Analysis of Periodic Clock Relations in Polychronous Systems

Hugo Metivier, Jean-Pierre Talpin, Thierry Gautier, and Paul Le Guernic

Abstract The *polychronous* (synchronous, multiclocked) language Signal is used for the design and analysis of reactive systems. For the purpose of modeling event-driven systems, we consider an extension of the polychronous model of computation of Signal with periodic equations denoted by ultimately periodic infinite words. These equations express periodic constraints on the signals of programs, that can be used to enrich the existing *clock calculus* of Signal. Thanks to this more powerful clock calculus, the communications between processes using periodic equations can be analysed to guarantee their correctness. In particular, the maximal size of buffers is formally evaluated. We illustrate the design of so-defined periodic systems using a 4-stroke engine example.

1 Introduction

While synchronous programming has extensively been applied to the design of control-intensive software for event-driven embedded systems [3], recent work [11, 5] has investigated extensions to symbolic calculus for synchrony in the aim of analyzing periodic systems and communications between them. Synchronous languages are appropriate specification formalisms to address the design of such architectures, as the otherwise symbolic model of time they support can be equipped with ad-hoc program analysis techniques to perform needed timing evaluation. As an example, polychrony, the synchronous multi-clocked model of computation of the data-flow specification formalism Signal, is dedicated to the specification of concurrent event-driven embedded software and for the main purpose of architecture exploration. Polychrony provides a discrete and partially ordered model of time that differs from the classical *synchronous hypothesis* where time is abstracted by totally ordered symbolic clocks. For the purpose of modeling periodic systems, we

H. Metivier (University of Rennes) · J-P. Talpin · T. Gautier · P. Le Guernic (INRIA)
Campus de Beaulieu, 35 042 Rennes Cedex France
e-mail: firstname.lastname@irisa.fr

consider an extension of the polychronous MoC with periodic clocks denoted by ultimately periodic infinite words. This defines a compositional specification structure to express periodic relations between signals. Applications are the design of periodic systems and the analysis of communication resources.

Related work Many approaches have been investigated for the design of hybrid hardware/software, event-driven/time-triggered, synchronous/asynchronous, embedded systems. Most of them are based on the principles of process networks and inherit from the pioneering work of Kahn [7]. Many examples could be cited, the Ptolemy [4] project, the Yapi [8] project, etc. While symbolic reasoning on process networks has essentially been developed for the purpose of embedded, control-dominated, software design and in the context of synchronous programming languages [3], such as Esterel, Lustre and Signal, analytical reasoning has mainly been studied in the context of high-performance, data-dominated systems.

Some approaches have arisen to combine concepts of both domains. 1) The first one is, to our knowledge, the work of Smarandache [11]. It aims at combining the multi-clocked synchronous model of computation of the language Signal with a topological model of the Alpha specification formalism [6]. Alpha manipulates convex polyhedral domains to describe high-performance, massively parallel algorithms over multi-dimensional data. Signal-Alpha [11] proposes a calculus of affine clock relations, associating symbolic signal clocks with affine functions: two periodic signals x, y are said in $(n, m, q - p)$-affine relation iff their respective clocks $\hat{x}, \hat{y}$ can be expressed as functions $\hat{x} = \{n.t + p \,|\, t \in \hat{z}\}$ and $\hat{y} = \{m.t + q \,|\, t \in \hat{z}\}$ of a common reference of discrete time $\hat{z}$ (m, n, p, q are integers). This yields a very expressive calculus for the specification and the analysis of time-triggered systems, while in fact most of the decidable and algorithmically affordable analysis concerns $(1, m, n)$-affine relations. 2) More recently, Cohen et al. [5] propose *an algebra of ultimately periodic clocks* to interpret synchronization in the synchronous language Lucid-synchrone. This yields the capability to model process networks: synchronous functions networked by bounded-buffering communication mechanisms. The algebra of periodic clock relations under consideration consists of associating a signal with a period described by a binary word (e.g. (01)). In turn, this defines a rich algebra in which a clock is itself the generator of an ideal consisting of any possible stretch of the generator (e.g. (0101), etc.). 3) The UML profile for *Modeling and Analysis of Real-Time and Embedded systems* (MARTE [1]) provides a general model of time in different aspects : physical/logical, dense/discrete, single/multiple. It offers basic operators and relations to combine timed events and clocks : subclocking, periodicity, etc. The calculus we propose could be a way of solving MARTE clock relations (in particular, periodic ones) in the case of discrete multiple time.

