An algebraic approach to time borrowing

Broomhead, David and Furber, Steve and Johnson, Marianne

2012

MIMS EPrint: **2012.60**

ISSN 1749-9097

AN ALGEBRAIC APPROACH TO TIME BORROWING

DAVID BROOMHEAD, STEVE FURBER, AND MARIANNE JOHNSON

Abstract. This paper is about a novel application of linear algebra to the timing of digital hardware. In particular we describe a rigorous, algorithmic approach to 'time borrowing'. Time borrowing is a technique whereby the use of a multiphase clock can allow for a more flexible, efficient use of time. In this approach the system is clocked periodically, but within each clock cycle processes are allowed to interact asynchronously allowing longer processes to be juxtaposed with shorter processes. We show that this problem can be solved completely using linear algebra defined over the max-plus semi-ring, and that the method so obtained conforms with an earlier, heuristic approach to the problem.

1. Introduction

In digital electronics, a latch is a piece of logic which acts as an element of memory. The basic property of a latch is that it maintains a fixed output signal, which has the value of the last input it received when its input was enabled. In clocked—as opposed to asynchronous—circuits, latches are usually *edge-triggered*, in the sense that their input is enabled on the rising (say) edge of the clock pulse. (The assumption here is that the clock is manifest as a piece-wise constant periodic signal distributed synchronously across the circuit.) It is possible to view edge-triggered latches in terms of simpler *transparent* latches, whose enabling function depends on the (constant) value of the clock signal rather than its dynamics at an edge. Early in the development of VLSI design methodologies the use of multi-phase clocks with level-sensitive transparent latches was a common approach [11], as this made very efficient use of limited transistor resources. However, as transistor resources increased and dependency on design automation became more prevalent, methodologies moved firmly towards single clocks and edge-triggered registers. The reasons here are clear: an edge-triggered register retimes its outputs, making timing analysis more straightforward both for human designers and, more importantly, for static timing analysis tools. Today nearly all complex digital chip designs exploit the timing analysis benefits of edge-triggered registers.

With the increasing importance of design for variability [1], however, it might be timely to review this trend. The feature of edge-triggered design that makes timing analysis straightforward also guarantees that the critical path will be one clock cycle long. A design methodology that results in multi-cycle critical paths will benefit from the reduced variability of those paths (in proportion to their mean delay) since this ratio goes down with the square root of the number of gates in series in the path.

Date: 23rd May 2012.

The authors would like to thank Sasha Borovik and Mark Kambites for their help.

This work was carried out as part of the Centre for Interdisciplinary Computational and Analysis (CICADA) project which is funded by the Engineering and Physical Sciences Research Council (EPSRC) .

Methodologies have been proposed to compensate for variability based on time stealing [12, 21], where edge-triggered registers are retained but the timing of their clocks adjusted. However level-sensitive latches automatically offer the possibility of time borrowing [19], which is arguably a more straightforward approach to the problem as it does not rely on clock retiming. It has, to date, carried the drawback of having no effective timing analysis tools, although there have been attempts to remedy this position, for example using linear programming [17]. Here we address this shortcoming, and offer a rigorous algorithmic approach to timing analysis in time borrowing circuits. Our approach formalizes an earlier algorithm based on graph analysis [7] and offers a firm basis for the exploitation of time borrowing in circuits, based on multi-phase clocks and level-sensitive latches, to reduce the adverse impact of variability on digital circuit performance.

We will show that the timing analysis problem for time borrowing circuits reduces to one of solving a system of linear equations over the max-plus semi-ring. 'Minimax algebra' was originally championed by Cuninghame-Green [6] and has applications in queueing theory and even railway time tabling. Nice introductions can be found in the texts by Butkovič [4] and by Heidergott, Olsder and van der Woude [9]. We shall try to make this paper reasonably self-contained, but the mathematical details we shall develop rest heavily on the development described in [9].

We note that the application of max-plus methods to timing analysis is not a new idea. In 1994 Gunawardena [8] proposed the use of min-max functions to model the timing of digital circuits, writing

> *"Max-plus algebra is a highly developed theory, which seems to be largely unknown to those working in timing analysis. It is a powerful tool for studying systems with only maximum constraints and is an essential foundation for the deeper results in the theory of min-max functions."*

Gunawardena noted that many of the results proved in [3, 18] are instances of general theorems in max-plus algebra. Nearly twenty years on, max-plus methods do not appear to be widespread amongst the timing analysis community, with only a handful of papers in this area [15, 16, 20]. We believe that these methods have great potential to be exploited. Thus, the aim of this paper is to illustrate some simple concepts in max-plus algebra and show how these can be brought to bear on the time borrowing problem.

2. Partially ordered time in digital circuitry

A graphical way to represent the relationships between the times at which the various signals generated within a device become valid is a *timing dependency graph.* This is basically a Hasse diagram (see, for example, Cameron [5]) representing the fact that the times at which the various signals become valid is a partially ordered set. Of course, times are non-negative real numbers, which are totally ordered using the familiar notion of the magnitude—we write $t_1 < t_2$ if t_1 is smaller in magnitude than t_2. However, in this context it is the *constraints* on the timings of the signals which are the issue. In principle, the timing of the individual signals is variable so that the constraints only impose a partial ordering on the timings. We will say $t_1 \prec t_2$[1] if t_1 is constrained to occur earlier than t_2.

[1]Which can be read "t_1 precedes t_2".

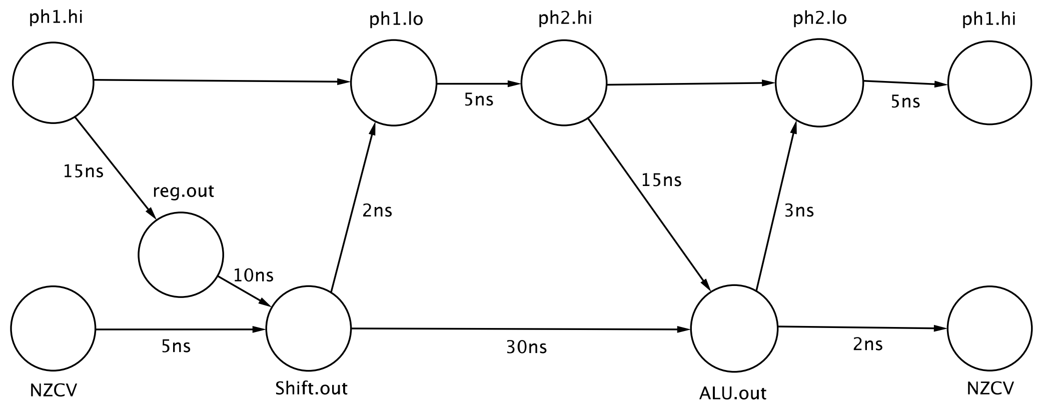

FIGURE 1. A timing dependency graph of a periodically clocked system with asynchronous processes such as might be found in the ARM6 core. The vertices of the graph (shown as circles) represent the times at which the various signals (given by the labels) become valid, and the directed edges give the causal links between the signals, with the labels representing delays (in nanoseconds) due to the linking logic. Note that the leftmost and rightmost processes are the same.

Figure 1 shows a simple example representing the kind of timings that were of importance in the design of the ARM6 core (see [7, page 105]) . The vertices of the graph (shown as circles) represent the times at which the various signals (given by the labels) become valid, and the directed edges give the causal links between the signals, with the labels representing delays (in nanoseconds) due to the linking logic. In the figure we see for example, vertices labelled: ph1.hi; ph1.lo; ph2.hi and ph2.lo, these represent the times at which the high and low signals of the two clock phases become valid. Clearly ph1.hi $\prec$ ph1.lo $\prec$ ph2.hi $\prec$ ph2.lo, that is, the phases of the clock represent a chain (a totally ordered subset) within the Hasse diagram. In contrast, we see that the time at which the register output becomes valid (reg.out) is not comparable with the time at which the output signal from the processor status register (NZCV) becomes valid although both must occur before the output of the shifter (Shift.out) becomes valid. The whole diagram—which is actually an infinite periodic structure—represents a pipelined process in which the next input is being prepared for the Arithmetic-Logic Unit (ALU) while the previous input is being operated on. The control of the whole is ensured by requiring that the transparent latches associated with these two parts are open during different phases of the clock.

3. The general problem

In this context, a Hasse diagram is an acyclic graph $G = \{V, E\}$ consisting of a set of vertices V representing the times in question, and E a set of directed arcs indicating the dependencies between the times. There is a directed arc $e_{ij} \in E$ if the time $v_i \in V$ *covers* $v_j \in V$; that is, there is no other $v_k \in V$ such that $v_j \prec v_k \prec v_i$. In this application, the basic structure of the Hasse diagram is augmented by associating with each arc in E a non-negative real number representing the delay due to the linking logic. For example, say that the signal representing the input to an adder becomes valid at v_j and the signal representing the output of the adder becomes valid at v_i, then τ_{ij} associated with $e_{ij} \in E$, tells us how long it takes for the adder to do its job. Formally, we can think of the function $\tau : E \to \mathbb{R}$ as giving the timing constraints on the whole system via local rules which say that a process signal cannot become valid until after all its input signals become valid.

Ultimately, the system we are interested in will be controlled by a periodic clock. This implies that the graphical representation of the system will have a specific structure; G will consist of an infinite repetition of an acyclic subgraph, as is illustrated in Figure 2. To the left and right of the central block of the figure (which represents a general acyclic graph) there are two sets of distinguished vertices. These are known as *cyclic nodes*(see [7, page 105]). They form an antichain—a set of times which are not comparable—which disconnects G. The set of cyclic nodes on the right corresponds to that on the left but one clock cycle later.

