UNIVERSITY NAME

DOCTORAL THESIS

Thesis Title

Author:
John SMITH

Supervisor:
Dr. James SMITH

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

April 16, 2021

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UNIVERSITY NAME

Abstract

Faculty Name
Department or School Name

Doctor of Philosophy

Thesis Title

by John SMITH

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgments and the people to thank go here, don’t forget to include your project advisor…

Contents

List of Figures

List of Tables

For/Dedicated to/To my…

Chapter 1

Proving semantic preservation in HILECOP

- Change σ_{injr} and σ_{injf} into σ_i.
- Define the $\texttt{Inject}_{\downarrow}$ and $\texttt{Inject}_{\uparrow}$ relations.
- Keep the *sitpn* argument in the SITPN full execution relation, but remove it from the SITPN execution, cycle and state transition relations.

1.1 Preliminary Definitions

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 design binder $\gamma \in WM(sitpn, d)$ is a tuple $<PMap, TMap, \mathcal{C}_{id}, \mathcal{A}_{id}, \mathcal{F}_{id}, CMap, AMap, FMap>$ where:*

- $sitpn = <P, T, pre, test, inhib, post, M_0, \succ, \mathcal{A}, \mathcal{C}, \mathcal{F}, \mathbb{A}, \mathbb{C}, \mathbb{F}, I_s>$
- $d = \texttt{design}\ id_{ent}\ id_{arch}\ gens\ ports\ sigs\ behavior$
- $PMap \in P \to P_{id}$ where $P_{id} = \{id \mid \texttt{comp}(id, "place", gm, ipm, opm) \in behavior\}$
- $TMap \in T \to T_{id}$ where $T_{id} = \{id \mid \texttt{comp}(id, "transition", gm, ipm, opm) \in behavior\}$
- $\mathcal{C}_{id} \subseteq \{id \mid (\texttt{in}, id, t) \in ports \wedge id \notin \{"clk", "rst"\}\}$
- $\mathcal{A}_{id} \subseteq \{id \mid (\texttt{out}, id, t) \in ports\}$
- $\mathcal{F}_{id} \subseteq \{id \mid (\texttt{out}, id, t) \in ports\}$
- $CMap \in \mathcal{C} \to \mathcal{C}_{id}$
- $AMap \in \mathcal{A} \to \mathcal{A}_{id}$
- $FMap \in \mathcal{F} \to \mathcal{F}_{id}$

Definition 2 (Similar Environments). *For a given $sitpn \in SITPN$, a $\mathcal{H}$-VHDL design $d \in design$, a design store $\mathcal{D} \in entity\text{-}id \nrightarrow design$, an elaborated version $\Delta \in ElDesign(d, \mathcal{D})$ of design d, and a binder $\gamma \in WM(sitpn, d)$, the environment $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, that yields the value of the primary input ports of Δ at a given simulation cycle and a given clock event, and the environment E_c, that yields the value of conditions of sitpn at a given execution cycle, are similar, noted $\gamma \vdash E_p \overset{env}{=} E_c$, iff for all $\tau \in \mathbb{N}$, $clk \in \{\uparrow, \downarrow\}$, $c \in \mathcal{C}$, $id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, $E_p(\tau, clk)(id_c) = E_c(\tau)(c)$.*

1.1.1 State Similarity

Definition 3 (General State Similarity). *For a given $sitpn \in SITPN$, a $\mathcal{H}$-VHDL design $d \in design$, an elaborated design $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, and a binder $\gamma \in WM(sitpn, d)$, an SITPN state $s \in S(sitpn)$ and a design state $\sigma \in \Sigma(\Delta)$ are similar, written $\gamma \vdash s \sim \sigma$ iff*

1. $\forall p \in P, id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$,
   $s.M(p) = \sigma(id_p)(''s\_marking'')$.
2. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
   $\big(upper(I_s(t)) = \infty \wedge s.I(t) \leq lower(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(''s\_time\_counter'')\big)$
   $\wedge\big(upper(I_s(t)) = \infty \wedge s.I(t) > lower(I_s(t)) \Rightarrow \sigma(id_t)(''s\_time\_counter'') = lower(I_s(t))\big)$
   $\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) > upper(I_s(t)) \Rightarrow \sigma(id_t)(''s\_time\_counter'') = upper(I_s(t))\big)$
   $\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) \leq upper(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(''s\_time\_counter'')\big)$.
3. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
   $s.reset_t(t) = \sigma(id_t)(''s\_reinit\_time\_counter'')$.
4. $\forall c \in \mathcal{C}, id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, $s.cond(c) = \sigma(id_c)$.
5. $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s.ex(a) = \sigma(id_a)$.
6. $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s.ex(f) = \sigma(id_f)$.

Definition 4 (Post Rising Edge State Similarity). *For a given $sitpn \in SITPN$, a $\mathcal{H}$-VHDL design $d \in design$, an elaborated design $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, and a binder $\gamma \in WM(sitpn, d)$, a clock cycle count $\tau \in \mathbb{N}$, and an SITPN execution environment $E_c \in \mathbb{N} \rightarrow \mathcal{C} \rightarrow \mathbb{B}$, an SITPN state $s \in S(sitpn)$ and a design state $\sigma \in \Sigma(\Delta)$ are similar after a rising edge happening at clock cycle count τ, written $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$ iff*

1. $\forall p \in P, id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$, $s.M(p) = \sigma(id_p)(''s\_marking'')$.
2. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
   $\big(upper(I_s(t)) = \infty \wedge s.I(t) \leq lower(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(''s\_time\_counter'')\big)$
   $\wedge\big(upper(I_s(t)) = \infty \wedge s.I(t) > lower(I_s(t)) \Rightarrow \sigma(id_t)(''s\_time\_counter'') = lower(I_s(t))\big)$
   $\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) > upper(I_s(t)) \Rightarrow \sigma(id_t)(''s\_time\_counter'') = upper(I_s(t))\big)$
   $\wedge\big(upper(I_s(t)) \neq \infty \wedge s.I(t) \leq upper(I_s(t)) \Rightarrow s.I(t) = \sigma(id_t)(''s\_time\_counter'')\big)$.
3. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
   $s.reset_t(t) = \sigma(id_t)(''s\_reinit\_time\_counter'')$.
4. $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s.ex(a) = \sigma(id_a)$.
5. $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s.ex(f) = \sigma(id_f)$.
6. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
   $t \in Sens(s.M) \Leftrightarrow \sigma(id_t)(''s\_enabled'') = \texttt{true}$.

7. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$,
$\sigma(id_t)(\text{"}s\_condition\_combination\text{"}) = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}$
where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

Definition 5 (Post Falling Edge State Similarity). *For a given* $sitpn \in SITPN$, *a* $\mathcal{H}$*-VHDL design* $d \in design$, *an elaborated design* $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, *and a binder* $\gamma \in WM(sitpn, d)$, *an SITPN state* $s \in S(sitpn)$ *and a design state* $\sigma \in \Sigma(\Delta)$ *are similar after a falling edge, written* $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$ *iff* $\gamma \vdash s \sim \sigma$ *(Def. 3, general state similarity) and*

1. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$,
$\sum_{t \in Fired(s)} pre(p,t) = \sigma(id_p)(\text{"}s\_output\_token\_sum\text{"})$.

2. $\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$,
$\sum_{t \in Fired(s)} post(t,p) = \sigma(id_p)(\text{"}s\_input\_token\_sum\text{"})$.

3. $\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$,
$t \in Fired(s) \Leftrightarrow \sigma(id_t)(\text{"}fired\text{"}) = \texttt{true}$.

Definition 6 (Execution Trace Similarity). *For a given* $sitpn \in SITPN$, *a* $\mathcal{H}$*-VHDL design* $d \in design$, *an elaborated design* $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, *and a binder* $\gamma \in WM(sitpn, d)$, *the execution trace* $\theta_s \in \texttt{list}(S(sitpn))$ *and the simulation trace* $\theta_\sigma \in \texttt{list}(\Sigma(\Delta))$ *are similar, written* $\gamma \vdash \theta_s \sim \theta_\sigma$, *according to the following rules:*

$$\textsc{SimTraceNil}\ \frac{}{\gamma \vdash [\,] \sim [\,]} \qquad \textsc{SimTraceCons}\ \frac{\gamma \vdash s \sim \sigma \quad \gamma \vdash \theta_s \sim \theta_\sigma}{\gamma \vdash (s :: \theta_s) \sim (\sigma :: \theta_\sigma)}$$

1.1.2 Equality between big operator expressions

Many times in the proceeding of the following proof, the equality between two sum or product expressions must be estbalished; for instance:
$\sum_{a \in A} f(a) = \sum_{b \in B} g(b)$ where A and B are finite sets, $f \in \mathbb{A} \to \mathbb{N}$ and $g \in B \to \mathbb{N}$

To prove such an equality, Theorem 1 is used, considering that the sum operator used in the above equation is a big operator over the triplet $<\mathbb{N}, 0, +>$. A big operator is defined as follows:

Definition 7 (Big Operator). *Given a triplet* $<A, *, e>$ *such that* A *is a set,* $* \in A \to A \to A$ *is a commutative and associative operator over* A, *and* $e \in A$ *is a neutral element of* $*$, *then for all finite set* B, *and application* $f \in B \to A$, *a big operator* Ω *is recursively defined as follows:* $\underset{b \in B}{\Omega} f(b) = \begin{cases} e & if\ B = \emptyset \\ f(b) * \underset{b' \in B \setminus \{b\}}{\Omega} f(b') & otherwise \end{cases}$

Then, we can prove the following theorem concerning the equality between two big operator expressions.

Theorem 1 (Big Operator Equality). *For all a triplet* $<A, *, e>$ *such that* A *is a set,* $* \in A \to A \to A$ *is a commutative and associative operator over* A, *and* $e \in A$ *is a neutral element of* $*$, *and for all finite sets* B *and* C, *and applications* $f \in B \to A$ *and* $g \in C \to A$, *assume that:*

- *there exists an injection $\iota \in B \rightarrow C$ s.t. $\forall b \in B, c \in C,\ \iota(b) = c \Rightarrow f(b) = g(c)$*
- $|B| = |C|$

then $\underset{b \in B}{\Omega} f(b) = \underset{c \in C}{\Omega} g(c)$.

Proof. Let us reason by induction over $\underset{b \in B}{\Omega} f(b)$:

- **BASE CASE** $B = \emptyset$:
  Then $|C| = |B| = 0$, and $C = \emptyset$. By definition of Ω:

$$\underset{b \in B}{\Omega} f(b) = e \tag{1.1}$$

$$\underset{c \in C}{\Omega} g(c) = e \tag{1.2}$$

  Rewriting the goal with (1.1) and (1.2), tautology .

- **INDUCTION CASE** $B \neq \emptyset$:

  For all finite set C' verifying:

  – $\exists \iota' \in B \setminus \{b\} \rightarrow C'$ s.t. ι' is injective and $\forall b' \in B \setminus \{b\}, c' \in C',\ \iota'(b') = c' \Rightarrow f(b') = g(c')$
  – $|B \setminus \{b\}| = |C'|$

  then $f(b) * \underset{b' \in B \setminus \{b\}}{\Omega} f(b') = f(b) * \underset{c' \in C'}{\Omega} g(c)$

  The goal is $\boxed{f(b) * \underset{b' \in B \setminus \{b\}}{\Omega} f(b') = \underset{c \in C}{\Omega} g(c)}$

  Let us take $\iota \in B \rightarrow C$ s.t. $\forall b \in B, c \in C,\ \iota(b) = c \Rightarrow f(b) = g(c)$. As $\iota(b) = \iota(b)$, then:

$$f(b) = g(\iota(b)) \tag{1.3}$$

  Also, by definition of Ω:

$$\underset{c \in C}{\Omega} g(c) = g(\iota(b)) * \underset{c' \in C \setminus \{\iota(b)\}}{\Omega} \tag{1.4}$$

  Rewriting the goal with (1.4) and (1.3),
  $\boxed{f(b) * \underset{b' \in B \setminus \{b\}}{\Omega} f(b') = f(b) * \underset{c' \in C \setminus \{\iota(b)\}}{\Omega} g(c')}$

  Let us apply the induction hypothesis with $C' = C \setminus \{\iota(b)\}$; then there are two points to prove:

  1. $\boxed{|B \setminus \{b\}| = |C \setminus \{\iota(b)\}|.}$ Trivial as $|B| = |C|$.
  2. $\boxed{\exists \iota' \in B \setminus \{b\} \rightarrow C \setminus \{\iota(b)\} \text{ s.t. } \iota' \text{ is injective and } \forall b' \in B \setminus \{b\}, c' \in C \setminus \{\iota(b)\},\ \iota'(b') = c' \Rightarrow f(b') = g(c')}$

Let us define a $\iota' \in B \setminus \{b\} \to C \setminus \{\iota(b)\}$ as follows: $\forall b' \in B \setminus \{b\},\ \iota'(b) = \iota(b)$. Let us show that this definition is correct by proving that
$\boxed{\forall b' \in B \setminus \{b\},\ \iota(b') \in C \setminus \{\iota(b)\}.}$

Given a $b' \in B \setminus \{b\}$, let us show $\boxed{\iota(b') \in C \setminus \{\iota(b)\}.}$

By definition of $\iota(b')$, there are 2 cases:

- **CASE** $\iota(b') = \iota(b)$, then by definition of ι as an injective function: $b' = b$. Then, $\boxed{b \in B \setminus \{b\} \text{ is a contradiction.}}$
- **CASE** $\boxed{\iota(b') \in C \setminus \{\iota(b)\}.}$

Now let us get back to the previous goal. Using ι' to prove it, there are 2 points to prove:

- $\boxed{\iota' \text{ is injective.}}$ Trivial by definition of ι'.
- $\boxed{\forall b' \in B \setminus \{b\}, c' \in C \setminus \{\iota(b)\},\ \iota'(b') = c' \Rightarrow f(b') = g(c').}$ Trivial by definition of ι'.

□

1.2 Behavior Preservation Theorem

1.2.1 Proof Notations

- Frame box for pending goals: $\boxed{\forall n \in \mathbb{N},\ n > 0 \lor n = 0}$
- Red frame box for completed goals: $\boxed{\texttt{true} = \texttt{true}}$
- Green frame box for induction hypotheses:

$$\boxed{\forall n \in \mathbb{N},\ n + 1 > 0}$$

- **CASE** to denote a case during a proof by case analysis.

1.2.2 Behavior Preservation Theorem and Proof

Theorem 2 (Behavior Preservation). *For all $sitpn \in SITPN, d \in design, \gamma \in WM(sitpn, d)$, $\tau \in \mathbb{N}$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $\theta_s \in \texttt{list}(S(sitpn))$ s.t.*

- *SITPN sitpn translates into design d: $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$*
- *SITPN sitpn yields the execution trace θ_s after τ execution cycles in environment E_c: $E_c, \tau \vdash sitpn \xrightarrow{full} \theta_s$.*

then there exists $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$ s.t. for all $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$ verifying

- *Simulation/Execution environments are similar: $\gamma \vdash E_p \overset{env}{=} E_c$.*

then there exists $\theta_\sigma \in \texttt{list}(\Sigma(\Delta))$ *s.t.*

- *Under the HILECOP design store* $\mathcal{D}_\mathcal{H}$ *and with an empty generic constant dimensioning function, design d yields the simulation trace* θ_σ *after* τ *simulation cycles, starting from its initial state:*
  $\mathcal{D}_\mathcal{H}, \Delta, \emptyset, E_p, \tau \vdash \mathrm{d} \xrightarrow{full} \theta_\sigma$
- *Traces* θ_s *and* θ_σ *are similar:* $\theta_s \sim \theta_\sigma$

Proof. $\boxed{\exists \Delta,\ \forall E_p,\ \gamma \vdash E_p \overset{env}{=} E_c,\ \exists \theta_\sigma, \mathcal{D}_\mathcal{H}, \Delta, \emptyset, E_p, \tau \vdash \mathrm{d} \xrightarrow{full} \theta_\sigma \wedge \theta_s \sim \theta_\sigma}$

By definition of the $\mathcal{H}$-VHDL full simulation relation:
$\mathcal{D}_\mathcal{H}, \Delta, \emptyset, E_p, \tau \vdash \mathrm{d} \xrightarrow{full} \theta_\sigma \equiv \exists \sigma_e, \sigma_0 \in \Sigma(\Delta),\ \mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$ and $\mathcal{D}_\mathcal{H}, \Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$
and $\mathcal{D}_\mathcal{H}, E_p, \Delta, \tau, \sigma_0 \vdash \mathrm{d.cs} \rightarrow \theta_\sigma$.