Contribution The results described in this paper can be used for the design and analysis of periodic systems specified using the polychronous model. Our approach consists first in the design of a clock calculus that balances the tradeoff between decidability and compositionality pointed out earlier [11]. Just as for the affine clock relations, we define a calculus of periodic clock relations to support the compositional modeling of multi-rate systems. Like in [5], periods are expressed here using

ultimately periodic infinite words. However, they are expressed using ternary logic instead of Boolean logic. The ternary logic allows to express periodicity on the values of boolean signals moreover periodicity on the presence/absence of signals. Our calculus corresponds to part of the domain of $(1, m, n)$-affine relations in [11], where most analyses are decidable and recasts these results in an extension of the algebra of [5] with ternary logic. Based on that algebra, we define a calculus of periodic clock relations to compositionally reason about real-time relations in multi-clocked and multi-rate systems. We provide a new Signal equation that allows the design of periodic processes. The clock calculus of Signal is extended to take it into account and an analysis can be applied on periodic processes to guarantee the communications using bounded buffers. The analysis gives as result the size of needed buffers.

Plan Section 2 gives a presentation of the polychronous language Signal. Section 3 presents our algebra of ultimately periodic words. Based on that calculus, section 4 defines a new Signal equation to write periodic processes using ultimately periodic infinite words; we illustrate this extension with a model of a 4-stroke engine. Section 5 describes the clock relation inference of Signal and its extension. Section 6 presents the analysis of periodic processes to guarantee communications using bounded buffers and the analysis is applied on the 4-stroke engine example.

2 Polychronous model of computation

We start with a definition of some required elements of the polychronous model of computation [9]. The set of *tags* $\mathscr{T}$ is the *discrete* time used in Signal and is partially ordered with the relation $t \leq u$. It stipulates that the tag t occurs before u. A *chain* $C \in \mathscr{C}$ is a subset of $\mathscr{T}$ which is totally ordered, $\mathrm{pred}_C(t)$ denotes the immediate predecessor tag in C. Signals, behaviors and processes are defined as follows :

- a *signal* $s \in \mathscr{S} = \mathscr{C} \to \mathscr{V}$ is a function from a *chain* of tags to a set of values,
- a *behavior* $b \in \mathscr{B} = \mathscr{X} \to \mathscr{S}$ is a function from a set of names x to signals, it represents a possible execution of a program.
- a *process* $p \in \mathscr{P}$ is a set of behaviors that have the same domain. This set represents all the possible executions of the program.

The set of possible values $\mathscr{V}$ is defined as union of sets of boolean values, integer values, etc. The type of a signal is that of its values. *Clock* signals take their values in the subset of $\mathscr{V}_B$ reduced to the { true } singleton. Type constraints are not described in this paper. We write $\mathrm{tags}(s)$ for the chain of tags of a signal s and $\min(\mathrm{tags}(s))$ for its first tag. We write $b|_X$ for the projection of a behavior b on a set of names X.

A Signal process consists of the synchronous composition of equations on signals. A signal x is an infinite flow of values that is discretely sampled according to the pace of its clock, noted $\hat{x}$. An equation partially relates signals with respect to an abstract timing model.

$$P ::= x = \mathsf{f}(y, z) \,|\, x = y \,\mathsf{pre}\, v \,|\, x = y \,\mathsf{when}\, z \,|\, x = y \,\mathsf{default}\, z \,|\, P \,|\, P$$

It is allowed to substitute a signal name by its definition to write concise processes. For example, $x = (y \neq z) \mid z = y$ pre true   may be written as $x = (y \neq (y$ pre true $))$.

Semantics of the Signal equations In the *functional equation* $x = f(y, z)$, the signals x, y and z are assumed to be synchronous, (their sets of tags are equal). For each tag the signals are present, x holds the result of the function f applied on the values of y and z. f can be an arithmetic or boolean classical function $(+, \neq, $ or $\ldots)$.

The *delay equation* $x = y$ pre v enforces the signals x and y to be synchronous too. The value of the signal x at a given tag is defined by the value of y from the previous tag in the chain. For its first tag, x holds the value v.

The *sampling equation* $x = y$ when z defines x by y when z is true. It means that the output signal x is present and takes the value of y iff both input signals y and z are present, and z holds the value *true*. In the following definition, the notation $[z]$ represents the clock which is true when z is present and true, and absent otherwise.

The *merge equation* $x = y$ default z defines x by y when y is present and by z otherwise. The signal x is present iff either of the signals y or z is present.