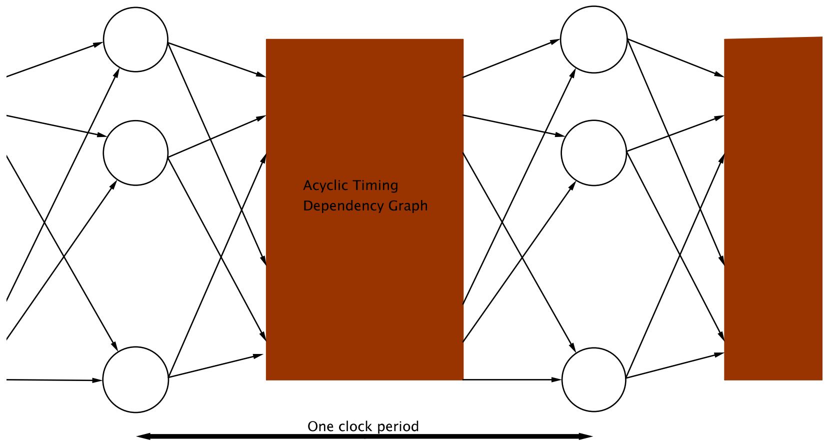

FIGURE 2. The general timing dependency graph of a periodically clocked system.

4. THE MAX-PLUS SEMI-RING

The natural algebraic structure that this kind of problem suggests is that associated with the binary operations of addition and taking the maximum of real numbers. Imagine that a given process is waiting for inputs from two other processes before it can begin. If these two input processes are taking place in parallel, then the output signal of our given process cannot become valid before the *maximum* of the times at which the signals from the two input processes become valid. On the other hand, if the two input processes are sequential, then the output signal of the given process cannot become valid before the *sum* of the delays on the corresponding sequence of dependency arcs.

Consider the set of real numbers $\mathbb{R}$ and augment this with a smallest (with respect to the total ordering on $\mathbb{R}$) element $\varepsilon = -\infty$. Let us say $\overline{\mathbb{R}} = \mathbb{R} \cup \{\varepsilon\}$. Then the binary operations $\oplus$ and $\otimes$ defined by

$$a \oplus b \overset{\text{def}}{=} \max\{a, b\} \quad \text{and} \quad a \otimes b \overset{\text{def}}{=} a + b \quad \text{for all} \;\; a, b \in \overline{\mathbb{R}},$$

give a rich arithmetic on $\overline{\mathbb{R}}$. (To avoid ambiguities we will sometimes refer to $\oplus$ and $\otimes$ as *max-plus* addition and multiplication respectively.) In particular, $\oplus$ and $\otimes$ are commutative and associative operations and $\otimes$ distributes over $\oplus$ since

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \iff a + \max\{b, c\} = \max\{a + b, a + c\}.$$

In addition, ε is the zero element in the sense that $a \oplus \varepsilon = a$ for any $a \in \overline{\mathbb{R}}$, and like zero in the usual number system, $a \otimes \varepsilon = \varepsilon$ for any $a \in \overline{\mathbb{R}}$. The number 0 on the other hand is not the zero element of $\overline{\mathbb{R}}$. In fact it acts as the unit element since $a \otimes 0 = a$ for any $a \in \overline{\mathbb{R}}$.

There are clear differences between this max-plus system and the arithmetic we were taught as children. In particular, $a \oplus a = a$ for all $a \in \overline{\mathbb{R}}$, so $\oplus$ is idempotent. A direct consequence of this is that there is no additive inverse[2]; that is, we don't have a nice analogue of the negative of a number[3]. This means that $(\overline{\mathbb{R}}, \oplus, \otimes)$ is a *semi*-ring, albeit one that is commutative and idempotent. This does not, however, mean that $(\overline{\mathbb{R}}, \oplus, \otimes)$ is useless. In the next section we will give the definition of a linear algebra over $(\overline{\mathbb{R}}, \oplus, \otimes)$ and summarize several important consequential results which we shall use to solve the general time borrowing problem.

As an aside, it is worth remarking on the analogy between max-plus arithmetic and the arithmetic of logarithms which is suggested by the correspondence between the operations $\otimes$ and $+$. In fact, max-plus arithmetic can be seen as a limit of a family of semi-rings isomorphic to the usual semi-ring with operations $+$ and $\times$ over the positive reals. These (semi-ring) isomorphisms, which depend continuously on h, are defined as $D_h : (\mathbb{R}_+ \setminus \{0\}, +, \times) \longrightarrow (\overline{\mathbb{R}}, \oplus_h, \otimes_h)$ sending $x \mapsto h \log x$, where the operations $\oplus_h$ and $\otimes_h$ are given by

$$a \otimes_h b \overset{\text{def}}{=} a + b \quad \text{and} \quad a \oplus_h b \overset{\text{def}}{=} \begin{cases} h \log(e^{a/h} + e^{b/h}) & h > 0, \\ \max\{a, b\} & h = 0. \end{cases}$$

There is a nice description of this and its relation to Litvinov and Maslov's formulation of the Quantum Mechanical Correspondence Principle in the paper by Viro [22].

5. Max-plus linear algebra

5.1. **Basic definitions.**

Given the semi-ring $(\overline{\mathbb{R}}, \oplus, \otimes)$, we can define matrices in the obvious way

$$(A)_{ij} \overset{\text{def}}{=} a_{ij}, \quad \text{with } a_{ij} \in \overline{\mathbb{R}}, \text{and } i = 1, \ldots m, j = 1, \ldots n$$

and operations on these which are a natural generalisation of $\oplus$ and $\otimes$:

$$\begin{aligned} (A \oplus B)_{ij} &\overset{\text{def}}{=} a_{ij} \oplus b_{ij} \quad \text{with } A, B \in \overline{\mathbb{R}}^{m \times n}, \\ (A \otimes B)_{ij} &\overset{\text{def}}{=} \bigoplus_{k=1}^{l} a_{ik} \otimes b_{kj} \quad A \in \overline{\mathbb{R}}^{m \times l},\ B \in \overline{\mathbb{R}}^{l \times n}. \end{aligned}$$

The lack of an additive inverse in $(\overline{\mathbb{R}}, \oplus, \otimes)$ suggests that the linear algebra resulting from these definitions might not be very useful. However, this is not the case. Our

[2]Assume on the contrary that for any given $a \neq \varepsilon$ there exists $b \in \overline{\mathbb{R}}$ such that $a \oplus b = \varepsilon$, then if we "add" a to both sides of this equation, the associativity of $\oplus$ and its idempotency give the contradictory result $a \oplus b = a$.

[3]Of course there are negative numbers in $\overline{\mathbb{R}}$, but these should be thought of in terms of the multiplicative inverse since $x \otimes (-x) = 0$

theory of time borrowing will require a few well-established results from the quite extensive linear algebra that can be developed; these will be recalled in the following.

5.2. **A graphical interpretation.** It will turn out to be useful to have a graphical interpretation of a square matrix. Suppose we have an $n \times n$ matrix A over $(\overline{\mathbb{R}}, \oplus, \otimes)$. We can associate this uniquely with a weighted directed graph G_A such that the order (the number of vertices) of G_A is n, and for each $A_{ij} = a_{ij} \neq \varepsilon$ there is an edge from the vertex labelled j to the vertex labelled i. Each edge is assigned a weight given by the corresponding matrix element a_{ij}. Conventionally, such graphs are known as *communication graphs*, but for the present purpose we see that the communication graph associated with a matrix of linking logic delay times is closely related to the timing dependency graph introduced earlier. Such differences as there are arise when we take an infinite, periodic, timing dependency graph—which must be acyclic—and represent it as a finite graph with cycles.

Given a matrix A of delay times, the above interpretation also provides a picture of powers of A. For example, the matrix element $A_{ij}^{\otimes 2} = \bigoplus_k A_{ik} \otimes A_{kj}$ tells us about all two-step processes in which signal j becoming valid leads to signal k becoming valid which then leads to signal i becoming valid. $A_{ij}^{\otimes 2}$ gives the longest of these times when all the possible intermediate signals, k, are considered. In terms of the timing dependency graph, this maximisation is over all logic paths of length two between vertex j and vertex i. Clearly, the mth power of A provides analogous information about m-step logic paths.

5.3. **Eigenvalues and eigenvectors.** The eigenvalue problem for square matrices is defined as one would expect. Given an $n \times n$ matrix A we look for a $\lambda \in \overline{\mathbb{R}}$ and a (non-trivial) vector $x \in \overline{\mathbb{R}}^n$ such that

$$A \otimes x = \lambda \otimes x. \tag{1}$$

The following theorem, well-known to max-plus experts, links the existence of a (unique) eigenvalue of a matrix to the structure of the corresponding communication graph (see, for example, [9]).

Theorem 1. *If G_A is strongly connected, then A possesses a unique eigenvalue which is finite and equal to the maximal average delay of circuits in G_A.*

Here, "finite" is taken to mean that the eigenvalue is not equal to ε. A circuit is a path in G_A which leads from a vertex back to itself. There are two obvious notions of length that might be associated with a circuit: one is the number of vertices (or edges) in the circuit (which we shall generally refer to as the length) and the other is the sum of the delays on the edges in the circuit. If we divide the latter by the former we get the average delay associated with (the edges of) a circuit.

5.3.1. *Example.* Figure 3 shows a small example of a timing dependency graph for a periodically forced system of asynchronous processes. As before, this should be thought of as a portion of an infinite periodic structure. If we identify each leftmost vertex in the figure with the corresponding rightmost vertex (we imagine the infinite graph as being the lift of one drawn on a cylinder) we can write down the following

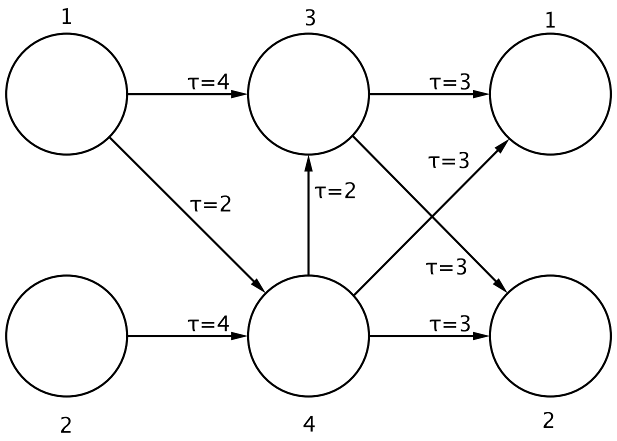

FIGURE 3. A simple example of a graph of timing dependencies for a periodically clocked system with asynchronous processes. The processes are represented by numbered discs and the timings are given by the edge labels. Note that the leftmost and rightmost processes are the same.

matrix of delay times:

$$A = \begin{pmatrix} \varepsilon & \varepsilon & 3 & 3 \\ \varepsilon & \varepsilon & 3 & 3 \\ 4 & \varepsilon & \varepsilon & 2 \\ 2 & 4 & \varepsilon & \varepsilon \end{pmatrix}. \tag{2}$$

Given the identification of vertices, it is clear that the timing dependency graph is strongly connected, and therefore, according to Theorem 1, it has a unique eigenvalue. There are circuits of length 2 (the one from vertex 1 to vertex 2 and back to vertex 1, for example) which have a total delay of $4 + 3 = 7$ and therefore an average delay of 3.5. Longer circuits cannot have a larger average than this (whenever a delay of 4 occurs in a circuit it must be followed by a delay of at most 3), therefore the eigenvalue is $\lambda = 3.5$.