Use Elaboration, Initialization and Simulation theorems to show that there exists a Δ, θ_σ, σ_e and σ_0 such that $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$ and $\mathcal{D}_\mathcal{H}, \Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$ and $\mathcal{D}_\mathcal{H}, E_p, \Delta, \tau, \sigma_0 \vdash \mathrm{d.cs} \rightarrow \theta_\sigma$.

Use Full Bisimulation theorem to show traces similarity.

□

Theorem 3 (Elaboration). *For all sitpn* $\in$ *SITPN, d* $\in$ *design,* $\gamma \in WM(sitpn, d)$ *s.t.*

- $\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$

then there exists $\Delta \in ElDesign(d, \mathcal{D}_\mathcal{H}), \sigma_e \in \Sigma(\Delta)$ *s.t.*

- $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$

Theorem 4 (Initialization). *For all sitpn* $\in$ *SITPN, d* $\in$ *design,* $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_\mathcal{H})$, $\sigma_e \in \Sigma(\Delta)$ *s.t.*

- $\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$ *and* $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$

then there exists $\sigma_0 \in \Sigma(\Delta)$ *s.t.*

- σ_0 *is the initial simulation state:* $\mathcal{D}_\mathcal{H}, \Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$

Theorem 5 (Simulation). *For all sitpn* $\in$ *SITPN, d* $\in$ *design,* $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_\mathcal{H})$, $\sigma_e, \sigma_0 \in \Sigma(\Delta)$ *s.t.*

- $\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$ *and* $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$ *and* $\mathcal{D}_\mathcal{H}, \Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$

then for all $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \rightarrow Ins(\Delta) \rightarrow value$, $\tau \in \mathbb{N}$, *there exists* $\theta_\sigma \in \texttt{list}(\Sigma(\Delta))$ *s.t.*

- *Design d yields the simulation trace* θ_σ *after* τ *simulation cycles, starting from initial state* σ_0*:*
  $\mathcal{D}_\mathcal{H}, E_p, \Delta, \tau, \sigma_0 \vdash \mathrm{d.cs} \rightarrow \theta_\sigma$

1.2.3 Bisimulation Theorem and Proof

Theorem 6 (Full Bisimulation)**.** *For all $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\tau \in \mathbb{N}$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $\theta_s \in \mathtt{list}(S(sitpn))$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, $\theta_\sigma \in \mathtt{list}(\Sigma(\Delta))$ s.t.*

- $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$
- $\gamma \vdash E_p \overset{env}{=} E_c$
- $E_c, \tau \vdash sitpn \xrightarrow{full} \theta_s$
- $\mathcal{D}_{\mathcal{H}}, \Delta, \emptyset, E_p, \tau \vdash \mathrm{d} \xrightarrow{full} \theta_\sigma$

then $\theta_s \sim \theta_\sigma$

Proof. Case analysis on τ (2 CASES).

- **CASE** $\tau = 0$. By definition of the SITPN full execution and the $\mathcal{H}$-VHDL full simulation relations:
  - $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$
  - $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$
  - $\theta_s = [s_0]$ and $\theta_\sigma = [\sigma_0]$

  $\boxed{\gamma \vdash s_0 \sim \sigma_0}$ (by def. of similar execution trace relation). Solved by applying Lemma Similar Initial States.

- **CASE** $\tau > 0$. By definition of the SITPN full execution and the $\mathcal{H}$-VHDL full execution relations:
  - $E_c, \tau \vdash s_0 \xrightarrow{\uparrow_0} s_0$
  - $E_c, \tau \vdash s_0 \xrightarrow{\downarrow} s$
  - $E_c, \tau - 1 \vdash sitpn, s \to \theta_s$
  - $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$
  - $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$
  - $E_p, \Delta, \tau, \sigma_0 \vdash \mathrm{d.cs} \to \theta$

  $\boxed{\gamma \vdash (s_0 :: s :: \theta_s) \sim (\sigma_0 :: \theta)}$

  By definition of the $\mathcal{H}$-VHDL full simulation relation, we know:
  - $E_p, \Delta, \tau, \sigma_0 \vdash d.cs \xrightarrow{\uparrow,\downarrow} \sigma$
  - $E_p, \Delta, \tau - 1, \sigma \vdash \mathrm{d.cs} \to \theta_\sigma$

where $\theta = \sigma :: \theta_\sigma$.

Rewriting θ as $\sigma :: \theta_\sigma$, $\boxed{\gamma \vdash (s_0 :: s :: \theta_s) \sim (\sigma_0 :: \sigma :: \theta_\sigma)}$

3 subgoals (by def. of Execution Trace Similarity).

1. $\gamma \vdash s_0 \sim \sigma_0$ (solved by applying Lemma Similar Initial States).
2. $\gamma \vdash s \sim \sigma$ (solved by applying Lemma First Cycle).
3. $\gamma \vdash \theta_s \sim \theta_\sigma$ (solved by applying Lemma Bisimulation).

□

Lemma 1 (First Cycle). *For all $sitpn \in SITPN, d \in design, \gamma \in WM(sitpn, d), s \in S(sitpn), \Delta \in ElDesign(d, \mathcal{D}_\mathcal{H}), \sigma_e, \sigma_0, \sigma \in \Sigma(\Delta)$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, assume that:*

- *$\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$ and $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$ and $\gamma \vdash E_p \overset{env}{=} E_c$*
- *σ_0 is the initial state of Δ: $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$*
- *First execution cycle for d: $E_p, \Delta, \tau, \sigma_0 \vdash d.cs \xrightarrow{\uparrow,\downarrow} \sigma$*
- *Particular first execution cycle for sitpn (first rising edge is idle):*

  $E_c, \tau \vdash s_0 \xrightarrow{\uparrow_0} s_0$ and $E_c, \tau \vdash s_0 \xrightarrow{\downarrow} s$

then $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$.

Proof. Let's show that the first execution cycle leads to two states verifying the Post Falling Edge State Similarity relation: $\boxed{\gamma \vdash s \overset{\downarrow}{\sim} \sigma.}$

By definition of the $\mathcal{H}$-VHDL cycle relation, we have:

- $\texttt{Inject}_\uparrow(\sigma_0, E_p, \tau, \sigma_{injr})$ and $\Delta, \sigma_{injr} \vdash \text{d.cs} \xrightarrow{\uparrow} \sigma_r$ and $\Delta, \sigma_r \vdash \text{d.cs} \xrightarrow{\theta} \sigma'$
- $\texttt{Inject}_\downarrow(\sigma', E_p, \tau, \sigma_{injf})$ and $\Delta, \sigma_{injf} \vdash \text{d.cs} \xrightarrow{\downarrow} \sigma_f$ and $\Delta, \sigma_f \vdash \text{d.cs} \xrightarrow{\theta'} \sigma$

Then, we can apply the Falling Edge lemma to solve $\boxed{\gamma \vdash s \overset{\downarrow}{\sim} \sigma.}$

One premise of the Falling Edge lemma remains to be proved: $\boxed{\gamma, E_c, \tau \vdash s_0 \overset{\uparrow}{\sim} \sigma'.}$

Then, we can apply the First Rising Edge lemma to solve $\boxed{\gamma, E_c, \tau \vdash s_0 \overset{\uparrow}{\sim} \sigma'.}$

□

Lemma 2 (Bisimulation). *For all sitpn, d, γ, E_p, E_c, τ, s, θ_s, σ, θ_σ, Δ, σ_e, assume that:*

- *$\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$ and $\gamma \vdash E_p \overset{env}{=} E_c$ and $\mathcal{D}_\mathcal{H}, \emptyset \vdash d \xrightarrow{elab} \Delta, \sigma_e$*

- *Starting states are similar as intended after a falling edge:* $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$
- $E_c, \tau \vdash sitpn, s \rightarrow \theta_s$
- $E_p, \Delta, \tau, \sigma \vdash d.cs \rightarrow \theta_\sigma$

then $\gamma \vdash \theta_s \sim \theta_\sigma$.

Proof. Induction on τ.

- Base case, $\tau = 0$: traces are empty, trivial.
- Induction case, $\tau > 0$:

  $\forall s, \sigma, \theta_s, \theta_\sigma$ s.t. $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$ and $E_c, \tau - 1 \vdash sitpn, s \rightarrow \theta_s$ and $E_p, \Delta, \tau - 1, \sigma \vdash d.cs \rightarrow \theta_\sigma$ then $\gamma \vdash \theta_s \sim \theta_\sigma$.

  By definition of the SITPN execution and the $\mathcal{H}$-VHDL simulation relations for $\tau > 0$:

  – $E, \tau \vdash sitpn, s \xrightarrow{\uparrow,\downarrow} s'$ and $E_c, \tau - 1 \vdash sitpn, s \rightarrow \theta_s$.

  – $E_p, \Delta, \tau, \sigma \vdash$ d.cs $\xrightarrow{\uparrow,\downarrow} \sigma'$ and $E_p, \Delta, \tau - 1, \sigma \vdash d.cs \rightarrow \theta_\sigma$.

  $\boxed{\gamma \vdash (s' :: \theta_s) \sim (\sigma' :: \theta_\sigma)}$.

  2 subgoals (by def. of Execution Trace Similarity).

  1. $\boxed{\gamma \vdash s' \sim \sigma'}$ (solved with Step).
  2. $\boxed{\gamma \vdash \theta_s \sim \theta_\sigma}$ (solved with Step and IH).

□

Lemma 3 (Step). *For all sitpn, d,* γ*,* E_p*,* E_c*,* τ*, s,* s''*,* σ*,* σ''*,* Δ*,* σ_e*, assume that:*

- $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$ *and* $E_p \overset{env}{=} E_c$ *and* $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash d \xrightarrow{elab} \Delta, \sigma_e$
- $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$
- *From state s to* s'' *in one execution cycle:* $E_c, \tau \vdash sitpn, s \xrightarrow{\uparrow,\downarrow} s''$
- *From state* σ *to* σ'' *in one simulation cycle:* $E_p, \Delta, \tau, \sigma \vdash d.cs \xrightarrow{\uparrow,\downarrow} \sigma''$

then $\gamma \vdash s'' \overset{\downarrow}{\sim} \sigma''$.

Proof. By def. of the SITPN and $\mathcal{H}$-VHDL cycle relations:

- $E_c, \tau \vdash sitpn, s \xrightarrow{\uparrow} s'$ and $E_c, \tau \vdash sitpn, s' \xrightarrow{\downarrow} s''$
- $\texttt{Inject}_{\uparrow}(\sigma, E_p, \tau, \sigma_{injr})$ and $\Delta, \sigma_{injr} \vdash$ d.cs $\xrightarrow{\uparrow} \sigma_r$ and $\Delta, \sigma_r \vdash$ d.cs $\xrightarrow{\theta} \sigma'$
- $\texttt{Inject}_{\downarrow}(\sigma', E_p, \tau, \sigma_{injf})$ and $\Delta, \sigma_{injf} \vdash$ d.cs $\xrightarrow{\downarrow} \sigma_f$ and $\Delta, \sigma_f \vdash$ d.cs $\xrightarrow{\theta'} \sigma''$

Solved by applying Rising Edge and then “Falling Edge” lemmas. □

1.3 Initial States

Definition 8 (Initial State Hypotheses). *Given an $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$, assume that:*

- *SITPN $sitpn$ translates into design d: $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$*
- *Δ is the elaborated version of d, σ_e is the default state of Δ, i.e, state of Δ where all signals have their default value:*
  $\mathcal{D}_{\mathcal{H}}, \varnothing \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$
- *σ_0 is the initial state of Δ: $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$*

Lemma 4 (Similar Initial States). *For all $sitpn \in SITPN, d \in design, \gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ that verify the hypotheses of Def. 8, then $\gamma \vdash s_0 \sim \sigma_0$.*

Proof. By definition of State Similarity, 6 subgoals.

1. $\forall p \in P, id_p \in Comps(\Delta), \sigma_p^0 \in \Sigma(\Delta(id_p))$ $s.t.$ $\gamma(p) = id_p$ and $\sigma_0(id_p) = \sigma_p^0$, $s_0.M(p) = \sigma_p^0(''s\_marking'')$.
2. $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t^0 \in \Sigma(\Delta(id_t))$ $s.t.$ $\gamma(t) = id_t$ and $\sigma_0(id_t) = \sigma_t^0$, $upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'') \wedge$ $upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = lower(I_s(t)) \wedge$ $upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t^0(''s\_tc'') = upper(I_s(t)) \wedge$ $upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(''s\_tc'')$.
3. $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t^0 \in \Sigma(\Delta(id_t))$ $s.t.$ $\gamma(t) = id_t$ and $\sigma_0(id_t) = \sigma_t^0$, $s_0.reset_t(t) = \sigma_t^0(''s\_reinit\_time\_counter'')$.
4. $\forall c \in \mathcal{C}, id_c \in Ins(\Delta)$ $s.t.$ $\gamma(c) = id_c$, $s_0.cond(c) = \sigma_0(id_c)$.
5. $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ $s.t.$ $\gamma(a) = id_a$, $s_0.ex(a) = \sigma_0(id_a)$.
6. $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ $s.t.$ $\gamma(f) = id_f$, $s_0.ex(f) = \sigma_0(id_f)$.

- Apply Lemma Initial States Equal Marking to solve 1.
- Apply Lemma Initial States Equal Time Counters to solve 2.
- Apply Lemma Initial States Equal Reset Orders to solve 3.
- Apply Lemma Initial States Equal Condition Values to solve 4.
- Apply Lemma Initial States Equal Action Executions to solve 5.
- Apply Lemma Initial States Equal Function Executions to solve 6.

□

1.3.1 Initial states and marking

Lemma 5 (Initial States Equal Marking). *For all* $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ *that verify the hypotheses of Def. 8, then* $\forall p \in P, id_p \in Comps(\Delta), \sigma_p^0 \in \Sigma(\Delta(id_p))$ *s.t.* $\gamma(p) = id_p$ *and* $\sigma_0(id_p) = \sigma_p^0$, $s_0.M(p) = \sigma_p^0(\text{"}s\_marking\text{"})$.

Proof. Given a $p \in P$, an $id_p \in Comps(\Delta)$ and a $\sigma_p^0 \in \Sigma(\Delta(id_p))$ s.t. $\gamma(p) = id_p$ and $\sigma_0(id_p) = \sigma_p^0$, let's show that

$\boxed{s_0.M(p) = \sigma_p^0(\text{"}s\_marking\text{"}).}$

By definition of id_p, there exist gm_p, ipm_p, opm_p s.t. $\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$.

By property of the $\mathcal{H}$-VHDL initialization relation, the P design behavior (process "`marking`"), and
$\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$, then $\sigma_p^0(\text{"}s\_marking\text{"}) = \sigma_p^0(\text{"}initial\_marking\text{"})$.

Rewriting $\sigma_p^0(\text{"}s\_marking\text{"})$ as $\sigma_p^0(\text{"}initial\_marking\text{"})$, $\boxed{\sigma_p^0(\text{"}initial\_marking\text{"}) = s_0.M(p).}$

By construction, $<\texttt{id}_\texttt{p}\texttt{.initial\_marking} \Rightarrow M_0(p)> \in ipm_p$. By property of the $\mathcal{H}$-VHDL initialization relation, and $\texttt{comp}(id_p, \text{"}place\text{"}, gm_p, ipm_p, opm_p) \in d.cs$, then $\sigma_p^0(\text{"}initial\_marking\text{"}) = M_0(p)$.

By definition of s_0, rewriting $s_0.M(p)$ as $M_0(p)$, $\sigma_p^0(\text{"}initial\_marking\text{"}) = s_0.M(p)$. ☐

1.3.2 Initial states and time counters

Lemma 6 (Initial States Equal Time Counters). *For all* $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ *that verify the hypotheses of Def. 8, then* $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t^0 \in \Sigma(\Delta(id_t))$ *s.t.* $\gamma(t) = id_t$ *and* $\sigma_0(id_t) = \sigma_t^0$,
$upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(\text{"}s\_tc\text{"}) \wedge$
$upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t^0(\text{"}s\_tc\text{"}) = lower(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t^0(\text{"}s\_tc\text{"}) = upper(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(\text{"}s\_tc\text{"})$.