The *synchronous composition* $P \mid Q$ is defined by the simultaneous solution of the equations P and Q at all times. If we note $\text{vars}(P)$ the domain of a process P, it can be defined by the union $b \uplus c$ of behaviors that match on the interface $\text{vars}(P) \cap \text{vars}(Q)$ between P and Q.

$$\llbracket x = f(y, z) \rrbracket = \left\{ b \in \mathscr{B}|_{x,y,z} \left| \begin{array}{l} \text{tags}(x) = \text{tags}(y) = \text{tags}(z), \\ \forall t \in \text{tags}(x), b(x)(t) = f(b(y)(t), b(z)(t)) \end{array} \right. \right\}$$

$$\llbracket x = y \text{ pre } v \rrbracket = \left\{ b \in \mathscr{B}|_{x,y} \left| \text{tags}(x) = \text{tags}(y) = C, b(x)(t) = \left| \begin{array}{ll} v & \text{if } t = \min(C) \\ b(y)(\text{pred}_C(t)) & \text{if } t \neq \min(C) \end{array} \right. \right. \right\}$$

$$\llbracket x = y \text{ when } z \rrbracket = \left\{ b \in \mathscr{B}|_{x,y,z} \left| \text{tags}(x) = \text{tags}(y) \cap \text{tags}([z]), \forall t \in \text{tags}(x), b(x)(t) = b(y)(t) \right. \right\}$$

$$\llbracket x = y \text{ default } z \rrbracket = \left\{ b \in \mathscr{B}|_{x,y,z} \left| \begin{array}{l} \text{tags}(x) = C \\ = \text{tags}(y) \cup \text{tags}(z) \end{array}, \forall t \in C, b(x)(t) = \left| \begin{array}{ll} b(y)(t) & \text{if } t \in \text{tags}(y) \\ b(z)(t) & \text{otherwise} \end{array} \right. \right. \right\}$$

$$\llbracket P \mid Q \rrbracket = \left\{ b \uplus c \mid b \in \llbracket P \rrbracket, c \in \llbracket Q \rrbracket, b|_{\text{vars}(P) \cap \text{vars}(Q)} = c|_{\text{vars}(P) \cap \text{vars}(Q)} \right\}$$

3 An algebra of ultimately periodic infinite words

We now present an algebra of ultimately periodic infinite words used in our extension of Signal for the design and the analysis of periodic processes. We use the three-value logic induced by $\mathbb{Z}/3\mathbb{Z}$ to denote boolean or clock signals by using atoms $a \in \mathbb{Z}/3\mathbb{Z} = \{-1, 0, 1\}$. The absence is denoted by 0, false by -1 and true by 1. We introduce this algebra to represent periodicity over boolean and clock signals.

Ultimately periodic infinite words The ultimately periodic infinite words (called *words* further) noted w or $u(v)$ under consideration are composed of a prefix $u \in (\mathbb{Z}/3\mathbb{Z})^*$ and a period $v \in (\mathbb{Z}/3\mathbb{Z})^+$.

$$\mathbb{W} = \{ w = u(v) \mid u \in (\mathbb{Z}/3\mathbb{Z})^* \text{ and } v \in (\mathbb{Z}/3\mathbb{Z})^+ \}$$

A word $u(v)$ represents the infinite sequence of atoms composed by the sequence u followed by the sequence v repeated infinitely. Words that represent the same sequence of atoms are equal. We now present a few required notations:

- $|u|$ the length of a sequence $u \in (\mathbb{Z}/3\mathbb{Z})^*$
- $\langle u \rangle$ the number of non-zero atoms of a sequence $u \in (\mathbb{Z}/3\mathbb{Z})^*$
- w_n the nth atom of a word w
- $|w|_n^a$ the number of atoms a in the nth first atoms of w
- $w[n]$ the nth non-zero atom in the word w
- "." the classical operation of concatenation.
- $\langle w \rangle_n^a$ the position of the nth atom a in the word w

  i.e. $\langle a.w \rangle_1^a = 1$ $\langle b.w \rangle_n^a = \begin{pmatrix} \langle w \rangle_n^a + 1 & \text{if } a \neq b \\ \langle w \rangle_{n-1}^a + 1 & \text{if } a = b \text{ and } n > 1 \end{pmatrix}$

  ex: $\langle 01(01\text{-}10) \rangle_1^1 = 2$ $\langle 01(01\text{-}10) \rangle_2^1 = 4$ $\langle 01(01\text{-}10) \rangle_2^{-1} = 9$

Operations and operators The partial order $w \sqsubseteq w'$ stipulates that the nth atom in w precedes the corresponding one in w', for all non-zero atoms. $w \sqsubseteq w'$ means that w' is stretched variant of w, since the non-zero atoms occur in the same order.

$$\forall n > 0, \forall a \neq 0, \qquad a.w @ b.w' = \begin{pmatrix} 0.(a.w @ w') & \text{if } b \neq 1 \\ a.(w @ w') & \text{if } b = 1 \end{pmatrix}$$
$$w \sqsubseteq w' \Leftrightarrow \langle w \rangle_n^a \leq \langle w' \rangle_n^a$$

We define the operator $w @ w'$ to resample a word w on an other one w'. The atoms of w being placed in correspondence with the atoms 1 of w', then the word $w @ w'$ has the corresponding atom of w when w' has the atom 1, and has the atom 0 otherwise. The @ operator distributes the values of the first stream to the positions where the second one holds the atom 1. This is an extension of the "w' on w" operator of [5]. This definition yields that $w \sqsubseteq w @ w'$ for all words w, w' and $(w @ w')_n = 0$ if $w'_n \neq 1$ and $w_{|w'|_n^1}$ if $w'_n = 1$.

