Formal Specification of the HILECOP Model-to-text Transformation

Vincent Iampietro

Formal specification of the HILECOP model-to-text transformation

VINCENT IAMPIETRO

May 9, 2022

1 Introduction

This document serves as a reference for the formal specification of the HILECOP model-to-text transformation. HILECOP is a methodology for the design and synthesis of safety-critical digital systems [1]. Many elements used in the following definitions are introduced in the following thesis [2].

2 Preliminary definitions

The HILECOP model-to-text transformation (HM2T) generates a $\mathcal{H}$-VHDL design [2, p.65] out of an SITPN model [2, p.37]. It also generates a structure that relates the elements of the SITPN model to the elements of the $\mathcal{H}$-VHDL design. This structure is called a SITPN-to-$\mathcal{H}$-VHDL binder. Its formal definition is as follows:

Definition 1 (SITPN-to-$\mathcal{H}$-VHDL design binder)**.** *Given a sitpn* $\in SITPN$ *and a* $\mathcal{H}$*-VHDL design* $d \in design$*, a SITPN-to-*$\mathcal{H}$*-VHDL binder* $\gamma \in WM(sitpn, d)$ *is a tuple* $<PMap, TMap, CMap, AFMap>$ *where:*

- $PMap \in P \rightarrow \{id_p \mid \texttt{comp}(id_p, \texttt{place}, g, i, o) \in d.beh\}$
- $TMap \in T \rightarrow \{id_t \mid \texttt{comp}(id_t, \texttt{transition}, g, i, o) \in d.beh\}$
- $CMap \in \mathcal{C} \rightarrow \{id_c \mid (\texttt{in}, id_c, \texttt{bool}) \in d.ports\}$
- $AFMap \in \mathcal{A} \cup \mathcal{F} \rightarrow \{id_{af} \mid (\texttt{out}, id_{af}, \texttt{bool}) \in d.ports\}$

Notation 1. *For a given binder* γ *and an element of an SITPN structure* $e \in P \cup T \cup \mathcal{C} \cup \mathcal{A} \cup \mathcal{F}$*, we write* $\gamma(e)$ *where* e *is looked up in the appropriate function. For instance, for a given* $f \in \mathcal{F}$*,* $\gamma(f)$ *is a shorthand notation for* $AFMap(f)$ *where* $\gamma = <\ldots, AFMap>$*.*

While presenting the specification of the HM2T, we refer to two pre-defined designs, namely: the place and transition designs. The details of the implementation of the place and transition designs in $\mathcal{H}$-VHDL are given in Appendices A and B. The HILECOP design store has been specifically defined for the elaboration and simulation the $\mathcal{H}$-VHDL designs generated by the HM2T. It holds the definitions of the place an transition designs.

3 Formal specification

In the following section, we present the formal specification of the HM2T. The specification is a relation between the inputs of the HM2T, namely a SITPN model and a bounding function, and the possible outputs which can be either a pair composed of a $\mathcal{H}$-VHDL design and a SITPN-to-$\mathcal{H}$-VHDL binder or the error value. The relation is written $HM2T_{\texttt{spec}} \subseteq SITPN \times (P \to \mathbb{N}) \times ((design \times WM(sitpn, d)) \cup \{\texttt{err}\})$.

Definition 2 (HILECOP model-to-text transformation specification)**.** *For all SITPN model $sitpn \in SITPN$, bounding function $b \in P \to \mathbb{N}$, $\mathcal{H}$-VHDL design $d \in design$, and SITPN-to-$\mathcal{H}$-VHDL binder $\gamma \in WM(sitpn, d)$, we have $HM2T_{\texttt{spec}}\big(sitpn, b, (d, \gamma)\big)$ if:*