If we define the trace of a square matrix to be the max-plus sum of its diagonal elements, we see that $\mathrm{Tr}A^{\otimes k}$ is the maximum delay of any k-circuit of the corresponding timing dependency graph. Thus, for a strongly connected graph G_A the eigenvalue can be found by calculating

$$\bigoplus_{k\geq 1} \frac{\mathrm{Tr}A^{\otimes k}}{k}.$$

Here the division $\frac{a}{k}$ has the usual meaning if $a \in \mathbb{R}$ and $\frac{a}{k} = \varepsilon$ if $a = \varepsilon$. In this example calculating the first few powers of A by hand we get:

$$\begin{aligned}
\mathrm{Tr}A &= \varepsilon,\ \frac{\mathrm{Tr}A}{1} = \varepsilon, \\
\mathrm{Tr}A^{\otimes 2} &= 7,\ \frac{\mathrm{Tr}A^{\otimes 2}}{2} = 3.5, \\
\mathrm{Tr}A^{\otimes 3} &= 9,\ \frac{\mathrm{Tr}A^{\otimes 3}}{3} = 3, \\
\mathrm{Tr}A^{\otimes 4} &= 14,\ \frac{\mathrm{Tr}A^{\otimes 4}}{4} = 3.5,
\end{aligned}$$

which is consistent with $\lambda = 3.5$.

This example shows clearly that the eigenvalue of the timing dependency graph is not directly relevant to the time borrowing problem. The problem is that the eigenvalue gives us the mean delay rather than the actual delay. Indeed, we see from the above computations of the traces that there is actually a path of length 3 from a cyclic node to the corresponding cyclic node one period later which has a delay of 9. This is the cycle from vertex 2 to vertex 4 to vertex 3 and back to vertex 2. Clearly this would control the timing of the clock rather than cycles (such as the one from vertex 1 to vertex 2 and back to vertex 1) which determine the eigenvalue.

5.4. **The Kleene star.** Given the lack of an additive inverse, it is not obvious that we can solve systems of linear equations in this algebra. In fact, it turns out that the only matrices which are invertible are monomial (see, for example [10]). However, things are not as bad as this suggests. In practice our problem will require that we solve a particular sort of linear equation in the form

$$x = (A \otimes x) \oplus b, \tag{3}$$

where x is an unknown vector in $\overline{\mathbb{R}}^n$ and A is a known $n \times n$ matrix and $b \in \overline{\mathbb{R}}^n$ is given. Formally, this equation can be solved by means of the Kleene star.

The Kleene star of the matrix A is defined as

$$A^* = \bigoplus_{j=0}^{\infty} A^{\otimes j},$$

that is, the max-plus sum of the powers of A where $A^{\otimes 0} = id$, the max-plus $n \times n$ identity matrix[4]. Given the Kleene star, the solution to equation (3) is

$$x = A^* \otimes b.$$

This is easily checked by substitution.

Evidently, the definition of A^* can only be useful if there is a finite upper bound to the entries of the $A^{\otimes j}$. In the application we describe here, there is indeed an upper bound and this is reached for some finite value of j. The following lemma will be relevant when we come to the solution of our problem:

Lemma 2. *Consider A, an $n \times n$ matrix of delays for which the corresponding graph G_A is finite and acyclic. Then there is an integer l_* such that*

$$A^* = \bigoplus_{j=0}^{l_*} A^{\otimes j}.$$

Proof. From the hypotheses that G_A is finite and acyclic it follows that it has a finite, longest directed path. Let us say that this has length l_*. Taking a matrix element $A_{ik}^{\otimes j}$, this gives the longest delay of all the j-step paths from k to i, or ε if no such path exists. Clearly, if $j > l_*$ all matrix elements of $A^{\otimes j}$ must be ε, and therefore powers of A greater than l_* do not contribute to A^*. $\square$

[4]That is, the $n \times n$ matrix with 0 in every diagonal entry and ε everywhere else.

5.5. **Orbits and asymptotics of autonomous linear dynamical systems.** The final issue that we need to consider is the asymptotics of iterations of the form

$$x(k+1) = A \otimes x(k), \tag{4}$$

where A is an $n \times n$ matrix over $(\overline{\mathbb{R}}, \oplus, \otimes)$, the $x(k)$ are vectors in $\overline{\mathbb{R}}^n$ and some $x(0) \in \overline{\mathbb{R}}^n$ is given. For the purposes of this paper we can assume that G_A has a single strong component[5]. Then, according to Theorem 1, A has a unique eigenvalue *lambda* and this eigenvalue is finite.

The simplest situation is if $x(0)$ is an eigenvector of A. Repeated use of equation (1) gives

$$x(k) = \lambda^{\otimes k} \otimes x(0),$$

where λ is the eigenvalue, so that in this case each component of $x_j(k)$ of $x(k)$ grows linearly with k giving

$$x_j(k) = x_j(0) + \lambda k$$

in usual arithmetic.

This suggests a change of coordinates (which might be termed a *moving frame*) which is useful even with a more general choice of $x(0)$. Let $y(k)$ be such that

$$x(k) = \lambda^{\otimes k} \otimes y(k).$$

Since the eigenvalue λ is finite, each of the powers $\lambda^{\otimes k}$ has a multiplicative inverse. Using this to rewrite equation (4) we get:

$$\begin{aligned} y(k+1) &= \lambda^{\otimes -(k+1)} \otimes A \otimes \lambda^{\otimes k} \otimes y(k) \\ &= \lambda^{\otimes -1} \otimes A \otimes y(k) \\ &= \hat{A} \otimes y(k), \end{aligned} \tag{5}$$

where we have used the invertibility of λ. The notation $\hat{A}$ represents the *normalised matrix* obtained by subtracting the eigenvalue of A from each of its matrix elements. The underlying graph for this matrix (ignoring edge weights) is the same as that for A, in this case however the maximum average delay of its circuits—and hence the eigenvalue of $\hat{A}$—is 0. Obviously, if $x(0) = y(0)$ is an eigenvector of A (and hence an eigenvector of $\hat{A}$), then $y(k) = y(0)$ for all positive k.

The situation is a little more complicated for other choices of $x(0)$. To understand the more general situation we need to introduce the idea of the *cyclicity* of A. We begin with the communication graph G_A. An *elementary circuit* of G_A is a circuit which, when considered as a subgraph of G_A, consists of a set of vertices each with in-degree and out-degree equal to one (that is we are only allowed to visit each vertex once). The length of an elementary circuit is the number of vertices it contains.

The cyclicity of a strongly connected graph G_A, is the greatest common divisor of the lengths of all its elementary circuits. In particular, we note that if the elementary cycles have lengths which are co-prime, then the cyclicity is equal to one. If the graph consists of more than one strong component, its cyclicity is the least common multiple of the cyclicities of its strong components.

Clearly the cyclicity is a topological property of the graph since it depends only on the circuits and their lengths. The matrix A contains more information than this since

[5]We will return to this point, but it will become clear that since the underlying timing dependency graph is periodic then if G_A is weakly connected it is also strongly connected. Conversely, if G_A is not weakly connected then the problem decomposes into several independent problems.

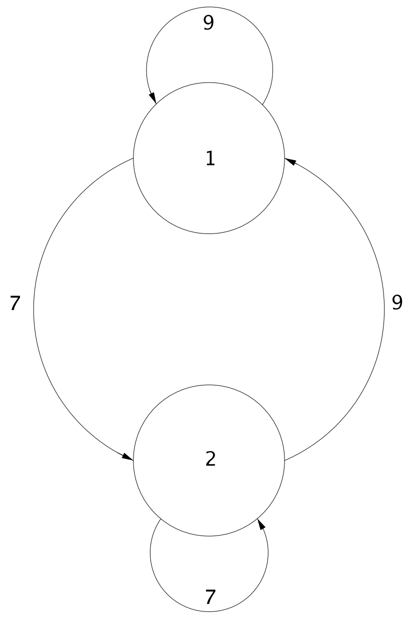

FIGURE 4. The communication graph corresponding to the matrix given in equation (6).

it associates with each edge of G_A a number representing the delay. This information can be captured by considering the *critical graph* associated with G_A. The critical circuits of G_A are those elementary circuits with maximum mean delay, the critical graph associated with G_A is the subgraph consisting of vertices and directed edges found in the critical circuits of G_A. The cyclicity of the matrix A, denoted $\sigma(A)$ is the cyclicity of the critical graph associated with G_A.

As an example consider:

$$A = \begin{pmatrix} 9 & 9 \\ 7 & 7 \end{pmatrix}. \tag{6}$$

The corresponding communication graph is shown in Figure 4. The graph has two kinds of elementary circuit: the self-loops (delays 9 and 7; length 1) and the cycle involving both vertices (delay 16; length 2). The eigenvalue of A in this example is therefore 9, and hence

$$\hat{A} = \begin{pmatrix} 0 & 0 \\ -2 & -2 \end{pmatrix}.$$

Moreover, the only critical circuit of the graph is the self-loop with delay 9. The critical graph therefore consists of the vertex labelled "1" in Figure 4 together with the self-loop. The cyclicity of this critical graph, and therefore of A, is one.