Proof. Given a $t \in T_i$, an $id_t \in Comps(\Delta)$ and a $\sigma_t^0 \in \Sigma(\Delta(id_t))$ s.t. $\gamma(t) = id_t$ and $\sigma_0(id_t) = \sigma_t^0$, let's show that:

1. $\boxed{upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(\text{"}s\_tc\text{"})}$
2. $\boxed{upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t^0(\text{"}s\_tc\text{"}) = lower(I_s(t))}$
3. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t^0(\text{"}s\_tc\text{"}) = upper(I_s(t))}$
4. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t^0(\text{"}s\_tc\text{"})}$

By definition of id_t, there exist gm_t, ipm_t, opm_t s.t. $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$.

Then, let's show the 4 previous subgoals.

1. Assume $upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t))$, then show $\boxed{s_0.I(t) = \sigma_t^0("s\_tc").}$

   Rewriting $s_0.I(t)$ as 0, by definition of s_0, $\boxed{\sigma_t^0("s\_tc") = 0.}$

   By property of the $\mathcal{H}$-VHDL initialization relation, the T design behavior (process "`time_counter`"), and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, then $\sigma_t^0("s\_tc") = 0$.

2. Assume $upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t))$, then show $\boxed{\sigma_t^0("s\_tc") = lower(I_s(t))}$.

   By definition, $lower(I_s(t)) \in \mathbb{N}^*$ and $s_0.I(t) = 0$. Then, $lower(I_s(t)) < 0$ is a contradiction.

3. Assume $upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t))$, then show $\boxed{\sigma_t^0("s\_tc") = upper(I_s(t))}$.

   By definition, $upper(I_s(t)) \in \mathbb{N}^*$ and $s_0.I(t) = 0$. Then, $upper(I_s(t)) < 0$ is a contradiction.

4. Assume $upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t))$, then show $\boxed{s_0.I(t) = \sigma_t^0("s\_tc")}$.

   Rewriting $s_0.I(t)$ as 0, by definition of s_0, $\boxed{\sigma_t^0("s\_tc") = 0.}$

   By property of the $\mathcal{H}$-VHDL initialization relation, the T design behavior (process "`time_counter`"), and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, then $\sigma_t^0("s\_tc") = 0$.

□

1.3.3 Initial states and reset orders

Lemma 7 (Initial States Equal Reset Orders). *For all $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ that verify the hypotheses of Def. 8, then $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t^0 \in \Sigma(\Delta(id_t))$ s.t. $\gamma(t) = id_t$ and $\sigma_0(id_t) = \sigma_t^0$, $s_0.reset_t(t) = \sigma_t^0("s\_reinit\_time\_counter")$.*

Proof. Given a $t \in T_i$, an $id_t \in Comps(\Delta)$ and a $\sigma_t^0 \in \Sigma(\Delta(id_t))$ s.t. $\gamma(t) = id_t$, let's show that
$\boxed{s_0.reset_t(t) = \sigma_t^0("s\_reinit\_time\_counter")}$.

Rewriting $s_0.reset_t(t)$ as $false$, by definition of s_0, $\boxed{\sigma_t^0("s\_reinit\_time\_counter") = false.}$

By definition of id_t, there exist gm_t, ipm_t, opm_t $s.t.$ $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$.

By property of the $\mathcal{H}$-VHDL initialization relation, the T design behavior (process `reinit_time_counter _evaluation`), and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, we know $\sigma_t^0(''s\_reinit\_time\_counter'') = \prod_{i=0}^{\Delta(id_t)(''in\_arcs\_nb'')-1} \sigma_t^0(''rt'')(i)$, where $\Delta(id_t)(''in\_arcs\_nb'')$ is the value of the generic constant $''in\_arcs\_nb''$ stored in the elaborated design $\Delta(id_t)$ (which, by property of the $\mathcal{H}$-VHDL elaboration relation, is an elaborated version of the T design).

Rewriting $\sigma_t^0(''s\_reinit\_time\_counter'')$ as $\prod_{i=0}^{\Delta(id_t)(''in\_arcs\_nb'')-1} \sigma_t^0(''rt'')(i)$,

$$\boxed{\prod_{i=0}^{\Delta(id_t)(''in\_arcs\_nb'')-1} \sigma_t^0(''rt'')(i) = false.}$$

For all $t \in T$ (resp. $p \in P$), let $input(t)$ (resp. $input(p)$) be the set of input places of t (resp. input transitions of p), and let $output(t)$ (resp. $output(p)$) be the set of output places of t (resp. output transitions of p).

Case analysis on $input(t)$ (2 CASES).

- **CASE** $input(t) = \emptyset$.

  By construction, $<\mathtt{id_t.in\_arcs\_nb} \Rightarrow 1> \in gm_t$, and by property of the elaboration relation,
  $\Delta(id_t)(''in\_arcs\_nb'') = 1$. By construction, $< \mathtt{id_t.rt(0)} \Rightarrow false > \in ipm_t$, and by property of the initialization relation, $\sigma_t^0(''rt'')(0) = false$.

  Rewriting $\Delta(id_t)(''in\_arcs\_nb'')$ as 1 and $\sigma_t^0(''rt'')(0)$ as $false$,

  $$\prod_{i=0}^{\Delta(''in\_arcs\_nb'')-1} \sigma_t^0(''rt'')(i) = \sigma_t^0(''rt'')(0) = false.$$

- **CASE** $input(t) \neq \emptyset$.

  We know $\prod_{i=0}^{\Delta(id_t)(''in\_arcs\_nb'')-1} \sigma_t^0(''rt'')(i) = false \equiv \exists i \in [0, \Delta(id_t)(''in\_arcs\_nb'') - 1]\ s.t.\ \sigma_t^0(''rt'')(i) = false$.

  $$\boxed{\exists i \in [0, \Delta(id_t)(''in\_arcs\_nb'') - 1]\ s.t.\ \sigma_t^0(''rt'')(i) = false.}$$

  Since $input(t) \neq \emptyset$, $\exists p\ s.t.\ p \in input(t)$. Let's take such a $p \in input(t)$.

  By construction, for all $p \in P$, there exist $id_p\ s.t.\ \gamma(p) = id_p$.

  By definition of id_p, there exist $gm_p, ipm_p, opm_p\ s.t.\ \mathtt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$.

  By construction, for all $p \in P$, $t \in T$ s.t. $p \in input(t)$ and $t \in output(p)$, for all id_p, id_t s.t. $\gamma(p) = id_p$ and $\gamma(t) = id_t$, for all gm_p, ipm_p, opm_p s.t. $\mathtt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$ and gm_t, ipm_t, opm_t s.t. $\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, op$
  $d.cs$, there exist $i \in [0, |input(t)| - 1], j \in [0, |output(p)| - 1], id_{ji}$ s.t. $<\mathtt{id_p.rtt(j)} \Rightarrow id_{ji}> \in opm_p$ and $<\mathtt{id_t.rt(i)} \Rightarrow id_{ji}> \in ipm_t$. Let's take such a i, j and id_{ji}.

By construction, for all $t \in T$ s.t. $input(t) \neq \emptyset, id_t, gm_t, ipm_t, opm_t$ s.t. $\gamma(t) = id_t$ and
$\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, then $<\texttt{id}_\texttt{t}\texttt{.in\_arcs\_nb} \Rightarrow |input(t)|> \in gm_t$.

By property of the $\mathcal{H}$-VHDL elaboration relation and $<\texttt{id}_\texttt{t}\texttt{.in\_arcs\_nb} \Rightarrow |input(t)|> \in gm_t$, we know $\Delta(id_t)("in\_arcs\_nb) = |input(t)|$.

Rewriting $\Delta(id_t)("in\_arcs\_nb)$ as $|input(t)|$, we have $i \in [0, \Delta(id_t)("in\_arcs\_nb) - 1]$. Let's take that i to prove the goal.

$\boxed{\sigma_t^0("rt")(i) = false.}$

By property of the $\mathcal{H}$-VHDL initialization relation and $<\texttt{id}_\texttt{t}\texttt{.rt(i)} \Rightarrow id_{ji}> \in ipm_t$, we know $\sigma_t^0("rt")(i) = \sigma_0("id_{ji}")$.

Rewriting $\sigma_t^0("rt")(i)$ as $\sigma_0("id_{ji}")$, $\boxed{\sigma_0("id_{ji}") = false.}$

By property of the $\mathcal{H}$-VHDL elaboration and initialization relations, and $\texttt{comp}(id_p, "place", gm_p, ip$
$d.cs$, there exists a $\sigma_p^0 \in \Sigma(\Delta(id_p))$ s.t. $\sigma_0(id_p) = \sigma_p^0$.

By property of the $\mathcal{H}$-VHDL initialization relation and $< \texttt{id}_\texttt{p}\texttt{.rtt(j)} \Rightarrow id_{ji} >\in opm_p$, we know $\sigma_0("id_{ji}") = \sigma_p^0("rtt")(j)$.

Rewriting $\sigma_0("id_{ji}")$ as $\sigma_p^0("rtt")(j)$, $\boxed{\sigma_p^0("rtt")(j) = false.}$

By property of the $\mathcal{H}$-VHDL initialization relation, the P design behavior (process `reinit_transitions_ti-`
`me_evaluation`), and $\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, we know that for all $j \in [0, \Delta(id_p)("out\_arcs\_nb") - 1]$, $\sigma_p^0("rtt")(j) = false$.

By construction, for all $p \in P$ s.t. $output(p) \neq \emptyset, id_p \in Comps(\Delta), gm_p, ipm_p, opm_p$
s.t. $\gamma(p) = id_p$ and $\texttt{comp}(id_p, "transition", gm_p, ipm_p, opm_p) \in d.cs$, then $<\texttt{id}_\texttt{p}\texttt{.out\_arcs\_nb} \Rightarrow |ou$
gm_p.

By property of the $\mathcal{H}$-VHDL elaboration relation and $<\texttt{id}_\texttt{p}\texttt{.out\_arcs\_nb} \Rightarrow |output(p)|> \in gm_p$, we know $\Delta(id_p)("out\_arcs\_nb") = |output(p)|$.

Rewriting $|output(p)|$ as $\Delta(id_p)("out\_arcs\_nb)$, we have $j \in [0, \Delta(id_p)("out\_arcs\_nb) - 1]$. Then, we can deduce $\sigma_p^0("rtt")(j) = false$.

$\square$

1.3.4 Initial states and condition values

Lemma 8 (Initial States Equal Condition Values)**.** *For all* $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ *that verify the hypotheses of Def. 8, then* $\forall c \in \mathcal{C}, id_c \in Ins(\Delta)$ *s.t.* $\gamma(c) = id_c$, $s_0.cond(c) = \sigma_0(id_c)$.

Proof. Given a $c \in \mathcal{C}$ and an $id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, let's show that $\boxed{s_0.cond(c) = \sigma_0(id_c).}$

Rewriting $s_0.cond(c)$ as $false$, by definition of s_0, $\boxed{\sigma_0(id_c) = false.}$
By construction, id_c is an input port identifier of boolean type in the $\mathcal{H}$-VHDL design d.
By property, of the $\mathcal{H}$-VHDL elaboration relation, $\sigma_e(id_c) = false$, where $false$ is the default value associated to signals of the boolean type during the elaboration (see definition of default value in chapter $\mathcal{H}$-VHDL semantics).
By property of the $\mathcal{H}$-VHDL initialization relation, we have $\sigma_e(id_c) = \sigma_0(id_c)$ (i.e, input ports are not assigned during the initialization phase).
Rewriting $\sigma_e(id_c)$ as $false$, $\sigma_0(id_c) = false$.

□

1.3.5 Initial states and action executions

Correction: id_f is assigned by the reset block of the function process

Lemma 9 (Initial States Equal Action Executions)**.** *For all* $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}}), \sigma_e, \sigma_0 \in \Sigma(\Delta)$ *that verify the hypotheses of Def. 8, then* $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ *s.t.* $\gamma(a) = id_a$, $s_0.ex(a) = \sigma_0(id_a)$.

Proof. Given a $a \in \mathcal{A}$ and an $id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, let's show that $\boxed{s_0.ex(a) = \sigma_0(id_a).}$

Rewriting $s_0.ex(a)$ as $false$, by definition of s_0, $\boxed{\sigma_0(id_a) = false.}$
By construction, id_a is an output port identifier of boolean type in the $\mathcal{H}$-VHDL design d.
By property, of the $\mathcal{H}$-VHDL elaboration relation, $\sigma_e(id_a) = false$, where $false$ is the default value associated to signals of the boolean type during the elaboration (see definition of default value in chapter $\mathcal{H}$-VHDL semantics).
By construction, we know that the output port identifier id_a is assigned in the generated `action` process, only at the falling edge phase of the simulation cycle (i.e, the assignment takes place in a `falling` statement block).
By property of the $\mathcal{H}$-VHDL initialization relation, and we have $\sigma_e(id_a) = \sigma_0(id_a)$ (i.e, process `action` is idle during the initialization phase).
Rewriting $\sigma_e(id_a)$ as $false$, $\sigma_0(id_a) = false$.

□

1.3.6 Initial states and function executions

Correction: id_f is assigned by the reset block of the function process

Lemma 10 (Initial States Equal Function Executions). *For all $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, $\sigma_e, \sigma_0 \in \Sigma(\Delta)$ that verify the hypotheses of Def. 8, then $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s_0.ex(f) = \sigma_0(id_f)$.*

Proof. Given a $f \in \mathcal{F}$ and an $id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, let's show that $\boxed{s_0.ex(f) = \sigma_0(id_f).}$

Rewriting $s_0.ex(f)$ as $false$, by definition of s_0, $\boxed{\sigma_0(id_f) = false.}$
By construction, id_f is an output port identifier of boolean type in the $\mathcal{H}$-VHDL design d.
By property, of the $\mathcal{H}$-VHDL elaboration relation, $\sigma_e(id_f) = false$, where $false$ is the default value associated to signals of the boolean type during the elaboration (see definition of default value in chapter $\mathcal{H}$-VHDL semantics).
By construction, we know that the output port identifier id_f is assigned in the generated `function` process (i.e, `function` is the process identifier), only at the rising edge phase of the simulation cycle (i.e, the assignment takes place in a `rising` statement block).
By property of the $\mathcal{H}$-VHDL initialization relation, and we have $\sigma_e(id_f) = \sigma_0(id_f)$ (i.e, process `function` is idle during the initialization phase).
Rewriting $\sigma_e(id_f)$ as $false$, $\sigma_0(id_f) = false.$

□

1.4 First Rising Edge

Definition 9 (First Rising Edge Hypotheses). *Given an $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_{\mathcal{H}})$, $\sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma \in \Sigma(\Delta)$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, $\tau \in \mathbb{N}$, assume that:*

- $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$ *and* $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash d \xrightarrow{elab} (\Delta, \sigma_e)$ *and* $\gamma \vdash E_p \overset{env}{=} E_c$
- σ_0 *is the initial state of* Δ: $\Delta, \sigma_e \vdash d.cs \xrightarrow{init} \sigma_0$
- $E_c, \tau \vdash s_0 \xrightarrow{\uparrow_0} s_0$
- $\texttt{Inject}_{\uparrow}(\sigma_0, E_p, \tau, \sigma_i)$ *and* $\Delta, \sigma_i \vdash \mathrm{d.cs} \xrightarrow{\uparrow} \sigma_{\uparrow}$ *and* $\Delta, \sigma_{\uparrow} \vdash \mathrm{d.cs} \xrightarrow{\theta} \sigma$

Lemma 11 (First Rising Edge). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma, E_c, E_p, \tau$ that verify the hypotheses of Def. 9, then $\gamma, E_c, \tau \vdash s_0 \overset{\uparrow}{\sim} \sigma$.*

Proof. By definition of Post Rising Edge State Similarity, 6 subgoals.

