From Multi-clocked Synchronous Processes to Latency-insensitive Modules

Jean-Pierre Talpin  Dimitru Potop-Butucaru  Julien Ouy  Benoit Caillaud

INRIA - IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
FirstNamn.LastName@inria.fr

ABSTRACT
We consider the problem of synthesizing correct-by-construction globally asynchronous, locally synchronous (GALS) implementations from modular synchronous specifications. This involves the synthesis of asynchronous wrappers that drive the synchronous clocks of the modules and perform input reading in such a fashion as to preserve, in a certain sense, the global properties of the system. Our approach is based on the theory of weakly endochronous systems, which gives criteria guaranteeing the existence of simple and efficient asynchronous wrappers. We focus on the transformation (by means of added signalling) of the synchronous modules of a multihit synchronous specification into weakly endochronous modules, for which simple and efficient wrappers exist.

Categories and Subject Descriptors: D.2.2: CASE
General Terms: Algorithms, Reliability, Verification.
Keywords: separate compilation, compositional mapping.

1. INTRODUCTION
We start our presentation with a summary of the theory under consideration. We refer the reader to the continuation of the present work in [4] for motivating examples, throughout relation to previous works and applications to separate compilation and distributed code generation. The micro-step automata theory under consideration is a particular class of concurrent automata equipped with synchronous product and synchronous and asynchronous FIFO models allowing to represent computation and communication causality as well as communication through read/write primitives over given communication channels. A detailed description of this theory can be found in [3].

Micro-step automata communicate through signals $x \in X$. The labels $l \in L_X$ generated by the set of names $X$ are represented by a partial map of domain from a set of signals $X$ noted $\text{vars}(l)$ to a set of values $V^l = V \cup \{\bot\}$ and tags. The label $\bot$ denotes the absence of communication during a transition of the automaton. We note $l' \leq l''$ iff there exists $l'$ disjoint from $l''$ such that $l = l' \sqcup l''$ and then $l' \n l'' = l''$. We say that $l$ and $l'$ (resp. $l$ and $l''$) are compatible, written $l \models l'$, iff $l(x) = l'(x)$ for all $x \in \text{vars}(l) \cap \text{vars}(l')$ and, if so, note $l \models l'$ their union. We write $\text{supp}(l) = \{x \in X \mid l(x) \neq \bot\}$ for the support of a label $l$ and $\bot_X$ for the empty support.

Definition 1. An automaton $A = (s^0, S, X, \rightarrow)$ is defined by an initial state $s^0$, a finite set of states $S$ noted $s$ or $x = v$, labels $L_X$ and by a transition relation $\rightarrow$ on $S \times L_X \times S$. The product $A \times A_2$ of $A = (s^0, S_1, X_1, \rightarrow_1)$ for $0 < i \leq 2$ is defined by $(s_1, s_2) \times_S S_1 \times_S S_2$ where $(s_1, s_2) \rightarrow (s_1', s_2')$ iff $s_1 \rightarrow_1 s_1'$ for $0 < i \leq 2$ and $l_1 X_1$, the projection of $l$ on $X_1$. An automaton $A = (s^0, S, X, \rightarrow)$ is concurrent iff $s \rightarrow^* s$ for all $s \in S$ and if $s \rightarrow^* s'$ and $l \leq l'$ then there exists $s''$ such that $s \rightarrow^* s''$ and $s'' \rightarrow^* l'$.

Synchronous automata account for primitive communications using read and write operations on directed communication channels pairing variables $x$ with directions represented by tags. Emitting a value $v$ along a channel $x$ is written $!x = v$ and receiving it $?x = v$. We write $\text{vars}(D)$ for the channel names associated to a set of directed channels $D$. The undirected or untagged variables of a synchronous automaton are its clocks noted $c$.

Definition 2. A synchronous automaton $(s^0, S, X, c, \rightarrow)$ of clock $c \in X$ is a concurrent automaton $(s^0, S, X, \rightarrow)$ s.t.
1. $s \rightarrow^l s$ implies $l = c$ or $c \leq l$
2. $s^0 \rightarrow^0 s^0$
3. $s \rightarrow^c s'$ implies $s' \rightarrow^c s'$
4. if $s_{i-1} \rightarrow^l s_i$ and $l_i \neq 1$ for $0 < i < n$ then $\text{vars}(l_i) \cap \text{vars}(l_{i+1}) = \emptyset$iff $i \neq j$.