The following theorem tells us about the asymptotic behaviour of $x(k)$ or $y(k)$ with general initial conditions (see for example [9])

Theorem 3. *Let G_A be strongly connected and let the matrix A have eigenvalue λ and cyclicity $\sigma(A)$. Then there exists a positive integer K such that*

$$A^{\otimes(k+\sigma(A))} = \lambda^{\otimes\sigma(A)} \otimes A^{\otimes k}$$

for all $k \geq K$.

If we consider $\hat{A}$, then for sufficiently large k

$$\hat{A}^{\otimes(k+\sigma(A))} = \hat{A}^{\otimes k}$$

or, using equation (5),

$$\begin{aligned} y(k+\sigma(A)) &= \hat{A}^{\otimes(k+\sigma(A))} \otimes y(0) \\ &= \hat{A}^{\otimes k} \otimes y(0) \\ &= y(k). \end{aligned}$$

Thus asymptotically $y(k)$ is periodic with period $\sigma(A)$.

Returning to our small example we see that, given an arbitrary choice of $y(0)$, for $k \geq 1$ the iterates $y(k)$ are constant, that is, $y(k)$ is periodic with period 1, which is equal to the cyclicity of A, as required by the theorem:

$$y(1) = \begin{pmatrix} 0 & 0 \\ -2 & -2 \end{pmatrix} \otimes \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \max\{u,v\} \\ \max\{u,v\} - 2 \end{pmatrix},$$

$$y(2) = \begin{pmatrix} 0 & 0 \\ -2 & -2 \end{pmatrix} \otimes \begin{pmatrix} \max\{u,v\} \\ \max\{u,v\} - 2 \end{pmatrix} = \begin{pmatrix} \max\{u,v\} \\ \max\{u,v\} - 2 \end{pmatrix}.$$

6. Petri nets

So far in this paper, a timing dependency graph has been taken to be an infinite Hasse diagram having a periodic structure or, equivalently, as a finite graph whose circuit structure captures the repetitive periodic form. Here we shall make this relationship more precise by introducing a kind of finite automaton which can be inferred directly from the finite form of the timing dependency graph and which can be thought of as being the generator of the infinite Hasse diagram. Generally such automata are known as *Petri nets* [14].

A Petri net is a directed, bipartite graph. The two classes of vertex of a Petri net are called "places" (which graphically we shall represent as circles) and "transitions" (which are drawn as vertical lines). The states of a Petri net correspond to distributions of tokens (represented as dots) across the places of the net. A given distribution of tokens is referred to as a "marking". Every transition has a set of directed edges incident to it from a set of input places, and a set of edges from it to output places. The dynamics of the Petri net follow from the rule that a transition can only fire if all of its input places have a token. When a transition fires, one token is removed from each of its input places and added to each of its output places[6].

The connection between Petri nets and timing dependency graphs can be made by making the identification shown in Figure 5: we identify the time at which a signal becomes valid with the time at which the corresponding transition fires, and the weights on the edges of the timing dependency graph (delays associated with linking logic) with waiting times at the places of the Petri net.

Returning to the example shown in Figure 3, we can draw the corresponding Petri net using these rules, the result is shown in Figure 6. Additionally, Figure 6 shows a possible initial marking for the system. Given this marking, we see that, initially, either of the transitions T_1 or T_2 can fire. Let us assume that it is T_1. Then the next marking will be as shown in Figure 7. Figures 8 and 9 show the subsequent evolution

[6]There is no suggestion here that tokens are conserved. In particular, the number of input places need not equal the number of output places.

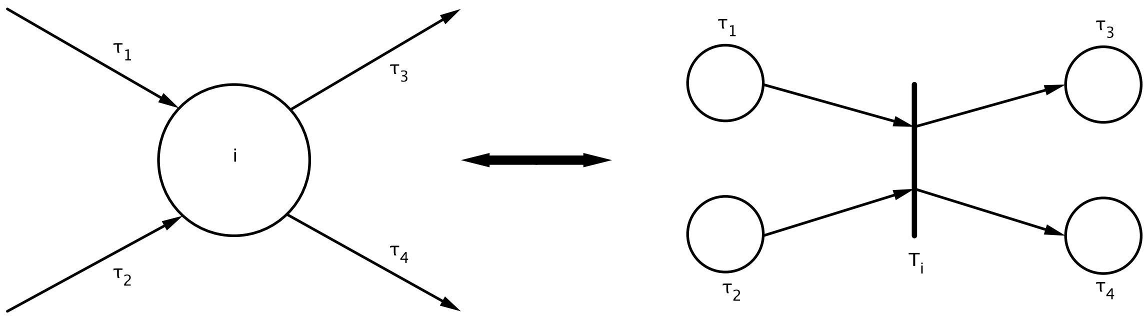

Figure 5. The relationship between elements of the timing dependency graph and the corresponding Petri net representation.

of the Petri net until the marking returns to the initial one. We note that in this case the order in which T_1 and T_2 fire has little effect on the overall dynamics.

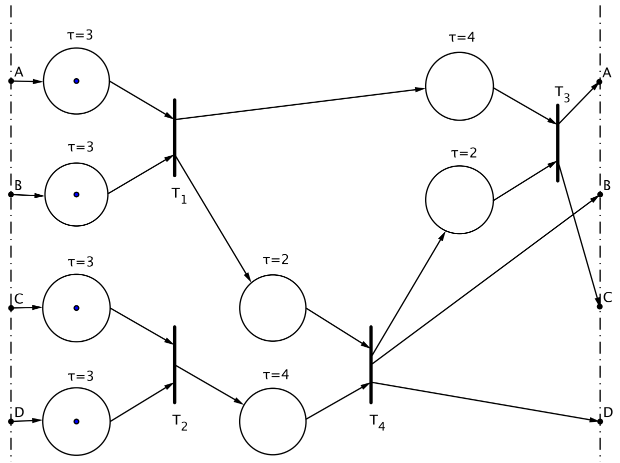

Figure 6. The Petri net corresponding to the timing dependency graph shown in Figure 3. This should be imagined drawn on the surface of a cylinder with the dash-dotted lines identified so that the pairs of points labelled A, B, C and D coincide. A possible initial marking is shown.

7. Timed event graphs and their analysis

7.1. **Timed event graphs.** Petri nets derived from timing dependency graphs by the translation rules given above have a particular structure; all places have exactly one upstream and one downstream transition. A Petri net having this property is known as a *timed event graph*. The number of tokens in each circuit of a timed event graph is conserved [9] and—importantly for the problem we are studying—the timing of events can be written as a linear process in the max-plus semi-ring.

7.2. **A worked example.** In the rest of this section we shall consider the example first introduced in Figure 3 and whose evolution as a Petri net is illustrated in Figures 6 to 9. The evolution of this system will be characterized by a vector-valued

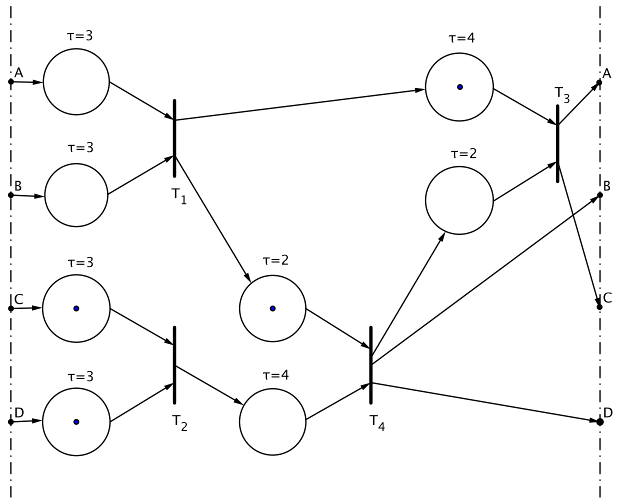

FIGURE 7. The Petri net corresponding to the timing dependency graph shown in Figure 3. The marking shown follows from that in Figure 6 assuming that transition T_1 has fired.

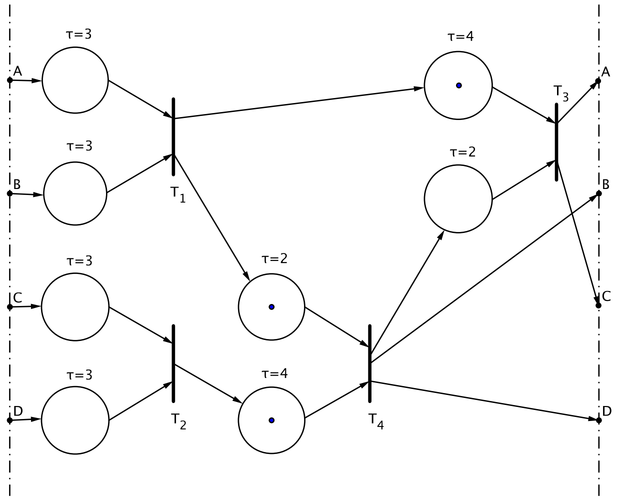

FIGURE 8. The Petri net corresponding to the timing dependency graph shown in Figure 3. The marking shown follows from that in Figure 7 (transition T_2 has fired).

sequence $x : \mathbb{N} \to \overline{\mathbb{R}}^n$, where $x_i(k)$ is the time at which signal i becomes valid for the kth time. In terms of the Petri net, $x_i(k)$ is the time that the transition T_i fires for the kth time. In our general setting, n is the number of signals we are keeping track of and hence also the size of the matrix of delay times and the vector x; in the example $n = 4$.

