Externalisation of Time-Triggered communication system in BIP high level models

Hela Guesmi, Belgacem Ben Hedia, Simon Bliudze, Saddek Bensalem

Externalisation of Time-Triggered communication system in BIP high level models

Hela Guesmi, Belgacem Ben Hedia
CEA, LIST
firstname.lastname@cea.fr

Simon Bliudze
EPFL
Simon.bliudze@epfl.ch

Saddek Bensalem
Verimag, UJF
Saddek.Bensalem@imag.fr

ABSTRACT

To target a wider spectrum of Time-Triggered(TT) implementations of hard real-time systems, we consider approaches for building component-based systems that provide a physical model from a high-level model of the system and TT specifications. The obtained physical model is thus suitable for direct transformation into languages of specific TT platforms. In addition, if these approaches provide correctness-by-construction, they can help to avoid the monolithic a posteriori validation.

In this paper, we focus on the TT interface concept of the TT paradigm. And we present a method that transforms the interaction in classic BIP (Behavior, Interaction, Priority) Model into a TT interface by source-to-source transformations. The method is based on the successive application of two types of source-to-source transformations; Transfer functions internalisation and $n+1$-ary connector to TT interface transformation. The first simplifies the connector transfer functions by modifying components automata. The second transforms connector with simple transfer function into TT interfaces.

Keywords

TT paradigm; correctness-by-construction; Source-to-source transformation; BIP; interaction expressions; connectors;

1. INTRODUCTION

In hard real time computer systems, correctness of a result depends on both the time and the value domains.
With the increasing complexity of these systems, ensuring their correctness using a posteriori verification becomes, at best, a major factor in the development cost and, at worst, simply impossible. An error in the specifications is not detectable. We must, therefore, define a structured and simplified design process which allows the development of correct-by-construction system. Thereby, monolithic a posteriori verification can be avoided as much as possible.

Two fundamentally different paradigms for the design of real-time systems are identified; Event-Triggered(ET) and TT paradigms. In ET paradigm, all communication and processing activities are initiated whenever a considerable event, i.e., change of state in the observed variable, is noticed. It doesn't cope with demands for predictability and determinism that must be met in hard real-time systems. Activities in TT paradigm are initiated periodically at predetermined points in time. These statically defined activation instants enforce regularity and make TT systems more predictable than ET systems. This approach is well-suited for hard real-time systems.

A system model of this paradigm is essential to speed-up understanding and smooth design task. It requires explicitly manipulating not only the value domain specifications, but also temporal constraints for which high abstraction level primitives are not provided. Kopetz [7] presents a TT-Model of computation, based on essential properties of the TT paradigm: **the global notion of time** that must be established by a periodic clock synchronization in order to enable a TT communication and computation, **the temporal structure of each task**, consisting of predefined start and worst-case termination instants attributed statically to each task and **TT interfaces** which is a memory element shared between two interfacing subsystems. TT-Model separates the design of interactions between components from the design of the components themselves.

To target a wider spectrum of TT implementations, we consider approaches for building component-based systems that provide a physical model from a high-level model of the system and TT specifications. In addition, if these approaches provide correctness-by-construction, they can avoid the monolithic a posteriori validation. We focus in particular on the framework BIP [1]. It is a component framework for constructing systems by the superposition of three layers: Behaviour, Interaction, and Priority. The Behaviour layer consists of a set of atomic components represented by transition systems. The second layer describes possible interactions between atomic component. Interactions are set of ports and are specified by a set of connectors. The third layer includes priorities between interactions using mechanisms for conflict resolution. In this paper, we consider Real-Time BIP version [2] where atomic components are represented by timed automata. We limit ourselves to connectors and leave priorities for future work.

From a high-level BIP system model, a physical model containing all TT concepts (such as TT interfaces, the global notion of time and the temporal structure of each task) is generated using a set of source-to-source transformations. This physical model (called also BIP-TT model) is then translated to the programming language specific to the particular TT platform. The program in this language is then compiled using the associated compilation chain. Thus, BIP-TT model is not dedicated to an exemplary architecture.

There have been a number of approaches exposing the relevant features of the underlying architectures at high level design tool. [8] presents a design framework based on UML diagrams for applications running on Time Triggered Architecture(TTA). This approach doesn't support earlier architectural design phase and needs a backward mechanisms for the generated code verification. Since BIP design flow is unique due to its single semantic framework used to support application modelling and to generate correct-by-construction code, many approaches tend to use it to translate high level models into physical models including architectural features. In [5], a distributed BIP model is generated from a high level

one. In [4], a method is presented for generating a mixed hardware/software system model for many-core platforms from an application software and a mapping. These two approaches take advantages from BIP framework but they do not address the TT paradigm. To the best of our knowledge, our approach is the first to address the problem of generating TT application from BIP high level models.