We assume that a channel $x$ connects at most one emitter with at most one receiver. Multicast will however be used in examples and is modeled by substituting variable names (one $!x = v$ and two $?x = w_{1,2}$ will be substituted by two $!x = v$, $!x_2 = v$ and two $?x = w_{1}$ $?x_2 = w_{2}$ by introducing a local signal $x_2$). For an automaton $A$, we define a trace $t \in T = L^*_X$ by a finite sequence of labels, write $|t|$ for its length and refer to $t_i$ as the ist label in $t$ and $T_A(s)$ as the set of traces accessible by $A$ from state $s$. For a synchronous automaton of clock $c$, we write $s \rightarrow^t s'$ iff there exists a series $(s_i)_{0 \leq i \leq n}$ with $n = |t|$ such that $s_0 = s$, $s_{i-1} \rightarrow^c s_i$ for $0 < i \leq n$ and $s_n = s'$. We note $s_0 \rightarrow^t s$ iff $s \rightarrow^t s'$ with $t_i \neq c$ for $0 < i \leq |t|$ and $s_{|t|} \rightarrow^c s$. For a synchronous automaton of clock $c$, we write $s \rightarrow^t s'$.
The composition of automata is defined by synchronized
product, using first-in-first-out buffer models to represent
communication through synchronous and asynchronous chan-
nels. Synchronous communication is modeled using 1-place
synchronous FIFO buffers. The synchronous FIFO of clock
c and channel x is noted s\(\text{fifo}^c\) and the asynchronous FIFO
buffer of channel x is written s\(\text{fifo}^x\). A synchronous FIFO
buffer serializes the emission event !x = v followed by the re-
ceipt event ?x = v within the same transition (the clock tick
c occurs after). Silent transitions s\(_i\) \(\rightarrow^s s_j\) for 0 \(\leq i \leq 2\) are
left implicit as s\(\text{fifo}^x\) is assumed to be a concurrent automa-
aton. The model of an unbounded and asynchronous FIFO
buffer is similarly defined by the repetition of the communi-
cation pattern of the synchronous buffer and by induction
on its storage size represented by a word w \(\in V^*\).

\[s\text{fifo}^c \overset{\text{def}}{=} (s_0, \{s_0, 2\}, \{(x, 1, x, c), c\} + \bigcup s_0 \rightarrow^x s_1 \rightarrow^x \cdots \rightarrow^x s_2)\]

\[s\text{fifo}^x \overset{\text{def}}{=} (v, \bigcup x \in V \{?x = v, !x = c\}, \bigcup x \in v \rightarrow^x v \rightarrow^x \cdots \rightarrow^x v \in V, !x \in V^*)\]

Two synchronous automata are composable if their tagged
variables are mutually disjoint. This rule enforces the point-
to-point communication restriction previously mentioned and
non-overlapping of clocks. The synchronous composition of
two automata consists of its synchronous product with the
synchronous FIFO buffer model instantiated for all channels
x common to the vocabulary of the composed automata.
The clocks c\(_1,2\) of both automata are synchronized to the
clock c of the FIFO (by the substitution A\(_1[c/c]\) of c\(_1\) by c
in A\(_1\)). Similarly, the asynchronous composition consists of
the synchronous product of the composed automata in
stances of asynchronous FIFO buffers for all common chan-
nels x and, this time, without clock synchronization.

**Definition 3.** Let \(A_1 = (s_0, S_1, X_1, c_1, \rightarrow_1), i=1, 2\) be two com-
posable synchronous automata and c a clock and write \(A[\rightarrow_2/c]\)
for the substitution of c\(_1\) by c\(_2\) in A. Synchronous com-
position \(A_1 \parallel A_2\) at clock c and asynchronous composition
\(A_1 \parallel A_2\) are defined by:

\[A_1 \parallel A_2 = (\bigotimes_{i=1}^2 A_1[i/c]) \otimes \left(\bigotimes_{i=1}^2 \text{vars}(X_1)\right) \text{s\(\text{fifo}^c\)}\]

\[A_1 \parallel A_2 = (\bigotimes_{i=1}^2 A_1[i/c]) \otimes \left(\bigotimes_{i=1}^2 \text{vars}(X_1)\right) \text{s\(\text{fifo}^x\)}\]

If \(A = (s, S, X, c, T)\) then the restriction \(A/x\) is defined by
\((s, S, X\{x\}, c, \{s_1 \rightarrow^x s_2 \in T^* | s_1 \rightarrow^x s_2 \in T\})\).