It is straightforward to conclude from Figures 6 to 9 that the initial marking shown in Figure 6 is recurrent. Let us imagine that we know $x_3(k)$ and $x_4(k)$, the times at which T_3 and T_4 fire for the kth time to give the marking shown in Figure 6 for the $(k+1)$th time. It follows that

$$\begin{aligned} x_1(k+1) &= \max\{x_3(k)+3, x_4(k)+3\}, \\ x_2(k+1) &= \max\{x_3(k)+3, x_4(k)+3\}, \end{aligned}$$

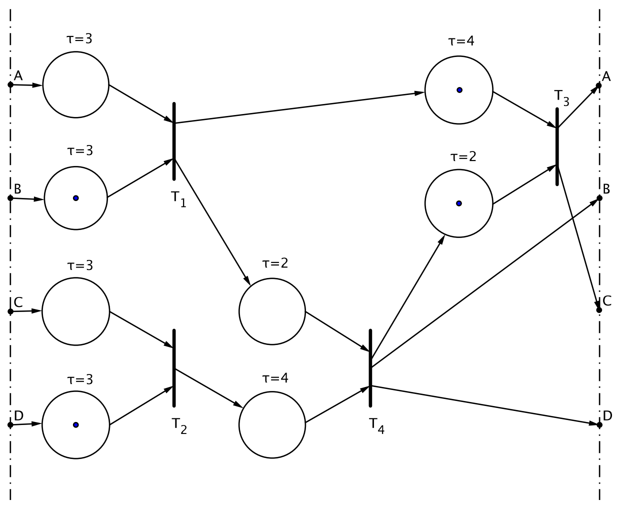

FIGURE 9. The Petri net corresponding to the timing dependency graph shown in Figure 3. The marking shown follows from that in Figure 8 (transition T_4 has fired). Note that given this marking the next transition to fire must be T_3. Following this, the marking is again that shown in Figure 6

which in max-plus notation can be written as

$$\begin{aligned} x_1(k+1) &= (3 \otimes x_3(k)) \oplus (3 \otimes x_4(k)), \\ x_2(k+1) &= (3 \otimes x_3(k)) \oplus (3 \otimes x_4(k)). \end{aligned}$$

Similarly we see that

$$\begin{aligned} x_4(k) &= \max\{x_1(k)+2, x_2(k)+4\}, \\ x_3(k) &= \max\{x_1(k)+4, x_4(k)+2\}, \end{aligned}$$

which is written as follows in max-plus notation

$$\begin{aligned} x_4(k) &= (2 \otimes x_1(k)) \oplus (4 \otimes x_2(k)), \\ x_3(k) &= (4 \otimes x_1(k)) \oplus (2 \otimes x_4(k)). \end{aligned}$$

We can combine these results into a system of linear equations which implicitly gives the evolution of x:

$$x(k) = (A_0 \otimes x(k)) \oplus (A_1 \otimes x(k-1)), \tag{7}$$

where

$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 4 & \varepsilon & \varepsilon & 2 \\ 2 & 4 & \varepsilon & \varepsilon \end{pmatrix}, A_1 = \begin{pmatrix} \varepsilon & \varepsilon & 3 & 3 \\ \varepsilon & \varepsilon & 3 & 3 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}.$$

We will show later that the general time borrowing problem results in an evolution equation in the form of equation (7). The details of a given problem are encoded in the form of the matrices A_0 and A_1. A comparison of A_0 and A_1 with A defined in equation (2)—the matrix of delay times for this example—shows that they are closely related. In fact $A = A_0 \oplus A_1$. The derivation given above has effectively decomposed A into two parts by taking account of the dependence of x on the variable k.

Although it is an implicit equation for the evolution of x, equation (7) is in the form of equation (3) and can be solved using the Kleene star of A_0 to give the following explicit dynamical system

$$x(k) = A_0^* \otimes A_1 \otimes x(k-1).$$

The communication graph G_{A_0} is the subgraph of that shown in Figure 3 obtained by removing the right hand pair of vertices labelled 1 and 2 (and the edges incident to them). We see that the result is an acyclic graph where the maximum path length is 2. According to Lemma 2 the Kleene star of A_0 is given by the sum

$$A_0^* = id \oplus A_0 \oplus A_0^{\otimes 2},$$

so

$$A_0^* = \begin{pmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 0 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 0 \end{pmatrix} \oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 4 & \varepsilon & \varepsilon & 2 \\ 2 & 4 & \varepsilon & \varepsilon \end{pmatrix} \oplus \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 4 & 6 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix} = \begin{pmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon \\ 4 & 6 & 0 & 2 \\ 2 & 4 & \varepsilon & 0 \end{pmatrix}.$$

Using this we get the following linear evolution equation

$$x(k) = \begin{pmatrix} \varepsilon & \varepsilon & 3 & 3 \\ \varepsilon & \varepsilon & 3 & 3 \\ \varepsilon & \varepsilon & 9 & 9 \\ \varepsilon & \varepsilon & 7 & 7 \end{pmatrix} \otimes x(k-1).$$

This reduces to a skew product dynamical system with the base dynamics given by

$$\begin{pmatrix} x_3(k) \\ x_4(k) \end{pmatrix} = \begin{pmatrix} 9 & 9 \\ 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} x_3(k-1) \\ x_4(k-1) \end{pmatrix} \tag{8}$$

and the following additional relation giving the behaviour of the other two components of x:

$$\begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix} \otimes \begin{pmatrix} x_3(k-1) \\ x_4(k-1) \end{pmatrix}.$$

To complete the specification of the problem we need some initial data. If we look at Figures 6 to 9 and assume that $(x_1(1), x_2(1)) = (0, 0)$, for example, then

$$\begin{pmatrix} x_3(1) \\ x_4(1) \end{pmatrix} = \begin{pmatrix} 6 \\ 4 \end{pmatrix}.$$

We have already analysed the asymptotics of equation (8) as an example (equation (6)) in section 5.5. There we found that, following an initial transient of one time step, the growth of x is determined by the eigenvalue, which is 9. In particular, given the initial condition assumed here we have:

$$\begin{pmatrix} x_3(2) \\ x_4(2) \end{pmatrix} = \begin{pmatrix} 9 & 9 \\ 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 6 \\ 4 \end{pmatrix} = \begin{pmatrix} 15 \\ 13 \end{pmatrix},$$

$$\begin{pmatrix} x_3(3) \\ x_4(3) \end{pmatrix} = \begin{pmatrix} 9 & 9 \\ 7 & 7 \end{pmatrix} \otimes \begin{pmatrix} 15 \\ 13 \end{pmatrix} = 9 \otimes \begin{pmatrix} 15 \\ 13 \end{pmatrix}.$$

7.3. **Remarks.** The first remark that should be made is that the eigenvalue of the matrix $A_0^* \otimes A_1$ really does tell us about the way that the system should be clocked because it is the maximum delay of all the elementary circuits of the timing dependency graph shown in Figure 3. In contrast, as it was remarked in section 5.3.1, the eigenvalue of A gives the maximum *average* delay of the circuits of the graph.

There is a nice interpretation of the matrix product $A_0^* \otimes A_1$ which gives a rationale to this. We know that A_0^* gives the maximum lengths of paths through the acyclic part of the timing dependency graph, and A_1 gives the delays associated with attaching this to the cyclic nodes. In particular, $(A_0^* \otimes A_1)_{ij}$ is the maximum delay associated with paths which leave cyclic node j and arrive at cyclic node i without passing through any other cyclic nodes (recall that the cyclic nodes are an antichain which separates the timing dependency graph).

7.4. **A slightly different example.** Figure 10 shows the timing dependency graph for a different example, with the corresponding Petri net representation, with a choice of initial marking, shown in Figure 11. This example has the form of a two stage asynchronous FIFO buffer, but we have wrapped it around to indicate the connection with an external clock signal. It can be thought of as a two-stage pipeline process which has much in common with the example shown in Figure 1. The mechanics

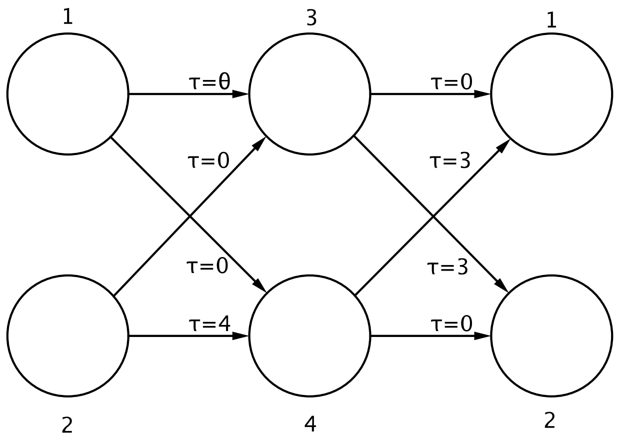

FIGURE 10. Another example of a graph of timing dependencies for a periodically clocked system with asynchronous processes. The processes are represented by numbered discs and the timings are given by the edge labels. Note that the leftmost and rightmost processes are the same.

of deriving the max-plus equations giving the timings of the process are essentially those described in the previous example. Again, we find a system of linear equations which implicitly give the evolution of x:

$$x(k) = (A_0 \otimes x(k)) \oplus (A_1 \otimes x(k-1)),$$

but in this case

$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \theta & 0 & \varepsilon & \varepsilon \\ 0 & 4 & \varepsilon & \varepsilon \end{pmatrix}, A_1 = \begin{pmatrix} \varepsilon & \varepsilon & 0 & 3 \\ \varepsilon & \varepsilon & 3 & 0 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}.$$

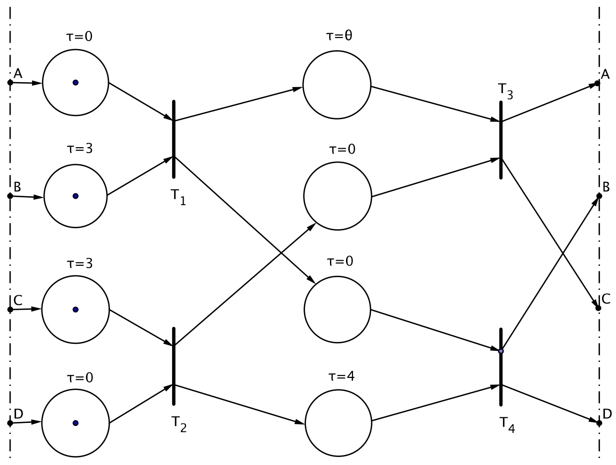

FIGURE 11. The Petri net corresponding to the timing dependency graph shown in Figure 10. This should be imagined drawn on the surface of a cylinder with the dash-dotted lines identified so that the pairs of points labelled A, B, C and D coincide. A possible initial marking is shown.