1. *Design d has an empty generic clause: $d.gens = \emptyset$.*
2. *The number of input ports of the design is equal to the number of conditions of the SITPN model: $|\{id \mid (\texttt{in}, id, \tau) \in d.ports\}| = |\mathcal{C}|$.*
3. *The number of output ports of the design is equal to the number of actions and functions of the SITPN model: $|\{id \mid (\texttt{out}, id, \tau) \in d.ports\}| = |\mathcal{A} \cup \mathcal{F}|$.*
4. *Design d is elaborable in the context of the HILECOP design store and given an empty dimensioning function: $\exists \Delta \in ElDesign, \sigma_e \in \Sigma$ s.t. $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash d \xrightarrow{elab} \Delta, \sigma_e$.*
5. *All design instantiation statement in the design's behavior either creates a PDI or a TDI: $\forall id_c, id_e, g, i, o$ s.t. $\texttt{comp}(id_c, id_e, g, i, o) \in d.beh,\ id_e = \texttt{place} \vee id_e = \texttt{transition}$.*
6. *The `actions` and `functions` processes are the only two processes in the design's behavior: $\forall id_p, vars, ss$ s.t. $\texttt{ps}(id_p, vars, ss) \in d.beh,\ id_p = \texttt{actions} \vee id_p = \texttt{functions}$.*
7. *All the fields of the SITPN-to-$\mathcal{H}$-VHDL binder are bijective functions: $PMap(\gamma)$ is bijective, $TMap(\gamma)$ is bijective,...*
8. *For all place of the input SITPN model, there exists a corresponding place design instance (PDI) identified through γ in the behavior of the output design: $\forall p \in P, \exists g_p, i_p, o_p$ s.t. $\texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh$.*
9. *For all place of the input SITPN model and its corresponding PDI, the generic map of the PDI holds the following associations: $\forall p \in P, g_p, i_p, o_p, \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow$*
$$g_p = \{(\texttt{mm} \Rightarrow b(p)), (\texttt{ian} \Rightarrow \begin{cases} 1 \text{ if } \texttt{input}(p) = \emptyset \\ |\texttt{input}(p)| \text{ otherwise} \end{cases}), (\texttt{oan} \Rightarrow \begin{cases} 1 \text{ if } \texttt{output}(p) = \emptyset \\ |\texttt{output}(p)| \text{ otherwise} \end{cases})\}$$
where $\texttt{input}(p) = \{t \mid post(t, p) = \lfloor \omega \rfloor\}$, the set of input transitions of place p, and $\texttt{output}(p) = \{t \mid pre(p, t) = \lfloor (\omega, a) \rfloor\}$, the set of output transitions of place p.
10. *For all place of the input SITPN model and its corresponding PDI, there is an association between the `im` input port and the initial marking of the place in the input port map of the PDI: $\forall p \in P, g_p, i_p, o_p, \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow (\texttt{im} \Rightarrow M_0(p)) \in i_p$.*

11. *For all place of the SITPN model with no input transition, the input port map of the corresponding PDI includes the following associations:*

$$\begin{aligned}&\forall p \in P, g_p, i_p, o_p,\\ &\quad \texttt{input}(p) = \emptyset \Rightarrow\\ &\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow\\ &\quad \{(\texttt{iaw}(0) \Rightarrow 0), (\texttt{itf}(0) \Rightarrow \texttt{false})\} \subseteq i_p.\end{aligned}$$

12. *For all place of the SITPN model with no output transition, the input port map and output port map of the corresponding PDI includes the following associations:*

$$\begin{aligned}&\forall p \in P, g_p, i_p, o_p,\\ &\quad \texttt{output}(p) = \emptyset \Rightarrow\\ &\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow\\ &\quad \{(\texttt{oaw}(0) \Rightarrow 0), (\texttt{oat}(0) \Rightarrow \texttt{basic}), (\texttt{otf}(0) \Rightarrow \texttt{false})\} \subseteq i_p\\ &\quad \wedge \{(\texttt{oav} \Rightarrow \texttt{open}), (\texttt{pah} \Rightarrow \texttt{open}), (\texttt{rtt} \Rightarrow \texttt{open})\} \subseteq o_p.\end{aligned}$$

13. *For all place of the SITPN model with no action, the* `marked` *output port is left unconnected in the output port map of the corresponding PDI:*

$$\begin{aligned}&\forall p \in P, g_p, i_p, o_p,\\ &\quad \texttt{acts}(p) = \emptyset \Rightarrow\\ &\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow\\ &\quad (\texttt{marked} \Rightarrow \texttt{open}) \in o_p\end{aligned}$$

where $\texttt{acts}(p) = \{a \mid \mathbb{A}(p, a) = \texttt{true}\}$*, the set of actions associated with place* p*.*

14. *For all transition of the input SITPN model, there exists a corresponding TDI identified through* γ *in the behavior of the output design:*
$\forall t \in T, \exists g_t, i_t, o_t$ *s.t.* $\texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh$*.*