1. $\forall p \in P, id_p \in Comps(\Delta), \sigma_p \in \Sigma(\Delta(id_p))$ s.t. $\gamma(p) = id_p$ and $\sigma(id_p) = \sigma_p$, $s_0.M(p) = \sigma_p(\text{"s\_marking"})$.

2. $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t \in \Sigma(\Delta(id_t))$ s.t. $\gamma(t) = id_t$ and $\sigma(id_t) = \sigma_t$,
$upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t(\text{"s\_tc"}) \wedge$
$upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t(\text{"s\_tc"}) = lower(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t(\text{"s\_tc"}) = upper(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t(\text{"s\_tc"})$.

3. $\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t \in \Sigma(\Delta(id_t))$ s.t. $\gamma(t) = id_t$ and $\sigma(id_t) = \sigma_t$, $s_0.reset_t(t) = \sigma_t(''s\_reinit\_time\_counter'')$.

4. $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s_0.ex(a) = \sigma(id_a)$.

5. $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s_0.ex(f) = \sigma(id_f)$.

6. $\forall t \in T_i, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, $t \in Sens(s.M) \Leftrightarrow \sigma(id_t)(''s\_enabled'') = \texttt{true}$.

7. $\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,
$$\sigma(id_t)(''s\_condition\_combination'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}$$
where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

- Apply Lemma First Rising Edge Equal Marking to solve 1.
- Apply Lemma First Rising Edge Equal Time Counters to solve 2.
- Apply Lemma First Rising Edge Equal Reset Orders to solve 3.
- Apply Lemma "First Rising Edge Equal Action Executions" to solve 4.
- Apply Lemma "First Rising Edge Equal Function Executions " to solve 5.
- Apply Lemma "Rising Edge Equal Sensitized" to solve 6.
- Apply Lemma "Rising Edge Equal Condition Combination" to solve 7.

□

1.4.1 First rising edge and marking

Lemma 12 (First Rising Edge Equal Marking). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_\uparrow, \sigma$, E_c, E_p, τ that verify the hypotheses of Def. 9, then $\forall p \in P, id_p \in Comps(\Delta), \sigma_p \in \Sigma(\Delta(id_p))$ s.t. $\gamma(p) = id_p$ and $\sigma(id_p) = \sigma_p$, $s_0.M(p) = \sigma_p(''s\_marking'')$.*

Proof. Given a p, id_p, σ_p s.t. $\gamma(p) = id_p$ and $\sigma(id_p) = \sigma_p$, let us show that $\boxed{s_0.M(p) = \sigma_p(''s\_marking'').}$ By definition of id_p, there exist gm_p, ipm_p, opm_p s.t. $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$.

By property of the $\mathcal{H}$-VHDL elaboration relation, the $\mathcal{H}$-VHDL initialization relation, the $\texttt{Inject}_\uparrow$ relation, the $\mathcal{H}$-VHDL rising edge relation and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$, there exist a $\sigma_p^e, \sigma_p^0, \sigma_p^{injr}, \sigma_p^r \in \Sigma(\Delta)$ s.t. $\sigma_e(id_p) = \sigma_p^e$ and $\sigma_0(id_p) = \sigma_p^0$ and $\sigma_i(id_p) = \sigma_p^{injr}$ and $\sigma_r(id_p) = \sigma_p^r$.

From the elaboration to the end of the first rising edge phase, an internal state is associated with the P component instance id_p in the component store of the top-level design d.

By property of the $\mathcal{H}$-VHDL rising edge relation, the P design behavior (process "`marking`"), and
$\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, then
$\sigma_p^r("s\_marking") = \sigma_p^{injr}("s\_marking") + \sigma_p^{injr}("s\_input\_token\_sum") - \sigma_p^{injr}("s\_output\_token\_sum")$.

> Result of the execution of the process "`marking`" that performs the signal assignment
> `s_marking` $\Leftarrow$ `s_marking` $+$ `s_input_token_sum` $-$ `s_output_token_sum`.

By property of the $\mathcal{H}$-VHDL stabilize relation, the P design behavior (process "`marking`"), and
$\texttt{comp}(id_p, "place", gm_p, ipm_p, opm_p) \in d.cs$, then $\sigma_p^r("s\_marking") = \sigma_p("s\_marking")$.

> As it is only assigned by the process "`marking`", and as the process "`marking`" is never executed during the stabilization phase, the "`s_marking`" signal has an invariant value during the stabilization phase.

Rewriting $\sigma_p("s\_marking")$ as $\sigma_p^r("s\_marking")$, and $\sigma_p^r("s\_marking")$ as
$\sigma_p^{injr}("s\_marking") + \sigma_p^{injr}("s\_input\_token\_sum") - \sigma_p^{injr}("s\_output\_token\_sum")$,
$\boxed{s_0.M(p) = \sigma_p^{injr}("s\_marking") + \sigma_p^{injr}("s\_input\_token\_sum") - \sigma_p^{injr}("s\_output\_token\_sum").}$

By property of the $\texttt{Inject}_\uparrow$ relation, $\sigma_p^{injr}("s\_marking") = \sigma_p^0("s\_marking")$ and
$\sigma_p^{injr}("s\_input\_token\_sum") = \sigma_p^0("s\_input\_token\_sum")$ and
$\sigma_p^{injr}("s\_output\_token\_sum") = \sigma_p^0("s\_output\_token\_sum")$. Rewriting the above,
$\boxed{s_0.M(p) = \sigma_p^0("s\_marking") + \sigma_p^0("s\_input\_token\_sum") - \sigma_p^0("s\_output\_token\_sum").}$

Detail the two lemmas giving this property.

By property of the $\mathcal{H}$-VHDL initialization relation, $\sigma_p^0("s\_input\_token\_sum") = 0$ and
$\sigma_p^0("s\_output\_token\_sum") = 0$. Rewriting the above, $\boxed{s_0.M(p) = \sigma_p^0("s\_marking").}$

Applying the Initial States Equal Marking lemma, $s_0.M(p) = \sigma_p^0("s\_marking").$

□

1.4.2 First rising edge and time counters

Lemma 13 (First Rising Edge Equal Time Counters). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_\uparrow, \sigma$, E_c, E_p, τ that verify the hypotheses of Def. 9, then*
$\forall t \in T_i, id_t \in Comps(\Delta), \sigma_t \in \Sigma(\Delta(id_t))$ *s.t.* $\gamma(t) = id_t$ *and* $\sigma(id_t) = \sigma_t$,
$upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t("s\_tc") \wedge$
$upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t("s\_tc") = lower(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t("s\_tc") = upper(I_s(t)) \wedge$
$upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t("s\_tc")$.

Proof. Given a $t \in T_i$, an $id_t \in Comps(\Delta)$ and a $\sigma_t \in \Sigma(\Delta(id_t))$ *s.t.* $\gamma(t) = id_t$ and $\sigma(id_t) = \sigma_t$, let's show that:

1. $\boxed{upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t("s\_tc")}$
2. $\boxed{upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t)) \Rightarrow \sigma_t("s\_tc") = lower(I_s(t))}$
3. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t)) \Rightarrow \sigma_t("s\_tc") = upper(I_s(t))}$
4. $\boxed{upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t)) \Rightarrow s_0.I(t) = \sigma_t("s\_tc")}$

By definition of id_t, there exist gm_t, ipm_t, opm_t s.t. $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$.

By property of the $\mathcal{H}$-VHDL elaboration relation, the $\mathcal{H}$-VHDL initialization relation, the $\texttt{Inject}_{\uparrow}$ relation, the $\mathcal{H}$-VHDL rising edge relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, there exist a $\sigma_t^e, \sigma_t^0, \sigma_t^{injr}, \sigma_t^r \in \Sigma(\Delta)$ s.t. $\sigma_e(id_t) = \sigma_t^e$ and $\sigma_0(id_t) = \sigma_t^0$ and $\sigma_i(id_t) = \sigma_t^{injr}$ and $\sigma_r(id_t) = \sigma_t^r$.

> From the elaboration to the end of the first rising edge phase, an internal state is associated with the T component instance id_t in the component store of the top-level design d.

Then, let's show the 4 previous subgoals.

1. Assume $upper(I_s(t)) = \infty \wedge s_0.I(t) \leq lower(I_s(t))$, then show $\boxed{s_0.I(t) = \sigma_t("s\_tc").}$
   By property of the $\texttt{Inject}_{\uparrow}$ relation, the $\mathcal{H}$-VHDL rising edge and stabilize relations, and
   $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs, \sigma_t("s\_tc") = \sigma_t^0("s\_tc")$.

   > The above equality is deduced from the two following facts:
   >
   > - The process "`time_counter`" is the only process that assigns signal `s_tc` in the T component behavior, and it is never executed during the rising edge and stabilization phases.
   > - The values of component instances' internal signals are invariant through the $\texttt{Inject}_{\uparrow}$ relation.

   Rewriting $\sigma_t("s\_tc")$ as $\sigma_t^0("s\_tc")$, $\boxed{s_0.I(t) = \sigma_t^0("s\_tc").}$

   Applying the Initial States Equal Time Counters lemma, $s_0.I(t) = \sigma_t^0("s\_tc").$

2. Assume $upper(I_s(t)) = \infty \wedge s_0.I(t) > lower(I_s(t))$, then show $\boxed{\sigma_t("s\_tc") = lower(I_s(t))}$.
   By definition, $lower(I_s(t)) \in \mathbb{N}^*$ and $s_0.I(t) = 0$. Then, $lower(I_s(t)) < 0$ is a contradiction.

3. Assume $upper(I_s(t)) \neq \infty \wedge s_0.I(t) > upper(I_s(t))$, then show $\boxed{\sigma_t("s\_tc") = upper(I_s(t))}$.
   By definition, $upper(I_s(t)) \in \mathbb{N}^*$ and $s_0.I(t) = 0$. Then, $upper(I_s(t)) < 0$ is a contradiction.

4. Assume $upper(I_s(t)) \neq \infty \wedge s_0.I(t) \leq upper(I_s(t))$, then show $\boxed{s_0.I(t) = \sigma_t("s\_tc")}$.

By property of the $\texttt{Inject}_{\uparrow}$ relation, the $\mathcal{H}$-VHDL rising edge and stabilize relations, and
$\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, $\sigma_t("s\_tc") = \sigma_t^0("s\_tc")$.

Rewriting $\sigma_t("s\_tc")$ as $\sigma_t^0("s\_tc")$, $\boxed{s_0.I(t) = \sigma_t^0("s\_tc").}$

Applying the Initial States Equal Time Counters lemma, $s_0.I(t) = \sigma_t^0("s\_tc").$

□

1.4.3 First rising edge and reset orders

Lemma 14 (First Rising Edge Equal Reset Orders). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma$, E_c, E_p, τ that verify the hypotheses of Def. 9, then*
$\forall t \in T, id_t \in Comps(\Delta)$ *s.t.* $\gamma(t) = id_t$,
$s_0.reset_t(t) = \sigma(id_t)("s\_reinit\_time\_counter")$.

Constants and signals reference			
Full name	*Alias*	*Category*	*Type*
$"s\_reinit\_time\_counter"$	$"srtc"$	internal signal (T)	$\mathbb{B}$
$"input\_arcs\_number"$	$"ian"$	generic constant (T)	$\mathbb{N}$
$"reinit\_time"$	$"rt"$	input port (T)	$\mathbb{B}$
$"reinit\_transition\_time"$	$"rtt"$	output port (P)	$\mathbb{B}$
$"output\_arcs\_types"$	$"oat"$	input port (P)	$\{\texttt{BASIC}, \texttt{TEST}, \texttt{INHIB}\}$
$"s\_marking"$	$"sm"$	internal signal (P)	$\mathbb{N}$
$"s\_output\_token\_sum"$	$"sots"$	internal signal (P)	$\mathbb{N}$
$"output\_arcs\_weights"$	$"oaw"$	input port (P)	$\mathbb{N}$
$"output\_transition\_fired"$	$"otf"$	input port (P)	$\mathbb{B}$

Proof. Given a $t \in T$ and an $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, let us show that $\boxed{s_0.reset_t(t) = \sigma(id_t)("srtc").}$
By definition of id_t, there exist gm_t, ipm_t, opm_t s.t.
$\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$.
By property of the $\mathcal{H}$-VHDL stabilize relation and $\texttt{comp}(id_t, "transition", gm_t, ipm_t, opm_t) \in d.cs$, then $\sigma(id_t)("srtc") = \sum\limits_{i=0}^{\Delta(id_t)("input\_arcs\_number")-1} \sigma(id_t)("reinit\_time")[i]$.

$$\boxed{s_0.reset_t(t) = \sum_{i=0}^{\Delta(id_t)("ian")-1} \sigma(id_t)("rt")[i].}$$

Case analysis on $input(t)$ (2 CASES):

- **CASE** $input(t) = \emptyset$:

  By construction, $<\texttt{input\_arcs\_number} \Rightarrow 1> \in gm_t$, and by property of the $\mathcal{H}$-VHDL elaboration relation, then $\Delta(id_t)("ian") = 1$. By construction, $<$

$\texttt{reinit\_time}(0) \Rightarrow \texttt{false} > \in ipm_t$, and by property of the $\mathcal{H}$-VHDL stabilize relation, $\sigma(id_t)(''rt'')[0] = false$.

Rewriting $\Delta(id_t)(''ian'')$ as 1 and $\sigma(id_t)(''rt'')[0]$ as $false$, and by definition of s_0, $s_0.reset_t(t) = \sum\limits_{i=0}^{\Delta(''ian'')-1} \sigma(id_t)(''rt'')[i] = \sigma(id_t)(''rt'')[0] = false.$

- **CASE** $input(t) \neq \emptyset$:

  By construction, $<\texttt{input\_arcs\_number} \Rightarrow |input(t)|> \in gm_t$, and by property of the $\mathcal{H}$-VHDL elaboration relation, then $\Delta(id_t)(''ian'') = |input(t)|$.

  Rewriting $\Delta(id_t)(''ian'')$ as $|input(t)|$, $\boxed{s_0.reset_t(t) = \sum\limits_{i=0}^{|input(t)|-1} \sigma(id_t)(''rt'')[i].}$

  By definition of s_0, $s_0.reset_t(t) = false$. Rewriting $s_0.reset_t(t)$ as $false$, $\boxed{\sum\limits_{i=0}^{|input(t)|-1} \sigma(id_t)(''rt'')[i] = false.}$

  Given a $i \in [0, |input(t)| - 1]$, let us show $\boxed{\sigma(id_t)(''rt'')[i] = false.}$

  By construction, and $input(t) \neq \emptyset$, there exist $p \in input(t)$ and $id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$.

  By definition of id_p, there exist gm_p, ipm_p, opm_p s.t. $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$.By construction for all $i \in [0, |input(t)| - 1]$, there exist $j \in [0, |output(p)| - 1]$ and $id_{ji} \in Sigs(\Delta)$ s.t. $<\texttt{reinit\_transition\_time}(\texttt{j}) \Rightarrow \texttt{id}_{\texttt{ji}}> \in$ opm_p and $<\texttt{reinit\_time}(\texttt{i}) \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$.

  By property of the $\mathcal{H}$-VHDL stabilize relation, $<\texttt{reinit\_transition\_time}(\texttt{j}) \Rightarrow \texttt{id}_{\texttt{ji}}> \in$ opm_p and $<\texttt{reinit\_time}(\texttt{i}) \Rightarrow \texttt{id}_{\texttt{ji}}> \in ipm_t$, then $\sigma(id_t)(''rt'')[i] = \sigma(id_{ji}) = \sigma(id_p)(''rtt'')[j]$.