Note that in this example we have an undetermined parameter, $\theta \geq 0$. We shall investigate changes in the behaviour of the system as θ is varied—we shall treat θ as a kind of bifurcation parameter.

It can be seen from a simple consideration of the block structure of A_0 that its Kleene star consists of just two terms:

$$A_0^* = id \oplus A_0,$$

so that

$$A_0^* = \begin{pmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon & \varepsilon \\ \theta & 0 & 0 & \varepsilon \\ 0 & 4 & \varepsilon & 0 \end{pmatrix}.$$

The explicit form of the evolution equation for $x(k)$ is then

$$\begin{aligned} x(k) &= A_0^* \otimes A_1 \otimes x(k-1) \\ &= \begin{pmatrix} \varepsilon & \varepsilon & 0 & 3 \\ \varepsilon & \varepsilon & 3 & 0 \\ \varepsilon & \varepsilon & 3 \oplus \theta & 3 \otimes \theta \\ \varepsilon & \varepsilon & 7 & 4 \end{pmatrix} \otimes x(k-1), \end{aligned}$$

which again reduces to a simpler skew product system:

$$\begin{pmatrix} x_3(k) \\ x_4(k) \end{pmatrix} = \begin{pmatrix} 3 \oplus \theta & 3 \otimes \theta \\ 7 & 4 \end{pmatrix} \otimes \begin{pmatrix} x_3(k-1) \\ x_4(k-1) \end{pmatrix} \tag{9}$$

and

$$\begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} = \begin{pmatrix} 0 & 3 \\ 3 & 0 \end{pmatrix} \otimes \begin{pmatrix} x_3(k-1) \\ x_4(k-1) \end{pmatrix}.$$

The latter equations are particularly transparent, since they state that signal 1 becomes valid three time units after signal 4 or immediately after signal 3, whichever is the later. Similarly, signal 2 becomes valid three time units after signal 3 or after signal 4 whichever is the later. This is evident in both Figure 10 and Figure 11.

The asymptotic behaviour of the base system given in equation (9) is more interesting and depends qualitatively on the value of θ. To understand this we need to calculate the cyclicity of the matrix

$$A_\theta = \begin{pmatrix} 3 \oplus \theta & 3 \otimes \theta \\ 7 & 4 \end{pmatrix}.$$

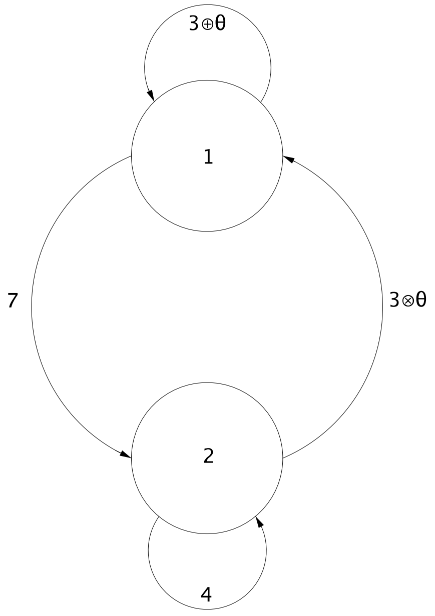

FIGURE 12. The communication graph corresponding to the matrix A_θ.

The communication graph for A_θ is shown in Figure 12. As with the earlier example, the graph has two kinds of elementary circuit: self-loops (delays 4 and $3 \oplus \theta$; length 1) and the cycle involving both vertices (delay $10 + \theta$; length 2). The eigenvalue of A_θ is therefore $\max\{4, 3 \oplus \theta, (10 + \theta)/2\} = \max\{\theta, (10 + \theta)/2\}$. So, if $\theta \in [0, 10]$, the eigenvalue is $(10 + \theta)/2$ and when $\theta \geq 10$ the eigenvalue is θ. The same calculation shows that when $\theta \in [0, 10)$ the cyclicity of A_θ is two (the critical graph contains only the cycle of length 2 connecting vertices 1 and 2), and when $\theta \geq 10$ the cyclicity of A_θ is one. (When $\theta > 10$ the critical graph is vertex 1 together with the self-loop. The particular case of $\theta = 10$ also has the cyclicity of A_θ equal to one; in this case the critical graph has both the length two cycle and the self loop.)

Using Theorem 3 we can describe the asymptotic behaviour of the vector of times $x(k)$. When $\theta \geq 10$, then there is a sufficiently large positive integer K such that

$$A_\theta^{\otimes(k+1)} = \theta \otimes A_\theta^{\otimes k}$$

for all $k \geq K$. That is, the times increase linearly as a function of k:

$$\begin{pmatrix} x_3(k) \\ x_4(k) \end{pmatrix} = \theta \otimes \begin{pmatrix} x_3(k-1) \\ x_4(k-1) \end{pmatrix}.$$

This is the same qualitative behaviour as in the example considered earlier. If, as in that case, we choose the initial data to be $(x_1(1), x_2(1)) = (0,0)$, then

$$\begin{pmatrix} x_3(1) \\ x_4(1) \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

and therefore

$$\begin{pmatrix} x_3(2) \\ x_4(2) \end{pmatrix} = \begin{pmatrix} \theta & 3 \otimes \theta \\ 7 & 4 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 3 \end{pmatrix} = \begin{pmatrix} 6 \otimes \theta \\ 10 \end{pmatrix}.$$

Iterating twice more shows that in this case when $k = 4$ the vectors have settled into the expected pattern:

$$\begin{pmatrix} x_3(3) \\ x_4(3) \end{pmatrix} = \begin{pmatrix} \theta & 3 \otimes \theta \\ 7 & 4 \end{pmatrix} \otimes \begin{pmatrix} 6 \otimes \theta \\ 10 \end{pmatrix} = \theta \otimes \begin{pmatrix} 6 \otimes \theta \\ 13 \end{pmatrix},$$

$$\begin{pmatrix} x_3(4) \\ x_4(4) \end{pmatrix} = \theta \otimes \begin{pmatrix} \theta & 3 \otimes \theta \\ 7 & 4 \end{pmatrix} \otimes \begin{pmatrix} 6 \otimes \theta \\ 13 \end{pmatrix} = \theta^{\otimes 2} \otimes \begin{pmatrix} 6 \otimes \theta \\ 13 \end{pmatrix}.$$

If, on the other hand, θ is decreased so that $\theta \in [0, 10)$, Theorem 3 indicates a quite different asymptotic behaviour for the $x(k)$. In this case the cyclicity of the matrix is two, so that for a sufficiently large positive integer K,

$$A_\theta^{\otimes(k+2)} = \left(\frac{10+\theta}{2}\right)^{\otimes 2} \otimes A_\theta^{\otimes k}$$

for all $k \geq K$. In this case the increase of the times as a function of k has a period-2 modulation and the uniform linear increase in the times is only observed if we look at every other value of k:

$$\begin{pmatrix} x_3(k) \\ x_4(k) \end{pmatrix} = \left(\frac{10+\theta}{2}\right)^{\otimes 2} \otimes \begin{pmatrix} x_3(k-2) \\ x_4(k-2) \end{pmatrix}. \tag{10}$$

If we choose the initial data as above, and assume that $\theta = 9$ in order to have a concrete example, then we get the following sequence of times

$$\begin{pmatrix} 3 \\ 3 \end{pmatrix} \mapsto \begin{pmatrix} 15 \\ 10 \end{pmatrix} \mapsto \begin{pmatrix} 24 \\ 22 \end{pmatrix},$$

which settles down to the expected period-2 behaviour

$$\begin{pmatrix} 24 \\ 22 \end{pmatrix} \mapsto \begin{pmatrix} 34 \\ 31 \end{pmatrix} \mapsto 19 \otimes \begin{pmatrix} 24 \\ 22 \end{pmatrix}.$$

The interpretation of this example is that the question of how to clock the device depends rather sensitively on the value of θ. When $\theta \geq 10$, following an initial transient, the minimum period of the clock that we can use to time the device is θ. On the other hand, if $\theta \in [0, 10)$ then although the eigenvalue is $(10 + \theta)/2$, this is not a useful clock period. In the numerical example, when $\theta = 9$, although the timings advance by $19 = 2 \times 9.5$ in two steps, after one step we see one process takes longer than 9.5 while the other takes less (for example, $x_3(4) - x_3(3) = 10$ and $x_4(4) - x_4(3) = 9$). Thus, in this example, the minimum suitable clock period is 10 (which is the shortest time it would take to complete all processes), unless the system is implemented using something like the transparent latch technology described in section 1. Using such technology we might arrange for the signals to be timed every two periods of a clock which has period 9.5. In each half of this two cycle period, a slow process borrows time from a corresponding faster process.

8. The General Case

So far, the theory has been illustrated through small examples. In this section we will establish that the structure we have identified is actually preserved in more general cases. We will do this graphically using block representations of the matrices involved. However, it should be clear that this is just a shorthand way of describing rigorous manipulations in the max-plus linear algebra.

Returning to the general timing dependency graph shown in Figure 2, we imagine translating this into a Petri net using the correspondence shown in Figure 5. The structure of the resulting Petri net is obvious: the region marked "acyclic timing dependency graph" in Figure 2 becomes an acyclic Petri net; for each cyclic node there is a transition; for each edge incident to a cyclic node there is a place with single edge which connects it to the transition and single edge incident from a transition within the acyclic Petri net. The overall periodic structure of the timing dependency graph is reflected in the cylindrical structure of the Petri net achieved by identifying the transitions associated with the cyclic nodes. A comparison between, say, Figure 3 and Figure 6 shows how this works. In this example the acyclic part of the timing dependency graph is vertices 3 and 4 and the edge connecting them. In the Petri net, the corresponding transitions T_3 and T_4 are connected via a single place. The cyclic nodes are vertices 1 and 2 in Figure 3. For convenience, in the Petri net we have lined up the input places to corresponding transitions, T_1 and T_2 to the left. It will be assumed that an initial marking with each of these places given one token will not lead to a "deadlock" in which the Petri net becomes frozen in the same state.