Example 1. $\forall w \in \mathbb{W}$, $w @ (1) = w$ and $w @ (0) = (0)$

w'	0 1 -1 1 1 0 1 -1 1 1 0 $\cdots = 0(1\text{-}1110)$
w	0 -1 1 0 -1 1 $\cdots = (0\text{-}11)$
$w @ w'$	0 0 0 -1 1 0 0 0 -1 1 0 $\cdots = 0(00\text{-}110)$

$(0\text{-}11) @ 0(1\text{-}1110) = 0(00\text{-}11)$

Remark 1. For any pair of words $u(v)$ and $x(y)$, it is always possible to build equivalent representations $u'(v') = u(v)$ and $x'(y') = x(y)$ such that $|u'| = |x'|$ and $|v'| = |y'|$ [5]. For example, $01(\text{-}11)$ and $(\text{-}101)$ can be respectively rewritten as $01(\text{-}11\text{-}11\text{-}11)$ and $\text{-}10(1\text{-}101\text{-}10)$. Such a rewriting is useful when pointwise operations have to be applied on words.

Synchronizable words

Definition 1. Two words w, w' are said *stretch-equivalent* if $\forall n > 0, w[n] = w'[n]$.

It means that the two stretch-equivalent words have the same order of non-zero atoms. Obviously, any two words the atoms of which take their values in $\{0, 1\}$ (called *binary words*) are stretch-equivalent.

In [5], it is shown that two binary words $u(v), u'(v')$ are *synchronizable* iff the lengths of their periods $(|v|, |v'|)$ and their numbers of presence $(\langle v \rangle, \langle v' \rangle)$ match $\langle v \rangle / |v| = \langle v' \rangle / |v'|$. This extends to stretch-equivalent ternary words.

Definition 2. Two stretch-equivalent ternary words w, w' are said *synchronizable*, written $w \sim w'$, iff there exists d, d' such that $w \sqsubseteq 0^d.w'$ and $w' \sqsubseteq 0^{d'}.w$.

The relation $w \sqsubseteq 0^d.w'$ means that we can delay the atoms of w' by d zero atoms so that the nth non-zero atom of w precedes the nth non-zero atom of w'. Then $w \sqsubseteq 0^d.w'$ (resp. $w' \sqsubseteq 0^{d'}.w$) implies that the distance between the nth non-zero value of w and the nth non-zero value of w' is bounded by d (resp. d'). This definition will be used to analyse the size of buffers required for communications (Section 6).

Property 1. Two stretch-equivalent words $u(v)$ and $u'(v')$ are synchronizable iff $\langle v \rangle / |v| = \langle v' \rangle / |v'|$. (the proof is similar to that of [5]).

4 Periodic processes

Periodic sampling equation We introduce a new operator in Signal, noted $@$, derived from the corresponding one defined on periodic words, in order to express periodic relations. A periodic sampling equation $x = w @ y$ relates two (clock or boolean) signals x, y, with a word w. It defines x to hold the successive non-zero atoms of w when y takes the value true. We note $t_x[n]$ for the nth tag of a signal x.

$$[\![x = w @ y]\!] = \left\{ b \in \mathcal{B}|_{x,y} \left| \begin{array}{l} t \in \text{tags}(x) \Leftrightarrow \exists n > 0,\ t = t_y[n] \text{ and } w_n \neq 0 \\ \forall n > 0,\ b(x)(t_x[n]) = w[n] \end{array} \right. \right\}$$

Example 2. $\forall w \in \mathbb{W}, x = (1) @ y \Rightarrow x = [y]$, and $x = (0) @ y \Rightarrow x$ is always absent. Recall that $[y]$ is a clock which is present iff the signal y is present and true.