### 2. MICRO-STEP SEMANTICS OF SIGNAL

Micro-step automata provide an expressive operational
semantics framework for the multi-clocked data-flow specifi-
cation formalism Signal under consideration. In Signal [2],
a process p is an infinite loop that consists of the synchronous
composition p\(\parallel q\) of simultaneous equations \(x = y f z\) over
signals noted x, y, z. Restricting the lexical scope of a signal
name x to a process p is noted p/x. A network of syn-
chronous processes is noted P and P \(\parallel Q\) stands for the
asynchronous composition of P and Q.

\(p, q \overset{\text{def}}{=} (x = y f z) \parallel p | q \parallel p | x \quad P, Q \overset{\text{def}}{=} p | P \parallel Q\)

A delay equation \(p = \text{def}(x = y \pre v_0)\) corresponds to an
automaton composed of four micro-states \(s^0, s^1, s^2, s^3\) and
for each value v of the signal x and starting from an initial state
\(s_0\) with the initial value \(v_0\). The automaton concurrently
performs both receive ?y = w and send !x = v actions and
then issues a clock transition c to the state \(s_0\) where the
next reaction takes place.

\[A_p = \left( \begin{array}{ccc}
 s^0 & s^0 & s^1 \\
 \downarrow \text{y=0} & \downarrow \text{y=1} & \downarrow \text{y=2} \\
 s^2 & s^3 & s^4
 \end{array} \right)\]

A sampling equation p\(\equiv\text{def}(x = y\ \text{default} z)\) performs two
concurrent receive actions ?y = v and ?z = w. If either y or z is absent
or if z equals 0 then the automaton performs a clock transi-
tion to the initial state \(s_0\). Otherwise, it performs the send
action !x = v. This means that x is causal to both y and z.
Notice that all intermediate states are parameterized with
the value v under consideration.

\[A_p = \left( \begin{array}{ccc}
 s^0 & s^0 & s^1 \\
 \downarrow \text{y=0} & \downarrow \text{y=1} & \downarrow \text{y=2} \\
 s^2 & s^3 & s^4
 \end{array} \right)\]

A merge equation p\(\equiv\text{def}(x = y\text{default} z)\) performs two
concurrent receive actions ?y = v and ?z = w. If y is present
then the send action is always !x = v to s4. If only z is present
(in s\(_2\)) then the send action is !x = w to s\(_3\) before a
clock transition back to s\(_0\). Otherwise the only choice is a
silent transition at s\(_0\). Again, all intermediate states are
parameterized with the possible pairs (v, w) \(\in V^2\).

**3. FORMAL PROPERTIES**

To address this issue, the property of weak endochrony [3]
defines the class of deterministic micro-automata which sat-
ify insensitivity to latency.

**Definition 4.** Let us write \(s \rightarrow^t s'\) iff there exists s'\(_t\) such
that s \(\rightarrow^t s'\). A micro-automaton A = (s\(_0\), S, X, c, \(\rightarrow\)) is
weakly-endochronous iff it is synchronous and:

1. deterministic: if s \(\rightarrow^t s_1\) and s \(\rightarrow^t s_2\) then \(s_1 = s_2\)
2. step-independent: if \( s \to s_1, s_1 \to s_2 \) and \( \text{supp}(l_1) \cap \text{supp}(l_2) = \emptyset \) then there exists \( s' \) such that \( s_1 \to s' \to s_2 \to s \) and \( s \to s_1 \to s_2 \).