In this paper we address the issue of source-to-source transformations that explicit TT communications in the physical model, in BIP framework. Other TT concepts (the global synchronized time and task temporal structure) transformations are beyond the scope of this paper.

The remainder of this paper is structured as follows: Section 2 introduces BIP framework and explains the relevant TT concepts. Section 3, presents a method using a set of source-to-source transformations for generating a BIP model expliciting TT communication interfaces, from a high level classic BIP model. In Section 4, we conclude the paper by discussing advantages and downsides of our method.

2. RELATED CONCEPTS

In this section, we present first the basic semantic model of BIP, and main TT concepts that must clearly appear in the final BIP-TT model.

2.1 The BIP component framework

In the BIP framework, for each layer, a concrete model is provided. Atomic components model the behaviour layer. The interaction layer is modelled with connectors and finally Priorities is a mechanism for scheduling interactions.

An atomic component consists of a timed automaton with local data and an interface consisting of ports. Transitions in the component automaton are labelled by ports and can execute C code to transform local data. Let $\mathcal{P}$ be a set of ports. We assume that every port $p \in \mathcal{P}$ has an associated data variable x_p. This variable is used to exchange data with other components, when interactions take place.

Definition 1. ***(atomic component):***
An atomic component B is defined by $B = (L, P, T, X, \{g_\tau\}_{\tau \in T}, \{f_\tau\}_{\tau \in T})$, where,

- *(L, P, T) is a labelled transition system, that is:*
  - *L is a set of control states*
  - *P is a set of communication ports,*
  - *$T \subseteq L \times 2^P \times L$ is a set of transitions*
- *X is a set of variables and for each transition $\tau \in T$, g_τ is a guard and f_τ is an update function that is state transformer defined on X.*

Interactions which are sets of ports allowing synchronizations between components, are defined and graphically represented by connectors. The execution of interactions may involve transfer of data between the participating components. For every interaction, data transfer functions of an interaction a are specified by an Up and a $Down$ actions. The action Up is supposed to update the local variables of the connector, using the values of variables associated with the ports. Conversely, the action $Down$ is supposed to update the variables associated with the ports, using the values of the connector variables.

Definition 2. *(Connector) A connector γ defines sets of ports of atomic components B_i which can be involved in an interaction a. It is formalized by $\gamma = (P, a, q, g, Up, Down)$ where:*

- *P is the support set of synchronized ports of γ with $P = a$*
- *q is its exported port.*
- *g is the boolean guard expression.*
- *Up is the upward transfer function of the form $x_q := Up(\{x_p\}_{p \in a})$,.*
- *and $Down$ is the downward transfer functions of the form $x_p := Down_p(x_q)$ for each $p \in a$.*

The interaction presented by this connector is of the form:

$$(q \leftarrow a).[g(\{x_p\}_{p \in a}) : x_q := Up(\{x_p\}_{p \in a}) // x_{p \in a} := Down_p^d(x_q)]$$

2.2 TT Paradigm [6, 7]

TT paradigm encompasses these 3 key concepts;

The global synchronized time: It allows definition of instances when communication and computation of tasks take place in a TT system. It is established by a periodic clock synchronization from which other clocks can be derived.

The temporal control structure of the task sequence: The TT paradigm is based on a set of static schedules. These schedules have to provide an implicit synchronization of the tasks at run time. This introduces a fixed task activation rates during system design. Thus to each task is allocated predefined start instant (T_b) and the worst-case termination instant (T_e). These instants are triggered by the progression of the global time.

Time-Triggered interface(Firewall): It is a data-sharing boundary between two communicating subsystems. Exchanged messages are state messages, informing about the state of the relevant variable at a particular point in time. A new version of a state message overwrites the previous version. State messages are not consumed on reading and they are produced periodically at predetermined points in real-time. Thus TT interfaces contain real-time data which is a valid image of the observed variable.

These three notions should clearly appear in the final BIP-TT model to facilitate its translation into the programming language specific to the particular TT platform.

3. TIME-TRIGGERED ARCHITECTURES IN BIP

The methodology that integrates TT concepts in BIP, is based on the transformation of an arbitrary BIP model with additional TT annotations (task, TT interfaces) into more restricted models called BIP-TT, which are suitable for direct transformation into languages of specific TT platforms.

In order to understand the transformation process of a BIP model into BIP-TT one, we present first the original BIP and final BIP-TT models and then we detail the transformation rules that transform the former into the latter.