A Petri net derived from a timing dependency graph is a timed event graph (see section 7.1), and can therefore be expressed as a linear max-plus system. In section 7.2, we derived such an expression—equation (7)—for our first example. The form of this equation is general for the problems that we will study—that is, timed event graphs with an initial marking which has no more than one token per place (see, for example, [9, Chapter 7])—and so it remains to establish what we know about the structures of the matrices A_0 and A_1. As with the simple examples given earlier, these matrices are a decomposition, based on the initial marking of the Petri net, of the matrix of delay times.

Let the total number of transitions in the Petri net be n, and of these, let there be n_c transitions corresponding to the cyclic nodes. The places of the Petri net will each have an associated delay time. Since each place of a timed event graph has a unique upstream transition and a unique downstream transition we can think of the delay times as being organised as an $n \times n$ matrix, τ where τ_{ij} is the delay associated with the place leading from T_j to T_i. Recall that the linear system will give the evolution of a vector-valued time series $x : \mathbb{N} \to \overline{\mathbb{R}}^n$, where $x_i(k)$ is the time at which signal i becomes valid for the kth time. This has the form

$$x(k) = (A_0 \otimes x(k)) \oplus (A_1 \otimes x(k-1)). \tag{11}$$

The matrices A_0 and A_1 have the following structure (see [9, Section 7.2])

$$(A_m)_{ij} = \begin{cases} \tau_{ij} & \text{if in the initial marking, there are exactly } m \text{ tokens} \\ & \text{in the place leading from } T_j \text{ to } T_i, \\ \varepsilon & \text{otherwise.} \end{cases}$$

If we label the components of $x \in \overline{\mathbb{R}}^n$ so that the first n_c correspond to the cyclic nodes, then (as we shall see in a moment) we obtain a simple block structure for A_0 and A_1:

$$A_0 = \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline b_0 & B_0 \end{array}\right), A_1 = \left(\begin{array}{c|c} \varepsilon & b_1 \\ \hline \varepsilon & \varepsilon \end{array}\right).$$

Firstly, it is easy to see that both A_0 and A_1 have an $n_c \times n_c$ block of ε entries in the upper left corner, which arises because the cyclic nodes have no mutual interdependencies—they were chosen to be an antichain of the original timing dependency graph.

By definition, the entries of A_0 not equal to ε correspond to the waiting times at places containing no tokens in the initial marking. Since the only places containing tokens in the initial marking are those leading to transitions corresponding to the cyclic nodes, it is easy to see that the $(n - n_c) \times (n - n_c)$ block B_0 in the lower right corner contains the delay times within the acyclic component of the graph, whilst the $(n - n_c) \times n_c$ block b_0 in the lower left corner contains the delays on edges from the cyclic nodes to the acyclic component. As already mentioned, the $n_c \times n_c$ block in the upper left corner of the matrix is null because there are no connections between the cyclic nodes. The remaining $n_c \times (n - n_c)$ block in the upper right corner is null because the initial marking has a token in the places leading from the acyclic component to the cyclic nodes (and hence these delays are found in A_1).

By similar reasoning we see that the only block of A_1 which is not null is the $n_c \times (n - n_c)$ block b_1 in the upper right corner, which contains the delays associated with the places leading from the acyclic component to the cyclic nodes (since these each have a single token in the initial marking).

Having established the form of the implicit equation (11) in this more general setting, we need to transform it to an explicit form as was done in the examples. To do this we need to compute the Kleene star of A_0, and to this end we need to investigate the form of powers of A_0. Let us first consider the form of the product $A_0 \otimes A_0$. The dimensions of the blocks are such that we can reduce the whole product to a set of block products:

$$A_0 \otimes A_0 = \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline b_0 & B_0 \end{array}\right) \otimes \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline b_0 & B_0 \end{array}\right) = \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline B_0 \otimes b_0 & B_0 \otimes B_0 \end{array}\right).$$

Notice that the top n_c rows of $A_0 \otimes A_0$ are all null. Moreover, by a simple induction, we can easily compute the block form of arbitrary powers of A_0:

$$A_0 \otimes A_0^{\otimes j} = \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline b_0 & B_0 \end{array}\right) \otimes \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline B_0^{\otimes(j-1)} \otimes b_0 & B_0^{\otimes j} \end{array}\right) = \left(\begin{array}{c|c} \varepsilon & \varepsilon \\ \hline B_0^{\otimes(j)} \otimes b_0 & B_0^{\otimes(j+1)} \end{array}\right).$$

The important point to note about the structure of $A_0^{\otimes j}$ is that the lower right, $(n - n_c) \times (n - n_c)$ diagonal block is $B_0^{\otimes j}$ and the $(n - n_c) \times n_c$ block is $B_0^{\otimes(j-1)} \otimes b_0$. When we form the Kleene star of A_0, we need to compute the max-plus sum of the corresponding blocks, ranging over all powers j. Thus the lower right hand block of A_0^* will be the Kleene star of B_0,

$$B_0^* = \bigoplus_{j=1}^{l_*} B_0^{\otimes j} \oplus id,$$

where l_* is the longest directed path in the acyclic component. Similarly, the lower left hand block of A_0^* will be given by

$$B_0^* \otimes b_0 = \bigoplus_{j=1}^{l_*+1} B_0^{\otimes(j-1)} \otimes b_0.$$

In other words,

$$A_0^* = \bigoplus_{j=0}^{l_*+1} A_0^{\otimes j} = \left(\begin{array}{c|c} id & \varepsilon \\ \hline B_0^* \otimes b_0 & B_0^* \end{array}\right).$$

The final step in this process is to form the product $A_0^* \otimes A_1$. This, again, can be done exploiting the block structures of the matrices, giving

$$A_0^* \otimes A_1 = \left(\begin{array}{c|c} id & \varepsilon \\ \hline B_0^* \otimes b_0 & B_0^* \end{array}\right) \otimes \left(\begin{array}{c|c} \varepsilon & b_1 \\ \hline \varepsilon & \varepsilon \end{array}\right) = \left(\begin{array}{c|c} \varepsilon & b_1 \\ \hline \varepsilon & B_0^* \otimes b_0 \otimes b_1 \end{array}\right).$$

The equations governing the evolution of $x(k)$ are then given by

$$x(k) = A_0^* \otimes A_1 \otimes x(k-1). \tag{12}$$

Since the first n_c columns of $A_0^* \otimes A_1$ are null, it is possible to reduce this system of equations. To do this, we shall think of $x \in \overline{\mathbb{R}}^n = \overline{\mathbb{R}}^{n_c} \times \overline{\mathbb{R}}^{n-n_c}$ and write $x = (x^{(c)} : \tilde{x})$, where $x^{(c)} \in \overline{\mathbb{R}}^{n_c}$ gives the timing of the cyclic nodes and $\tilde{x} \in \overline{\mathbb{R}}^{n-n_c}$ gives the timing of the rest. Substituting this into equation (12) and using the block matrix form of $A_0^* \otimes A_1$ gives the following system of equations

$$x^{(c)}(k) = b_1 \otimes \tilde{x}(k-1), \tag{13}$$
$$\tilde{x}(k) = \tilde{A} \otimes \tilde{x}(k-1), \tag{14}$$

where

$$\tilde{A} = B_0^* \otimes b_0 \otimes b_1. \tag{15}$$

In effect, this has reduced the dynamics by eliminating the cyclic nodes. Mathematically, equations (13) and (14) have the form of a skew product. The dynamics takes place in $\overline{\mathbb{R}}^{n-n_c}$ and is a linear evolution governed by the matrix $\tilde{A}$. The timings of the cyclic nodes are obtained from the orbit $\{\tilde{x}(k) : k \in \mathbb{N}\}$ by applying the fixed linear mapping $b_1 : \overline{\mathbb{R}}^{n-n_c} \to \overline{\mathbb{R}}^{n_c}$ to each $\tilde{x}(k)$.

Looking back at the examples described earlier, it is clear that the rather special-looking block structures which arose there were actually a consequence of the structure of our basic problem rather than a peculiarity of the examples. As with those examples, the timing of the system can be explained entirely in terms of the eigenvalue and cyclicity of the matrix $\tilde{A}$.

9. Numerical Methods

The ARM6 example taken from [7] and shown in section 1 is greatly simplified. The full design had of the order of 100 nodes, 20 to 25 cyclic nodes with the timing dependency graph having around 300 edges. Clearly problems of this size require numerical methods. Here we shall give a brief review of some of the available algorithms and the estimates of how they scale with problem size. This is not intended to be an exhaustive review, our main source of information is the two texts we have used extensively in writing this paper [4] and [9]. Our main point will be that numerical algorithms do exist. Indeed, the numerical analysis of linear algebra problems over the max-plus semi-ring—frequently called *tropical* problems—is an active field of research.

The kinds of numerical computations that we need for this work involve matrix multiplication, the computation of the Kleene star and techniques for finding the eigenvalue and cyclicity of matrices.

The product of a pair of $n \times n$ matrices uses $O(n^3)$ $\oplus$ and $\otimes$ operations (that is, comparisons and additions). This is the basic expectation, it is as if we were to get the computer to reproduce the process that we might do by hand. In conventional arithmetic there are faster ways to do matrix multiplication which use $O(n^\alpha)$ ($2 < \alpha < 3$) floating point operations, but it appears that nothing analogous has been found for max-plus computations [4]. Nevertheless, we might expect that the practical tricks for speeding up matrix multiplication using parallelisation will carry over to the max-plus case.

Given this limitation on matrix multiplication, it seems that the computation of the Kleene star of an $n \times n$ matrix would take $O(n^4)$ operations. However, a dynamic programming algorithm (the so-called Floyd-Warshall algorithm—see for example [2]) for finding the longest paths in a weighted graph can be used to do compute the Kleene star in $O(n^3)$ operations.