15. *For all transition of the input SITPN model and its corresponding TDI, the generic map of the TDI holds the following associations:*

$$\begin{aligned}&\forall t \in T, g_t, i_t, o_t,\\ &\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow\\ &\quad g_t = \{(\texttt{tt} \Rightarrow \begin{cases}\texttt{NOT\_TEMPORAL if } t \notin \texttt{dom}(I_s)\\ \texttt{TEMPORAL\_A\_A if } I_s(t) = [a, a]\\ \texttt{TEMPORAL\_A\_B if } I_s(t) = [a, b]\\ \texttt{TEMPORAL\_A\_INF if } I_s(t) = [a, \infty]\end{cases}), (\texttt{mtc} \Rightarrow \begin{cases}1 \text{ if } t \notin \texttt{dom}(I_s)\\ b \text{ if } I_s(t) = [a, b]\\ a \text{ if } I_s(t) = [a, \infty]\end{cases}),\\ &\quad (\texttt{ian} \Rightarrow \begin{cases}1 \text{ if } \texttt{input}(t) = \emptyset\\ |\texttt{input}(t)| \text{ otherwise}\end{cases}), (\texttt{cn} \Rightarrow \begin{cases}1 \text{ if } \texttt{conds}(t) = \emptyset\\ |\texttt{conds}(t)| \text{ otherwise}\end{cases})\}.\end{aligned}$$

where $\texttt{input}(t) = \{p \mid pre(p, t) = \lfloor(\omega, a)\rfloor\}$*, the set of input places of transition* t*, and* $\texttt{conds}(t) = \{c \mid \mathbb{C}(t, c) = 1 \vee \mathbb{C}(t, c) = -1\}$*, the set of conditions associated with transition* t*.*

16. *For all transition of the input SITPN model and its corresponding TDI, the input port map of the TDI holds the following associations:*

$$\begin{aligned}
&\forall t \in T, g_t, i_t, o_t, \\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\quad \{(\texttt{A} \Rightarrow \begin{cases} 0 \text{ if } t \notin \texttt{dom}(I_s) \\ l(I_s(t)) \text{ otherwise} \end{cases}), (\texttt{B} \Rightarrow \begin{cases} 0 \text{ if } t \notin \texttt{dom}(I_s) \vee u(I_s(t)) = \infty \\ u(I_s(t)) \text{ otherwise} \end{cases})\} \subseteq i_t.
\end{aligned}$$

17. *For all transition of the input SITPN model with no input place, the input port map and output port map of the corresponding TDI holds the following associations:*

$$\begin{aligned}
&\forall t \in T, g_t, i_t, o_t, \\
&\quad \texttt{input}(t) = \emptyset \Rightarrow \\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\quad (\exists id_s \ s.t. \ (id_s, \texttt{bool}) \in d.sigs \wedge (\texttt{rt}(0) \Rightarrow id_s) \in i_t \wedge (\texttt{fired} \Rightarrow id_s) \in o_t) \\
&\quad \wedge \{(\texttt{iav}(0) \Rightarrow \texttt{true}), (\texttt{pah}(0) \Rightarrow \texttt{true})\} \subseteq i_t.
\end{aligned}$$

18. *For all transition of the input SITPN model with no condition, the input port map of the corresponding TDI holds the following association:*

$$\begin{aligned}
&\forall t \in T, g_t, i_t, o_t, \\
&\quad \texttt{conds}(t) = \emptyset \Rightarrow \\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\quad (\texttt{ic}(0) \Rightarrow \texttt{true}) \in i_t.
\end{aligned}$$

19. *For all post arc of the input SITPN model, the TDI and PDI corresponding to the source transition and target place of the arc are connected as follows:*

$$\begin{aligned}
&\forall t \in T, p \in P, g_t, i_t, o_t, g_p, i_p, o_p, \omega \in \mathbb{N}^*, \\
&\quad post(t, p) = \lfloor \omega \rfloor \Rightarrow \\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow \\
&\quad \exists i \in [0, |\texttt{input}(p)| - 1] \ s.t. \ (\texttt{iaw}(i) \Rightarrow \omega) \in i_p \\
&\qquad \wedge \exists id_s \ s.t. \ (id_s, \texttt{bool}) \in d.sigs \wedge (\texttt{fired} \Rightarrow id_s) \in o_t \wedge (\texttt{itf}(i) \Rightarrow id_s) \in i_p.
\end{aligned}$$