3.1 The original BIP Model

We assume that the considered original BIP model consists only of atomic components and flat connectors, example cf.Figure1. Indeed, these assumptions do not impose restrictions on the components since we can use the "component flattening" transformation [5] to replace every composite component by its equivalent set of atomic components.

Figure 1 shows a BIP model, made up of five atomic components executing four different tasks. We assume that a task is a set of elementary actions. Thus two or more components can execute separately elementary actions belonging to the same task. Each component is annotated by the

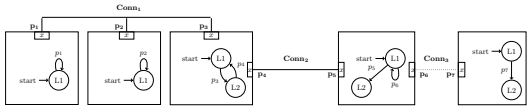
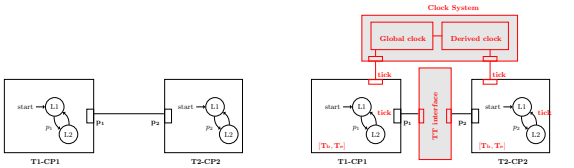


Figure 1: High level BIP model

task it is executing and the component identifier. Take for example the first component, annotated by "T1-CP1", i.e., "CP1" is its identifier and "T1" is the executed task identifier. Two different components may execute the same task, e.g components "CP4" and "CP5". The connector relating such components is shown by dotted lines. To simplify the presentation of figures' automata in this paper, the temporal aspect is not displayed.

3.2 BIP_TT Model

The final BIP-TT Model presents a hand-made translation of the TT paradigm, introduced by Kopetz, into a BIP model. It clearly includes TT three main concepts.
Figure 2b shows roughly how should be the BIP-TT model of the BIP model of Figure 2a. Red components are BIP components and presents TT concepts.



(a) Original BIP model (b) Final BIP-TT model

Figure 2: Modelling TT paradigm in BIP

As here we settle for studying source-to-source transformations to obtain TT interfaces from BIP connectors, we model in Figure 3 the TT interface in BIP. It is an atomic two-port component which behaviour is modelled by a labelled automaton with one state and two transitions, one for reading action (labelled by the port W_{ITT}) and one for writing(labelled by the port R_{ITT}).

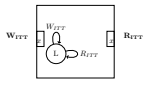


Figure 3: BIP model of the TT interface

3.3 Transformations from BIP classic model to BIP-TT model: from the communication concept point of view

The high level BIP model refinement process is based on the operational semantics of BIP [3] which allows to compute the meaning of a BIP model with simple connectors as a behaviourally equivalent BIP model that contains TT interfaces cf.Figure 3. The transformation process follows these two steps: ***1)*** Transfer functions internalisation and ***2)*** $n+1$-ary connector to TT interface transformation $\mathcal{F}_n$.

These two transformations are described in reverse, from the most specific to the most general $N-$ary connector case. We use the high level BIP model in Figure 1 as a running example throughout the paper to illustrate these transformation rules.

3.3.1 $\mathbf{n+1}$-ary connector to TT interface transformation ($\mathcal{F}_\mathbf{n}$)

This transformation is applied only on $n+1-$ary connector with only one writer, n readers, and with simple assignation transfer functions, i.e., we just copy the value of the associated variable to the writer port in the local variable of the connector (the Up function), and copy the latter in readers' ports' variables($Down$ functions). Note that this behaviour is similar to the TT interface one which is used to make and transfer copy from the producer to consumers. We denote this transformation function by $\mathcal{F}_n$, it transforms an $n+1-$ary connector $C = (P_C, a_C, q_C, g_C, Up_C, Down_C)$, in the source model, into the triplet; binary connector C_1, TT interface I_{TT}, and an $n-$ary connector C_2^n, in the resulting model. These are defined below in function of the initial connector C. Let P_C be the set of ports of the connector C such as $P_C = \{p_{W_C}, \{p_{R_{Ci}}\}_{i\in[1..n]}\}$.

Rule 1. $\mathbf{C_1}$

C_1 is formalized by $C_1 = (P_1, a_1, q_1, g_1, Up_1, Down_1)$. The interaction presented by this connector is then of the form:

$$(q_1 \leftarrow a_1 = \{p_{W_C}, p_{W_{ITT}}\}) \ . \ [g_C(x_{p_{W_C}}) : x_{q_1} := Up_1(x_{p_{W_C}}) = x_{p_{W_C}} // x_{p\in a_1} := Down_1(x_{q_1}) = x_{q_1}]$$

Rule 2. $\mathbf{I_{TT}}$

The atomic component $I_{TT} = (L, P, T, X, \{g_\tau\}_\tau \in T, \{f_\tau\}_\tau \in T)$ where $L = \{l\}$, $P = \{p_{W_{ITT}}, p_{R_{ITT}}\}$, T is a set of the two possible transitions, each labeled by one of the two ports.

