in Theorem 2 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, there is an L-periodic maximal–rate schedule for G where L denotes the number of tokens in a simple cycle which determines the maximal cycle mean of $\hat{G}$.

Due to space limitations, we omit the proof of Theorem 3. Instead, we explain its essence by introducing a simple, illustrative example.

Example 4 *Given the MASS G in Fig. 4 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. 4. 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 3. For example, we obtain the following schedule:*

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

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

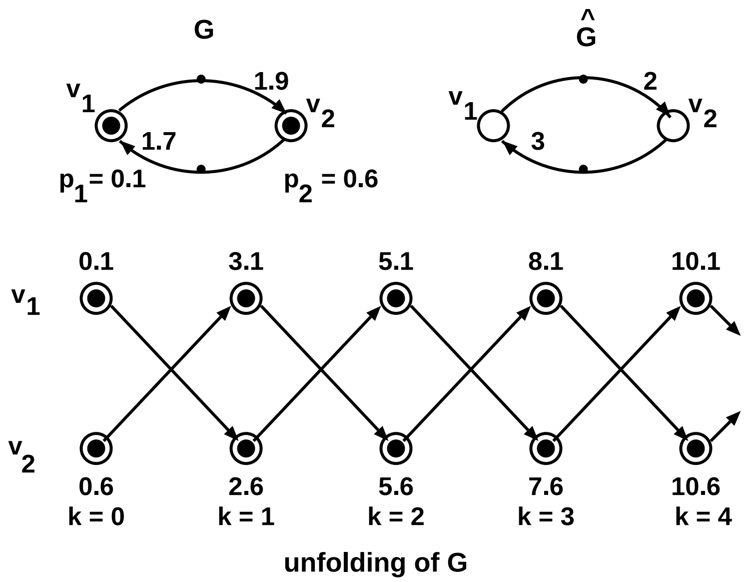

Figure 4: MASS G, associated marked graph $\hat{G}$ and unfolding of G

PERSPECTIVES

In a future publication, we will show how the presented analysis can be used for optimization of mixed asynchronous–synchronous systems, e.g., by adjusting the phases of the synchronous modules for purpose of optimizing the throughput rate, etc. We would also like to extend the MASS model to consider multiple clock rates and deal with issues of synthesis.

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 the paper, and to the Ptolemy project.

References

[BCOQ92] F. Baccelli, G. Cohen, G.J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity.* John Wiley, Sons, New York, 1992.

[GJ93] 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.

[Kar78] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1978.

[LRS83] 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.

[Pet81] J.L. Peterson. *Petri Net Theory and Modeling of Systems.* Prentice Hall, Reading, Mass., 1981.

[Rei68] R. Reiter. Scheduling parallel computations. *J. of the ACM*, 15:590–599, 1968.

[Sei80] 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.

[Sha84] D. M. Shapiro. *Globally Asynchronous, Locally Synchronous Systems.* PhD thesis, Stanford University, 1984.

[Sut89] I. E. Sutherland. Micropipelines. *Communication of the ACM*, 32:720–738, 1989.

[WB93] K. Wolinski and M. Belhadj. Vers la synthese automatique de programmes signal. Technical Report 746, IRISA, Rennes, France, 1993.