20. *For all pre arc of the input SITPN model, the PDI and TDI corresponding to the source place and target*

transition of the arc are connected as follows:

$$
\begin{array}{l}
\forall t \in T, p \in P, g_t, i_t, o_t, g_p, i_p, o_p, \omega \in \mathbb{N}^*, a \in \{\texttt{basic}, \texttt{test}, \texttt{inhib}\}, \\
\quad pre(p,t) = \lfloor (\omega, a) \rfloor \Rightarrow \\
\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow \\
\quad \exists i \in [0, |\texttt{output}(p)| - 1] \ s.t. \ \{(\texttt{oaw}(i) \Rightarrow \omega), (\texttt{oat}(i) \Rightarrow a)\} \subseteq i_p \\
\quad \wedge \exists j \in [0, |\texttt{input}(t)| - 1], id_{av}, id_{rt}, id_{frd}, id_{pah} \ s.t. \\
\qquad \{(id_{av}, \texttt{bool}), (id_{rt}, \texttt{bool}), (id_{frd}, \texttt{bool}), (id_{pah}, \texttt{bool})\} \subseteq d.sigs \\
\qquad \wedge (\texttt{oav}(i) \Rightarrow id_{av}) \in o_p \wedge (\texttt{iav}(j) \Rightarrow id_{av}) \in i_t \\
\qquad \wedge (\texttt{rtt}(i) \Rightarrow id_{rt}) \in o_p \wedge (\texttt{rt}(j) \Rightarrow id_{rt}) \in i_t \\
\qquad \wedge (\texttt{otf}(i) \Rightarrow id_{frd}) \in i_p \wedge (\texttt{fired} \Rightarrow id_{frd}) \in o_t \\
\qquad \wedge (\texttt{pah}(i) \Rightarrow id_{frd}) \in o_p \\
\qquad \wedge (a = \texttt{test} \vee a = \texttt{inhib} \vee \texttt{confl}(p) = \emptyset \Rightarrow (\texttt{pah}(j) \Rightarrow \texttt{true}) \in i_t) \\
\qquad \wedge (a = \texttt{basic} \wedge \texttt{confl}(p) \neq \emptyset \ or \ \texttt{err} \Rightarrow (\texttt{pah}(j) \Rightarrow id_{pah}) \in i_t).
\end{array}
$$

where $\texttt{confl} \in P \rightarrow 2^T \cup \{\texttt{err}\}$ *takes a place* p *as input and yields either an error or an ordered set of transitions computed as follows:*

(a) If all conflicts between the output transitions of p *are solved by mutual exclusion, or if the set of conflicting transitions of* p *is a singleton, then* `confl` *returns an empty set.*

(b) Otherwise, the function tries to establish a total ordering over the set of conflicting transitions of p *w.r.t the firing priority relation:*

- *If no such ordering can be established (in that case, the firing priority relation is ill-formed, and the input SITPN is not well-defined,* `confl` *returns the* `err` *value.*
- *Otherwise, the function returns the set in a decreasing priority order.*

21. For all place of the input SITPN model for which conflicts in its output transitions are not solved by mutual exclusion, the port indices of the corresponding PDI reflect the priority order established between

the conflicting output transitions:

$$\begin{aligned}
&\forall p \in P, t, t' \in \texttt{confl}(p), g_p, i_p, o_p, g_t, i_t, o_t, g_{t'}, i_{t'}, o_{t'},\\
&\quad t \succ t' \Rightarrow\\
&\quad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow\\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow\\
&\quad \texttt{comp}(\gamma(t'), \texttt{transition}, g_{t'}, i_{t'}, o_{t'}) \in d.beh \Rightarrow\\
&\quad (\forall i, j \in \mathbb{N}, id_{frd}, id_{frd'}, id_s, id_{s'}, name_t, name_{t'}, id_{out},\\
&\qquad (\texttt{fired} \Rightarrow id_{frd}) \in o_t \Rightarrow\\
&\qquad (\texttt{fired} \Rightarrow id_{frd'}) \in o_{t'} \Rightarrow\\
&\qquad \{(\texttt{otf}(i) \Rightarrow id_{frd}), (\texttt{otf}(j) \Rightarrow id_{frd'})\} \subseteq i_p \Rightarrow\\
&\qquad (name_t \Rightarrow id_s) \in i_t \Rightarrow\\
&\qquad (name_{t'} \Rightarrow id_{s'}) \in i_{t'} \Rightarrow\\
&\qquad \{(id_{out}(i) \Rightarrow id_s), (id_{out}(j) \Rightarrow id_{s'})\} \subseteq o_p \Rightarrow\\
&\qquad i < j)
\end{aligned}$$

22. *There exists an* **`actions`** *process that assigns a value to the output port representing the activation status of the actions (referred to as action ports) of the input SITPN model:*