Rule 3. $\mathbf{C_2^n}$

C_2^n is formalized by $C_2^n = (P_2^n, a_2, q_2, g_2, Up_2, Down_2)$. The interaction presented by this connector is of the form:

$$(q_2 \leftarrow a_2 = \{p_{R_{ITT}}, \{p_{R_{Ci}}\}_{i\in[1..n]}\}) \ . \ [g_C(\{x_{p_{R_{Ci}}}\}_{i\in[1..n]}) : x_{q_2} := Up_2(x_{p_{R_{ITT}}}) = x_{p_{R_{ITT}}} // x_{p\in a_2} := Down_2(x_{q_2}) = x_{q_2}]$$

Example 1. *If we suppose that there exists only one writer among the first three components in the example of Figure 1 (for example CP1), then this transformation will transform connectors $Conn_1$ (using $\mathcal{F}_2$) and $Conn_2$ (using $\mathcal{F}_1$) as shown in Figure 4.*

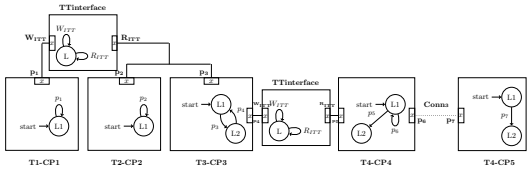


Figure 4: $Conn_1$ and $Conn_2$ connectors to TT interfaces transformation

3.3.2 Transfer functions internalisation

This transformation takes an arbitrary $N-$ary connector with transfer functions different from the simple assignation and produces a connector with simple assignations transfer functions. Then the transformation function $\mathcal{F}_n$ can be applied on the obtained connector. Up and $Down$ functions are internalised by modifying components' automata. In this transformation readers and writers are detected. Suppose that there are m writers, $m \geq 1$ and n readers, $N \leq n+m$, i.e., a component can be both reader and writer. One component writer is randomly chosen to be "the maestro" (in the rest of the paper the maestro is the m^{th} writer). It is then connected to all the rest of writers via $m-1$ binary connectors, so that to aggregate all their data, and to readers via an $n-$ary connector.

Automata of Writers $W_{j,j\in[1,m-1]}$ and readers $R_{i,i\in[1,n]}$, are modified so that to internalize their concerned $Down$ functions. The maestro M component and automaton are modified by adding ports, variables, states and transitions. We denote their refined models respectively by $W^r_{j,j\in[1,m-1]}$,

$R^r_{i,i\in[1,n]}$and M^r. Thus the initial connector $C = (P_C, a_C, q_C, g_C, Up_C, Down_C)$ is split into $m-1$ binary connectors $C^b_{i,i\in[1,m-1]}$ if $m > 1$, and an $n-$ary connector C^n.

We denote the sets of ports and interactions of the initial connector C respectively by $P_C = \{\{p_{W_i}\}_{i\in[1..m]} \bigcup \{p_{R_j}\}_{j\in[1..n]}\}$ and a_C. Ports p_M and p_{R_i} are respectively ports of the maestro and the component R_i involved in the interaction a_C. The derived connectors after transformation and the refined components are defined below.

Rule 4. ***Connector*** $\mathbf{C^b_{i,i\in[1,m-1]}}$

C^b_i is formalized by $C^b_i = (P^b_i, a^b_i, q^b_i, g^b_i, Up^b_i, Down^b_i)$.

The interaction presented by this connector is then of the form: $(q^b_i \leftarrow a^b_i = \{p_{W_i}, p_{M_i}\}) \cdot [g_C(x_{p_{W_i}}) : x_{q^b_i} := Up^b_i(x_{p_{W_i}}) = x_{p_{W_i}} // x_{p_{M_i}} := Down^b_i(x_{q^b_i}) = x_{q^b_i}]$

C^n connector, relating the maestro writer component to the n reader components is defined below;

Rule 5. ***Connector*** $\mathbf{C^n}$

C^n is formalized by $C^n = (P^n, a^n, q^n, g^n, Up^n, Down^n)$, where:

The interaction presented by this connector is then of the form: $(q^n \leftarrow a^n = \{p_M, \{p_{R_j}\}_{j\in[1..n]}\}) \cdot [g_C(\{p_{R_j}\}_{j\in[1..n]}) : x_{q^n} := Up^n(x_{p_M}) = x_{p_M} // x_{p_{R_j}} := Down^n(x_{q^n}) = x_{q^n}]$