$x = 0(10\text{-}1) @ y$

y	1	-1	1	1	1	-1	1	1	-1	1	$\cdots$
$0(10\text{-}1)$	0		1	0	-1		1	0		-1	$\cdots$
x			1		-1		1			-1	$\cdots$

Example 3. To outline the use of a periodic sampling equation in Signal, we consider the specification of a one-place buffer which is constrained to behave as a mailbox (every message has to be read once and only once before there is a new one). The process buffer uses two subprocesses, alternate and current. The process alternate stipulates that the signals x and y should have exclusive periods (01) and (10) with respect to the clock c. The process current stores the value of an input signal y and loads it into the output signal x upon request. The buffer has a main clock equivalence class $\hat{r} = c$ and two exclusive samples $\hat{x} = (01) @ c$ and $\hat{y} = (10) @ c$.

$$x = \text{buffer}(y) \stackrel{\text{def}}{=} (x = \text{current}(y, c) \,|\, \text{alternate}(x, y, c))$$
$$\text{alternate}(x, y, c) \stackrel{\text{def}}{=} (\hat{x} = (01) @ c \,|\, \hat{y} = (10) @ c)$$
$$x = \text{current}(y, c) \stackrel{\text{def}}{=} (r = y \,\text{default}\, (r \,\text{pre}\, initValue) \,|\, x = r \,\text{when}\, \hat{x} \,|\, \hat{r} = c)$$

Use case In a 4-stroke engine, each cylinder performs a cycle with four phases: intake, compression, combustion and exhaust. During the intake phase, a blend of fuel and air is put in the cylinder, the compression compresses this blend, the combustion is the action of burning the blend because of an ignition and the exhaust

evacuates the gas of the cylinder. To model this periodic system, like [2] we use a clock *clkShaft* which represents the rotation of the crankshaft. In figure 1, it appears that the crankshaft turns $720°$ for each cycle of the engine.
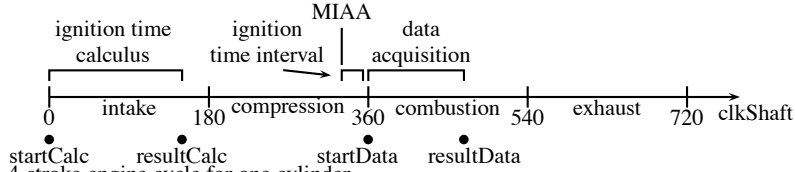
Fig. 1 4-stroke engine cycle for one cylinder

The date of ignition occurs during the 'ignition time interval' and must be computed for each cycle of the engine for a better efficiency. This date depends on several measures made during the combustion phase of the previous cycle. 'Data acquisition' is the period where the measures are done and 'ignition time calculus' is the one where the ignition date is calculated. The Maximal Ignition Advance Angle (MIAA) is the beginning of the 'ignition time interval'.

The process **oneCylinder** represents the model of a cylinder in Signal using periodic sampling equations. The signal *clkShaft* is a clock which is present each time the crankshaft turns one degree. The clocks *clkIntake, clkCompress, clkCombust, clkExhaust* represent the beginning of the four phases of the 4-stroke engine (1). For example, the equation $clkCompress = (0^{180}10^{539})\,@\,clkShaft$ constrains *clkCompress* to be present when *clkShaft* is present for the $(181 \bmod 720)^{th}$ times that correspond to a turn of $180°$ from each beginning of a cycle. The clock *clkMIAA* corresponds to the MIAA.

We assume two external components represented by two processes: **dataAcquisition** for the acquisition of the data (2) and **ignitionTime** for the computation of the date of ignition (3). For each cycle, the beginning and the end of the running of the two processes are constrained by the periodic clocks of the signals *startData* and *resultData* for **dataAcquisition** and the signals *startCalc* and *resultCalc* for **ignitionTime**.

The ignition must occur at the MIAA plus a number of degrees contained in the result *resultCalc* of the process **ignitionTime**. The clock *clkIgnition* (4) is the result of the process **Ignition** (5) that computes this date. A buffer is needed to delay the acquisition of the data at the clock *resultData* to the use of this data at the clock *startĈalc* (in the next cycle of the engine). The management of the buffer to operate this delay is shown in the global process: **4cylinderEngine**. The model of a four cylinder engine is a composition of four processes **oneCylinder**, the basic clocks of which are 'shifted' (6): the first cylinder starts running at the first presence of the clock *clk* of the crankshaft, the second cylinder starts with a delay of 180 degrees on the clock *clk*, the third one with a delay of 360 degrees and the fourth one with a delay of 540 degrees. The process **bufferFIFO** is a classical FIFO in which an element is added when one of the signals $d1, d2, d3$ or $d4$ (corresponding to the respective *resultData* signal of each cylinder) is present; an element is taken from the FIFO