$\exists ss_{ra}, ss_a$ *s.t.* $\texttt{ps}(\texttt{actions}, \emptyset, \texttt{rst}\{ss_{ra}\}\texttt{else}\{\texttt{falling}\{ss_a\}\}) \in d.beh$ *and*

   (a) *The length of the* ss_{ra} *and* ss_a *sequences is equal to the number of actions of the input SITPN model:*
   $|ss_{ra}|_; = |ss_a|_; = |\mathcal{A}|$ *where* $|ss|_; = \begin{cases} |ss_1|_; + |ss_2|_; \text{ if } ss = ss_1; ss_2 \\ 1 \text{ otherwise} \end{cases}$

   (b) *During the initialization, all action ports are assigned to* **`false`** *by the* **`actions`** *process:*
   $\forall a \in \mathcal{A},\ \gamma(a) \Leftarrow \texttt{false} \in ss_{ra}$

   (c) *An action port corresponding to an action associated with no place is assigned to* **`false`** *during the falling edge phase:*
   $\forall a \in \mathcal{A}$ *s.t.* $\texttt{pls}(a) = \emptyset, \gamma(a) \Leftarrow \texttt{false} \in ss_a$ *where* $\texttt{pls}(a) = \{p \mid \mathbb{A}(p, a) = \texttt{true}\}$*, the set of places associated with action* a*.*

   (d) *Otherwise, the value of the action port is the result of the Boolean sum between the* **`marked`** *output port of all PDIs representing the places associated with the corresponding action:*

$$\begin{aligned}
&\forall a \in \mathcal{A}, \texttt{pls}(a) \neq \emptyset \Rightarrow\\
&\qquad \exists e_{or} \text{ s.t. } \gamma(a) \Leftarrow e_{or} \in ss_a \wedge \texttt{is_bsum}(e_{or}, |\texttt{pls}(a)|)\\
&\qquad \wedge \forall p \in \texttt{pls}(a), g_p, i_p, o_p,\\
&\qquad\qquad \texttt{comp}(\gamma(p), \texttt{place}, g_p, i_p, o_p) \in d.beh \Rightarrow\\
&\qquad\qquad \exists id_m \text{ s.t. } (id_m, \texttt{bool}) \in d.sigs \wedge id_m \in e_{or} \wedge (\texttt{marked} \Rightarrow id_m) \in o_p
\end{aligned}$$

where `is_bsum` *is defined as follows:* $\dfrac{e \in \{id, b\}}{\texttt{is\_bsum}(e, 1)}$ $\qquad$ $\dfrac{\texttt{is\_bsum}(e_1, n) \quad \texttt{is\_bsum}(e_2, m)}{\texttt{is\_bsum}(\texttt{or}(e_1, e_2), n + m)}$

23. There exists an **`functions`** *process that assigns a value to the output port representing the execution status of the functions (referred to as function ports) of the input SITPN model:*

$\exists ss_{rf}, ss_f$ *s.t.* $\texttt{ps}(\texttt{functions}, \emptyset, \texttt{rst}\{ss_{rf}\}\texttt{else}\{\texttt{rising}\{ss_f\}\}) \in d.beh$ *and*

(a) The length of the ss_{rf} and ss_f sequences is equal to the number of functions of the input SITPN model:
$|ss_{rf}|; = |ss_f|; = |\mathcal{F}|$

(b) During the initialization, all function ports are assigned to **`false`** *by the* **`functions`** *process:*
$\forall f \in \mathcal{F},\ \gamma(f) \Leftarrow \texttt{false} \in ss_{rf}$

(c) An function port corresponding to an function associated with no transition is assigned to **`false`** *during the rising edge phase:*
$\forall f \in \mathcal{F}$ *s.t.* $\texttt{trs}(f) = \emptyset, \gamma(f) \Leftarrow \texttt{false} \in ss_f$ *where* $\textbf{\textit{trs}}(f) = \{t \mid \mathbb{F}(t, f) = \texttt{true}\}$*, the set of transitions associated with function f.*

(d) Otherwise, the value of the function port is the result of the Boolean sum between the **`fired`** *output port of all TDIs representing the transitions associated with the considered function:*

$$\begin{aligned}
&\forall f \in \mathcal{F}, \texttt{trs}(f) \neq \emptyset \Rightarrow \\
&\quad \exists e_{or} \text{ s.t. } \gamma(f) \Leftarrow e_{or} \in ss_f \wedge \texttt{is_bsum}(e_{or}, |\texttt{trs}(f)|) \\
&\quad \wedge \forall t \in \texttt{trs}(f), g_t, i_t, o_t, \\
&\qquad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\qquad \exists id_m \text{ s.t. } (id_m, \texttt{bool}) \in d.sigs \wedge id_m \in e_{or} \wedge (\texttt{fired} \Rightarrow id_m) \in o_t
\end{aligned}$$