  Rewriting $\sigma(id_t)(''rt'')[i]$ as $\sigma(id_{ji})$ and $\sigma(id_{ji})$ as $\sigma(id_p)(''rtt'')[j]$, $\boxed{\sigma(id_p)(''rtt'')[j] = false.}$

  By property of the $\mathcal{H}$-VHDL rising edge and stabilize relations,

  $$\begin{aligned} \sigma(id_p)(''rtt'')[j] =& ((\sigma_0(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma_0(id_p)(''oat'')[j] = \texttt{TEST}) \\ & .(\sigma_0(id_p)(''sm'') - \sigma_0(id_p)(''sots'') < \sigma_0(id_p)(''oaw'')[j]) \\ & .(\sigma_0(id_p)(''sots'') > 0)) \\ & + (\sigma_0(id_p)(''otf'')[j]) \end{aligned}$$

  Rewriting the goal with the above equation,

  $$\boxed{\begin{aligned} false =& ((\sigma_0(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma_0(id_p)(''oat'')[j] = \texttt{TEST}) \\ & .(\sigma_0(id_p)(''sm'') - \sigma_0(id_p)(''sots'') < \sigma_0(id_p)(''oaw'')[j]) \\ & .(\sigma_0(id_p)(''sots'') > 0)) \\ & + (\sigma_0(id_p)(''otf'')[j]) \end{aligned}}$$

Add a lemma + proof in section initial states for fired = false after initialization.

By property of the $\mathcal{H}$-VHDL initialization and the $\texttt{Inject}_{\uparrow}$ relations, then $\sigma_0(id_p)(''otf'')[j] = false$. Rewriting $\sigma_0(id_p)(''otf'')[j]$ as $false$ and simplifying the goal,

$$\boxed{\begin{aligned} false =& ((\sigma_0(id_p)(''oat'')[j] = \texttt{BASIC} + \sigma_0(id_p)(''oat'')[j] = \texttt{TEST}) \\ & .(\sigma_0(id_p)(''sm'') - \sigma_0(id_p)(''sots'') < \sigma_0(id_p)(''oaw'')[j]) \\ & .(\sigma_0(id_p)(''sots'') > 0)) \end{aligned}}$$

Add a lemma + proof in section initial states for output token sum = 0 after initialization.

By property of the $\mathcal{H}$-VHDL initialization and the $\texttt{Inject}_{\uparrow}$ relations, then $\sigma_0(id_p)(''sots'') = 0$. Rewriting $\sigma_0(id_p)(''sots'')$ as 0 and simplifying the goal, $false = false$

□

1.4.4 First rising edge and action executions

Lemma 15 (First Rising Edge Equal Action Executions). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma$, E_c, E_p, τ that verify the hypotheses of Def. 9, then $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, $s_0.ex(a) = \sigma(id_a)$.*

Proof. Given an $a \in \mathcal{A}$ and an $id_a \in Outs(\Delta)$ s.t. $\gamma(a) = id_a$, let us show that $\boxed{s_0.ex(a) = \sigma(id_a).}$

Rewriting $s_0.ex(a)$ as $false$, by definition of s_0, $\boxed{\sigma(id_a) = false.}$
By construction, id_a is an output port identifier of boolean type in the $\mathcal{H}$-VHDL design d assigned only during a falling edge phase in the ‘‘action’’ process.
By property of the $\mathcal{H}$-VHDL $\texttt{Inject}_{\uparrow}$, rising edge and stabilize relations, then $\sigma(id_a) = \sigma_0(id_a)$.
Thanks to the Lemma Initial States Equal Action Executions, $\sigma_0(id_a) = false$.
Rewriting $\sigma(id_a)$ as $\sigma_0(id_a)$, and $\sigma_0(id_a)$ as $false$, $false = false.$

□

1.4.5 First rising edge and function executions

Lemma 16 (First Rising Edge Equal Function Executions). *For all $sitpn, d, \gamma, \Delta, \sigma_e, \sigma_0, \sigma_i, \sigma_{\uparrow}, \sigma$, E_c, E_p, τ that verify the hypotheses of Def. 9, then $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, $s_0.ex(f) = \sigma(id_f)$.*

Proof. Given an $f \in \mathcal{F}$ and an $id_f \in Outs(\Delta)$ s.t. $\gamma(f) = id_f$, let us show that $\boxed{s_0.ex(f) = \sigma(id_f).}$

Rewriting $s_0.ex(f)$ as $false$, by definition of s_0, $\boxed{\sigma(id_f) = false.}$
By construction, the ‘‘`function`’’ process is a part of design d's behavior, i.e $\texttt{ps}("function", \emptyset, sl, ss) \in d.cs$.
By construction id_f is an output port of design d, and it is only assigned in the body of the ‘‘`function`’’ process. Let $trs(f)$ be the set of transitions associated to function f, i.e $trs(f) = \{t \in T \mid \mathbb{F}(t,f) = true\}$. Then, depending on $trs(f)$, there are two cases of assignment of output port id_f:

- **CASE** $trs(f) = \emptyset$:
  By construction, $\texttt{id}_\texttt{f} \Leftarrow \texttt{false} \in ss_\uparrow$ where $ss_\uparrow$ is the part of the ‘‘`function`’’ process body executed during the rising edge phase.

  By property of the $\mathcal{H}$-VHDL rising edge and the stabilize relation, then $\sigma(id_f) = false.$

- **CASE** $trs(f) \neq \emptyset$:
  By construction, $\texttt{id}_\texttt{f} \Leftarrow \texttt{id}_{\texttt{ft}_0} + \cdots + \texttt{id}_{\texttt{ft}_\texttt{n}} \in ss_\uparrow$ where $ss_\uparrow$ is the part of the ‘‘`function`’’ process body executed during the rising edge phase, and $n = |trs(f)| - 1$, and for all $i \in [0, n-1]$, id_{ft_i} is a internal signal of design d.

  By property of the $\texttt{Inject}_\uparrow$, the $\mathcal{H}$-VHDL rising edge and stabilize relation, then $\sigma(id_f) = \sigma_0(id_{ft_0}) + \cdots + \sigma_0(id_{ft_n})$.

  Rewriting $\sigma(id_f)$ as $\sigma_0(id_{ft_0}) + \cdots + \sigma_0(id_{ft_n})$, then $\boxed{\sigma_0(id_{ft_0}) + \cdots + \sigma_0(id_{ft_n}) = false.}$

  By construction, for all id_{ft_i}, there exist a $t_i \in trs(f)$ and an id_{t_i} s.t. $\gamma(t_i) = id_{t_i}$.

  By definition of id_{t_i}, there exist gm_{t_i}, ipm_{t_i} and opm_{t_i} s.t. $\texttt{comp}(id_{t_i}, "transition", gm_{t_i}, ipm_{t_i}, opm_{t_i}) \in d.cs$.

  By construction, $<\texttt{fired} \Rightarrow \texttt{id}_{\texttt{ft}_\texttt{i}}> \in opm_{t_i}$, and by property of the initialization relation $\sigma_0(id_{ft_i}) = \sigma_0(id_{t_i})("fired")$.

  Rewriting $\sigma_0(id_{ft_i})$ as $\sigma_0(id_{t_i})("fired")$, then $\boxed{\sigma_0(id_{t_0})("fired") + \cdots + \sigma_0(id_{t_n})("fired") = false.}$

  By property of the initialization relation, we know that for all $t \in T$ and $id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$, then $\sigma_0(id_t)("fired") = false$.

  Rewriting all $\sigma_0(id_{t_i})("fired")$ as $false$ and simplifying the goal, then $false = false.$

□

1.5 Rising Edge

Definition 10 (Rising Edge Hypotheses). *Given an* $sitpn \in SITPN$, $d \in design$, $\gamma \in WM(sitpn, d)$, $E_c \in \mathbb{N} \to \mathcal{C} \to \mathbb{B}$, $\Delta \in ElDesign(d, \mathcal{D}_\mathcal{H})$, $E_p \in (\mathbb{N} \times \{\uparrow, \downarrow\}) \to Ins(\Delta) \to value$, $\tau \in \mathbb{N}$, $s, s' \in S(sitpn)$, $\sigma_e, \sigma, \sigma_i, \sigma_\uparrow, \sigma' \in \Sigma(\Delta)$, $\theta \in \texttt{list}(\Sigma(\Delta))$, *assume that:*

- $\lfloor sitpn \rfloor_{\mathcal{H}} = (d, \gamma)$ *and* $\gamma \vdash E_p \overset{env}{=} E_c$ *and* $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash \mathrm{d} \xrightarrow{elab} \Delta, \sigma_e$
- $\gamma \vdash s \overset{\downarrow}{\sim} \sigma$
- $E_c, \tau \vdash s \xrightarrow{\uparrow} s'$
- $\mathtt{Inject}_{\uparrow}(\sigma, E_p, \tau, \sigma_i)$ *and* $\Delta, \sigma_i \vdash \mathrm{d.cs} \xrightarrow{\uparrow} \sigma_{\uparrow}$ *and* $\Delta, \sigma_{\uparrow} \vdash \mathrm{d.cs} \xrightarrow{\theta} \sigma'$

Lemma 17 (Rising Edge). *For all $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_{\uparrow}$, σ', θ that verify the hypotheses of Def. 10, then $\gamma, E_c, \tau \vdash s' \overset{\uparrow}{\sim} \sigma'$.*

Proof. By definition of Post Rising Edge State Similarity, there are 7 points to prove.

1. $\forall p \in P, id_p \in Comps(\Delta)$ $s.t.$ $\gamma(p) = id_p$, $s'.M(p) = \sigma'(id_p)(''s\_marking'')$.
2. $\forall t \in T_i, id_t \in Comps(\Delta)$ $s.t.$ $\gamma(t) = id_t$,
   $(upper(I_s(t)) = \infty \wedge s'.I(t) \leq lower(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))$
   $\wedge (upper(I_s(t)) = \infty \wedge s'.I(t) > lower(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = lower(I_s(t)))$
   $\wedge (upper(I_s(t)) \neq \infty \wedge s'.I(t) > upper(I_s(t)) \Rightarrow \sigma'(id_t)(''s\_time\_counter'') = upper(I_s(t)))$
   $\wedge (upper(I_s(t)) \neq \infty \wedge s'.I(t) \leq upper(I_s(t)) \Rightarrow s'.I(t) = \sigma'(id_t)(''s\_time\_counter''))$.
3. $\forall t \in T_i, id_t \in Comps(\Delta)$ $s.t.$ $\gamma(t) = id_t$,
   $s'.reset_t(t) = \sigma'(id_t)(''s\_reinit\_time\_counter'')$.
4. $\forall a \in \mathcal{A}, id_a \in Outs(\Delta)$ $s.t.$ $\gamma(a) = id_a$, $s'.ex(a) = \sigma'(id_a)$.
5. $\forall f \in \mathcal{F}, id_f \in Outs(\Delta)$ $s.t.$ $\gamma(f) = id_f$, $s'.ex(f) = \sigma'(id_f)$.
6. $\forall t \in T, id_t \in Comps(\Delta)$ $s.t.$ $\gamma(t) = id_t$,
   $t \in Sens(s'.M) \Leftrightarrow \sigma'(id_t)(''s\_enabled'') = \mathtt{true}$.
7. $\forall t \in T, id_t \in Comps(\Delta)$ $s.t.$ $\gamma(t) = id_t$,
   $$\sigma'(id_t)(''s\_condition\_combination'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \mathtt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}$$
   where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

Each point is proved by a separate lemma:

– Apply Lemma Rising Edge Equal Marking to solve 1.

– Apply "Rising Edge Equal Time Counter" lemma to solve 2.

– Apply "Rising Edge Equal Reset Order" lemma to solve 3.

– Apply "Rising Edge Equal Action" lemma to solve 4.

– Apply "Rising Edge Equal Function" lemma to solve 5.

– Apply "Rising Edge Equal Sensitized" lemma to solve 6.

- Apply Lemma Rising Edge Equal Condition Combination to solve 7.

□

1.5.1 Rising Edge and Marking

Lemma 18 (Rising Edge Equal Marking). *For all $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ', θ that verify the hypotheses of Def. 10, then $\forall p, id_p$ s.t. $\gamma(p) = id_p$ and $\sigma'(id_p) = \sigma'_p$, $s'.M(p) = \sigma'_p(''s\_marking'')$.*

Constants and signals reference			
Full name	*Alias*	*Category*	*Type*
$''s\_marking''$	$''sm''$	internal signal (P)	$\mathbb{N}$
$''s\_output\_token\_sum''$	$''sots''$	internal signal (P)	$\mathbb{N}$
$''s\_input\_token\_sum''$	$''sits''$	internal signal (P)	$\mathbb{N}$

Proof. Given a $p \in P$, let us show $\boxed{s'.M(p) = \sigma'(id_p)(''s\_marking'').}$
By definition of id_p, there exist gm_p, ipm_p, opm_p s.t.
$\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$.By definition of the SITPN state transition relation on rising edge:

$$s'.M(p) = s.M(p) - \sum_{t \in Fired(s)} pre(p,t) + \sum_{t \in Fired(s)} post(t,p) \tag{1.5}$$

By property of the $\texttt{Inject}_\uparrow$, the $\mathcal{H}$-VHDL rising edge and the stabilize relations, and $\texttt{comp}(id_p, ''place'', gm_p, ipm_p, opm_p) \in d.cs$:

$$\begin{aligned}\sigma'(id_p)(''sm'') = \sigma(id_p)(''sm'') - \sigma(id_p)(''s\_output\_token\_sum'')\\ +\sigma(id_p)(''s\_input\_token\_sum'')\end{aligned} \tag{1.6}$$

By the definition of Post Falling Edge State Similarity relation:

$$s.M(p) = \sigma(id_p)(''sm'') \tag{1.7}$$

$$\sum_{t \in Fired(s)} pre(p,t) = \sigma(id_p)(''sots'') \tag{1.8}$$

$$\sum_{t \in Fired(s)} post(t,p) = \sigma(id_p)(''sits'') \tag{1.9}$$

Rewriting the goal with 1.5, 1.6, 1.7, 1.8 and 1.9, tautology .

□

1.5.2 Rising edge and condition combination

Lemma 19 (Rising Edge Equal Condition Combination). *For all $sitpn$, d, γ, E_c, E_p, τ, Δ, σ_e, s, s', σ, σ_i, $\sigma_\uparrow$, σ', θ that verify the hypotheses of Def. 10, then*
$\forall t \in T, id_t \in Comps(\Delta)$ s.t. $\gamma(t) = id_t$,

$$\sigma'(id_t)(''s\_condition\_combination'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \mathtt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}$$
where $conds(t) = \{c \in \mathcal{C} \mid \mathbb{C}(t,c) = 1 \vee \mathbb{C}(t,c) = -1\}$.

Constants and signals reference			
Full name	*Alias*	*Category*	*Type*
"s_condition_combination"	*"scc"*	internal signal (T)	$\mathbb{B}$
"conditions_number"	*"cn"*	generic constant (T)	$\mathbb{N}$
"input_conditions"	*"ic"*	input port (T)	$\mathbb{B}$

Proof. Given a t, id_t, σ_t s.t. $\gamma(t) = id_t$, let us show

$$\boxed{\sigma'(id_t)(''s\_condition\_combination'') = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \mathtt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}.}$$

By definition of id_t, there exist gm_t, ipm_t, opm_t s.t.
$\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs.$
By property of the $\mathcal{H}$-VHDL stabilize relation, and
$\mathtt{comp}(id_t, ''transition'', gm_t, ipm_t, opm_t) \in d.cs$:

$$\sigma'(id_t)(''scc'') = \prod_{i=0}^{\Delta(id_t)(''conditions\_number'')-1} \sigma'(id_t)(''input\_conditions'')[i] \tag{1.10}$$

Rewriting the goal with 1.10,

$$\boxed{\prod_{i=0}^{\Delta(id_t)(''cn'')-1} \sigma'(id_t)(''ic'')[i] = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \mathtt{not}(E_c(\tau, c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}.}$$

Case analysis on $conds(t)$ (2 CASES):

- **CASE** $conds(t) = \emptyset$:

  $$\boxed{\prod_{i=0}^{\Delta(id_t)(''cn'')-1} \sigma'(id_t)(''ic'')[i] = \mathtt{true}.}$$

  By construction, $<\mathtt{conditions\_number} \Rightarrow 1> \in gm_t$ and $<\mathtt{input\_conditions}(0) \Rightarrow \mathtt{true}> \in ipm_t$.

  By property of the stabilize relation, $<\mathtt{conditions\_number} \Rightarrow 1> \in gm_t$ and $<\mathtt{input\_conditions}(0) \Rightarrow \mathtt{true}> \in ipm_t$:

  $$\Delta(id_t)(''cn'') = 1 \tag{1.11}$$
  $$\sigma'(id_t)(''ic'')[0] = \mathtt{true} \tag{1.12}$$

  Rewriting the goal with 1.11 and 1.12, tautology.