$$\text{oneCylinder}(clkShaft, resultData, startCalc) \stackrel{\text{def}}{=}$$

$$\begin{pmatrix} clkIntake \quad = (10^{719})@clkShaft \quad | clkCompress = (0^{180}10^{539})@clkShaft \\ | clkCombust = (0^{360}10^{359})@clkShaft | clkExhaust \quad = (0^{540}10^{179})@clkShaft \\ | clkMIAA \quad = (0^{330}10^{389})@clkShaft \end{pmatrix} (1)$$

$$| \begin{pmatrix} resultData = \text{dataAcquisition}(startData) \\ | start\hat{D}ata \ = clkCombust \ | resul\hat{t}Data = (0^{470}10^{249})@clkShaft \end{pmatrix} (2)$$

$$| \begin{pmatrix} resultCalc = \text{ignitionTime}(startCalc) \\ | start\hat{C}alc \ = 0^{720}(10^{719})@clkShaft \ | resul\hat{t}Calc = 0^{720}(0^{150}10^{569})@clkShaft \end{pmatrix} (3)$$

$$| \ (\ clkIgnition = \text{Ignition}(resultCalc, clkShaft, clkMIAA)) \quad (4)$$

$$clkIgnition = \text{Ignition}(resultCalc, clkShaft, clkMIAA) \stackrel{\text{def}}{=}$$

$$\begin{pmatrix} ignitionDelay = resultCalc \text{ default } (ignitionDelay \text{ pre } firstIgnitionDate) \\ | i \quad\quad\quad = 0 \text{ when } clkMIAA \\ \quad\quad\quad\quad \text{default -1 when } (zi == ignitionDelay) \\ \quad\quad\quad\quad \text{default } zi+1 \text{ when } (zi \neq \text{-1}) \\ | zi \quad\quad\quad = i \text{ pre -1} \quad | \quad \hat{i} = clkShaft \\ | clkIgnition \quad = \text{true when } (i == ignitionDelay) \end{pmatrix} (5)$$

$$\text{4cylinderEngine}(clk) \stackrel{\text{def}}{=}$$

$$(6) \begin{pmatrix} \text{oneCylinder}(clk, d1, c1) \\ | \text{oneCylinder}(0^{180}(1)@clk, d2, c2) \\ | \text{oneCylinder}(0^{360}(1)@clk, d3, c3) \\ | \text{oneCylinder}(0^{540}(1)@clk, d4, c4) \end{pmatrix} | \begin{pmatrix} add = d1 \text{ default } d2 \text{ default } d3 \text{ default } d4 \\ | take = \text{bufferFIFO}(add) \\ | c1 \ = take \text{ when } \hat{c1} | c2 = take \text{ when } \hat{c2} \\ | c3 \ = take \text{ when } \hat{c3} | c4 = take \text{ when } \hat{c4} \end{pmatrix} (7)$$

when one of the signals $c1, c2, c3$ or $c4$ (corresponding to the respective *startCalc*) is present (7). The analysis of the necessary size of this buffer is presented in Section 6. The figure 2 shows a chronogram of this model.
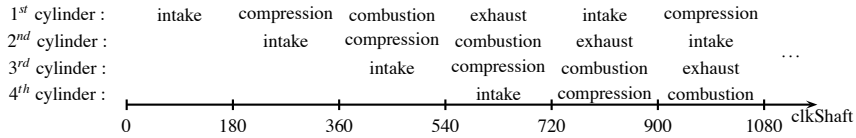
Fig. 2 a 4 cylinder engine chronogram

5 Inference of clock relations

In this section, we present the clock relations extracted from Signal equations and the associated clock calculus extended with the periodic sampling equation. In general, a clock c denotes a set of tags in the polychronous model of computation. The clock $\hat{x}$ of a signal x denotes the tags at which the signal x is present. The clock $[x]$ (resp. $[\neg x]$) denotes the tags at which a boolean signal x is present and true (resp. false). The intersection $\cap$, union $\cup$ and the complement $\setminus$ are the usual operators on sets applied on the sets of tags representing clocks. The expression $w @ c$ denotes the clock obtained by the periodic sampling defined in Section 4. When we are talking

about clocks, all the words are binary words. Any Signal process P corresponds to a system of implicit clock relations C that denotes its implied timing structure.

$$c, d ::= \hat{x} \mid [x] \mid [\neg x] \mid c \cap d \mid c \cup d \mid c \setminus d \mid w @ c \qquad \text{(clock)}$$
$$C, D ::= c = d \mid C \mid D \qquad\qquad\qquad \text{(clock relation)}$$