3. clock-independent: \( s \to s' \) implies
(a) if \( s \to s' \) and \( c \not\in \text{supp}(t) \) then \( s' \to s \).
(b) if \( s \rtarrows s' \) then \( s' \to s \).
(c) if \( s \rtarrows s' \) then \( s \rtarrows s'' \) and \( t' \leq t'' \).
(d) if \( s \to * s' \) and \( s \to t \) then \( s \to (\tau(t')) \).

4. choice-independent: if \( \text{supp}(l_i) = \text{supp}(l_j) \), \( s \rtarrows s' \to s_1 \) and \( t_1 \bowtie t' \) then \( s \to s_1 \).

Let \( (A_i)_{0 \leq i \leq n} \) be weakly endochronous automata, if the synchronization \( \{ A_i \}, 0 \leq i \leq n \) is non-blocking (i.e. \( s \to s' \Rightarrow \) then \( \{ A_i \}_{0 \leq i \leq n} \) are weakly isochronous. Weakly isochronous automata satisfy the de-synchronization correctness criterion of \([3]\).

**Theorem 1.** If the automata \( A_i, 0 \leq i \leq n \) are weakly isochronous then their desynchronization is correct.

**4. FLOW ANALYSIS**

In order to define decision criteria, section 5, to validate definition 4 and meet the property of theorem 1, consider an intermediate representation of multi-clocked specification that exposes its control and data-flow properties for the purpose of their analysis.

A process \( p \) is represented as a data-flow graph \( G \). In this graph, a vertex \( y \) is a data-flow relation that partially defined a clock or a signal. A signal vertex \( c \Rightarrow x = f(y_{1..n}) \) partially defines \( x \) by \( f(y_{1..n}) \) at \( c \). We note \( y \) the clock \( c \) of a signal vertex \( y \), \( \text{use}(g) \) its set of used signal names \( \{ y_{1..n} \} \), \( \text{def}(g) \) its defined signal name \( x \), \( \text{vars}(g) \) its \( \text{use}(g) \cup \text{def}(g) \) and \( \text{fun}(g) \) its function \( f \), which can either be the identity, a boolean function \( \land \lor \) or the delay operator \( \text{pre} \).

For the sake of a clear separation of concerns, the form of the graph \( G_p \) of a process \( p \) is decomposed into its clock vertices \( C_p \), that defines its control and timing model, and signal vertices \( S_p \), that define its (untimed) data-flow. The set of bound signal names \( X_p \) of \( G_p \) is further extracted by commutativity and for any substitution \( \{ G \} / x \) \( \{ \text{supp}(x) \} \) \( \text{supp} \). We denote the decomposition of the graph \( G_p \) of a process \( p \) as \( G_p = (C_p, S_p) / X_p \).

The remainder requires a couple of notations to be defined: we write \( G \models e = f \) iff the system of Boolean equations \( C_p \) in \( G \) implies that \( e = f \) always holds. In addition, and for all processes \( p \) and all boolean signal \( x \) in \( p \), we assume that \( C_p \models \hat{x} = x \lor \neg \hat{x} \) and \( C_p \models x \lor \neg \hat{x} = 0 \). We note \( c \leq d \) for syntactic clock inclusion: \( x < \hat{x} \) or \( x < \hat{x} \) and \( \hat{x} < \hat{x} \).

We note \( \text{vars}(p) \) for the set of free signal names of a process \( p \). The free signals of a process are those which appear in equations and whose scope is not bound by restriction. The free output signals \( \text{out}(p) \) of a process \( p \) are free signal names occurring on the left-hand side of equations. The free input signals \( \text{in}(p) \) of a process \( p \) are the remaining \( \text{vars}(p) \setminus \text{out}(p) \).

**Definition 5.** The clock \( c \) is computable from the input signals \( \text{in}(p) \), written \( c \models \text{pre} = C \), iff
- if \( x \in \text{in}(p) \) and \( c \leq x \) then \( c \models \{ c \} \).
- if \( c \in \text{in}(p) \) and \( \hat{x} \models C \) and \( c \models \hat{x} \).
- if \( c_1 < c_2 \) and \( C_p \models c_1 = c_2 \), then \( c_2 < C \).
- if \( c_1 < C \) and \( C_p \models d = c_1 \lor c_2 \) or \( d = c_1 \land c_2 \) then \( d < C \).