- **CASE** $conds(t) \neq \emptyset$:
  By construction, $<\texttt{conditions\_number} \Rightarrow |\texttt{conds(t)}|> \in gm_t$, and by property of the stabilize relation:

  $$\Delta(id_t)(''cn'') = |conds(t)| \tag{1.13}$$

  Rewriting the goal with (1.13),

  $$\boxed{\prod_{i=0}^{|conds(t)|-1} \sigma'(id_t)(''ic'')[i] = \prod_{c \in conds(t)} \begin{cases} E_c(\tau, c) & if\ \mathbb{C}(t,c) = 1 \\ \texttt{not}(E_c(\tau,c)) & if\ \mathbb{C}(t,c) = -1 \end{cases}.}$$

  Then, 2 subgoals to prove the equation:

  1. $\boxed{\forall c \in conds(t),\ \exists i \in [0, \Delta(id_t)(''cn'') - 1]\ s.t.\ \mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$
     Given a $c \in conds(t)$, let us show that
     $\boxed{\exists i \in [0, \Delta(id_t)(''cn'') - 1]\ s.t.\ \mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$
     By definition of $c \in conds(t)$, there are 2 cases:
     - **CASE** $\mathbb{C}(t,c) = 1$:
       By construction, there exists $id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, and there exists $i \in [0, |conds(t)| - 1]$ s.t. $<\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$.
       As $\Delta(id_t)(''cn'') = |conds(t)|$, then we have $i \in [0, \Delta(id_t)(''cn'') - 1]$.
       Let us take this i to prove the goal,
       $\boxed{\mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$
       The right part of the goal is proved by contradiction, then what is left to prove is: $\boxed{\mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c).}$

       Assuming $\mathbb{C}(t,c) = 1$, let us show $\boxed{\sigma'(id_t)(''ic'')[i] = E_c(\tau,c).}$

       By property of the stabilize relation and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$, then $\sigma'(id_t)(''ic'')[i] = \sigma(id_c)$.
       By property of the $\mathcal{H}$-VHDL $\texttt{Inject}_\uparrow$, the rising edge, the stabilize relations, and $id_c \in Ins(\Delta)$, then $\sigma(id_c) = E_p(\tau, \uparrow)(id_c)$.
       By property of $\gamma \vdash E_p \overset{env}{=} E_c$, then $E_p(\tau, \uparrow)(id_c) = E_c(\tau, c)$.
       Rewriting the goal with the above equations, $\sigma'(id_t)(''ic'')[i] = E_c(\tau,c).$
     - **CASE** $\mathbb{C}(t,c) = -1$:
       By construction, there exists $id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, and there exists $i \in [0, |conds(t)| - 1]$ s.t. $<\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$.
       As $\Delta(id_t)(''cn'') = |conds(t)|$, then we have $i \in [0, \Delta(id_t)(''cn'') - 1]$.
       Let us take this i to prove the goal,

$\boxed{\mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$

The left part of the goal is proved by contradiction, then what is left to prove is: $\boxed{\mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$

Assuming $\mathbb{C}(t,c) = -1$, let us show $\boxed{\sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$

By property of the stabilize relation and $<\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$, then $\sigma'(id_t)(''ic'')[i] = \texttt{not}\ \sigma(id_c)$.
By property of the $\mathcal{H}$-VHDL $\texttt{Inject}_\uparrow$, the rising edge, the stabilize relations, and $id_c \in Ins(\Delta)$, then $\sigma(id_c) = E_p(\tau,\uparrow)(id_c)$.
By property of $\gamma \vdash E_p \overset{env}{=} E_c$, then $E_p(\tau,\uparrow)(id_c) = E_c(\tau,c)$.
Rewriting the goal with the above equations, $\sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c)$.

2. $\boxed{\forall i \in [0,\Delta(id_t)(''cn'')-1],\ \exists c \in conds(t),\ s.t.\ \mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$

Given a $i \in [0,\Delta(id_t)(''cn'')-1]$, let us show
$\boxed{\exists c \in conds(t),\ s.t.\ \mathbb{C}(t,c) = 1 \Rightarrow \sigma'(id_t)(''ic'')[i] = E_c(\tau,c) \wedge \mathbb{C}(t,c) = -1 \Rightarrow \sigma'(id_t)(''ic'')[i] = \texttt{not}\ E_c(\tau,c).}$

By construction, there exists $c \in conds(t)$ and $id_c \in Ins(\Delta)$ s.t. $\gamma(c) = id_c$, and $\mathbb{C}(t,c) = 1 \Rightarrow <\texttt{input\_conditions(i)} \Rightarrow \texttt{id}_\texttt{c}> \in ipm_t$ and $\mathbb{C}(t,c) = -1 \Rightarrow <\texttt{input\_conditions(i)} \Rightarrow \texttt{not id}_\texttt{c}> \in ipm_t$.
Let us take such an $c \in conds(t)$ to prove the goal. By definition of $c \in conds(t)$, there are 2 cases: see 1 for the remainder of the proof.

□

1.6 Falling Edge

Definition 11 (Falling Edge Hypotheses). *Given an sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ', assume that:*

- $\lfloor sitpn \rfloor_{\mathcal{H}} = (d,\gamma)$ *and* $\gamma \vdash E_p \overset{env}{=} E_c$ *and* $\mathcal{D}_{\mathcal{H}}, \emptyset \vdash \mathrm{d} \xrightarrow{elab} \Delta, \sigma_e$
- $\gamma, E_c, \tau \vdash s \overset{\uparrow}{\sim} \sigma$
- $E_c, \tau \vdash sitpn, s \xrightarrow{\downarrow} s'$
- $\texttt{Inject}_\downarrow(\sigma, E_p, \tau, \sigma_i)$ *and* $\Delta, \sigma_i \vdash \mathrm{d.cs} \xrightarrow{\downarrow} \sigma_\downarrow$ *and* $\Delta, \sigma_\downarrow \vdash \mathrm{d.cs} \xrightarrow{\theta} \sigma'$

Lemma 20 (Falling Edge). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, then $\gamma \vdash s' \overset{\downarrow}{\sim} \sigma'$.*

1.6.1 Falling Edge and Marking

Lemma 21 (Falling Edge Prepare Marking Update). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, then $\forall p, id_p$ s.t. $\gamma(p) = id_p$ and $\sigma'(id_p) = \sigma'_p$:*

- $\sum\limits_{t \in Fired(s')} pre(p,t) = \sigma'_p(\text{"}s\_output\_token\_sum\text{"})$
- $\sum\limits_{t \in Fired(s')} post(t,p) = \sigma'_p(\text{"}s\_input\_token\_sum\text{"})$

Proof. First, by reasoning on the VHDL falling and stabilize relation, and on the VHDL Place component behavior, we can unfold the value of signals "s_input_token_sum" and "s_output_token_sum" at state σ'_p.

- $\sigma'_p(\text{"}s\_input\_token\_sum\text{"}) = \sum\limits_{i \in \texttt{FIIdx}(\sigma'_p)} \sigma'_p(\text{"}input\_arcs\_weights\text{"})(i)$
  where $i \in \texttt{FIIdx}(\sigma) \equiv i \in [0, \sigma(\text{"}input\_arcs\_number\text{"}) - 1]$
  $\wedge\, \sigma(\text{"}input\_transition\_fired\text{"})(i) = \texttt{true}$.
- $\sigma'_p(\text{"}s\_output\_token\_sum\text{"}) = \sum\limits_{i \in \texttt{FOIdx}(\sigma'_p)} \sigma'_p(\text{"}output\_arcs\_weights\text{"})(i)$
  where $i \in \texttt{FOIdx}(\sigma) \equiv i \in [0, \sigma(\text{"}output\_arcs\_number\text{"}) - 1]$
  $\wedge\, \sigma(\text{"}output\_transition\_fired\text{"})(i) = \texttt{true}$.

Then, we need to prove the two following equalities:

- $\sum\limits_{t \in Fired(s')} pre(p,t) = \sum\limits_{i \in \texttt{FOIdx}(\sigma'_p)} \sigma'_p(\text{"}output\_arcs\_weights\text{"})(i)$.
- $\sum\limits_{t \in Fired(s')} post(t,p) = \sum\limits_{i \in \texttt{FIIdx}(\sigma'_p)} \sigma'_p(\text{"}input\_arcs\_weights\text{"})(i)$.

We can deduce that:

- $\sum\limits_{t \in Fired(s')} pre(p,t) = \sum\limits_{t \in FI(s',p)} pre(p,t)$
  where $t \in FI(s',p) \equiv t \in Fired(s') \wedge \exists \omega$ s.t. $pre(p,t) = (\omega, \texttt{basic})$.
- $\sum\limits_{t \in Fired(s')} post(t,p) = \sum\limits_{t \in FO(s',p)} post(t,p)$
  where $t \in FO(s',p) \equiv t \in Fired(s') \wedge \exists \omega$ s.t. $post(t,p) = \omega$.

Then, we have $|FI(s',p)| = |\texttt{FIIdx}(\text{œ}'_{\text{p}})|$ and $|FO(s',p)| = |\texttt{FOIdx}(\text{œ}'_{\text{p}})|$.
Then, it easier to show the two equalities by showing that the sets $FI(s',p)$ (resp. $FO(s',p)$) and $\texttt{FIIdx}(\sigma'_p)$ (resp. $\texttt{FOIdx}(\sigma'_p)$) are in bijection.
There are 4 subgoals to prove.

1. $\forall t \in FI(s',p), \exists i \in \texttt{FOIdx}(\sigma'_p)$ s.t. $pre(p,t) = \sigma'_p(\text{"}output\_arcs\_weights\text{"})(i)$.
   Given a transition $t \in FI(s',p)$, by definition:

- $\exists\omega$ *s.t.* $pre(p,t) = (\omega, \texttt{basic})$.
  Then, by construction, there exists a Transition component $id_t \in Comps(\Delta)$ implementing transition t, and there exists an index $j \in [0, |output(p)| - 1]$, and a signal $sig \in Sigs(\Delta)$ such that
  $\texttt{id}_\texttt{t}\texttt{.fired} \Rightarrow \texttt{sig} \Rightarrow \texttt{id}_\texttt{p}\texttt{.output\_transitions\_fired(j)}$
  and $\texttt{id}_\texttt{p}\texttt{.output\_arcs\_weights(j)} \Rightarrow \texttt{!}$
  and $\texttt{id}_\texttt{p}\texttt{.output\_arcs\_number} \Rightarrow |\texttt{output(p)}|$.
  Then, by reasoning on the VHDL stabilize relation, we can deduce $j \in [0, \sigma'_p(''output\_arcs\_number'') - 1]$
  and $\sigma'_p(''output\_arcs\_weights'')(j) = \omega$.
- $t \in Fired(s')$.
  Thanks to Lemma Falling Edge Equal Fired, we know that $\sigma'_t(''fired'') = \texttt{true}$.
  Then, by reasoning on the VHDL stabilize relation, we can deduce $\sigma'_p(''output\_transitions\_f$
  $\sigma'(sig) = \sigma'_t(''fired'') = \texttt{true}$.

Then, choose index j to solve the goal.

2. $\forall i \in \texttt{FOIdx}(\sigma'_p), \exists t \in FI(s', p)$ *s.t.* $pre(p,t) = \sigma'_p(''output\_arcs\_weights'')(i)$.
3. $\forall t \in FO(s', p), \exists i \in \texttt{FIIdx}(\sigma'_p)$ *s.t.* $post(t,p) = \sigma'_p(''input\_arcs\_weights'')(i)$.
4. $\forall i \in \texttt{FIIdx}(\sigma'_p), \exists t \in FO(s', p)$ *s.t.* $post(t,p) = \sigma'_p(''input\_arcs\_weights'')(i)$.

□

Lemma 22 (Falling Edge Computes Output Token Sum). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, then*
$\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$,
$\sigma'(id_p)(''s\_output\_token\_sum'') = \sum_{i=0}^{n} (\texttt{if } \sigma'(id_p)(''out\_t\_fired'')[i] \texttt{ then } \sigma'(id_p)(''out\_arcs\_weights'')[i]$
where $n = \Delta(id_p)(''output\_arcs\_number'') - 1$

Proof. Given a $p \in P$ and a $id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$, let us show that

$$\boxed{\begin{array}{l}\sigma'(id_p)(''s\_output\_token\_sum'') = \sum_{i=0}^{n} (\texttt{if } \sigma'(id_p)(''out\_t\_fired'')[i] \texttt{ then } \sigma'(id_p)(''out\_arcs\_weights'')[i \\ \text{where } n = \Delta(id_p)(''output\_arcs\_number'') - 1\end{array}}$$

Applying the Stabilize Computes Output Token Sum lemma,

$$\sigma'(id_p)(''s\_output\_token\_sum'') = \sum_{i=0}^{n} (\texttt{if } \sigma'(id_p)(''out\_t\_fired'')[i] \texttt{ then } \sigma'(id_p)(''out\_arcs\_weights'')[$$

□

Lemma 23 (Stabilize Computes Output Token Sum). *For all sitpn $\in$ SITPN, d $\in$ design, $\gamma \in WM(sitpn, d)$, $\Delta \in ElDesign(d, \mathcal{D}_\mathcal{H})$, $\sigma_e, \sigma, \sigma' \in \Sigma(\Delta)$, $\tau \in \mathbb{N}$, $\theta \in$* $\texttt{list}(\Sigma(\Delta))$, *assume that:*

- $\lfloor sitpn \rfloor_\mathcal{H} = (d, \gamma)$ *and* $\mathcal{D}_\mathcal{H}, \emptyset \vdash \mathrm{d} \xrightarrow{elab} \Delta, \sigma_e$

- $\Delta, \sigma \vdash \text{d.cs} \xrightarrow{\theta} \sigma'$

then
$\forall p \in P, id_p \in Comps(\Delta)$ *s.t.* $\gamma(p) = id_p$,

$$\sigma'(id_p)(\text{"}s\_output\_token\_sum\text{"}) = \sum_{i=0}^{n} \big(\texttt{if } \sigma'(id_p)(\text{"}output\_transition\_fired\text{"})[i] \texttt{ then } \sigma'(id_p)(\text{"}output\_arcs\_weights\text{"})[i] \texttt{ else } 0\big)$$

where $n = \Delta(id_p)(\text{"}output\_arcs\_number\text{"}) - 1$

Full signal name	Alias
$\text{"}s\_output\_token\_sum\text{"}$	$\text{"}sots\text{"}$
$\text{"}output\_transition\_fired\text{"}$	$\text{"}otf\text{"}$
$\text{"}output\_arcs\_weights\text{"}$	$\text{"}oaw\text{"}$
$\text{"}output\_arcs\_number\text{"}$	$\text{"}oan\text{"}$

Proof. Given a $p \in P$ and a $id_p \in Comps(\Delta)$ s.t. $\gamma(p) = id_p$, let us show that

$$\boxed{\sigma'(id_p)(\text{"}sots\text{"}) = \sum_{i=0}^{n} \big(\texttt{if } \sigma'(id_p)(\text{"}otf\text{"})[i] \texttt{ then } \sigma'(id_p)(\text{"}oaw\text{"})[i] \texttt{ else } 0\big) \text{ where } n = \Delta(id_p)(\text{"}oan\text{"}) - 1}$$

Induction on $\Delta, \sigma \vdash \text{d.cs} \xrightarrow{\theta} \sigma'$.