The clock relations are specified by the inference system $P : C$ that is recursively defined by induction on the syntax of P. In a delay equation $x = y \text{ pre } v$, the signals are synchronous ($\hat{x} = \hat{y}$). In a sampling equation $x = y \text{ when } z$, the clock $\hat{x}$ is defined by the clock $\hat{y}$ at the sampling condition $[z]$. In a merge equation $x = y \text{ default } z$, $\hat{x}$ is present if either one of the clock y, z is. In a functional equation $x = \text{f}(y, z)$ the signals x, y, z are synchronous. In a periodic sampling equation $x = w @ y$, x is defined by the atoms of w when y is true. The rule for composition $P \mid Q$ is defined to be $C \mid D$ and by induction on the deductions $P : C$ and $Q : D$ made on its sub-terms.

$$
\begin{array}{lll}
x = y \text{ pre } v & : (\hat{x} = \hat{y}) & \qquad x = w @ y : (\hat{x} = w^2 @ [y]) \quad (*) \\
x = y \text{ when } z & : (\hat{x} = \hat{y} \cap [z]) & \qquad\qquad \| [x] = [w] @ [y] \\
x = y \text{ default } z & : (\hat{x} = \hat{y} \cup \hat{z}) & \qquad\qquad \| [\neg x] = [\neg w] @ [y])
\end{array}
\qquad
\dfrac{P : C \quad Q : D}{P \mid Q : C \mid D}
$$

(*) $w^2, [w]$ and $[\neg w]$ are binary words respectively defined by

$$(a.w)^2 = \begin{pmatrix} 1, a = 1 \\ 1, a = -1 \\ 0, a = 0 \end{pmatrix} . w^2 \quad [a.w] = \begin{pmatrix} 1, a = 1 \\ 0, a = -1 \\ 0, a = 0 \end{pmatrix} . [w] \quad [\neg a.w] = \begin{pmatrix} 0, a = 1 \\ 1, a = -1 \\ 0, a = 0 \end{pmatrix} . [\neg w]$$

Intuitively, the binary words $w^2, [w]$ and $[\neg w]$ are used to represent respectively presence, presence and true, presence and false of a boolean signal constrained by a periodic sampling equation using a word w.

Property 2. Let c, d and e be clocks of a process P and w and w' two binary words,
$P : (c = w @ d \mid d = w' @ e) \Rightarrow P : c = (w @ w') @ e$
Proof : We know that the operator $@$ is associative for binary words [5].
$c = w @ (w' @ e) \Longrightarrow c = (w @ w') @ e.$

6 Communication analysis

Most of reactive systems use communications between their different components. In safety-critical domains, like car industry, there is a need for guaranteeing bounded communications. In the particular case of reactive systems described with our model of periodic processes, we provide a formal analysis allowing to determine minimal buffer size required for communications.

Buffer size analysis We consider a definition of periodically equivalent clocks based on the definition of synchronizable words (definition 2). Two clocks c and d are periodically equivalent if there exists a clock r such that the tags of c and d are included in the set of tags of r, and there are constraints $c = w_c @ r$, $d = w_d @ r$ where

w_c, w_d are synchronizable. It means that the difference of the number of presence of the clocks c and d is bounded all times the clock r is present.

Definition 3. Two clocks c, d are periodically equivalent in P, noted $c \sim d$, iff $\exists r$ a clock, such that $P : (c = w_c @ r, \mid d = w_d @ r)$ and $w_c \sim w_d$

From this definition, we define the synchronization of a clock c on another clock d, noted $c \succ d$. It constrains the nth occurrence of the clock c to precede the corresponding one of the clock d for all $n > 0$. c and d must be periodically equivalent. The operator $\succ$ on the clocks is not commutative.

Definition 4. A clock c is synchronizable on a clock d in P, noted $c \succ d$, iff
1. $c \sim d$,   (i.e. $\exists r$ a clock, such that $P : (c = w_c @ r, \mid d = w_d @ r)$, $w_c \sim w_d$)
2. $w_c \sqsubseteq w_d$

When a clock c is synchronizable on a clock d, it means that values can be delayed from the clock c to the clock d using a bounded buffer. The maximal size of the buffer used by the delay can be computed. We thus obtain the minimal size of the buffer to delay a signal of clock c for its use on clock d.

Property 3. Minimal buffer size to guarantee the communication between two synchronizable clocks c and d in a process P.
From definition 4, we have $c \succ d \Leftrightarrow P : (c = w_c @ r, \mid d = w_d @ r)$, $w_c \sim w_d$ and $w_c \sqsubseteq w_d$. The minimal size of the buffer to guarantee the delay from the clock c to the clock d is : $size(c, d) = \max_{n>0}(|w_d|_n^1 - |w_c|_n^1)$
The $size(c, d)$ is the maximal difference between the numbers of presence of the clocks d and c. This size is computable thanks to remark 1 (section 3).