Of the commonly-used algorithms for computing the eigenvalue of a max-plus matrix, perhaps the best known is Karp's algorithm [13, 4, 9]. This finds the maximum cycle mean of an $n \times n$ matrix in $O(n|E|)$ operations where $|E|$ is the number of edges of the communication graph associated with the matrix. Clearly this can be fast if the matrix is sparse, but, at worst, the number of operations scales as $O(n^3)$.

The limitation of Karp's algorithm is that although it finds the eigenvalue, it does not give a corresponding eigenvector (which is desirable if we can control the initial conditions and wish to obtain periodic behaviour). A method that does give both eigenvalue and eigenvector is the *power method* [9], which, as with its counterpart in standard linear algebra involves computing powers of the matrix in question. In effect, the power method corresponds to the naive approach to solving problems like equation (14), that is, iteration. Simple iteration eventually tells us all we need to know, that is, the asymptotic growth rate of the times (hence the eigenvalue) and any periodic modulation on the asymptotic growth (hence the cyclicity). The problem with simple iteration and the power method is that there are as yet no good estimators of the transient times which precede the desired asymptotic evolution. Actually, this limitation corresponds to a physical issue with the kind of digital devices we have in mind, since the transient times will correspond to the times that a device will take to settle down into its normal behaviour after it has been switched on.

10. Conclusion

The main conclusion of this work is that given a timing dependency graph representing the interaction of asynchronous processes within a periodically clocked system, it is possible to write down a max-plus matrix which determines the behaviour of the system. Moreover, the information required to find the minimum feasible period of the clock is contained in the eigenvalue and cyclicity of this matrix. Thus, the problem has been reduced to an algorithmic one, one that can be solved using suitable numerical methods.

It is interesting to compare this approach with the heuristic method for time borrowing designs described in [7][7]. We quote directly:

> *The algorithm to identify the critical paths begins by reducing the graph to one with just the cyclic nodes, removing all the inner detail by recursively searching the graph for the longest path from each input to each output. Then the reduced graph is repeated to form graphs of length 1 to N clock cycles, where N is the number of cyclic nodes. Finally each of these graphs is searched for the longest route from any input to its corresponding output, and the longest of these over all the repeated graphs, scaled by the number of clock cycles the graph represents, gives the minimum clock period for the design. Tracing this path back to the original graph yields the critical path.*

If we consider the definition of $\tilde{A}$ given in equation (15), many of the elements of the heuristic method are clear. The matrix b_1 contains the delays associated with edges leading from the acyclic component to the cyclic nodes and b_0, contains the delays on edges from the cyclic nodes to the acyclic component. The matrix element $(b_0 \otimes b_1)_{jk}$ is, therefore (using the definition of the matrix product) the maximum delay along all paths which leave the acyclic component from the kth node and enter the acyclic component again at the jth node going via any one of the cyclic nodes. Similarly, we know that the matrix element of Kleene star $(B_0^*)_{ij}$ gives the maximum delay of all paths through the acyclic component going from node i to node j. The interpretation of a general matrix element of $\tilde{A}$, is that $\tilde{A}_{ik}$ is the maximum delay of all possible paths from acyclic node k to acyclic node i where the paths pass through a single cyclic node. That is, all paths which cross one period of the timing dependency graph.

In effect, the communication graph of $\tilde{A}$ plays the role of the reduced graph described in the heuristic. Albeit a reduced graph based on the nodes of the acyclic component of the timing dependency graph, rather than the cyclic nodes. The analysis of $\tilde{A}$ in terms of its eigenvalue and cyclicity, provides a description of the form of powers of the matrix. For instance, $\tilde{A}^{\otimes p}$ gives the maximum delays associated with paths which go around p times. (In the heuristic this corresponds to the analysis of the repetitions of the reduced graph.) Thus the eigenvalue of $\tilde{A}$, gives the minimum clock period as found in the heuristic. The cyclicity gives the number of times the critical path goes around. The critical path itself can be identified by looking at the critical circuits of $\tilde{A}$.

The discrepancy between the two approaches—that the approach derived in this paper is based on the acyclic nodes whereas the heuristic approach is based on the cyclic nodes—turns out to be cosmetic. To see this consider equation (14) and expand $\tilde{A}$ using equation (15). Using equation (13) and associativity we have

$$\tilde{x}(k) = B_0^* \otimes b_0 \otimes x^{(c)}(k).$$

[7]Note that this is to be found on page 105 of the 1996 edition, but was taken out of the subsequent edition.

Multiplication from the left by b_1, using equation (13) again, now gives an evolution equation for $x^{(c)}(k)$

$$x^{(c)}(k+1) = b_1 \otimes B_0^* \otimes b_0 \otimes x^{(c)}(k).$$

Now we have an evolution determined by iteration of the matrix

$$A^{(c)} \overset{\text{def}}{=} b_1 \otimes B_0^* \otimes b_0, \tag{16}$$

which is $n_c \times n_c$, that is, we have a theory based on the cyclic nodes.

This equivalence is not as surprising as it might seem at first sight. It is a consequence of the overall periodicity of the problem. Indeed, if we take powers of these two matrices—as when generating $x^{(c)}(k)$ or $\tilde{x}(k)$ by iteration—we see that they are very closely related (semi-conjugate):

$$A^{(c)\otimes k} \otimes b_1 = b_1 \otimes \tilde{A}^{\otimes k}.$$

The result is, however, clear. We now have a theory based on the cyclic nodes, and the argument made above which interpreted the matrix elements of $\tilde{A}$ in terms of the heuristic, applies equally well to the matrix elements of $A^{(c)}$ (since the matrices are cyclic permutations of the same factors). We have, therefore, a mathematically rigorous approach to time borrowing which is exactly equivalent to the 'heuristic method' used by the early ARM designers.

References

[1] A. Asenov, S. Roy, R.A. Brown, G. Roy, C. Alexander, C. Riddet, C.Millar, B. Cheng, A.Martinez, N.Seoane, D.Reid, M.F. Bukhori, X.Wang, U.Kovac, *Advanced simulation of statistical variability and reliability in nano CMOS transistors*, Electron Devices Meeting, 2008. IEDM 2008. IEEE International.

[2] Jørgen Bang-Jensen, Gregory Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag, 2002.

[3] S. M. Burns, *Performance analysis of asynchronous systems*, PhD Thesis, 1990.

[4] Peter Butkovič, *Introduction to max-algebra*, American Institute of Mathematics, Palo Alto, 2008. (This text can be downloaded at `http://web.mat.bham.ac.uk/P.Butkovic/max.html`)

[5] Peter J. Cameron, *Combinatorics: Topics, Techniques, Algorithms*, Cambridge University Press, 1994

[6] R.A. Cuninghame-Green, *Minimax Algebra*, Lecture Notes in Economics and Mathematical Systems, **166**, Springer-Verlag, Berlin, 1979.

[7] Steve Furber, *ARM System Architecture*, Addison-Wesley, 1996.

[8] Jeremy Gunawardena, *Timing Analysis of Digital Circuits and the Theory of Min-Max Functions*, HP Labs Technical Report, 1994.

[9] Bernd Heidergott, Geert Jan Olsder, Jacob van der Woude, *Max Plus at Work*, Princeton Series in Applied Mathematics, 2006.

[10] M. Johnson, M. Kambites, *Multiplicative structure of* 2×2 *tropical matrices*, Linear Algebra Appl. (2010), doi:10.1016/j.laa.2009.12.030

[11] C. Mead, L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[12] Asit K. Mishra, Reetuparna Das, Soumya Eachempati, Ravi Iyer, N. Vijaykrishnan, Chita R. Das, *A case for dynamic frequency tuning in on-chip networks*, Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, ACM, New York, 2009.

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

[14] W. Riesig, *Petri Nets: An Introduction*, Monographs in Theoretical Computer Science, Springer-Verlag, 1985.

[15] Jian Ruan, Zhiying Wang, Kui Dai and Yong Li, *Latency Estimation of the Asynchronous Pipeline Using the Max-Plus Algebra*, ICCS 2007, Part IV, LNCS 4490, pp 251-258, 2007.

[16] Ruibing Lu and Cheng-kok Koh *Performance Analysis of Latency-Insensitive Systems*, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, Vol 25, No 3, March 2006.

[17] Karem A. Sakallah, , Trevor N. Mudge, Oyekunle A. Olukotun, *Analysis and Design of Latch-Controlled Synchronous Digital Circuits*, IEEE Transactions on Computer-Aided Design, Vol. II , No. 3, 1992.

[18] T. Szymanski and N. Shenoy, *Verifying Clock Schedules*, Digest of Technical Papers of the IEEE International Conference on Computer-Aided Design of Integrated Circuits, pages 124-131. IEEE Computer Society 1992.

[19] Baris Taskin, Ivan S. Kourtev, *Performance optimization of single-phase level-sensitive circuits using time borrowing and non-zero clock skew*, Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems, 111-118, 2002.

[20] Bill Teng and Jason H. Anderson, *Latch-Based Performance Optimization for FPGAs*, Proceedings of the 34th annual international symposium on 21st International COnference on Field Programmable Logic and Applications, 2011.

[21] Abhishek Tiwari, Smruti R. Sarangi, Josep Torrellas, *ReCycle: pipeline adaptation to tolerate process variation*, Proceedings of the 34th annual international symposium on Computer architecture ACM New York, 2007.

[22] O. Viro, *Dequantization of Real Algebraic Geometry on Logarithmic Paper*, arXiv:math/0005163v3 [math.AG] 6 Jun 2000.

School of Mathematics, Alan Turing Building, The University of Manchester, Oxford Road, Manchester M13 9PL
E-mail address: `david.broomhead@manchester.ac.uk`

School of Computer Science, Kilburn Building, University of Manchester, Oxford Road, Manchester, M13 9PL
E-mail address: `steve.furber@manchester.ac.uk`

School of Mathematics, Alan Turing Building, The University of Manchester, Oxford Road, Manchester M13 9PL
E-mail address: `Marianne.Johnson@maths.manchester.ac.uk`