The clock of \( p \) are computable iff for all bound signals \( x \in X_p \) of \( p \) and \( c \leq x \) there exists \( C \) such that \( c < C \).

**Definition 6.** Two signal vertices \( g, h \in S_p \) are causal, written \( g \models h \), iff \( \text{def}(g) \in \text{use}(h) \), \( \text{fun}(h) \neq \text{pre} \) and \( C_p \models \hat{g} \land \hat{h} \neq 0 \); independent, written \( g \models h \), iff \( g \models \hat{h} \) and \( h \models \hat{g} \). Two independent vertices \( g \) and \( h \) are exclusive, written \( g \hash h \), if \( C_p \models \hat{g} \land \hat{h} = 0 \); synchronous, written \( g \models h \), if \( g \models \hat{h} \) or \( g \models \hat{h} \); concurrent, \( g \models h \), if \( g \models h \) or \( g \models \hat{h} \).

**Definition 6** provides a complete structure for the vertices of a data-flow graph: two vertices are either causal or independent. Two independent vertices are either concurrent, synchronous or exclusive.

A notion of grammar establishes the duality between the automaton and the graph of a process \( p \) by linking these representations. The grammar \( M_p \) of a process \( p \) consists of signal vertices related by the connectors \( \models \), \( \models \sim \) and \( \models \). It
can be viewed as a refinement of data-flow graph $G_p$ which make the event structure implied by the clock relations $C_p$ explicit.

$$M, N := g | M \succ N | M \# N | M \sim N | M \circ N$$

To construct the event grammar of a process, we make use of a few functions to manipulate the graph structure implied by the relation of causality. The immediate successors and predecessors of a vertex $g$ in a graph $G$ are noted $\text{succ}_g(G)$ and $\text{pred}_g(G)$ and their transitive closures $\text{succ}_g^*(G)$ and $\text{pred}_g^*(G)$, respectively. The minimal and maximal vertices of a graph $g$ are noted $\text{min}(G)$ and $\text{max}(G)$. The neighbors of $g$ in $G$ are $\text{next}_g(G) = \bigcup_{h \in \text{pred}_g^*(G)} \text{succ}_h(G)$.

$$\text{pred}_g(G) = \{ h \in G | h \succ g \}$$
$$\text{min}(G) = \{ g \in G | \forall h \in G, h \not\succ g \}$$
$$\text{next}_g(G) = \{ g \in G | \forall h \in G, g \not\prec h \}$$

Definition 7. The grammar $M$ is defined from the signal vertices of a graph $S$ by the function fork$(S)$. We write fork$_g(S)$ for the prefix of $g$ in the grammar fork$(S)$.

$$\text{fork}(S) = \text{let } o_{m=1}^{\text{fork}(S)} = (\text{fork}(S)) \text{in } o_{m=1}^{\text{fork}(S)}$$

The partition $S_{\#}$ of $(S_i)_{i=1}^{\text{vars}(p)}$ of a set of independent signal vertices $S$ into exclusive vertices is defined by $S = \bigcup_{i=1}^{\text{vars}(p)} S_i$ and, for all $0 < i, j < n$, for all $(g, b) \in S_i \times S_j$, $(g\# h \Leftrightarrow i = j)$ and $(g \# h \Leftrightarrow i \neq j)$.

A sequential component of $M$ is a sub-grammar that does not contain a concurrency or synchrony relation (only $\succ$ or $\#$). A thread $T$ is a sequence of signal vertices that represents the possible scheduling of a transition. We call $|T|$ its length and $T_i$ its $i$th element. The threads $\text{join}(M)$ of a grammar $M$ are constructed by structural induction.

$$\text{join}(g) = g \text{ join}(M \# N) = \text{join}(M) \cup \text{join}(N)$$
$$\text{join}(M \circ N) = \text{join}(T \cup T) \cup \text{join}(M \times \text{join}(N))$$
$$\text{join}(M \sim N) = \text{join}(M \# N \# (N \sim M))$$
$$\text{join}(M \sim N) = \text{join}(M \# N \# (N \# M))$$

5. DECISION PROCEDURES

The control and data-flow analysis of a process define an event structure represented by a grammar $M_p$ that allows us criteria upon which a process $p$ can be guaranteed to be weakly-endochronous. First, we check that a process $p$ deterministic and step independent amounts to the satisfaction of clock relations in $C_p$, some of which being related to the causal structure implied by its graph $G_p$.