Analysis of the minimal buffer size required for the process bufferFIFO The inference system $(P : C)$ defined in Section 5 applied on the process 4cylinderEngine gives the following relations:
    4cylinderEngine : $(add = 0^{291}(0^{179}1) @ clk \mid take = 0^{541}(0^{179}1) @ clk)$
The two clocks *add* and *take* are periodically equivalent (definition 3) and synchronizable (definition 4). So we can calculate the size of the buffer needed to delay values from the clock *add* to the clock *take* : $size(add, take) = 2$. The communications between the acquisition of the data from the previous cycle of the engine to the computation of the ignition date are guaranteed with a two place buffer.

Analysis of the number of components used for the computation of the ignition date. In the 4cylinderEngine process, each process oneCylinder uses a ignitionTime process to compute its ignition date. But this process could be shared between several oneCylinder processes (ignitionTime is only running during the intake phases). Just like for the above analysis of the minimal size of the FIFO buffer, the same kind of analysis can be done to calculate the number of ignitionTime processes required for the overall application. The ignitionTime process is running between an occurrence of the *startCalc* signal to the next occurrence of the *resultCalc* signal. The *size* function (between the clocks corresponding to the start and the end of the running of the ignitionTime process) can be used here to determine the maximal number of ignitionTime processes running at the same time.

$$(\bigcup startCalc = 0^{541}(0^{179}1)@clk \mid \bigcup resultCalc = 0^{690}(0^{179}1)@clk)$$

The result is $size(\bigcup startCalc, \bigcup resultCalc) = 1$. Therefore one single component can be used for the computation of the ignition date of the four processes oneCylinder. The previous computation is always finished when a new computation starts.

Applying the same analysis for a 4-stroke engine with 6 cylinders defined on a similar model than the 4cylinderEngine (with a delay of 120 degrees between the clocks *clkShaft*) gives a result size of 2, so that two components are needed for the computation of the ignition date of the six processes oneCylinder.

7 Conclusion

We have presented an extension of the polychronous model of computation of the Signal formalism with a periodic sampling equation using ultimately periodic infinite words. We have shown that periodic clock relations provide a calculus to compositionally reason about real-time relations in multi-clocked systems. Our main contributions are the adaptation of periodic relations of [5] to $\mathbb{Z}/3\mathbb{Z}$ and their handling in the clock calculus associated with the polychronous model. This extension is used for the design of periodic systems in Signal and we illustrated it with an example of a 4-stroke engine. We gave an analysis allowing to guarantee the communications using bounded buffers; the maximal size of buffers is formally evaluated.

The work that has been done allows to analyse periodic systems that are designed using the new specific periodic sampling equation introduced in Signal. A further study would be to define a static analysis that would extract periodic relations from implicit periodicities of the systems. This will allow to analyse a greater set of periodic systems.

References

1. C. André, F. Mallet, R. de Simone. Modeling time in UML. Research report ISRN I3S/RR-2007-16-FR, I3S Laboratory, 2007.
2. C. André, F. Mallet, M-A. Peraldi. A multiform time approach to real-time system modeling. SIES, 2007.
3. A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The Synchronous Languages Twelve Years Later. *Proceedings of the IEEE*. IEEE, 2003.
4. J. Buck, S. Ha, E. Lee, D. Messerschmitt. Ptolemy: a framework for simulating and prototyping hetrogeneous systems. *International Journal in Compuer Simulation*, v4(2), 1994.
5. A. Cohen, M. Duranton, C. Eisenbeis, C. Pagetti, F. Plateau, and M. Pouzet. N-synchronous Kahn networks: a relaxed model of synchrony for real-time systems. In *POPL*. ACM, 2006.
6. C. Dezan, P. Quinton. Verification of regular architectures using Alpha. In *Adaptative Sensor Array Processing Workshop*. IEEE Press, 1994.
7. G. Kahn. The semantics of a simple language for parallel programming. *Information Processing*, North Holland, 1974.
8. E. Kock, et al. Yapi: Application modeling for signal processing systems. *Design Automation Conference*. ACM, 2000.
9. P. Le Guernic, J.-P. Talpin, and J.-C. Le Lann. Polychrony for system design. In *Journal for Circuits, Systems and Computers*. World Scientific, 2003

10. H. Marchand, E. Rutten, M. Le Borgne and M. Samaan. Formal Verification of programs specified with Signal : application to a power transformer station controller. In *Science of Computer Programming*. Elsevier, 2001.
11. I. Smarandache, T. Gautier, P. Le Guernic. Validation of mixed Signal-Alpha real-time systems through an affine calculus on clock synchronization constraints. In *Formal Methods Europe*. Springer, 1999.