We now present how we transform a writer component M in original BIP model, into a maestro component M^r that is capable to aggregate all other writers data, to internalize Up transfer function of the initial connector and then to send the result of this function to readers. The maestro component M^r, has $m-1$ ports p_{M_i} allowing its connection with the rest of writers and a port p_M relating the maestro to readers. Old exported variable x is kept as a local variable, and a new variable z is associated with the port p_M. To be able to internalize the Up function of the initial connector, $m-1$ states and transitions are added before each transition labelled by the port p_M. Each new transition is labelled by a port P_{M_i}. Then Up function is executed in the last new transition. Then, after executing the interaction involving P_M port, we copy z variable to x variable. Figure 5 shows an example of the maestro transformation, in case of a connector with 2 writers $m = 2$, and which transfer functions are Up and $Down$.

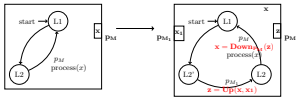


Figure 5: Example of a writer to a maestro transformation with $m = 2$

Example 2. *Based on the example model of figure1, we suppose that the connector $Conn_1$ have the following transfer functions. The transfer functions of the connector C_2 are simple assignations; Up :* $x_{Conn_1} = U(x_{p_1})$ *and* $Down : x_{p_1} = D_1(x_{Conn_1}), x_{p_2} = D_2(x_{Conn_1})$ *and* $x_{p_3} = D_3(x_{Conn_1})$.

By applying the transformation to this connector, we obtain the model of Figure 6. Since the initial model contains just one writer, the connector topology remains intact, only its transfer functions and component behaviours are modified in that example. Functions U and D_1 will be integrated to $CP1$ component. D_2 (resp. D_3) function will be internalized in $CP2$ (resp. $CP3$). In each component, we export a new variable z (instead of x) in ports p_i, $i \in [1,3]$. For down functions (D_1, D_2 and D_3), we add a C function in every transition labelled by port p_i. This function is of the form $x = D_i(z)$, $i \in [1,3]$. Concerning Up function, a state and a transition are added before each transition labelled by the writing port p_1 in the component $T1 - CP1$. The new transition executes a C function of the form $z = U(x)$. The new connector $Conn'_1$ have the following transfer functions: $Up :$ $x_{Conn'_1} = z_{p_1}$ *and* $Down : z_{p_1} = z_{p_2} = z_{p_3} = x_{Conn'_1}$.

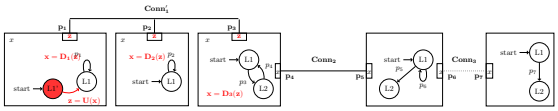


Figure 6: $Conn_1$ transfer functions internalisation

4. DISCUSSION & CONCLUSION

BIP connectors, can be transformed into TT interfaces by successive application of two types of source-to-source transformations; **Transfer functions internalisation** and **n + 1-ary connector to TT interface transformation**. The first simplifies the connector transfer functions by modifying components automata while keeping the same general behaviour of the model. The second transforms connector with simple transfer functions to TT interfaces.

The major asset of these source-to-source transformations, is that we don't add new components requiring adding new tasks, a part from TT interfaces. These transformations focus on transforming atomic components by adding new ports, new variables and extending automata with new states and transitions. The number of added states strongly depends on the number of writers in the model and the number of transitions labeled by the port involved in the interaction.

For that we propose in our future work to study different cases and to decide whether to modify components automata or add a task that orchestrates all interactions without altering components' automata. Then, based on system constraints, a trade-off can be defined.

5. REFERENCES

[1] *BIP2 Documentation*, July 2012.

[2] T. Abdellatif, J. Combaz, and J. Sifakis. Model-based implementation of real-time applications. pages 229–238, May 2010.

[3] A. Basu, P. Bidinger, M. Bozga, and J. Sifakis. Distributed semantics and implementation for systems with interaction and priority. In *Formal Techniques for Networked and Distributed Systems–FORTE 2008*, pages 116–133. Springer, 2008.

[4] P. Bourgos. Rigorous design flow for program-ming manycore platforms.

[5] M. Bozga, M. Jaber, and J. Sifakis. Source-to-source architecture transformation for performance optimization in bip. *Industrial Informatics, IEEE Transactions on*, 6(4):708–718, 2010.

[6] H. Kopetz. The time-triggered approach to real-time system design. *Predictably Dependable Computing Systems. Springer*, 1995.

[7] H. Kopetz. The time-triggered model of computation. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pages 168–177. IEEE, 1998.

[8] K. D. Nguyen, P. Thiagarajan, and W.-F. Wong. A uml-based design framework for time-triggered applications. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 39–48. IEEE, 2007.