24. For all association between a transition and a condition, the input port reflecting the Boolean value of the given condition (referred to as a condition port) is connected as follows to the input port map of the TDI corresponding to the associated transition:

$$\begin{aligned}
&\forall t \in T, g_t, i_t, o_t, c \in \mathcal{C}, \\
&\quad \texttt{comp}(\gamma(t), \texttt{transition}, g_t, i_t, o_t) \in d.beh \Rightarrow \\
&\quad (\mathbb{C}(t, c) = 1 \Rightarrow \exists i \in [0, |\texttt{conds}(t)| - 1] \text{ s.t. } (\texttt{ic}(i) \Rightarrow \gamma(c)) \in i_t) \\
&\quad (\mathbb{C}(t, c) = -1 \Rightarrow \exists i \in [0, |\texttt{conds}(t)| - 1] \text{ s.t. } (\texttt{ic}(i) \Rightarrow \texttt{not}(\gamma(c))) \in i_t).
\end{aligned}$$

To conclude the definition of the $HM2T_{\texttt{spec}}$ relation, we have $HM2T_{\texttt{spec}}(sitpn, b, \texttt{err})$ if sitpn is not a well-defined SITPN model [2].

A The place design in abstract $\mathcal{H}$-VHDL syntax

```
{

-- Generic constant declarations
gens={
  (input_arcs_number, nat(0,NATMAX), 1),
  (output_arcs_number, nat(0,NATMAX), 1),
  (maximal_marking, nat(0,NATMAX), 1)},

-- Port declarations
ports ={
  (in, initial_marking, nat(0, maximal_marking)),
  (in, input_arcs_weights, array (nat(0, 255), 0, sub(input_arcs_number, 1))),
  (in, output_arcs_types, array (nat(0, 2), 0, sub(output_arcs_number, 1))),
  (in, output_arcs_weights, array (nat(0, 2), 0, sub(output_arcs_number, 1))),
  (in, input_transitions_fired, array (bool, 0, sub(input_arcs_number, 1))),
  (in, output_transitions_fired, array (bool, 0, sub(output_arcs_number, 1))),
  (out, output_arcs_valid, array (bool, 0, sub(output_arcs_number, 1))),
  (out, priority_authorizations, array (bool, 0, sub(output_arcs_number, 1))),
  (out, reinit_transitions_time, array (bool, 0, sub(output_arcs_number, 1))),
  (out, marked, bool)},

-- Internal signal declarations
sigs={(s_input_tokens_sum, nat(0, maximal_marking)),
        (s_marking, nat(0, maximal_marking)),
        (s_output_tokens_sum, nat(0, maximal_marking))},

-- Behavior
beh=
ps (input_tokens_sum,
  {(v_tokens_sum, nat(0, maximal_marking))},
  v_tokens_sum := 0;
  for (i, 0, sub(input_arcs_number, 1)){
     if (input_transitions_fired(i)) {
        v_tokens_sum := add(v_tokens_sum, input_arcs_weights(i))
     } else { null }
  };
  s_input_tokens_sum ⇐ v_tokens_sum)
||
ps (output_tokens_sum,
  {(v_tokens_sum, nat(0, maximal_marking))},
  v_tokens_sum := 0;
  for (i, 0, sub(output_arcs_number, 1)) {
    if (and(output_transitions_fired(i), eq(output_arcs_types(i), 0))) {
      v_tokens_sum := add(v_tokens_sum, output_arcs_weights(i))
    } else { null }
  };
  s_output_tokens_sum ⇐ v_tokens_sum)
```

```
||
ps (marking, ∅,
  rst { s_marking ⇐ initial_marking }
  else {
    rising {
      s_marking ⇐ add(s_marking, sub(s_input_tokens_sum, s_output_tokens_sum))
    }
  })
||
ps (determine_marked, ∅, marked ⇐ gt(s_marking, 0))
||
ps (marking_validation_evaluation, ∅,
  for (i, 0, sub(output_arcs_number, 1)) {
    output_arcs_valid(i) ⇐
    or(and(or(eq(output_arcs_types(i), 0),
              eq(output_arcs_types(i), 1)),
           ge(s_marking, output_arcs_weights(i))),
       and(eq(output_arcs_types(i), 2),
           lt(s_marking, output_arcs_weights(i))))
  })
||
ps (priority_evaluation,
  {(v_saved_output_tokens_sum, nat(0, maximal_marking))},
  v_saved_output_tokens_sum := 0;
  for (i, 0, sub(output_arcs_number, 1)) {
    priority_authorizations(i) ⇐ ge(s_marking, add(v_saved_output_tokens_sum, output_arcs_weights
        (i));
    if (and(output_transitions_fired(i), eq(output_arcs_types(i), 0))) {
      v_saved_output_tokens_sum := add(v_saved_output_tokens_sum, output_arcs_weights(i))
    } else { null }
  })
||
ps (reinit_transitions_time_evaluation, ∅,
  rst {
    for (i, 0, sub(output_arcs_number, 1)) {
      reinit_transitions_time(i) ⇐ false
    }
  } else {
    rising {
      for (i, 0, sub(output_arcs_number, 1)) {
        reinit_transitions_time(i) ⇐
        or(and(and(or(eq(output_arcs_types(i), 0),
                      eq(output_arcs_types(i), 1)),
                   lt(sub(s_marking, s_output_tokens_sum),
                      output_arcs_weights(i)))
               gt(s_output_tokens_sum, 0)),
               output_transitions_fired(i))
      }
    }
```

```
96      })
97  }
```