Definition 8. Let $(g, h)$ two signal vertices of a graph $G_p$. If $g \succ h$ then $(g \land h) \Rightarrow \text{def}(g) \Rightarrow \text{def}(h)$. If $e_1 \Rightarrow x \Rightarrow y$ and $e_2 \Rightarrow x \Rightarrow y$ then $(e_1 \land e_2) \Rightarrow x \Rightarrow y$. If $e_1 \Rightarrow x \Rightarrow y$ and $e_2 \Rightarrow y \Rightarrow z$ then $(e_1 \land e_2) \Rightarrow x \Rightarrow z$. A process $p$ is schedulable iff $e \Rightarrow x \Rightarrow x$ implies $C_p \models e = 0$ for all $x$.

A process $p$ is predictable iff its clocks are computable and all distinct vertices $g \neq h$ of $S_p$ such that $\text{def}(g) = \text{def}(h)$ are exclusive $g\# h$.

Second, grammars provide the necessary framework to finitely explore the state-space of a process and define decision procedures for the properties of clock and choice-independence.

Definition 9. Thread $T$ is compatible with $U$ on $I$, written $T \triangleright_I U$, iff for all $x \in I$ and $0 < i \leq |T|$, there exists $0 < j \leq |U|$ s.t. $T_i \Rightarrow x \land U_j \Rightarrow x$ and $T_i \land U_j \neq 0$.

Two threads $T$ and $U$ are compatible on $I$, written $T \triangleright_I U$, iff $T \triangleright_I U$ and $U \triangleright_I T$. Two grammars are compatible, written $M \triangleright_I N$, iff $T \triangleright_I U$ for all $(T, U) \in \text{join}(M) \times \text{join}(N)$. Two processes $p$ and $q$ are compatible $I = \text{vars}(p) \cap \text{vars}(q)$ iff $M_p \triangleright_I M_q$.

We write $S_I$ for the restriction of $S$ on vertices that have used or defined variables in $I$.

Definition 10. A process $p$ is conflit-free iff, for all $g, h \in S = (S_i)_{i=1}^{\text{vars}(p)}$, (1) if $g \prec h$ then for all $(g', h') \in \text{pred}_p(S) \times \text{pred}_p(S)$, $g' \not\succ h'$, use$(g') \cap \text{use}(h') \cap \text{vars} = \emptyset$ and $g' \not\leq h'$ and (2) if $g\# h$ and for$k_S(S) \Rightarrow \text{vars}(p)$ for$k_S(S)$ then there exists $(g', h') \in \text{next}_p(S) \times \text{next}_k(S)$, $g \sim h'$ and $h \sim g'$.

A conflict-free process $p$ is clock and choice independent: for instance, the grammar of the switch restricted to its free variables is conflict-free: $(g^1 \# g^2) \circ (g^1 \# g^2)$). At last, Definition 11 summarizes the above and defines a notion of weak controllability that is sufficient for a process to be weakly endochronous and a pair of processes weakly isochronous.

Definition 11. A process $p$ is weakly controllable iff it is schedulable, predictable and conflict-free. Two processes $p$ and $q$ are mutually controllable iff $p$ and $q$ are compatible and $p \mid q$ is schedulable.

Our main result is in accordance to this definition.

Theorem 2. If $p$ is weakly controllable then $A_p$ is weakly endochronous. If $p$ and $q$ are weakly and mutually controllable then $A_p$ and $A_q$ are weakly isochronous.

6. CONCLUSIONS

The micro-step automata theory of [3], and with the data-structure and analysis we introduce, allow us to attack a central issue of desynchronization: to locally and compositionally minimize synchronization constraints. It precludes optimized code generation and communication synthesis schemes to be accordingly defined. In this aim, a subsequent technical report [4] defines a procedure for compositionally synthesizing a weakly-endochronous automaton and modularly generating C-like code starting from the intermediate representation of a weakly controllable process $p$.

7. REFERENCES