- **BASE CASE:**
  - $\Delta, \sigma \vdash \text{d.cs} \xrightarrow{[\,]} \sigma$
  - $\mathcal{E}(\sigma) = \emptyset$
  - $\sigma = \sigma'$

  $$\boxed{\sigma(id_p)(\text{"}sots\text{"}) = \sum_{i=0}^{n} \big(\texttt{if } \sigma(id_p)(\text{"}otf\text{"})[i] \texttt{ then } \sigma(id_p)(\text{"}oaw\text{"})[i] \texttt{ else } 0\big) \text{ where } n = \Delta(id_p)(\text{"}oan\text{"}) - 1}$$

  ➔ first pb, what's the value of $\sigma(id_p)(\text{"}sots\text{"})$?
  ➔ let's have it as an hypothesis that

  $$\boxed{\sigma(id_p)(\text{"}sots\text{"}) = \sum_{i=0}^{n} \big(\texttt{if } \sigma(id_p)(\text{"}otf\text{"})[i] \texttt{ then } \sigma(id_p)(\text{"}oaw\text{"})[i] \texttt{ else } 0\big) \text{ where } n = \Delta(id_p)(\text{"}oan\text{"}) - 1}$$

- **INDUCTION CASE:**
  - $\Delta, \sigma \vdash \text{d.cs} \rightarrow \sigma_1$ and $\Delta, \sigma \vdash \text{d.cs} \xrightarrow{\theta} \sigma$
  - $\mathcal{E}(\sigma) \neq \emptyset$ and $\mathcal{E}(\sigma') = \emptyset$

  ➔ Problem: our hypothesis is taken in the induction process.

$$\left(\sigma_1(id_p)(''sots'') = \sum_{i=0}^{n} (\texttt{if}\ \sigma_1(id_p)(''otf'')[i]\ \texttt{then}\ \sigma_1(id_p)(''oaw'')[i]\ \texttt{else}\ 0)\right) \Rightarrow$$
$$\sigma'(id_p)(''sots'') = \sum_{i=0}^{n} (\texttt{if}\ \sigma'(id_p)(''otf'')[i]\ \texttt{then}\ \sigma'(id_p)(''oaw'')[i]\ \texttt{else}\ 0)$$

$$\boxed{\sigma'(id_p)(''sots'') = \sum_{i=0}^{n} (\texttt{if}\ \sigma'(id_p)(''otf'')[i]\ \texttt{then}\ \sigma'(id_p)(''oaw'')[i]\ \texttt{else}\ 0)}$$

Applying the induction hypothesis to prove the goal,

$$\boxed{\sigma_1(id_p)(''sots'') = \sum_{i=0}^{n} (\texttt{if}\ \sigma_1(id_p)(''otf'')[i]\ \texttt{then}\ \sigma_1(id_p)(''oaw'')[i]\ \texttt{else}\ 0)}$$

By property of $\Delta, \sigma \vdash \text{d.cs} \rightarrow \sigma_1$,
$\sigma_1(id_p)(''sots'') = \sum_{i=0}^{n} (\texttt{if}\ \sigma(id_p)(''otf'')[i]\ \texttt{then}\ \sigma(id_p)(''oaw'')[i]\ \texttt{else}\ 0)$

➔ We can only prove the goal if we know that $\sigma(id_p)(''otf'') = \sigma_1(id_p)(''otf'')$ and $\sigma(id_p)(''oaw'') = \sigma_1(id_p)(''oaw'')$.

□

1.6.2 Falling Edge and Fired

Lemma 24 (Falling Edge Equal Fired). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, then*
$\forall t, id_t$ s.t. $\gamma(t) = id_t$ and $\sigma'(id_t) = \sigma'_t$, $t \in Fired(s') \Leftrightarrow \sigma'_t(''fired'') = true$.

Proof. Given a $t \in T$, and an id_t, prove both senses of the equivalence.

1. $t \in Fired(s') \Rightarrow \sigma'_t(''fired'') = true$.

   By definition of $t \in Fired(s')$.

   - $t \in Fired(s') \equiv \exists fset \subseteq T$, s.t., $IsFiredSet(s', fset)\ \wedge\ t \in fset$.

   Then, apply Lemma Falling Edge Equal Fired Set.

2. $\sigma'_t(''fired'') = true \Rightarrow t \in Fired(s')$

   We can prove that $\forall sitpn, s, \exists fset$ s.t. $IsFiredSet(s, fset)$.

   Then, by specializing the above lemma, we can apply Lemma Falling Edge Equal Fired Set to complete the goal.

□

Lemma 25 (Falling Edge Equal Fired Set). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, then*
$\forall t, id_t$ s.t. $\gamma(t) = id_t$ and $\sigma'(id_t) = \sigma'_t$ and $\forall fset \subseteq T$,
$IsFiredSet(s', fset) \Rightarrow t \in fset \Leftrightarrow \sigma'_t(''fired'') = true$.

Proof. Given a $t \in T$, a $fset \subseteq T$ and a proof of $IsFiredSet(s', fset)$. Unfold the definition of the $IsFiredSet$ relation:

- $IsFiredSet(s', fset) \equiv IsFiredSetAux(s', \emptyset, T, fset)$.

Then, apply Lemma Falling Edge Equal Fired Set Aux.

□

Lemma 26 (Falling Edge Equal Fired Set Aux)**.** *For all* $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' *that verify the hypotheses of Def. 11, and*
$\forall t, id_t$ *s.t.* $\gamma(t) = id_t$ *and* $\sigma'(id_t) = \sigma'_t$ *and* $\forall fired \subseteq T,\ T_i \subseteq T,\ fset \subseteq T$, *assume that:*

- $IsFiredSetAux(s', fired, T_i, fset)$
- *EH (Extra. Hypothesis):*
  $\forall t', id_{t'}, (t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true})$
  $\wedge\ (\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired\ \vee\ t' \in T_i)$.

then $t \in fset \Leftrightarrow \sigma'_t(''fired'') = \texttt{true}$.

Proof. Given a $t, id_t, fired, T_i, fset$, reason by induction on $IsFiredSetAux$.

- BASE CASE. Trivial.
- IND. CASE.
  - $IsTopPriorityList(T_i, \emptyset, \emptyset, tp)$
  - $ElectFired(s', fired, tp, fired')$
  - $FiredAux(s', fired', T_i \setminus tp, fset)$
  - IH: $\big(\forall t' \in T, id_{t'}, (t' \in fired' \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true})$
    $\wedge\ (\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired'\ \vee\ t' \in T_i \setminus tp)\big) \Rightarrow$
    $t \in fset \Leftrightarrow \sigma'_t(''fired'') = true$.

  Apply IH, then, the new goal is:
  $\forall t', id_{t'},\ (t' \in fired' \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true})$
  $\wedge\ (\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired'\ \vee\ t' \in T_i \setminus tp)$
  Apply Lemma Elect Fired Equal Fired to solve the goal.

□

Lemma 27 (Elect Fired Equal Fired)**.** *For all* $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' *that verify the hypotheses of Def. 11,and*
$\forall t, id_t,\ fired, fired' \subseteq T,\ T_i,\ tp \subseteq T_i,\ fset$, *assume that:*

- $IsTopPriorityList(T_i, \emptyset, \emptyset, tp)$
- $ElectFired(s', fired, tp, fired')$
- $FiredAux(s', fired', T_i \setminus tp, fset)$
- *EH (Extra. Hypothesis):*
  $\forall t', id_{t'},$
  $(t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true}) \wedge (\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired\ \vee\ t' \in T_i)$

then
$(t \in fired' \Rightarrow \sigma'_t("fired") = \texttt{true}) \wedge (\sigma'_t("fired") = \texttt{true} \Rightarrow t \in fired' \vee t \in T_i \setminus tp)$.

Proof. Reason by induction on the $ElectFired$ relation.
BASE CASE. Trivial.
2 INDUCTIVE CASES.

1. CASE t_0 is elected to be fired.

   - $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t_0\} \cup tp)$
   - $ElectFired(s', fired \cup \{t_0\}, tp, fired')$
   - $t_0 \in Firable(s')$
   - $t_0 \in Sens(s'.M - \sum_{t_i \in Pr(t_0, fired)} pre(t_i))$
   - $Pr(t_0, fired) = \{t_i | t_i \succ t_0 \wedge t_i \in fired\}$
   - EH: $\forall t' \in T, id_{t'}, (t' \in fired \Rightarrow \sigma'_{t'}("fired") = \texttt{true})$
     $\wedge\ (\sigma'_{t'}("fired") = \texttt{true} \Rightarrow t' \in fired\ \vee\ t' \in T_i)$
   - IH:
     $\forall T_i \subseteq T,\ (\forall t' \in T, id_{t'}, (t' \in fired \cup \{t_0\} \Rightarrow \sigma'_{t'}("fired") = \texttt{true}) \wedge$
     $(\sigma'_{t'}("fired") = \texttt{true} \Rightarrow t' \in fired \cup \{t_0\}\ \vee\ t' \in T_i)) \Rightarrow$
     $IsTopPriorityList(T_i, \emptyset, \emptyset, tp) \Rightarrow$
     $\forall t \in T,\ (t \in fired' \Rightarrow \sigma'_t("fired") = \texttt{true})$
     $\wedge\ (\sigma'_t("fired") = \texttt{true} \Rightarrow t \in fired' \vee t \in T_i \setminus tp)$

   GOAL:
   $\forall t \in T,\ (t \in fired' \Rightarrow \sigma'_t("fired") = \texttt{true})$
   $\wedge\ (\sigma'_t("fired") = \texttt{true} \Rightarrow t \in fired' \vee t \in T_i \setminus \{t_0\} \cup tp)$

   Apply IH with $T_i \setminus \{t_0\}$, then, the hard case to prove is:
   $\forall t' \in T, id_{t'},\ (t' \in fired \cup \{t_0\} \Rightarrow \sigma'_{t'}("fired") = \texttt{true}) \wedge$
   $(\sigma'_{t'}("fired") = \texttt{true} \Rightarrow t' \in fired \cup \{t_0\}\ \vee\ t' \in T_i \setminus \{t_0\})$

   (a) Assume $t' \in fired \cup \{t_0\}$, prove $\sigma'_{t'}("fired") = \texttt{true}$.
   - If $t' \in fired$, then assumption.
   - If $t' = t_0$, then, introduce the expression qualifying "fired": $\sigma_{t'}("fired") = \sigma_{t'}("s\_firable").\sigma_{t'}("s\_priority\_combination")$

     Then, we can show that:
     – $\sigma_{t'}("s\_firable") = \texttt{true}$ by applying Lemma Falling Edge Equal Firable
     – $\sigma_{t'}("s\_priority\_combination") = \texttt{true}$ by applying Lemma Stabilize Compute Priority Combination After Falling Edge.

     Then, it is trivial to show that $\sigma_{t'}("fired") = \texttt{true}$.

   (b) Assume $\sigma'_{t'}("fired") = \texttt{true}$, prove $t' \in fired \cup \{t_0\}\ \vee\ t' \in T_i \setminus \{t_0\}$.

   Thanks to EH, we know that: $t' \in fired \vee t' \in T_i$.

- CASE $t' \in fired$, trivial to show $t' \in fired \cup \{t_0\}$.
- CASE $t' \in T_i$. We know that $t_0 \in T_i$, therefore, either $t' \in T_i \setminus \{t_0\}$ (assumption) or $t' = t_0$ (then, $t' \in fired \cup \{t_0\}$).

2. CASE t_0 is not elected to be fired.

- $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t_0\} \cup tp)$
- $ElectFired(s', fired, tp, fired')$
- $\neg\big(t_0 \in Firable(s') \land t_0 \in Sens(s'.M - \sum_{t_i \in Pr(t_0, fired)} pre(t_i))\big)$
- $Pr(t_0, fired) = \{t_i | t_i \succ t_0 \land t_i \in fired\}$
- EH:
  $\forall t' \in T, id_{t'},$
  $(t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true}) \land (\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired \ \lor \ t' \in T_i).$
- IH:
  $\forall T_i \subseteq T,$
  $\big(\forall t' \in T, id_{t'}, (t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true}) \land$
  $(\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired \ \lor \ t' \in T_i)\big) \Rightarrow$
  $IsTopPriorityList(T_i, \emptyset, \emptyset, tp) \Rightarrow$
  $\forall t \in T, \ \big(t \in fired' \Rightarrow \sigma'_t(''fired'') = \texttt{true}\big)$
  $\land \ \big(\sigma'_t(''fired'') = \texttt{true} \Rightarrow t \in fired' \lor t \in T_i \setminus tp\big)$

GOAL:
$\forall t \in T, \ \big(t \in fired' \Rightarrow \sigma'_t(''fired'') = \texttt{true}\big)$
$\land \ \big(\sigma'_t(''fired'') = \texttt{true} \Rightarrow t \in fired' \lor t \in T_i \setminus \{t_0\} \cup tp\big)$

Apply IH with $T_i \setminus \{t_0\}$, then, the hard case to prove is:
$\forall t' \in T, id_{t'}, \ (t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true}) \land$
$(\sigma'_{t'}(''fired'') = \texttt{true} \Rightarrow t' \in fired \ \lor \ t' \in T_i \setminus \{t_0\})$

(a) Prove $t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \texttt{true}$ (assumption).

(b) Assume $\sigma'_{t'}(''fired'') = \texttt{true}$, prove $t' \in fired \ \lor \ t' \in T_i \setminus \{t_0\}$.

Thanks to EH, we know that: $t' \in fired \lor t' \in T_i$.

- CASE $t' \in fired$ (assumption).
- CASE $t' \in T_i$. We know that $t_0 \in T_i$, therefore, either $t' \in T_i \setminus \{t_0\}$ (assumption) or $t' = t_0$.

  Then, we need to show a contradiction by proving
  $t' \in Firable(s') \land t' \in Sens(s'.M - \sum_{t_i \in Pr(t', fired)} pre(t_i))$
  based on $\sigma'_{t'}(''fired'') = \texttt{true}$.
  We know

$$\begin{aligned} \sigma'_{t'}(''fired'') = & \ \sigma'_{t'}(''s\_firable'').\sigma'_{t'}(''s\_priority\_combination'') \\ = & \ \texttt{true} \end{aligned}$$

- Show $t' \in Firable(s')$ by applying Lemma Falling Edge Equal Firable.
- Show $t' \in Sens(s'.M - \sum_{t_i \in Pr(t', fired)} pre(t_i))$ by applying Lemma Stabilize Compute Priority Combination After Falling Edge (needs a proof of $t \in Firable(s')$ to be applied).

□

Lemma 28 (Stabilize Compute Priority Combination After Falling Edge)**.** *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, and*
$\forall t, id_t, \sigma'_t$ s.t. $\gamma(t) = id_t$ and $\sigma'(id_t) = \sigma'_t$,
$\forall fired, fired', T_i, tp, fset$, assume that:

- $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t\} \cup tp)$
- $ElectFired(s', fired, tp, fired')$
- $FiredAux(s', fired', T_i \setminus \{t\} \cup tp, fset)$
- *EH (Extra. Hypothesis):*
  $\forall t' \in T, id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'_{t'}("fired") = \texttt{true}) \wedge (\sigma'_{t'}("fired") = \texttt{true} \Rightarrow t' \in fired \vee t' \in T_i)$.
- $t \in Firable(s')$

then
$t \in Sens(s'.M - \sum_{t_i \in Pr(t, fired)} pre(t_i)) \Leftrightarrow \sigma'_t("s\_priority\_combination") = \texttt{true}$

Proof. We know $\sigma'_t("s\_priority\_combination") = \prod_{i=0}^{|input(t)|-1} \sigma'_t("pauths")(i)$. Then, apply Lemma Stabilize Compute Priority Authorizations After Falling Edge to solve the goal. □

Lemma 29 (Stabilize Compute Priority Authorizations After Falling Edge)**.** *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, and*
$\forall t, id_t, \sigma'_t$ s.t. $\gamma(t) = id_t$ and $\sigma'(id_t) = \sigma'_t$,
$\forall fired, fired', T_i, tp, fset$, assume that:

- $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t\} \cup tp)$
- $ElectFired(s', fired, tp, fired')$
- $FiredAux(s', fired', T_i \setminus \{t\} \cup tp, fset)$
- *EH (Extra. Hypothesis):*
  $\forall t' \in T, id_{t'}$,
  $(t' \in fired \Rightarrow \sigma'_{t'}("fired") = \texttt{true}) \wedge (\sigma'_{t'}("fired") = \texttt{true} \Rightarrow t' \in fired \vee t' \in T_i)$.
- $t \in Firable(s')$

then
$t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i)) \Leftrightarrow$
$\forall i \in [0, \sigma'_t(''input\_arcs\_number'') - 1],\ \sigma'_t(''pauths'')(i) = \texttt{true}$

Proof. Show the two sides of the equivalence.

1. Assume $t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i))$,
   show $\forall i \in [0, \sigma'_t(''input\_arcs\_number'') - 1],\ \sigma'_t(''pauths'')(i) = \texttt{true}$.