Listing 1: The `place` design in $\mathcal{H}$-VHDL abstract syntax.

B The transition design in abstract $\mathcal{H}$-VHDL syntax

```
{

-- Generic constant declarations
gens={
  (transition_type, nat(0, 3), 0),
  (input_arcs_number, nat(0, NATMAX), 1),
  (conditions_number, nat(0, NATMAX), 1),
  (maximal_time_counter, nat(0, NATMAX), 1)},

-- Port declarations
ports={
  (in, input_conditions, array(bool, 0, sub(conditions_number, 1)),
  (in, time_A_value, nat(0, maximal_time_counter)),
  (in, time_B_value, nat(0, maximal_time_counter)),
  (in, input_arcs_valid, array(bool, 0, sub(input_arcs_number, 1))),
  (in, reinit_time, array(bool, 0, sub(input_arcs_number, 1))),
  (in, priority_authorizations, array(boolean, 0, sub(input_arcs_number, 1))),
  (out, fired, bool)},

-- Internal signal declarations
sigs={
  (s_condition_combination, bool),
  (s_enabled, bool),
  (s_firable, bool),
  (s_firing, bool),
  (s_priority, bool),
  (s_reinit, bool),
  (s_time_counter, nat(0, maximal_time_counter)},

-- Behavior
beh=
ps (condition_evaluation, {(v_internal_condition, bool)},
  v_internal_condition := true;
  for (i, 0, sub(conditions_number, 1)) {
    v_internal_condition := and(v_internal_condition, input_conditions(i))
  };
  s_condition_combination <= v_internal_condition)
||
ps (enable_evaluation, {(v_internal_enabled, bool)},
  v_internal_enabled := true;
  for (i, 0, sub(input_arcs_number, 1)) {
    v_internal_enabled := and(v_internal_enabled, input_arcs_valid(i))
  };
  s_enabled <= v_internal_enabled)
||
ps (reinit_time_counter_evaluation, {(v_internal_reinit_time_counter, bool)},
  v_internal_reinit_time_counter := false;
```

```
  for (i, 0, sub(input_arcs_number, 1)) {
    v_internal_reinit_time_counter := or(v_internal_reinit_time_counter, reinit_time(i))
  };
  s_reinit_time_counter ⇐ v_internal_reinit_time_counter)
||
ps (time_counter, ∅,
  rst { s_time_counter ⇐ 0 }
  else {
    falling {
      if (and(s_enabled, neq(transition_type, 0))) {
        if (eq(s_reinit_time_counter, false)) {
          if (s_time_counter < maximal_time_counter) {
            s_time_counter ⇐ s_time_counter + 1
          } else { null }
        }
        else { s_time_counter ⇐ 1 }
      }
      else { s_time_counter ⇐ 0 }
    }
  })
||
ps (firing_condition_evaluation, ∅,
  s_firing_condition ⇐
  s_condition_combination
  and s_enabled
  and (eq(transition_type, 0)
       or (eq(s_reinit_time_counter, false)
           and ((eq(transition_type, 1)
                 and ge(s_time_counter, sub(time_A_value, 1))
                 and (s_time_counter < time_B_value))
                or (eq(transition_type, 2)
                    and eq(s_time_counter, sub(time_A_value, 1)))
                or (eq(transition_type, 3)
                    and ge(s_time_counter, sub(time_A_value, 1)))))
       or (s_reinit_time_counter
           and neq(transition_type, 0)
           and eq(time_A_value, 1))))
||
ps (priority_authorization_evaluation, {(v_priority_combination, bool)},
  v_priority_combination := true;
  for (i, 0, sub(input_arcs_number, 1)) {
    v_priority_combination := and(v_priority_combination, priority_authorizations(i))
  };
  s_priority_combination ⇐ v_priority_combination)
||
ps (firable, ∅,
  rst { s_firable ⇐ false }
  else { falling { s_firable ⇐ s_firing_condition } })
||
```

```
97  ps (fired_evaluation, ∅, fired ⇐ and(s_firable, s_priority_combination))
```

Listing 2: The `transition` design in $\mathcal{H}$-VHDL abstract syntax.

C Generic constant and signal names reference

Generic constants and signals reference			
Full name	*Alias*	*Category*	*Type*
input_arcs_number	ian	generic constant (T)	nat(0, NATMAX)
transition_type	tt	generic constant (T)	{not_temp, temp_a_b, temp_a_a, temp_a_inf}
conditions_number	cn	generic constant (T)	nat(0, NATMAX)
maximal_time_counter	mtc	generic constant (T)	nat(0, NATMAX)
time_A_value	A	input port (T)	nat(0, mtc)
time_B_value	B	input port (T)	nat(0, mtc)
input_conditions	ic	input port (T)	array(bool, 0, cn − 1)
reinit_time	rt	input port (T)	array(bool, 0, ian − 1)
input_arcs_valid	iav	input port (T)	array(bool, 0, ian − 1)
priority_authorizations	pah	input port (T)	array(bool, 0, ian − 1)
fired	f	output port (T)	bool
s_condition_combination	scc	internal signal (T)	bool
s_reinit_time_counter	srtc	internal signal (T)	bool
s_priority_combination	spc	internal signal (T)	bool
s_firable	sfa	internal signal (T)	bool
s_enabled	se	internal signal (T)	bool
s_time_counter	stc	internal signal (T)	nat(0, mtc)
s_firing_condition	sfc	internal signal (T)	bool
input_arcs_number	ian	generic constant (P)	nat(0, NATMAX)
output_arcs_number	oan	generic constant (P)	nat(0, NATMAX)
maximal_marking	mm	generic constant (P)	nat(0, NATMAX)
initial_marking	im	input port (P)	nat(0, mm)
output_arcs_types	oat	input port (P)	array({basic, test, inhib}, 0, oan − 1)
output_arcs_weights	oaw	input port (P)	array(nat(0, 255), 0, oan − 1)
output_transitions_fired	otf	input port (P)	array(bool, 0, oan − 1)
input_arcs_weights	iaw	input port (P)	array(nat(0, 255), 0, ian − 1)
input_transitions_fired	itf	input port (P)	array(bool, 0, ian − 1)
output_arcs_valid	oav	output port (P)	array(bool, 0, oan − 1)
reinit_transitions_time	rtt	output port (P)	array(bool, 0, oan − 1)
priority_authorizations	pah	output port (P)	array(bool, 0, oan − 1)
marked	m	output port (P)	bool
s_marking	sm	internal signal (P)	nat(0, mm)
s_output_token_sum	sots	internal signal (P)	nat(0, mm)
s_input_token_sum	sits	internal signal (P)	nat(0, mm)

Table 1: Constants and signals reference for the $\mathcal{H}$-VHDL transition and place designs. In the *Category* column, T (resp. P) indicates a generic constant, input port, output port or internal signal defined in the `transition` (resp. `place`) design.

References

[1] David Andreu, David Guiraud, and Guillaume Souquet. "A Distributed Architecture for Activating the Peripheral Nervous System". In: *Journal of Neural Engineering* 6.2 (Apr. 1, 2009), p. 026001. ISSN: 1741-2560, 1741-2552. DOI: 10.1088/1741-2560/6/2/026001. URL: https://iopscience.iop.org/article/10.10 (visited on 06/08/2020).

[2] Vincent Iampietro. "Vérification formelle d'une méthodologie pour la conception et la production de systèmes numériques critiques". Theses. Université Montpellier, Dec. 2021. URL: https://tel.archives-ouvertes