   Reason on the cardinality of the set of input places of t. 2 CASES.

   - CASE $|input(t)| = 0$.
     Then $\sigma'_t(''input\_arcs\_number'') = 1$ and $i = 0$.
     Then, by construction, $\texttt{id}_\texttt{t}\texttt{.pauths(0)}$ is connected to the `true` constant in the input map of Transition component id_t.
     Then, $\sigma'_t(''pauths'')(0) = \texttt{true}$.
   - CASE $|input(t)| > 0$.
     Then, for all $i \in [0, \sigma'_t(''input\_arcs\_number'') - 1]$, exists a place p and an arc a such that $pre(p, t) = a$.
     Then, by construction, there exists a Place component id_p implementing place p.
     Reason on a.
     - CASE $a = (\omega, \texttt{test})$ or $a = (\omega, \texttt{inhib})$.
       Then, by construction, $\texttt{id}_\texttt{t}\texttt{.pauths(i)}$ is connected to the `true` constant in the input map of Transition component id_t.
       Then, $\sigma'_t(''pauths'')(i) = \texttt{true}$.
     - CASE $a = (\omega, \texttt{basic})$, then 2 CASES.
       * CASE For all pair of transitions in $output_c(p)$, all conflicts are solved by mutual exclusion.
         Then, by construction, $\texttt{id}_\texttt{p}\texttt{.pauths}$ is an unconnected (i.e, `open`) port, and $\texttt{id}_\texttt{t}\texttt{.pauths(i)}$ is connected to the `true` constant.
         Then, $\sigma'_t(''pauths'')(i) = \texttt{true}$.
       * CASE The priority relation is a strict total order over the set $output_c(p)$.
         Then, by construction, there exists an index j and a signal sig connecting $\texttt{id}_\texttt{p}\texttt{.pauths(j)}$ to $\texttt{id}_\texttt{t}\texttt{.pauths(i)}$.
         Then, we can deduce that $\sigma'_t(''pauths'')(i) = \sigma'(''sig'') = \sigma'_p(''pauths'')(j)$.
         Then, we can specialize the definition of $t \in Sens(s'.M - \sum_{t_i \in Pr(t,fired)} pre(t_i))$ with place p, and $pre(p, t) = (\omega, \texttt{basic})$ to get $s'.M(p) - \sum_{t_i \in Pr(t,fired)} pre(p, t_i) \geq \omega$.
         Then, we can show that $\sigma'_p(''pauths'')(j) = \texttt{true}$ by applying Lemma Stabilize Compute Individual Priority Authorization After Falling Edge.
         Then, the goal is trivially solved by rewriting.

2. Assume $\forall i \in [0, \sigma'_t(''input\_arcs\_number'') - 1]$, $\sigma'_t(''pauths'')(i) = \texttt{true}$, show $t \in Sens(s'.M - \sum\limits_{t_i \in Pr(t,fired)} pre(t_i))$.

   Then, unfold the definition of the *Sens* relation.
   $\forall p \in P, \omega \in \mathbb{N}^*$,
   $\big(pre(p,t) = (\omega, \texttt{basic}) \vee pre(p,t) = (\omega, \texttt{test}) \Rightarrow$
   $s'.M(p) - \sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega\big)$
   $\wedge \big(pre(p,t) = (\omega, \texttt{inhib})\big) \Rightarrow s'.M(p) - \sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) < \omega\big)$

   Then, treat the 3 different cases.

   (a) Assume $pre(p,t) = (\omega, \texttt{test})$,
   show $s'.M(p) - \sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega$.
   Then, by assuming that the priority relation is well-defined, there exists no transition t_i connected by a `basic` arc to p that verified $t_i \succ t$. This is because t is connected to p by a `test` arc; thus, t is not in conflict with the other output transitions of p; thus, there is no relation of priority between t and the output of p.
   Then, we can deduce that $\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = 0$.
   Then, the new goal is $s'.M(p) \geq \omega$.
   That we can prove because we know $t \in Firable(s')$, thus, $t \in Sens(s'.M)$, thus, $s'.M(p) \geq \omega$.

   (b) Assume $pre(p,t) = (\omega, \texttt{inhib})$,
   show $s'.M(p) - \sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) < \omega$.
   Use the same strategy as above.

   (c) Assume $pre(p,t) = (\omega, \texttt{basic})$,
   show $s'.M(p) - \sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) \geq \omega$.
   Then, there are 2 CASES.

      i. CASE For all pair of transitions in $output_c(p)$, all conflicts are solved by mutual exclusion.
      Then, assuming that the priority relation is well-defined, it must not be defined over the set $output_c(t)$, and we know that $t \in output_c(p)$ since $pre(p,t) = (\omega, \texttt{basic})$.
      Then, there exists no transition t_i connected to p by a `basic` arc that verifies $t_i \succ t$.
      Then, we can deduce $\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = 0$.
      Then, the new goal is $s'.M(p) \geq \omega$.
      We know $t \in Firable(s')$, thus, $t \in Sens(s'.M)$, thus, $s'.M(p) \geq \omega$.

      ii. CASE The priority relation is a strict total order over the set $output_c(p)$.
      Assuming $pre(p,t) = (\omega, \texttt{basic})$, then, by construction, there exist:
      - a Place component id_p implementing place p

- two indexes $i \in [0, \sigma'_t(''input\_arcs\_number'') - 1]$ and $j \in [0, \sigma'_p(''output\_arcs\_number'') -$ $1]$
- a signal sig connecting $\mathtt{id_p.pauths(j)}$ to $\mathtt{id_t.pauths(i)}$

Then, we can deduce that $\sigma'_t(''pauths'')(i) = \sigma'(''sig'') = \sigma'_p(''pauths'')(j)$.
Then, by specializing $\forall i \in [0, \sigma'_t(''input\_arcs\_number'') - 1]$, $\sigma'_t(''pauths'')(i) =$ $\mathtt{true}$ with i, we can deduce $\sigma'_t(''pauths'')(i) = \sigma'(''sig'') = \sigma'_p(''pauths'')(j) =$ $true$.
Then, we have all the premises necessary to apply Lemma Stabilize Compute Individual Priority Authorization After Falling Edge, and thus to solve the goal.

□

Lemma 30 (Stabilize Compute Individual Priority Authorization After Falling Edge). *For all $sitpn$, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, and*
$\forall t, id_t, \sigma'_t$, *s.t.* $\gamma(t) = id_t$ *and* $\sigma'(id_t) = \sigma'_t$,
$\forall p, id_p, \sigma'_p$, *s.t.* $\gamma(p) = id_p$ *and* $\sigma'(id_p) = \sigma'_p$,
$\forall fired, fired', T_i, tp, fset, sig \in Sigs(\Delta), i, j \in \mathbb{N}, \omega \in \mathbb{N}$, *assume that:*

- $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t\} \cup tp)$
- $ElectFired(s', fired, tp, fired')$
- $FiredAux(s', fired', T_i \setminus \{t\} \cup tp, fset)$
- *EH (Extra. Hypothesis):*
  $\forall t' \in T, id_{t'},$
  $(t' \in fired \Rightarrow \sigma'_{t'}(''fired'') = \mathtt{true}) \land (\sigma'_{t'}(''fired'') = \mathtt{true} \Rightarrow t' \in fired \ \lor \ t' \in T_i)$.
- $\mathtt{id_p.pauths(j)} \Rightarrow \mathtt{sig} \Rightarrow \mathtt{id_t.pauths(i)}$
- $pre(p, t) = (\omega, \mathtt{basic})$

then $\sigma'_p(''pauths'')(j) = \mathtt{true} \Leftrightarrow s'.M(p) - \sum_{t_i \in Pr(t, fired)} pre(p, t_i) \geq \omega$.

Proof. From the behavior of the VHDL Place component, we can deduce:
$\sigma'_p(''pauths'')(j) = true \Leftrightarrow \sigma'_p(''s\_marking'') - \sum_{k \in HPF(\sigma'_p, j)} \sigma'_p(''out\_arc\_w'')(k) \geq \sigma'_p(''out\_arc\_w'')(j)$ where $k \in HPF(\sigma'_p, j) \equiv k \in [0, j-1] \land \sigma'_p(''out\_arc\_t'')(k) = \mathtt{basic} \land \sigma'_p(''out\_t\_fired'')(k) = \mathtt{true}$
Then, the new goal is:
$\sigma'_p(''s\_marking'') - \sum_{k \in HPF(\sigma'_p, j)} \sigma'_p(''out\_arc\_w'')(k) \geq \sigma'_p(''out\_arc\_w'')(j) \Leftrightarrow s'.M(p) -$ $\sum_{t_i \in Pr(t, fired)} pre(p, t_i) \geq \omega$.
Proof by reflexivity. 3 subgoals.

1. Show $s'.M(p) = \sigma'_p(''s\_marking'')$.

   From $\gamma \vdash s \sim \sigma$, we know $s.M(p) = \sigma_p(''s\_marking'')$.

   From $E_c, \tau \vdash sitpn, s \xrightarrow{\downarrow} s'$, we know $s.M(p) = s'.M(p)$.

By reasoning on the VHDL falling and stabilize relations, and on the Place component behavior, we know that the "s_marking" is idle from state σ_p to state σ'_p; thus, $\sigma_p(''s\_marking'') = \sigma'_p(''s\_marking'')$.

Then, the goal is trivially proved by using the rewriting rules.

2. Show $\omega = \sigma'_p(''out\_arc\_w'')(j)$.

   We know that $pre(p,t) = (\omega, \texttt{basic})$ and $\texttt{id}_\texttt{p}\texttt{.pauths(j)} \Rightarrow \texttt{sig} \Rightarrow \texttt{id}_\texttt{t}\texttt{.pauths(i)}$.

   Then, by construction, $\texttt{id}_\texttt{p}\texttt{.output\_arcs\_weights(j)}$ is connected to the constant ω in the input map of Place component id_p.

   Then, the goal is trivially solved by showing that ports that are mapped to constant are idle during the simulation of a VHDL design.

3. Show $\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = \sum\limits_{k \in HPF(\sigma'_p,j)} \sigma'_p(''out\_arc\_w'')(k)$.

   We can show $\sum\limits_{t_i \in Pr(t,fired)} pre(p,t_i) = \sum\limits_{t_i \in Pr(p,t,fired)} pre(p,t_i)$
   where $t_i \in Pr(p,t,fired) \equiv t_i \succ t \wedge t_i \in fired \wedge \exists \omega \in \mathbb{N},\ s.t.,\ pre(p,t_i) = (\omega, \texttt{basic})$.

   Then, we can show that the sets $Pr(p,t,fired)$ and $HPF(\sigma'_p,j)$ are in bijection, and that for each $t_i \in Pr(p,t,fired)$ mapped to a $k \in HPF(\sigma'_p,j)$, we have $pre(p,t_i) = \sigma'_p(''out\_arc\_w'')(k)$.

   2 subgoals to solve.

   (a) $\forall t_i \in Pr(p,t,fired), \exists k \in HPF(\sigma'_p,j)\ s.t.\ pre(p,t_i) = \sigma'_p(''out\_arc\_w'')(k)$.
   Given a transition $t_i \in Pr(p,t,fired)$, show $\exists k \in HPF(\sigma'_p,j)\ s.t.\ pre(p,t_i) = \sigma'_p(''out\_arc\_w'')(k)$.
   Unfold the definition of $t_i \in Pr(p,t,fired)$:

   - $\exists \omega \in \mathbb{N}\ s.t.\ pre(p,t_i) = (\omega, \texttt{basic})$.
     Let us call ω' the element of $\mathbb{N}^*$ verifying $pre(p,t_i) = (\omega', \texttt{basic})$.
     Then, by construction, there exists a Transition component id_{t_i} implementing transition t_i and an index $n \in \mathbb{N}^*$ such that $\texttt{id}_\texttt{p}\texttt{. output\_arcs\_weights(n)}$ is connected to ω' and
     $\texttt{output\_arcs\_types(n)}$ is connected to $\texttt{basic}$.
     Then, by reasoning on the VHDL falling and stabilize relation, we can show that $\sigma'_p(''output\_arcs\_weights'')(n) = \omega'$.
   - $t_i \succ t$.
     By construction, there exists an index $m \in \mathbb{N}^*$ and a signal $sig' \in \texttt{Declared}(\Delta)$ such that $\texttt{id}_\texttt{p}\texttt{.pauths(n)} \Rightarrow \texttt{sig}' \Rightarrow \texttt{id}_{\texttt{t}_\texttt{i}}\texttt{.pauths(m)}$
     Then, by construction, and since $t_i \succ t$, we know that $n < j$. Then, $n \in [0, j-1]$.
   - $t_i \in fired$.
     Thanks to the EH, we know that $\sigma'_{t_i}(''fired'') = \texttt{true}$.
     By construction, there exists a signal $sig'' \in \texttt{Declared}(\Delta)$ such that $\texttt{id}_{\texttt{t}_\texttt{i}}\texttt{.fired} \Rightarrow \texttt{sig}'' \Rightarrow \texttt{id}_\texttt{p}\texttt{.output\_transitions\_fired(n)}$.
     Then, by reasoning on the VHDL stabilize relation, we can deduce $\sigma'_p(''output\_transitions\_fired'')(n) = \sigma'_{t_i}(''fired'') = \texttt{true}$.

Then, we have $n \in HPF(\sigma'_p, j)$ and $pre(p, t_i) = \sigma'_p(''output\_arcs\_weights'')(n)$. Thus, let us take n to prove the goal by assumption.

(b) $\forall k \in HPF(\sigma'_p, j), \exists t_i \in Pr(p, t, fired)\ s.t.\ pre(p, t_i) = \sigma'_p(''out\_arc\_w'')(k)$.
Given an index $k \in HPF(\sigma'_p, j)$, show $\exists t_i \in Pr(p, t, fired)\ s.t.\ pre(p, t_i) = \sigma'_p(''out\_arc\_w'')(k)$.
Unfold the definition of $k \in HPF(\sigma'_p, j)$:

- $k \in [0, j-1]$.
  By construction, there exists a $t_i \in T$ and an $\omega' \in \mathbb{N}^*$ such that $pre(p, t_i) = (\omega', \texttt{basic})$ and $t_i \succ t$ and $\mathtt{id_p.output\_arcs\_weights(k)} \Rightarrow \mathtt{!}'$ and $\mathtt{id_p.output\_arcs\_types(k)} \Rightarrow \mathtt{basic}$.
- $\sigma'_p(''output\_transitions\_fired'')(k) = \mathtt{true}$.
  By construction, there exists a Transition component id_{t_i} implementing transition t_i such that $\mathtt{id_{t_i}.fired} \Rightarrow \mathtt{id_p.output\_transitions\_fired(k)}$. Then, by reasoning on the VHDL falling and stabilize relations, we can deduce $\sigma'_p(''output\_transitions\_fired'')(k) = \sigma'_{t_i}(''fired'') = \mathtt{true}$. Then, thanks to EH, we know that $t_i \in fired$ or $t_i \in T_i$.
  - CASE $t_i \in fired$. Then, take t_i to prove the goal by assumption.
  - CASE $t_i \in T_i$.
    Since t is a *top-priority* transition of set T_i (given by $IsTopPriorityList(T_i, \emptyset, \emptyset, \{t\} \cup tp)$, then there exists no transition $t' \in T_i$ such that $t' \succ t$. Since $t_i \in T_i$, then we have $t_i \not\succ t$ contradicting $t_i \succ t$.

□

1.6.3 Falling Edge and Firable

Lemma 31 (Falling Edge Equal Firable). *For all sitpn, d, γ, Δ, σ_e, E_c, E_p, τ, s, s', σ, σ_i, $\sigma_\downarrow$, θ, σ' that verify the hypotheses of Def. 11, and $\forall t, id_t$ s.t. $\gamma(t) = id_t$ and $\sigma'(id_t) = \sigma'_t$, then $t \in Firable(s') \Leftrightarrow \sigma'_t(''s\_firable'') = \mathtt{true}$.*

Proof. □

Appendix A

Reminder on natural semantics

Appendix B

Reminder on induction principles

- Present all the material that will be used in the proof, and that needs clarifying for people who do not come from the field (e.g, automaticians and electronicians)
  - structural induction
  - induction on relations
  - ...